DaoCloud Enterprise

云原生容器平台 · 应用商店模块

应 用 商 店

用 户 手 册

文档版本　v1.0

内容来源　docs.daocloud.io/kpanda/user-guide/helm

整理日期　2026 年 10 月 27 日

共 5 章 • 11 节

道客 DaoCloud

关于本手册

本手册讲的是容器管理模块中的应用商店：一个把 Helm 模板集中陈列、把安装与运维收进界面的地方。 全书的组织方式与一次真实的使用顺序一致——先理解商店的构成（第 1 章），再备好货架（第 2 章）， 然后把要用的应用上架（第 3 章），最后才是安装、更新与卸载（第 4 章）；第 5 章把入口、命令与常见失败收在一处速查。

- **与界面术语的对应**：书中沿用「应用商店」的说法——仓库是**货架**、Chart 是**商品**（应用模板）、Release 是**我的应用**（已安装应用）； 凡涉及点击路径的地方仍按界面原文写，例如 **Helm 应用->Helm 模板**，两者在 1.2 的对照表里一一对应。
- **编排方式**：章（第 X 章）对应用商店的一段使用流程，节（X.Y）对应官网的一个文档页面，页面内小节保留原文层级。
- **交叉引用**：原文中指向其他文档的链接已改为纯文本，并在其后标注「见 X.Y 节」，可按编号在本手册内检索。
- **提示框**：提示块以「说明 / 提示 / 警告 / 注意」加粗标签开头，「警告」多为会引发故障的操作风险。
- **配图**：截图取自官方文档，用于示意界面位置与字段，实际界面可能随产品版本略有差异。

内容版权归 DaoCloud 所有，本手册仅用于产品学习、交付与培训参考。

目　录

第 1 章　走进应用商店 5

1.1　应用商店是什么 5

1.2　三个核心概念 5

1.3　商店的整体结构 6

1.4　一次完整的使用旅程 7

第 2 章　货架管理：应用仓库 9

2.1　管理应用仓库 9

前提条件 10

引入第三方 Helm 仓库 10

更新 Helm 仓库 12

删除 Helm 仓库 14

第 3 章　商品上架：上传、导入与离线交付 17

3.1　上传应用模板 17

方式一：客户端推送 17

方式二：页面上传 18

同步仓库：让新模板出现在目录里 18

3.2　将自定义应用导入内置 addon 仓库 19

离线环境 20

前提条件 20

同步 Helm Chart 20

在线环境 22

3.3　多架构应用的导入与升级 22

导入 23

单架构导入 23

多架构融合 23

升级 23

单架构升级 23

多架构融合 23

注意事项 24
磁盘空间 24
失败后重试 24
镜像空间 24
架构冲突 24
第 4 章　选购与运维：安装、更新与卸载应用 26
4.1　安装、更新与卸载应用 26
前提条件 26
安装 Helm 应用 26
更新 Helm 应用 29
注意事项： 31
查看 Helm 操作记录 31
删除 Helm 应用 33
第 5 章　速查与排障 35
5.1　入口与命令速查 35
5.2　常见失败与处理 36

第 1 章　走进应用商店

装一个 Kubernetes 组件通常要三步：找到 Chart、改好 values、在命令行里跑 `helm install`。应用商店把这三步收进了一个界面：模板集中陈列、版本一目了然、参数在页面上改完就能装。本章先讲清它的定位、构成与使用顺序，具体操作留给后面三章。

1.1　应用商店是什么

应用商店是容器管理模块对 Helm 能力的产品化封装。Helm 是 Kubernetes 的包管理工具，方便用户快速发现、共享和使用 Kubernetes 上构建的应用；第五代容器管理模块提供了上百个 Helm 模板，涵盖存储、网络、监控、数据库等主要场景。借助这些模板，用户可以在界面上完成应用的部署与日常管理，不必自己维护一套 Helm 命令行环境。

对使用者来说，它提供三样东西：

- **一个可浏览的目录**：模板按仓库分组陈列，点进去能看到版本与参数配置界面，选定即可安装。
- **一条可扩展的供货渠道**：除系统预置仓库外，还能接入第三方 Helm 仓库，把自有应用推进商店。
- **一本可追溯的台账**：每次安装、更新、删除都会留下一条操作记录，并可在线展开运行日志。

> **先说清边界**
>
> 商店本身不生产应用。它能装什么，取决于仓库里有什么——这也决定了本手册的顺序：先备货架（第 2 章），再上架（第 3 章），最后才是安装与运维（第 4 章）。
>
> **三个入口**
>
> 应用商店没有单独的顶级菜单，入口都在集群详情页的 **Helm 应用** 之下：**Helm 模板** 是商品目录，**Helm 仓库** 是货架管理，**Helm 应用** 是已安装应用的台账。三者对应第 1.3 节的三层结构。

商店里的动作分三类人来做，本手册的章节划分与这三类动作一致：

角色	在商店里做什么	对应章节
平台管理员	接入或移除仓库，把离线包与自有应用上架，保证目录完整可用	第 2、3 章
应用运维 / 开发者	浏览模板、选版本、配参数，安装并持续更新应用	第 4 章
交付与支持人员	离线导入与多架构融合，并按操作记录定位失败原因	3.2、3.3、5.2

1.2　三个核心概念

Helm 有三个概念，界面上各有一个中文叫法，混着用很容易对不上。先把对照关系立起来：

应用商店的说法	界面上的位置	Helm 术语	说明
货架 / 应用仓库	Helm 应用 → Helm 仓库	Repository	发布和存储模板包的 HTTP(S) 存储库
商品 / 应用模板	Helm 应用 → Helm 模板	Chart	一个安装包，包含运行应用所需的镜像、依赖与资源定义，还可能包含集群中的服务定义
我的应用 / 已安装应用	Helm 应用	Release	模板在本集群内的一次安装实例
版本	安装页右上角的版本选择	chart version	同一模板的不同发布版本，可在仓库中并存
上架	推送 Chart + 同步仓库	helm push / repo sync	让模板出现在商店目录中的那一步动作

三个概念里最容易被忽略的是 Release：**同一个模板可以在同一集群内安装多次，每次安装都生成一个独立的 Release**。例如在不同的命名空间各装一套 cert-manager，得到的是两个互不影响的应用，可以分别更新、分别卸载。

Chart 的类比也值得记一句：它类似 Homebrew 中的 formula、APT 的 deb 包或 Yum 的 rpm 文件——都是「一个可安装的包」，区别只在于 Chart 描述的是 Kubernetes 资源。更多细节可查阅 Helm 官网。

1.3 商店的整体结构

三层从上到下叠起来，就是商店的全貌：

1. **应用仓库——货架**。它只保存地址与认证信息，本身不占运行时资源。系统内置 4 个仓库（partner、system、addon、community），分别覆盖生态伙伴、系统核心组件、通用业务场景与社区热门组件，各自的定位与示例见 2.1 节。
2. **应用模板——商品目录**。商店把各仓库中的 Chart 索引成列表，按仓库分组展示，同一模板的多个版本在这里聚合。
3. **已安装应用——我的应用**。选定模板与版本、填好名称与命名空间之后，安装结果就是一条应用记录，状态从「安装中」变为「运行中」。

三层之间只有两条通道，也是本手册的两条主线：**上架**（模板进仓库，第 3 章）与**安装**（模板变应用，第 4 章）。其余操作——建仓库、改参数、看日志——都挂在这两条线上。

还有三点结构性的事实，值得动手前记住：

- **作用域是集群**：仓库、模板、应用都归属于某一个集群，切换集群等于走进另一间商店，目录也随之改变。
- **来源与产物分离**：仓库只是目录来源。删除仓库会让对应模板从列表中消失，但已经安装的应用不受影响；反过来，卸载应用也不会动仓库里的模板。
- **权限照常生效**：安装产物是一组 Kubernetes 资源，仍受命名空间与 RBAC 约束，因此各节的前提条件里都要求目标命名空间的 NS Admin 或更高权限。

1.4 一次完整的使用旅程

把五个步骤串成一条线：

1. **备货架**：确认集群已接入、权限已就绪，按需引入第三方仓库（第 2 章）。
2. **上架**：把模板包推进仓库并同步，让它出现在目录里；离线环境改用 charts-syncer 或离线包导入（第 3 章）。
3. **浏览**：在 **Helm 模板** 页面按仓库分组找到目标模板，在右上角选定版本（4.1 节）。
4. **安装**：填写名称、命名空间，必要时修改参数 YAML，确定后等待状态变为「运行中」（4.1 节）。
5. **运维**：更新参数并对比变化、回看操作记录与日志、必要时卸载（4.1 节）。

前两步是「把东西放进商店」，通常由管理员做一次；后三步是「从商店取东西」，每个使用者都会反复做。如果只在系统预置仓库里挑应用，第 2 章和第 3 章可以整章跳过，直接从 4.1 节开始。

> **读法建议**
>
> 第一次接触这套界面，建议按 1 → 2 → 3 → 4 的顺序通读；只负责装应用的同学，读完本章可直接跳到 4.1 节；负责私有化交付的同学，重点看 3.2 与 3.3 节；现场排障时查第 5 章。

第 2 章　货架管理：应用仓库

商店开张前先把货架备好。一个仓库就是一条供货渠道：地址指向哪里、用什么身份认证、多久刷新一次，决定了商店里能看到哪些模板、以及上游更新后多久能被发现。

本章只有一节，把仓库的引入、更新与删除讲完；系统内置的 4 个仓库作为起点，也在这里交代清楚。

引入仓库与安装应用的前置条件其实是同一套，先合并成一张表，各节仍保留原文表述：

前置条件	要求
集群	已接入 Kubernetes 集群或已创建集群，且能够访问集群的 UI 界面
命名空间	已创建命名空间，安装应用时在其中选择
用户权限	已创建用户并授予 NS Admin 或更高权限
私有仓库	当前操作用户对该仓库拥有读写权限

2.1　管理应用仓库

位置：应用商店 › 货架管理

在应用商店里，仓库就是货架：它存放和发布模板包，商店通过 HTTP(S) 协议读取仓库中的 Chart，再索引成模板目录。系统默认内置了下表所示的 4 个仓库，覆盖企业生产中最常见的四类来源。

仓库	描述	示例
partner	由生态合作伙伴所提供的各类优质特色 Chart	tidb
system	系统核心功能组件及部分高级功能所必需依赖的 Chart，如必需安装 insight-agent 才能够获取集群的监控信息	Insight
addon	业务场景中常见的 Chart	cert-manager
community	Kubernetes 社区较为热门的开源组件 Chart	Istio

除上述预置仓库外，您也可以把第三方 Helm 仓库接入商店，作为自有货架。下文的操作分引入、更新、删除三步。

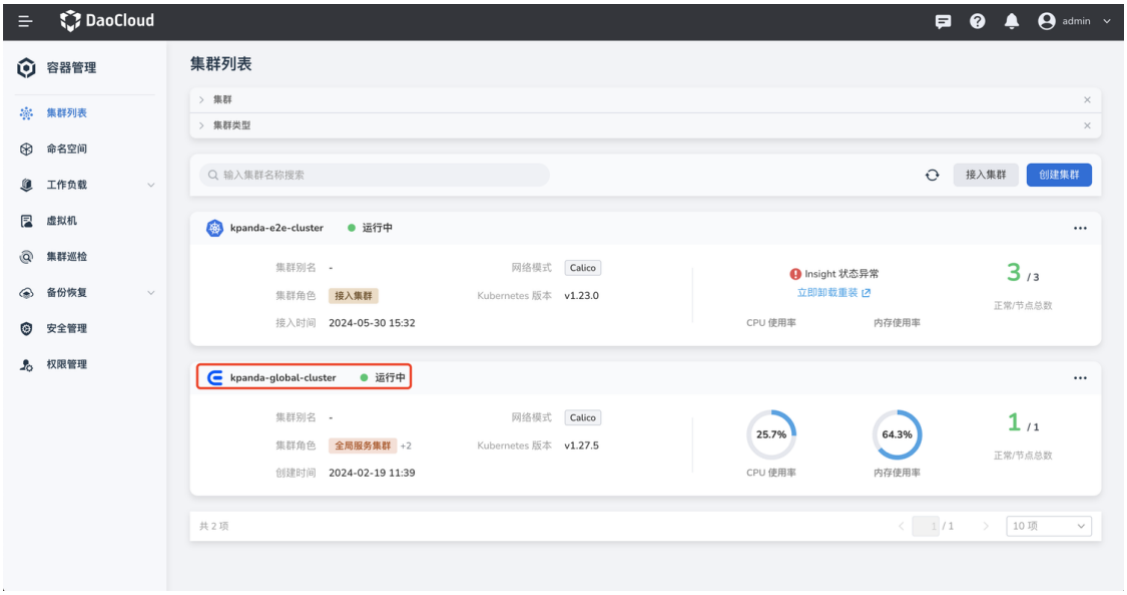
前提条件

- 容器管理模块已接入 Kubernetes 集群或者已创建 Kubernetes 集群，且能够访问集群的 UI 界面
- 已完成一个命名空间的创建、用户的创建，并为用户授予 NS Admin 或更高权限 ，详情可参考命名空间授权。
- 如果使用私有仓库，当前操作用户应拥有对该私有仓库的读写权限。

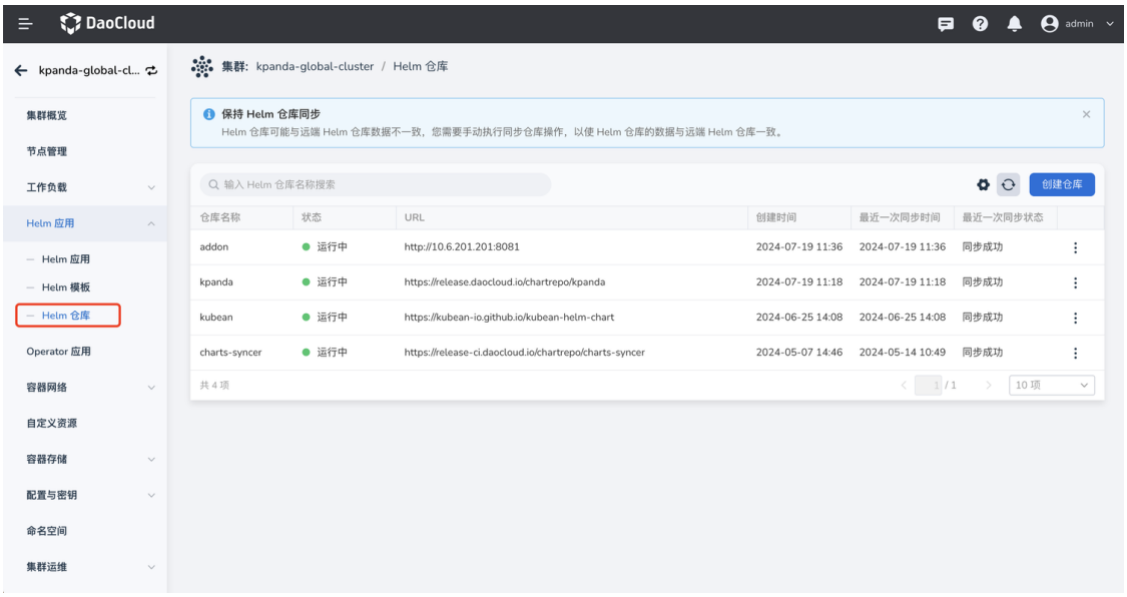
引入第三方 Helm 仓库

下面以 Kubevela 公开的镜像仓库为例，引入 Helm 仓库并管理。

1. 找到需要引入第三方 Helm 仓库的集群，点击集群名称，进入 **集群详情**

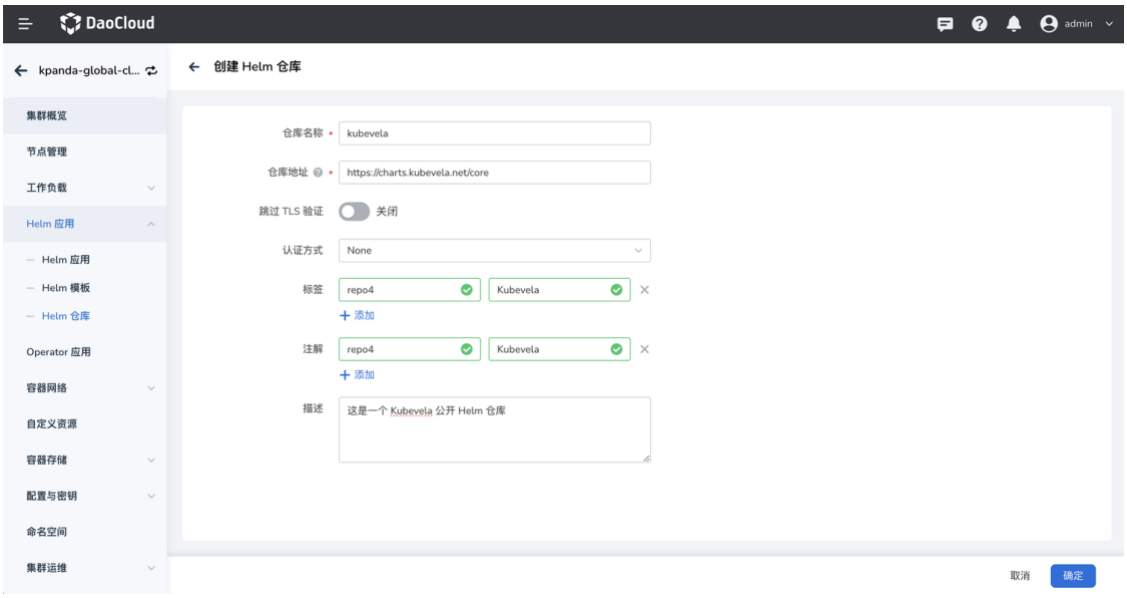


2. 在左侧导航栏，依次点击 **Helm 应用 -> Helm 仓库** ，进入 Helm 仓库页面。

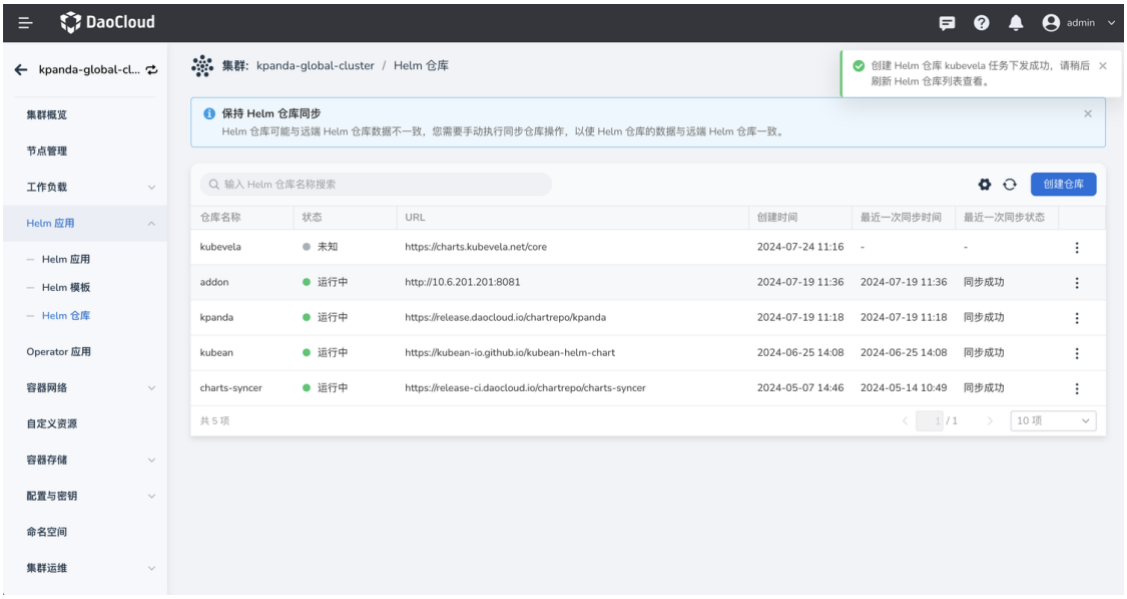


3. 在 Helm 仓库页面点击 **创建仓库** 按钮，进入创建仓库页面，按照下表配置相关参数。

   - 仓库名称：设置仓库名称。最长 63 个字符，只能包含小写字母、数字及分隔符 **-**，且必须以小写字母或数字开头并结尾，例如 kubevela
   - 仓库地址：用来指向目标 Helm 仓库的 http（s）地址。例如 https://kubevela.github.io/charts
   - 跳过 TLS 验证: 如果添加的 Helm 仓库为 https 地址且需跳过 TLS 验证，可以勾选此选项，默认为不勾选
   - 认证方式：连接仓库地址后用来进行身份校验的方式。对于公开仓库，可以选择 **None**，私有的仓库需要输入用户名/密码以进行身份校验
   - 标签：为该 Helm 仓库添加标签。例如 key: repo4；value: Kubevela
   - 注解：为该 Helm 仓库添加注解。例如 key: repo4；value: Kubevela
   - 描述：为该 Helm 仓库添加描述。例如：这是一个 Kubevela 公开 Helm 仓库



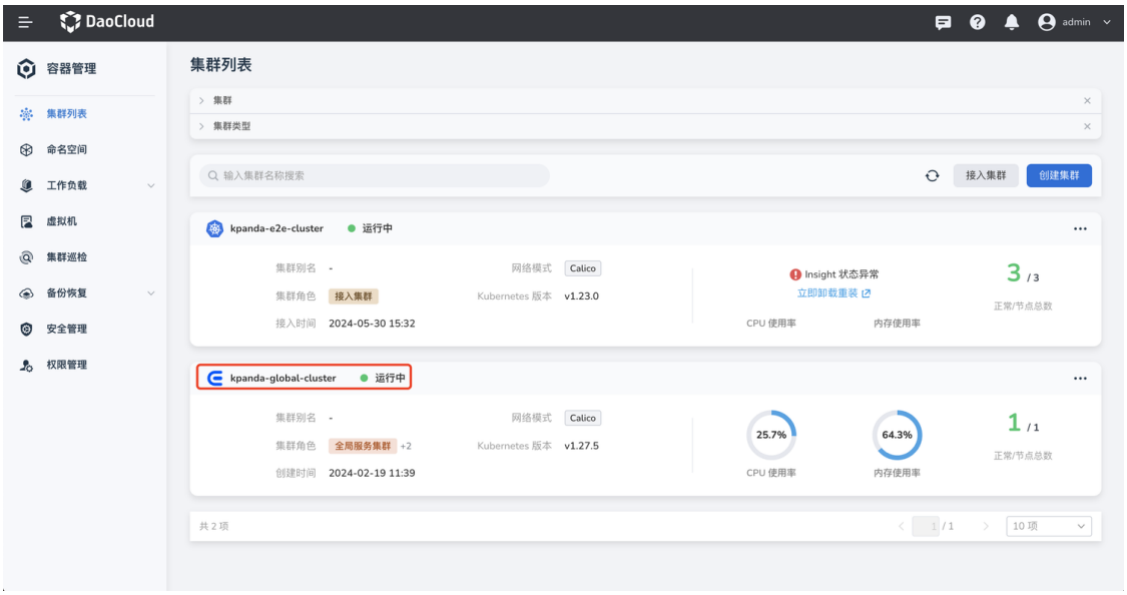
4. 点击 **确定** ，完成 Helm 仓库的创建。页面会自动跳转至 Helm 仓库列表。



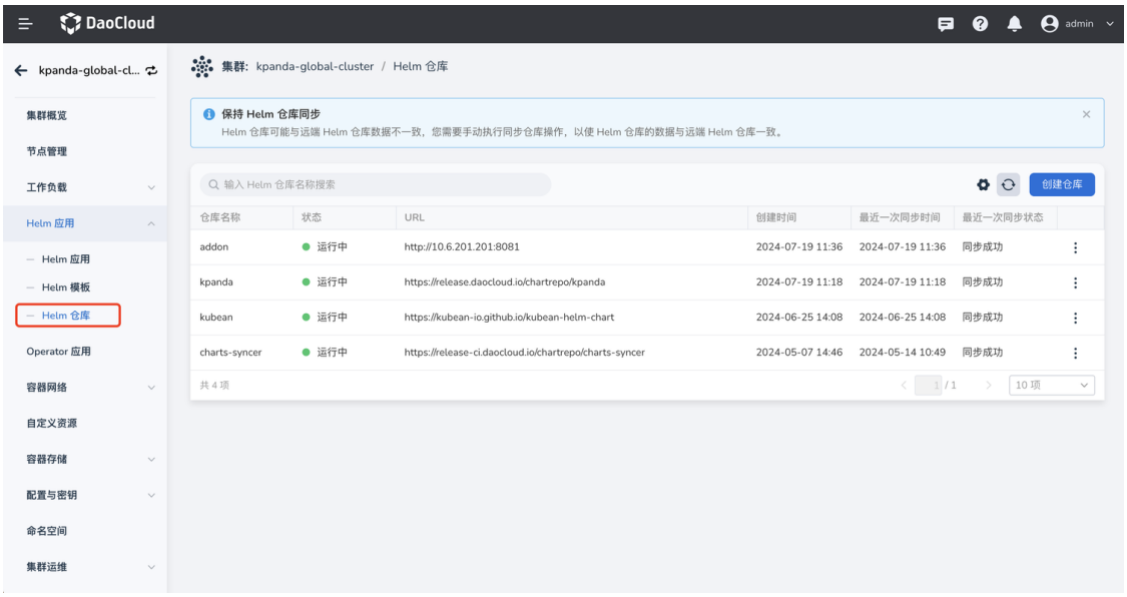
更新 Helm 仓库

当 Helm 仓库的地址信息发生变化时，可以更新 Helm 仓库的地址、认证方式、标签、注解及描述信息。

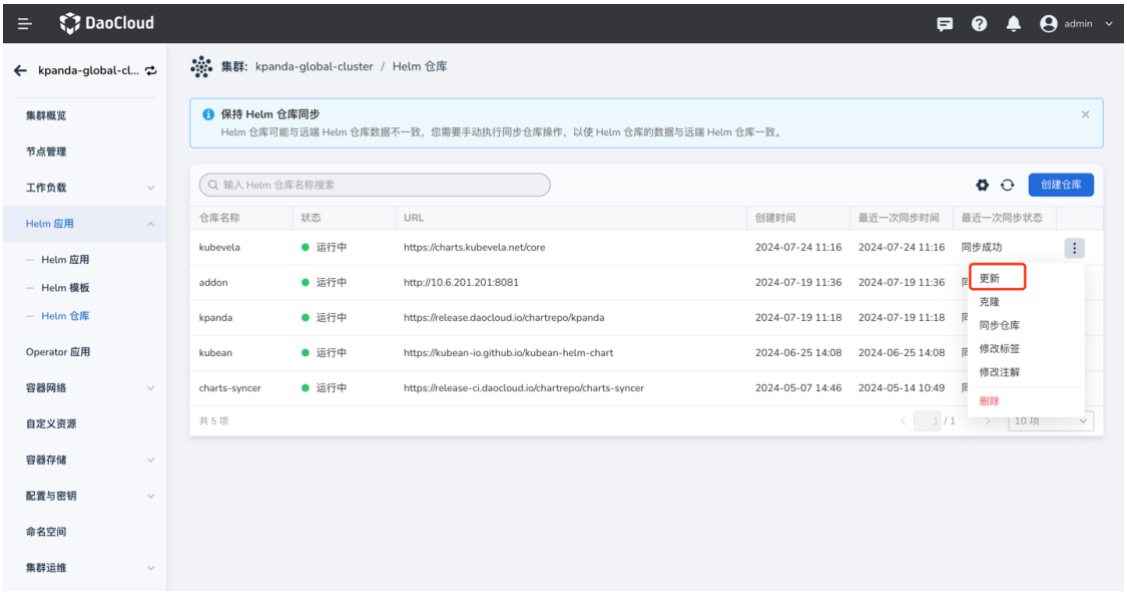
1. 找到待更新仓库所在的集群，点击集群名称，进入 **集群详情**



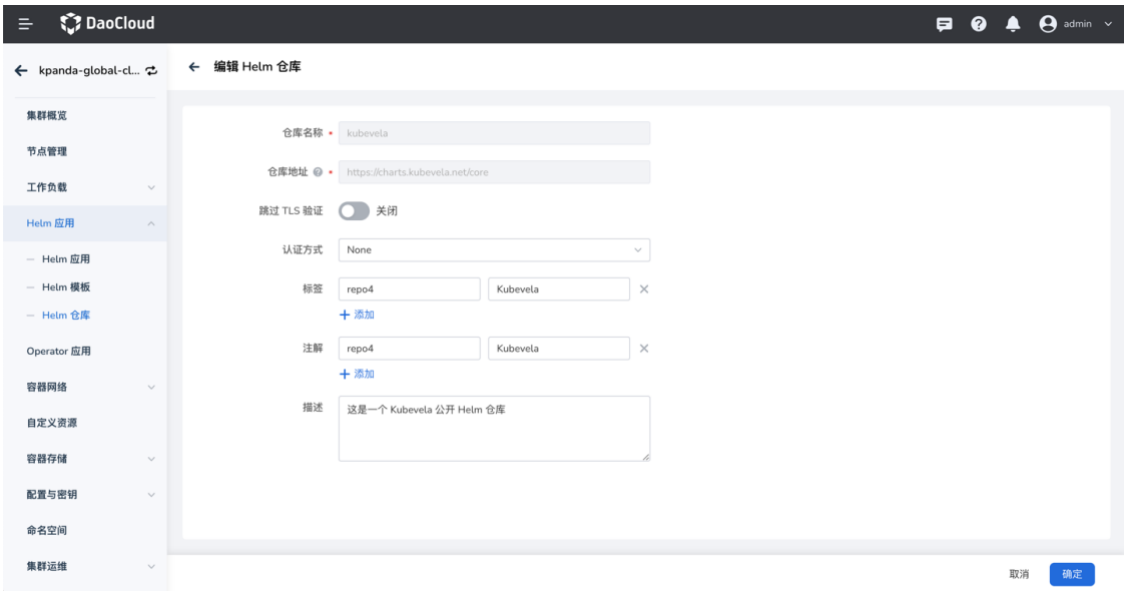
2. 在左侧导航栏，依次点击 **Helm 应用 -> Helm 仓库** ，进入 Helm 仓库列表页面。



3. 在仓库列表页面找到需要更新的 Helm 仓库，在列表右侧点击 ⋮ 按钮，在弹出菜单中点击 **更新**



4. 在 **编辑 Helm 仓库** 页面进行更新，完成后点击 **确定**

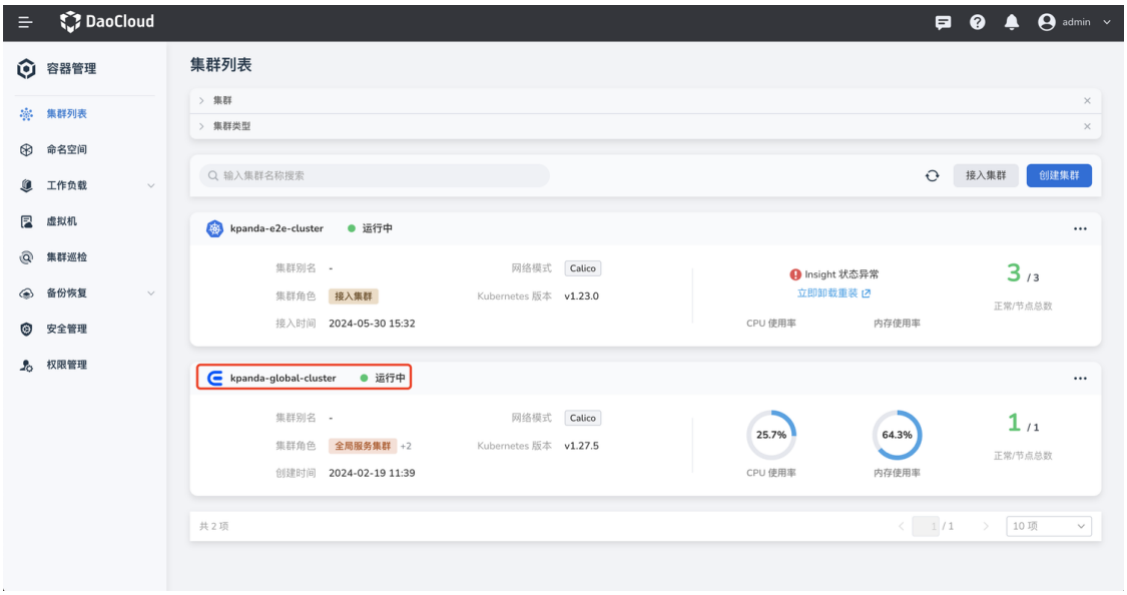


5. 返回 Helm 仓库列表，屏幕提示更新成功。

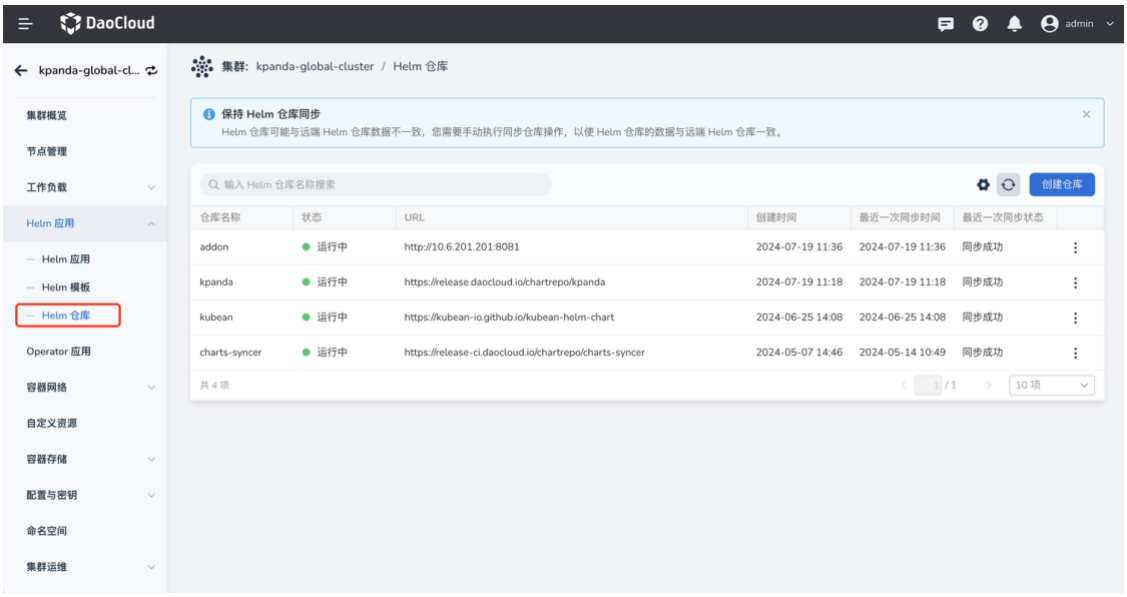
删除 Helm 仓库

除了引入、更新仓库外，您也可以将不需要的仓库删除，包括系统预置仓库和第三方仓库。

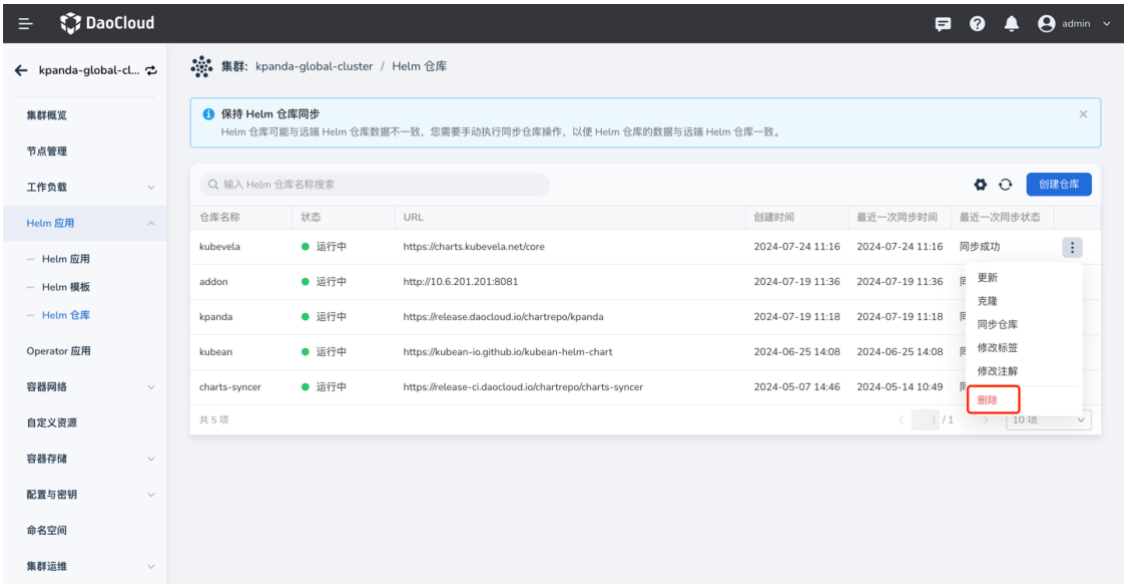
1. 找到待删除仓库所在的集群，点击集群名称，进入 **集群详情**



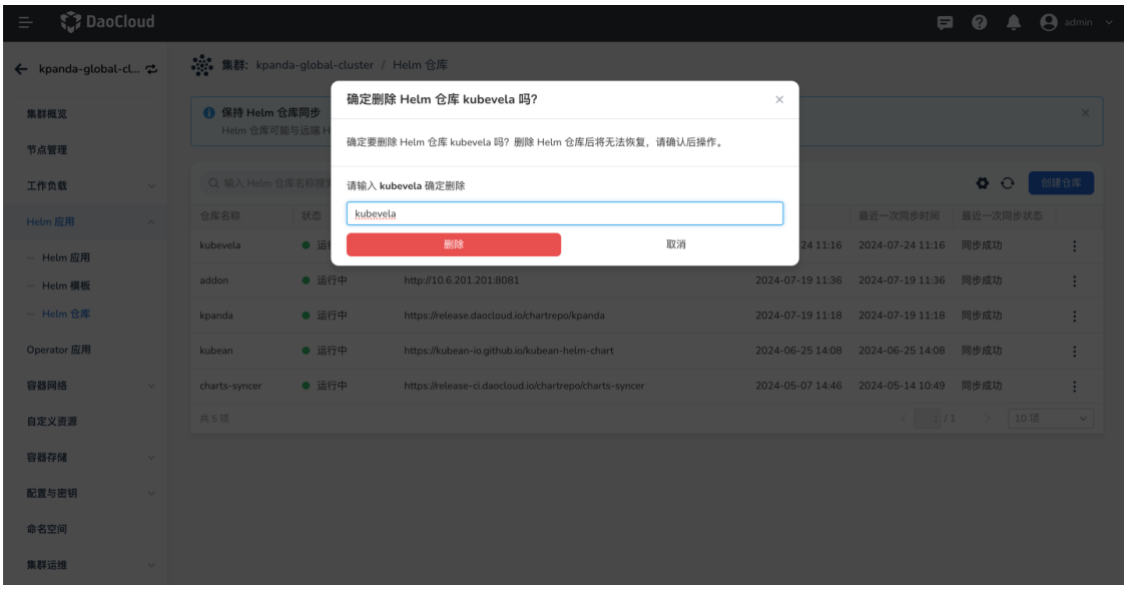
2. 在左侧导航栏，依次点击 **Helm 应用 -> Helm 仓库** ，进入 Helm 仓库列表页面。



3. 在仓库列表页面找到需要更新的 Helm 仓库，在列表右侧点击 ⋮ 按钮，在弹出菜单中点击 **删除**



4. 输入仓库名称进行确认，点击 **删除**



5. 返回 Helm 仓库列表，屏幕提示删除成功。

第 3 章　商品上架：上传、导入与离线交付

目录里没有想要的应用，就要自己上架。上架的本质是让模板包落进某个仓库，并让商店的索引认得它——在线环境靠推送，离线环境靠 charts-syncer 与离线包，多架构集群还要再做一次融合。

三条路径怎么选，先看这张表：

场景	走哪条路	小节
有外网，自有 Harbor、ChartMuseum 或 JFrog 仓库	用 `helm cm-push` 推送，或直接在 Harbor 页面上传	3.1
要让自定义应用出现在系统内置 addon 仓库，且环境离线	用 charts-syncer 把 Chart 与镜像一并同步进 addon	3.2
整包离线交付，需要 amd64 与 arm64 同时可用	导入离线包后再做多架构融合	3.3
在线环境，但无权限向内置仓库写入	自建仓库，再按 3.2 的同步步骤接入	3.2

无论走哪条路，最后都要回到 **Helm 仓库**页面点一次**同步仓库**，模板才会出现在目录里——这一步最容易被忘。

3.1　上传应用模板

位置：应用商店 › 商品上架

上架一个应用分三步：先把仓库接入商店（第 2 章），再把模板包推进仓库，最后同步仓库让目录看见它。本节的两种上传方式按仓库类型选一种即可，第三步则两种方式都要做。

方式一：客户端推送

> **适用范围**
>
> 适用于 Harbor、ChartMuseum、JFrog 类型仓库。

1. 准备节点与插件。登录一台可以访问到 Helm 仓库的节点，把 Helm 二进制文件上传到该节点，并安装 cm-push 插件（需要连通外网，并提前安装 Git）。插件安装流程参考 安装 cm-push 插件。
2. 推送模板包到仓库，执行如下命令：

```
helm cm-push ${charts-dir} ${HELM_REPO_URL} --username ${username} --password
${password}
```

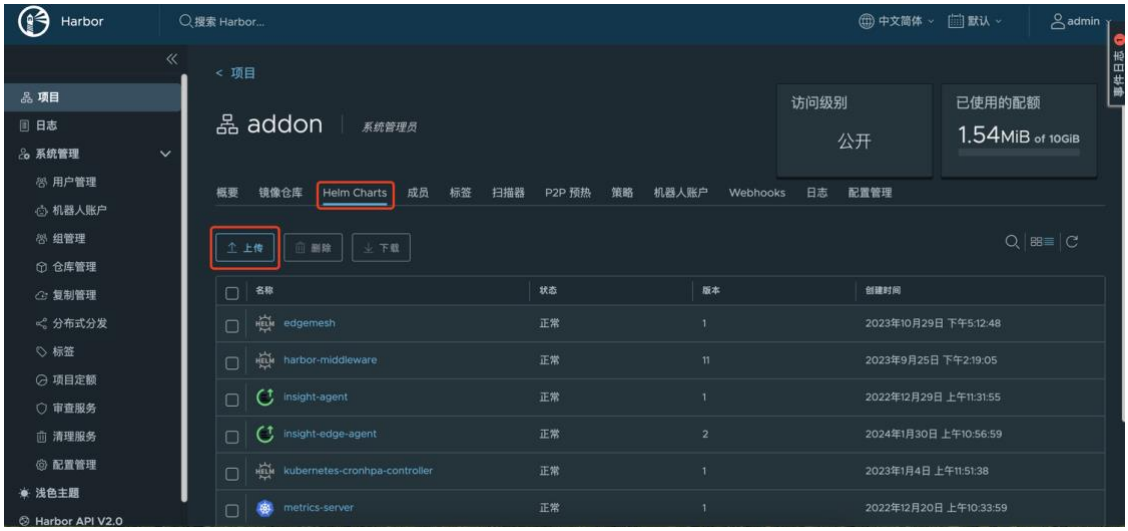
字段说明：

- `charts-dir`：Helm Chart 的目录，或者是打包好的 Chart（即 .tgz 文件）。
- `HELM_REPO_URL`：Helm 仓库的 URL。
- `username` / `password`：有推送权限的 Helm 仓库用户名和密码。
- 如果采用 https 访问且需要跳过证书验证，可添加参数 `--insecure`。

方式二：页面上传

> **适用范围**
>
> 仅适用于 Harbor 类型仓库。

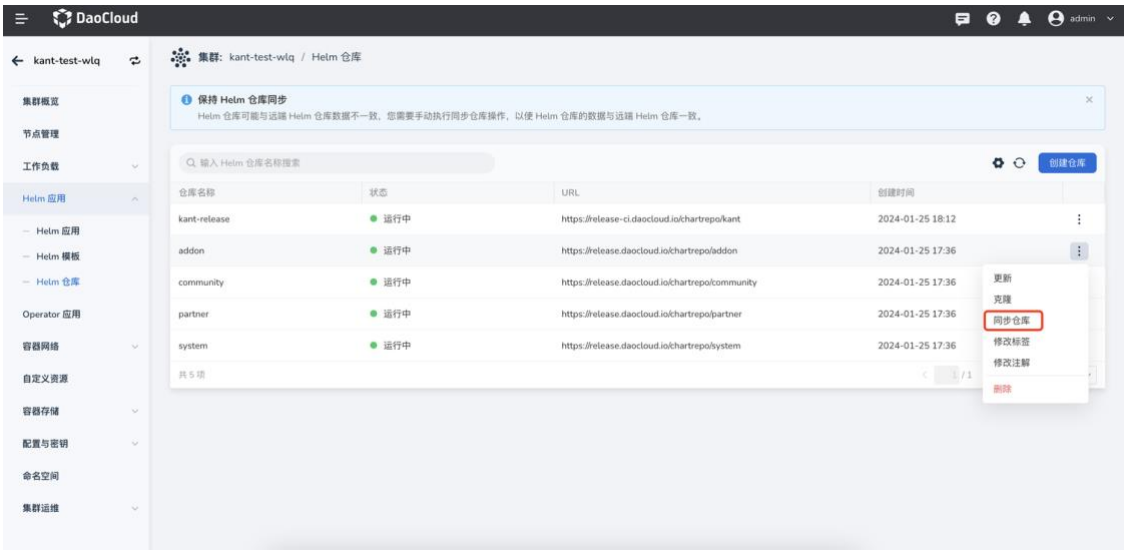
1. 登录网页版 Harbor 仓库，请确保登录用户有推送权限。
2. 进入对应项目，选择 **Helm Charts** 页签，点击页面**上传** 按钮，完成 Helm Chart 上传。



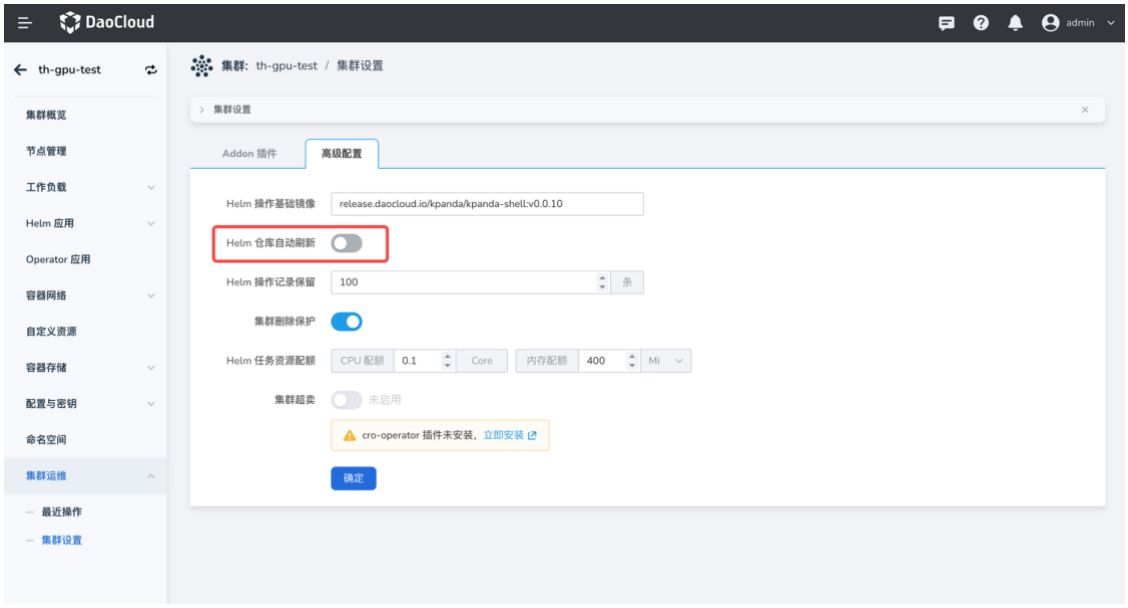
同步仓库：让新模板出现在目录里

推送只是把包放进了仓库，商店并不会自动知道。默认情况下集群未开启 **Helm 仓库自动刷新**，需要手动同步：

1. 进入 **Helm 应用->Helm 仓库**。
2. 在仓库列表右侧点击 ⋮ 按钮，选择**同步仓库**，等待同步完成。



如果希望上游仓库更新后自动可见，可以打开自动刷新：进入 **集群运维->集群设置->高级配置**，开启 **Helm 仓库自动刷新** 开关。



> **上架完成后的验证**
>
> 回到 **Helm 应用->Helm 模板**，在左侧切换到刚推送的仓库，能看到新模板即表示上架成功；点进去还能看到它的版本列表，接下来的安装步骤见 4.1 节。

3.2 将自定义应用导入内置 addon 仓库

位置：应用商店 › 商品上架

上架的最终目的是让模板出现在目录里。若目标货架是系统内置的 **addon** 仓库，就要把 Chart 连同它引用的镜像一起同步进去；本节分离线与在线两种环境说明。

离线环境

离线环境指的是无法连通互联网或封闭的私有网络环境，此时既拿不到公网仓库，也不能依赖安装器自动拉取，只能靠 charts-syncer 把 Chart 与镜像一起搬进 addon 仓库。

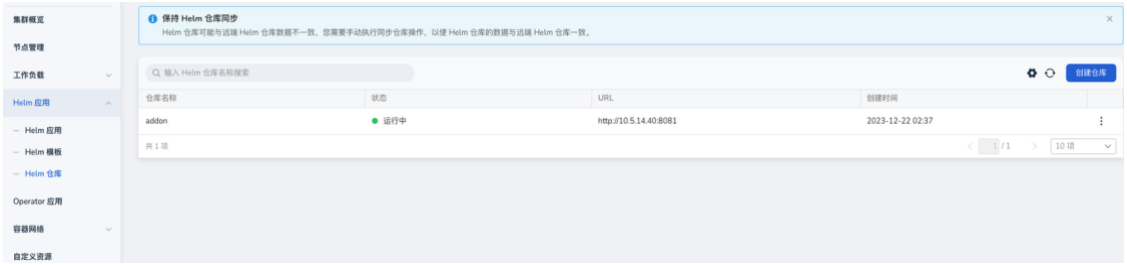
前提条件

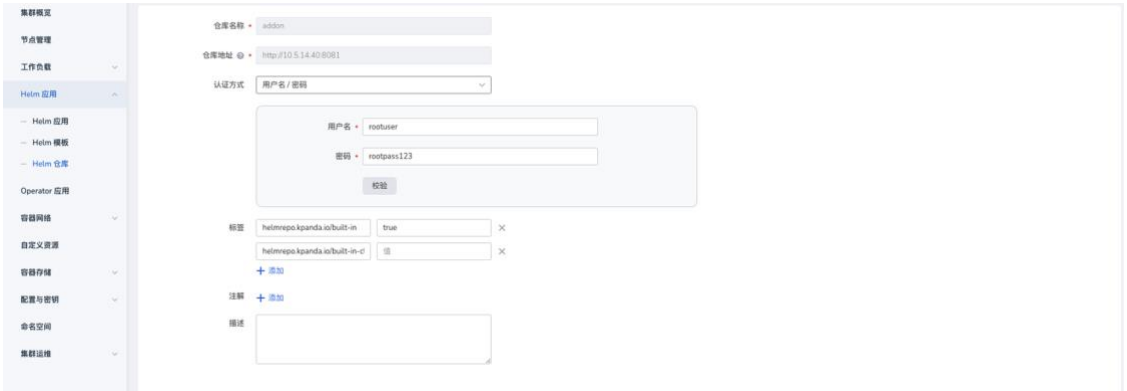
- 存在可以运行的 charts-syncer。 若没有，可点击下载。
- Helm Chart 已经完成适配 charts-syncer。 即在 Helm Chart 内添加了 `.relok8s-images.yaml` 文件。该文件需要包含 Chart 中所有使用到镜像， 也可以包含 Chart 中未直接使用的镜像，类似 Operator 中使用的镜像。

> **说明**
>
> - 如何编写 Chart 可参考 image-hints-file。 要求镜像的 registry 和 repository 必须分开，因为 load 镜像时需替换或修改 registry/repository。
> - 安装器所在的火种集群已安装 charts-syncer。 若将自定义 Helm 应用导入安装器所在火种集群，可跳过下载直接适配； 若未安装 charts-syncer 二进制文件， 可立即下载。

同步 Helm Chart

1. 进入 **容器管理->Helm 应用->Helm 仓库** ，搜索 addon， 获取内置仓库地址和用户名/密码（系统内置仓库默认用户名/密码为 rootuser/rootpass123）。





2. 同步 Helm Chart 到容器管理内置仓库 Addon

   ◦ 编写如下配置文件，可以根据具体配置修改，并保存为 `sync-dao-2048.yaml`。

```yaml
source:  # helm charts 源信息
  repo:
```

```
    kind: HARBOR # 也可以是任何其他支持的 Helm Chart 仓库类别，比如
CHARTMUSEUM
    url: https://release-ci.daocloud.io/chartrepo/community #  需更改为
chart repo url
    #auth: # 用户名/密码,若没有设置密码可以不填写
      #username: "admin"
      #password: "Harbor12345"
charts:  # 需要同步
  - name: dao-2048 # helm charts 信息，若不填写则同步源 helm repo 内所有
charts
    versions:
      - 1.4.1
target:  # helm charts 目标信息
  containerRegistry: 10.5.14.40 # 镜像仓库 url
  repo:
    kind: CHARTMUSEUM # 也可以是任何其他支持的 Helm Chart 仓库类别，比如
HARBOR
    url: http://10.5.14.40:8081 #  需更改为正确 chart repo url，可以通过
helm repo add $HELM-REPO 验证地址是否正确
    auth: # 用户名/密码，若没有设置密码可以不填写
      username: "rootuser"
      password: "rootpass123"
  containers:
    # kind: HARBOR # 若镜像仓库为 HARBOR 且希望 charts-syncer 自动创建镜像
Repository 则填写该字段
    # auth: # 用户名/密码，若没有设置密码可以不填写
      # username: "admin"
      # password: "Harbor12345"

# leverage .relok8s-images.yaml file inside the Charts to move the
container images too
relocateContainerImages: true
```

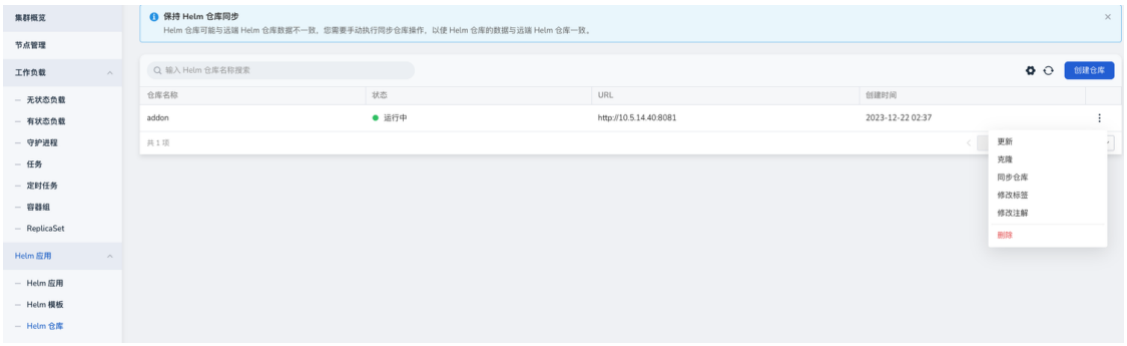
◦ 执行 charts-syncer 命令同步 Chart 及其包含的镜像

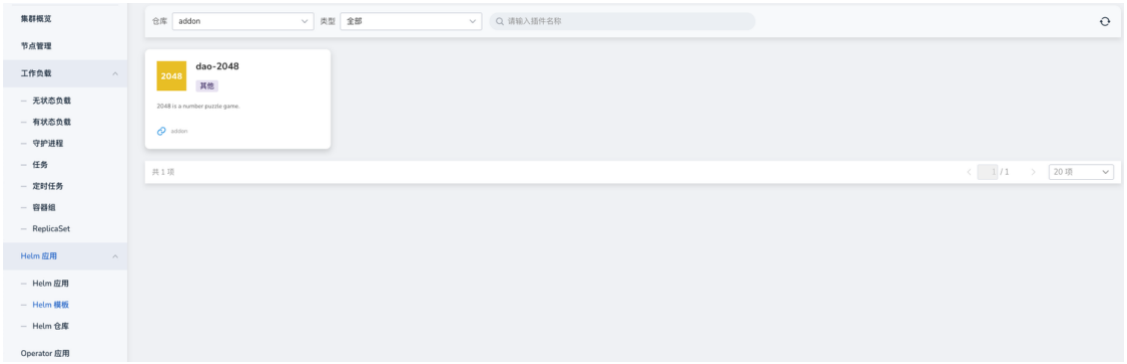
```
charts-syncer sync --config sync-dao-2048.yaml --insecure --auto-
create-repository
```

预期输出为：

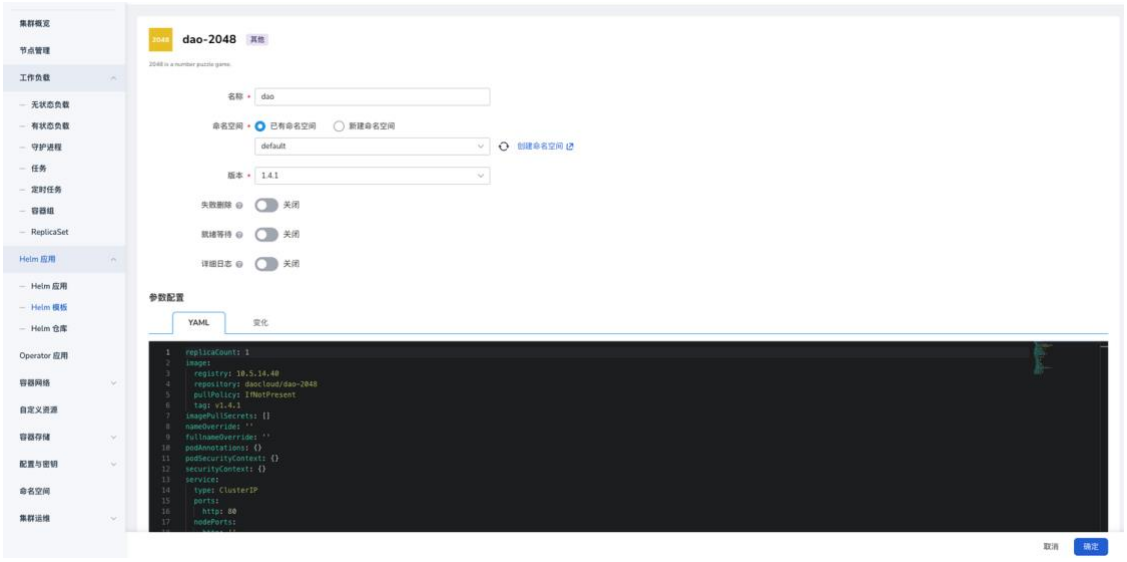
```
I1222 15:01:47.119777    8743 sync.go:45] Using config file:
"examples/sync-dao-2048.yaml"
W1222 15:01:47.234238    8743 syncer.go:263] Ignoring skipDependencies
option as dependency sync is not supported if container image
relocation is true or syncing from/to intermediate directory
I1222 15:01:47.234685    8743 sync.go:58] There is 1 chart out of sync!
I1222 15:01:47.234706    8743 sync.go:66] Syncing "dao-2048_1.4.1"
chart...
.relok8s-images.yaml hints file found
Computing relocation...

Relocating dao-2048@1.4.1...
Pushing 10.5.14.40/daocloud/dao-2048:v1.4.1...
Done
Done moving /var/folders/vm/08vw0t3j68z9z_4lcqyhg8nm0000gn/T/charts-
syncer869598676/dao-2048-1.4.1.tgz
```

3. 待上一步执行完成后，进入 **容器管理->Helm 应用->Helm 仓库** ，找到对应 Addon， 在操作栏点击 **同步仓库** ，回到 Helm 模板就可以看到上传的 Helm 应用





4. 后续可正常进行安装、升级、卸载



在线环境

在线环境的 Helm Repo 地址为 `release.daocloud.io`。 如果用户无权限添加 Helm Repo，则无法将自定义 Helm 应用导入系统内置 Addon。 您可以添加自己搭建的 Helm 仓库，然后按照离线环境中同步 Helm Chart 的步骤将您的 Helm 仓库集成到平台使用。

3.3 多架构应用的导入与升级

位置：应用商店 › 商品上架

在多架构集群里，同一批应用往往要同时覆盖 amd64 与 arm64，整包交付的 addon 离线包因此不能只按一种架构导入。 本节说明单架构离线包的导入与升级、多架构融合的做法，以及几个容易踩到的坑。

导入

单架构导入

准备好待导入的离线包 `addon-offline-full-package-${version}-${arch}.tar.gz` ，可从下载中心下载。 把路径填写至 **clusterConfig.yml** 配置文件，例如：

```
addonPackage:
  path: "/home/addon-offline-full-package-v0.9.0-amd64.tar.gz"
```

然后执行导入命令：

```
~/dce5-installer cluster-create -c /home/offline/sample/clusterConfig.yaml -m
/home/offline/sample/manifest.yaml -d -j13
```

多架构融合

准备好待融合的离线包 `addon-offline-full-package-${version}-${arch}.tar.gz` ，可从下载中心下载。

以 addon-offline-full-package-v0.9.0-arm64.tar.gz 为例，执行导入命令：

```
~/dce5-installer import-addon -c /home/offline/sample/clusterConfig.yaml --addon-
path=/home/addon-offline-full-package-v0.9.0-arm64.tar.gz
```

升级

单架构升级

准备好待导入的离线包 `addon-offline-full-package-${version}-${arch}.tar.gz` ，可从下载中心下载。

把路径填写至 clusterConfig.yml 配置文件，例如：

```
addonPackage:
  path: "/home/addon-offline-full-package-v0.11.0-amd64.tar.gz"
```

然后执行导入命令：

```
~/dce5-installer cluster-create -c /home/offline/sample/clusterConfig.yaml -m
/home/offline/sample/manifest.yaml -d -j13
```

多架构融合

准备好待融合的离线包 `addon-offline-full-package-${version}-${arch}.tar.gz` ，可从下载中心下载。

以 addon-offline-full-package-v0.11.0-arm64.tar.gz 为例，执行导入命令：

```
~/dce5-installer import-addon -c /home/offline/sample/clusterConfig.yaml --addon-path=/home/addon-offline-full-package-v0.11.0-arm64.tar.gz
```

注意事项

下面是离线导入与融合时最常见的四类问题，处理办法都来自实际交付现场，建议动手前先扫一眼。

磁盘空间

离线包比较大，且过程中需要解压和 load 镜像，需要预留充足的空间，否则可能在过程中报 “no space left” 而中断。

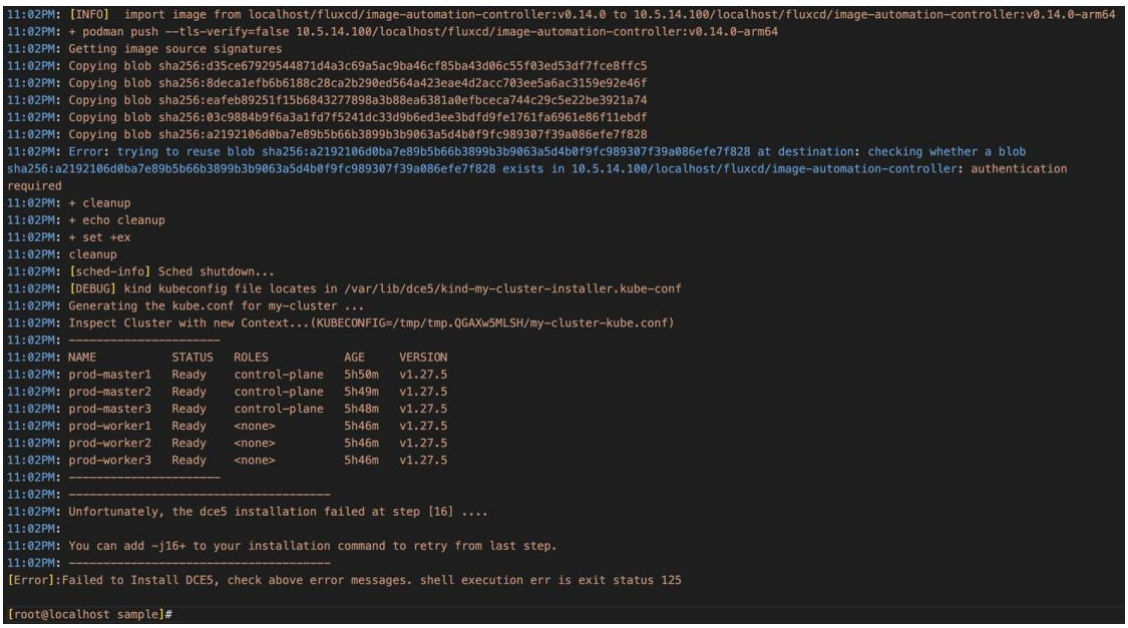
失败后重试

如果在多架构融合步骤执行失败，重试前需要清理一下残留：

```
rm -rf addon-offline-target-package
```

镜像空间

如果融合的离线包中包含了与导入的离线包不一致的镜像空间，可能会在融合过程中因为镜像空间不存在而报错：



解决办法：只需要在融合之前创建好该镜像空间即可，例如上图报错可通过创建镜像空间 localhost 提前避免。

架构冲突

升级至低于 0.12.0 版本的 addon 时，由于目标离线包里的 charts-syncer 没有检查镜像存在则不推送功能，因此会在升级的过程中会重新把多架构冲成单架构。 例如：在 v0.10 版本将 addon 实现为多架构，此时若升级为 v0.11 版本，则多架构 addon 会被覆盖为单架构；若升级为 0.12.0 及以上版本则仍能够保持多架构。

第 4 章　选购与运维：安装、更新与卸载应用

最后一步，也是用得最多的一步。安装本身只有几步，但「装完以后怎么办」决定了这套能力好不好用：参数能不能改、改错了能不能看见差异、失败能不能查日志、不用了能不能干净卸载。

本章按这四件事组织：安装 → 更新（含变化对比与配额调整）→ 操作记录 → 删除。文中用 cert-manager 与 dao-2048 两个模板举例，步骤对任何模板都通用；前提条件与 2.1 节相同——集群已接入、命名空间已创建、用户具备 NS Admin 或更高权限。

4.1　安装、更新与卸载应用

位置：应用商店 › 选购与运维

应用商店把 Helm 的安装与运维搬到了界面上：从模板创建应用、自定义参数，并对应用做全生命周期管理。

本节以 cert-manager 为例，完整走一遍从商店挑模板、装成应用，到更新、回看记录与卸载的过程。

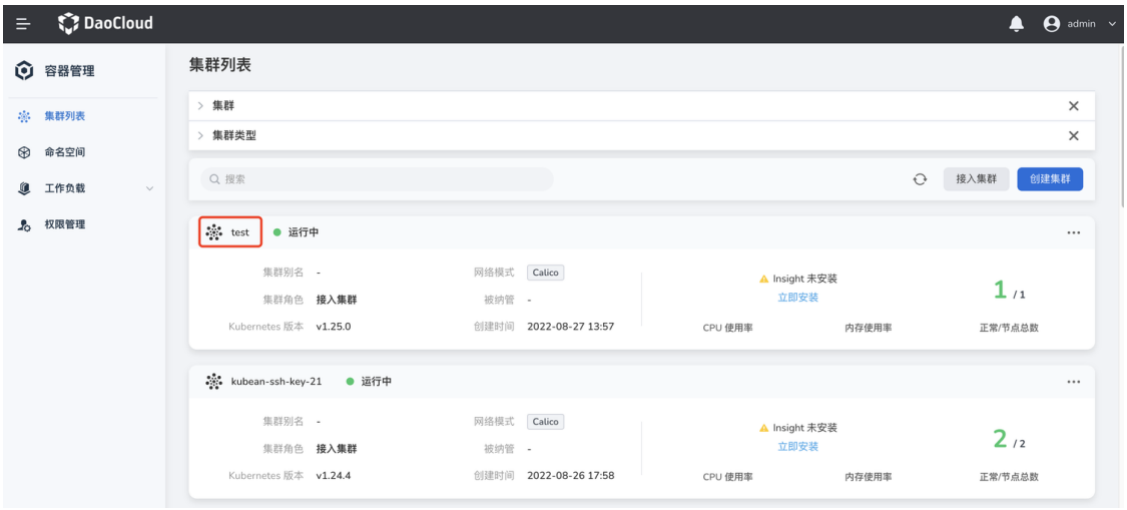
前提条件

- 容器管理模块已接入 Kubernetes 集群或者已创建 Kubernetes 集群，且能够访问集群的 UI 界面。
- 已完成一个命名空间的创建、用户的创建，并为用户授予 NS Admin 或更高权限，详情可参考命名空间授权。

安装 Helm 应用

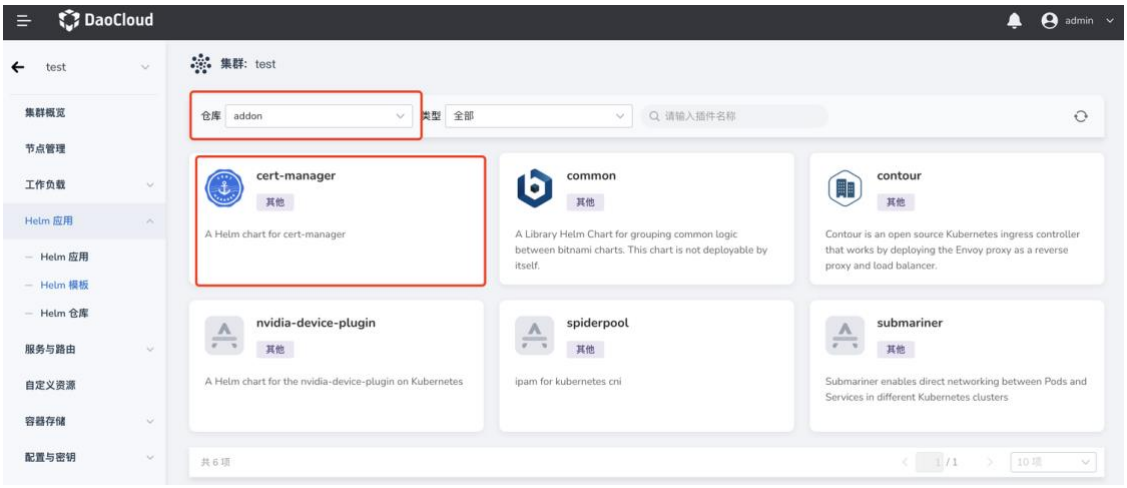
参照以下步骤安装 Helm 应用。

1. 点击一个集群名称，进入 **集群详情** 。

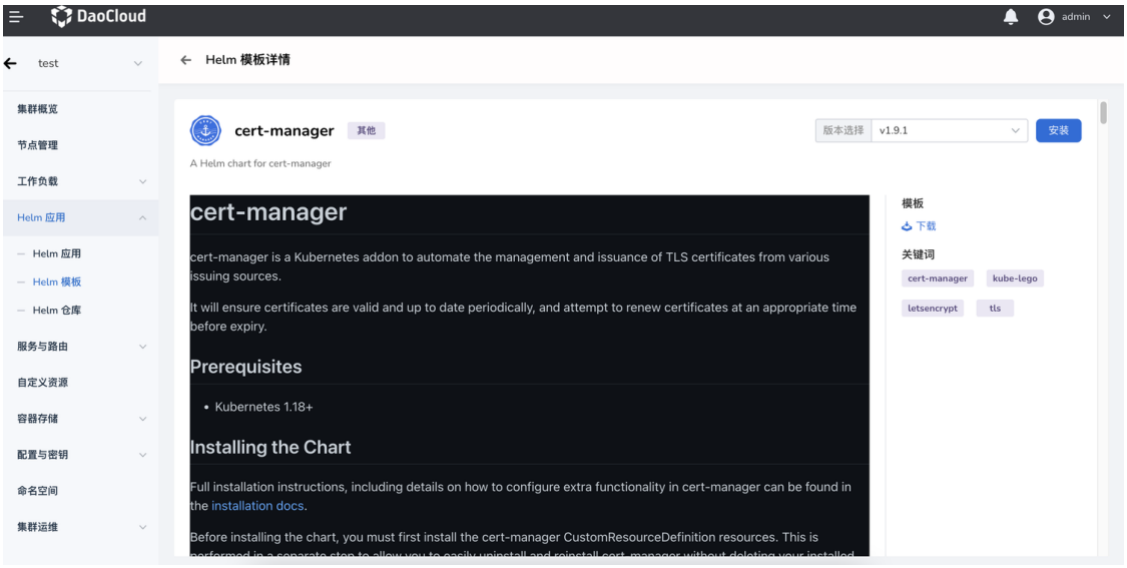


2. 在左侧导航栏，依次点击 **Helm 应用 -> Helm 模板** ，进入 Helm 模板页面。

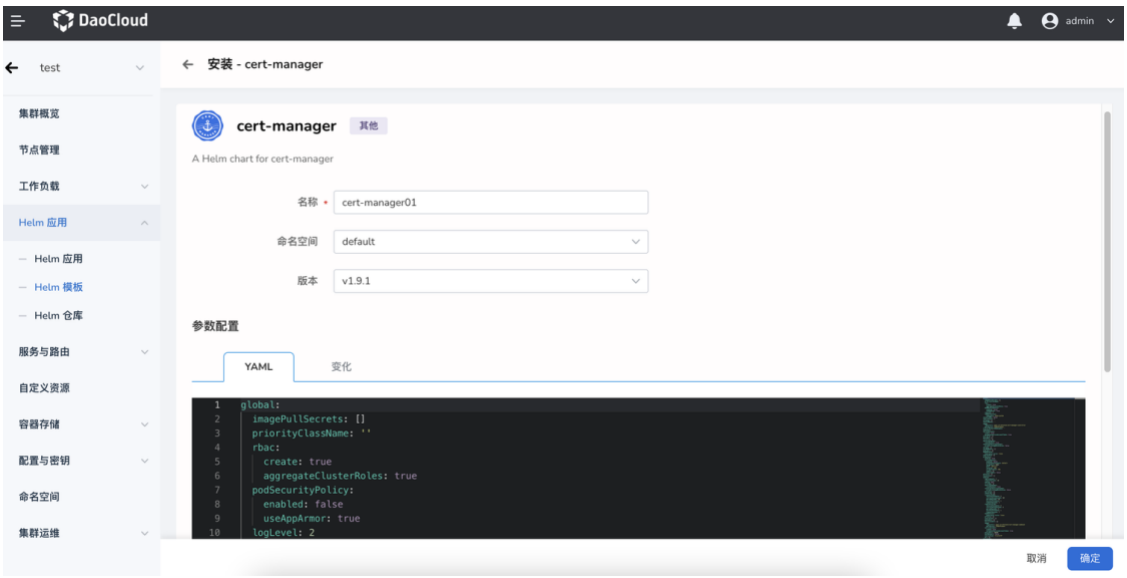
   在 Helm 模板页面选择名为 **addon** 的 Helm 仓库（见 2.1 节），此时界面上将呈现 **addon** 仓库下所有的 Helm chart 模板。 点击名称为 **cert-manager** 的 Chart。



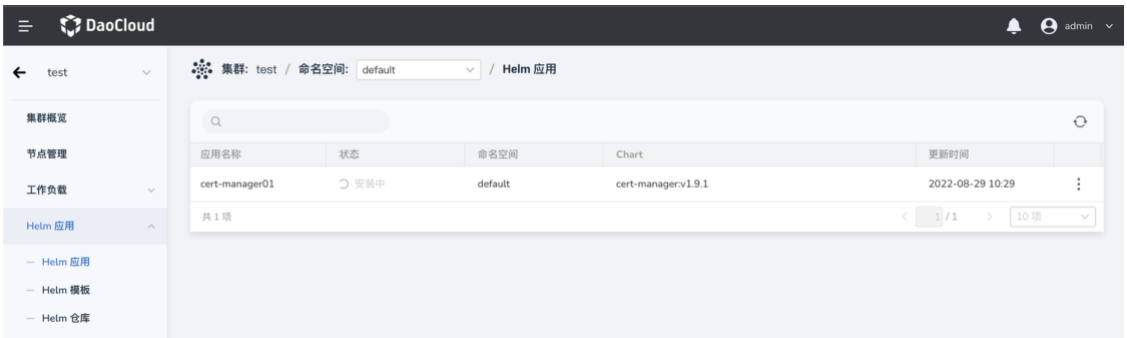
3. 在安装页面，能够看到 Chart 的相关详细信息，在界面右上角选择需要安装的版本，点击 **安装** 按钮。此处选择 v1.9.1 版本进行安装。



4. 配置 **名称** 、 **命名空间** 及 **版本信息** ， 也可以在下方的 **参数配置** 区域通过修改 YAML 来自定义参数。点击 **确定** 。



5. 系统将自动返回 Helm 应用列表，新创建的 Helm 应用状态为 **安装中** ， 等待一段时间后状态变为 **运行中** 。

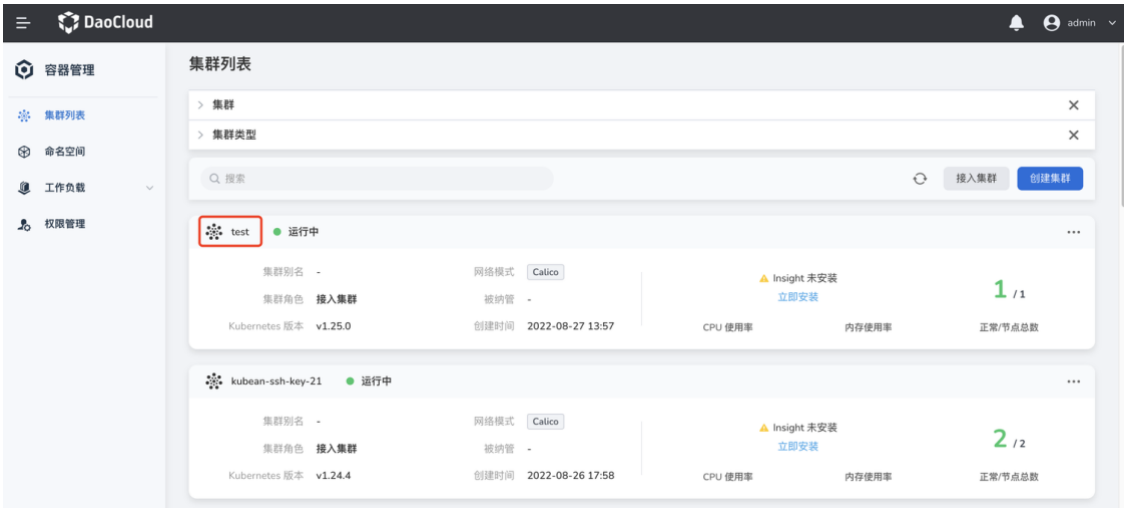


更新 Helm 应用

当我们通过界面完成一个 Helm 应用的安装后，我们可以对 Helm 应用执行更新操作。注意：只有通过界面安装的 Helm 应用才支持使用界面进行更新操作。

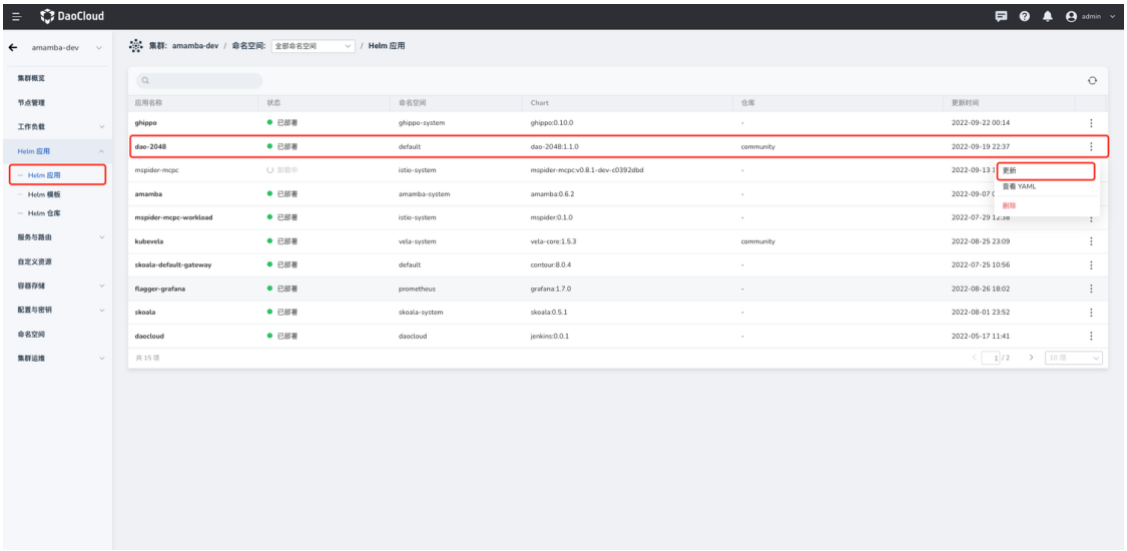
参照以下步骤更新 Helm 应用。

1. 点击一个集群名称，进入 **集群详情** 。

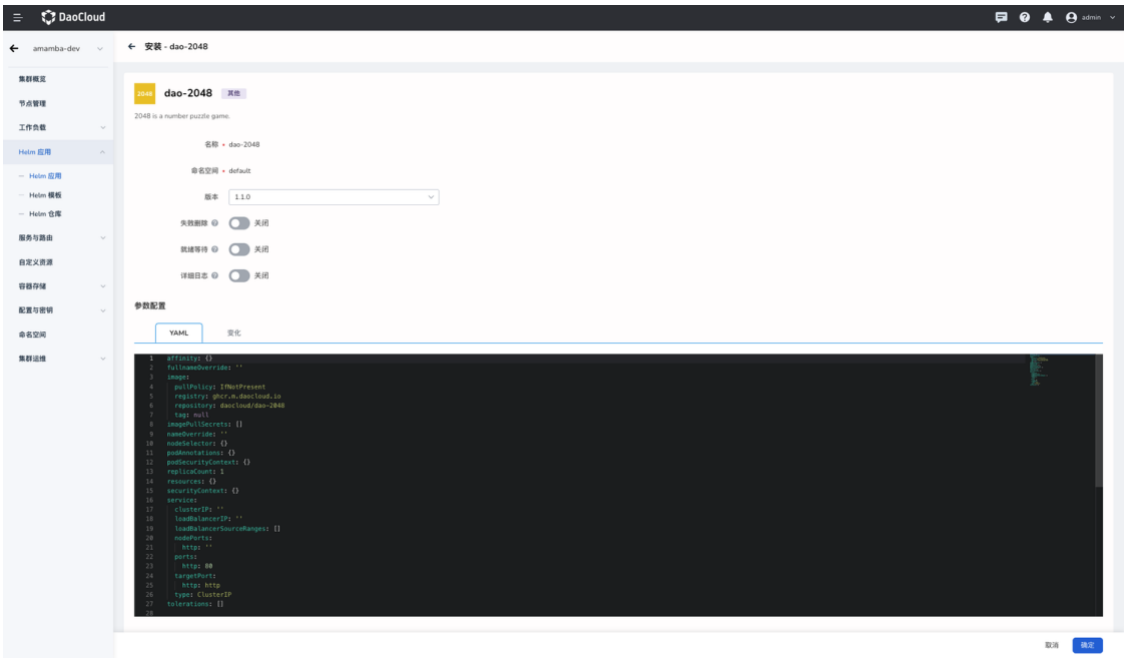


2. 在左侧导航栏，点击 **Helm 应用** ，进入 Helm 应用列表页面。

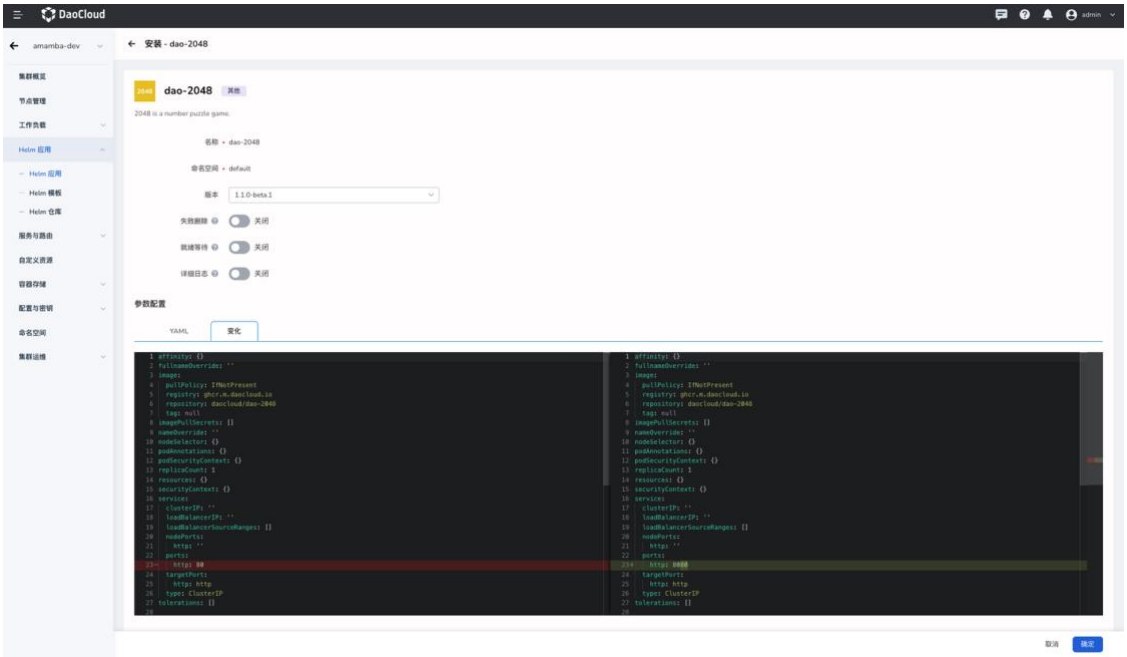
   在 Helm 应用列表页选择需要更新的 Helm 应用，点击列表右侧的 ⋮ 操作按钮，在下拉选择中选择 **更新** 操作。



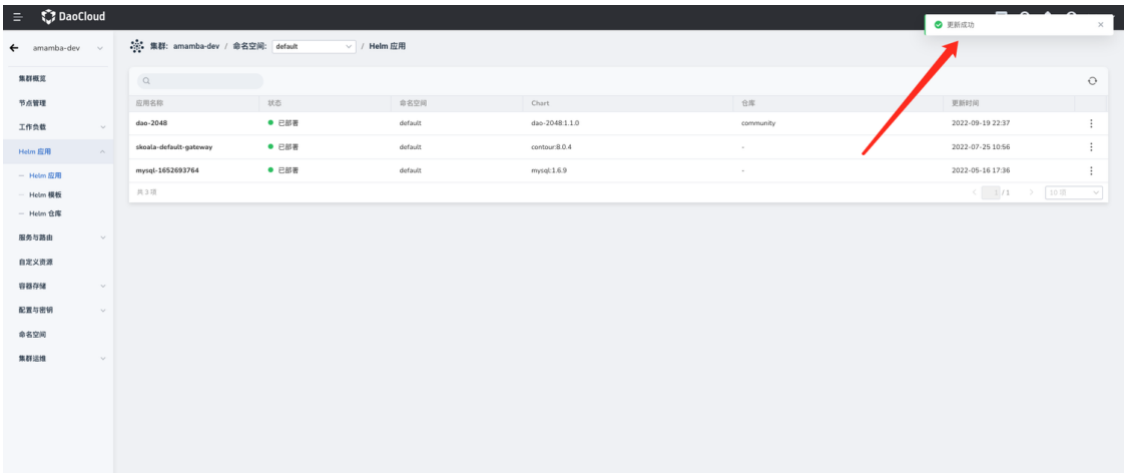
3. 点击 **更新** 按钮后，系统将跳转至更新界面，您可以根据需要对 Helm 应用进行更新，此处我们以更新 **dao-2048** 这个应用的 http 端口为例。



4. 修改完相应参数后。您可以在参数配置下点击 **变化** 按钮，对比修改前后的文件，确定无误后，点击底部 **确定** 按钮，完成 Helm 应用的更新。



5. 系统将自动返回 Helm 应用列表，右上角弹窗提示 **更新成功** 。



注意事项：

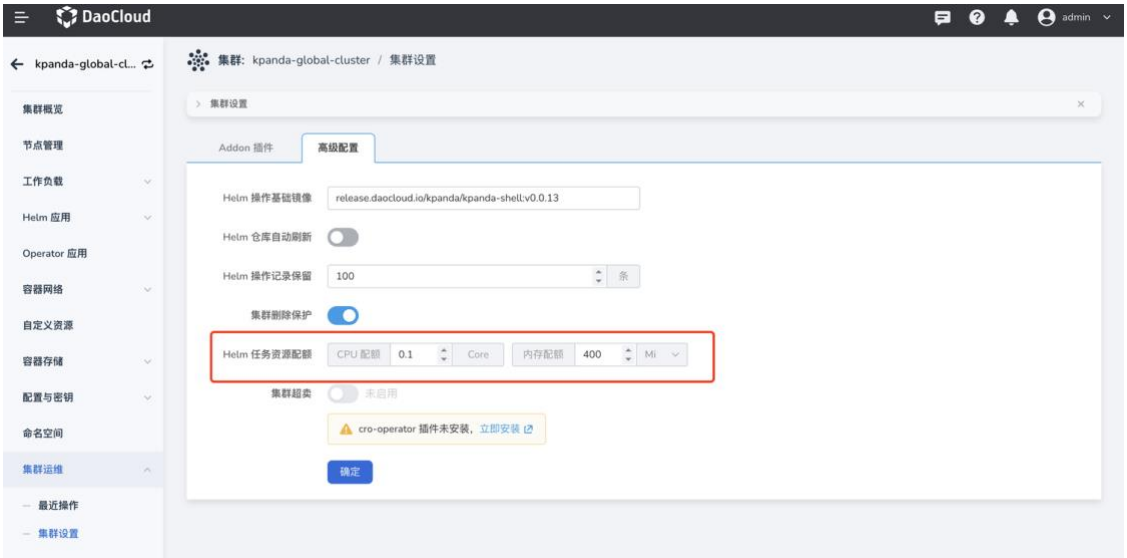
若更新耗时较长，可能是由于资源限制导致。可通过以下步骤调整 Helm 任务资源配额：

1. 查看更新该 helm 应用所占用的资源

```
# 获取更新 helm 应用的 job name
kubectl -n default get job

# 查看更新该 helm 应用的这个 job 所占用的资源
kubectl -n zestu-system get job **换成 jobname ** -oyaml|grep resources -A 6
```

2. 调整 Helm 任务的资源配额

   进入 **集群详情->集群运维->集群设置->高级设置**，调整 Helm 任务资源配额，并点击 **确定** 完成保存。

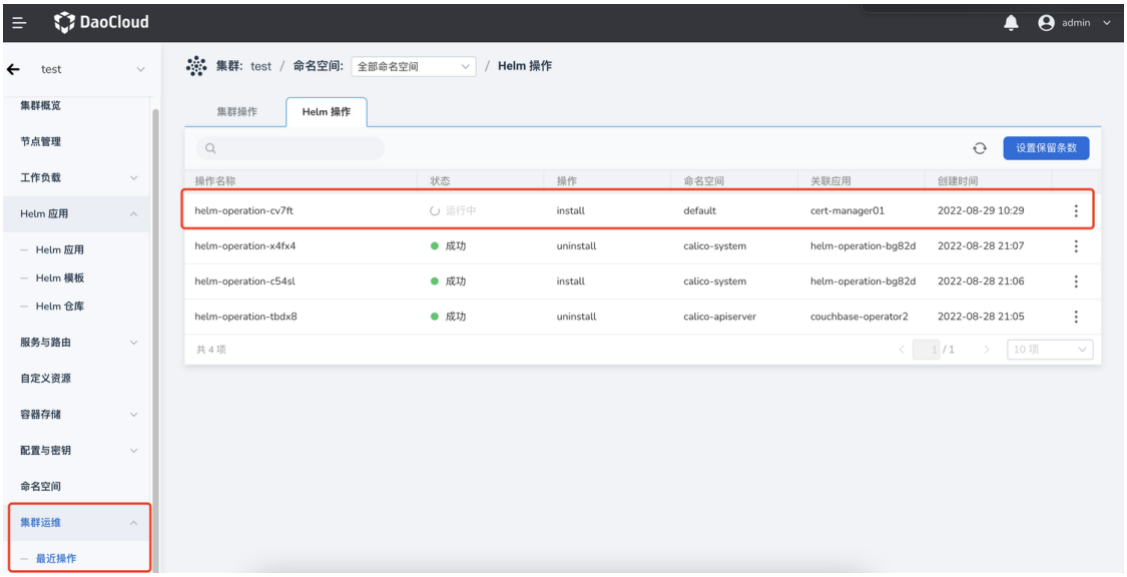


注：Helm 应用更新完成后，可将资源配额修改成默认值，以减少资源开支。

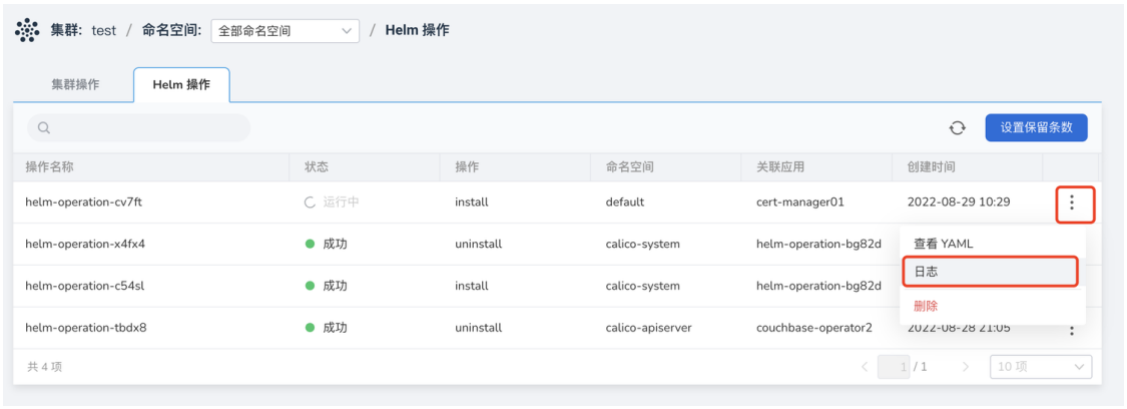
查看 Helm 操作记录

Helm 应用的每次安装、更新、删除都有详细的操作记录和日志可供查看。

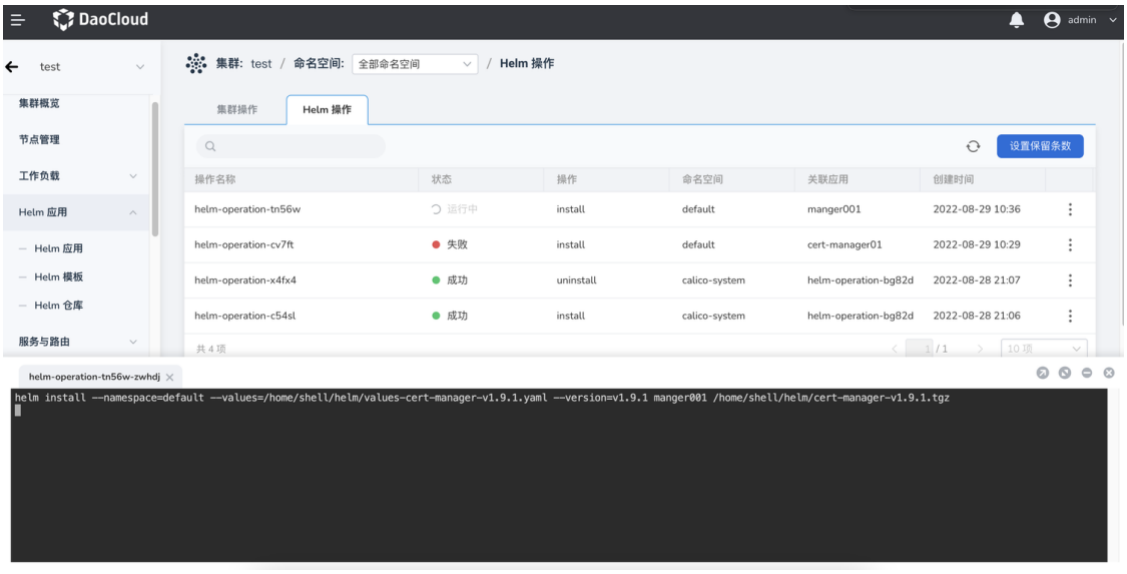
1. 在左侧导航栏，依次点击 **集群运维 -> 最近操作** ，然后在页面上方选择 **Helm 操作** 标签页。每一条记录对应一次安装/更新/删除操作。



2. 如需查看每一次操作的详细日志：在列表右侧点击 ⋮，在弹出菜单中选择 **日志** 。



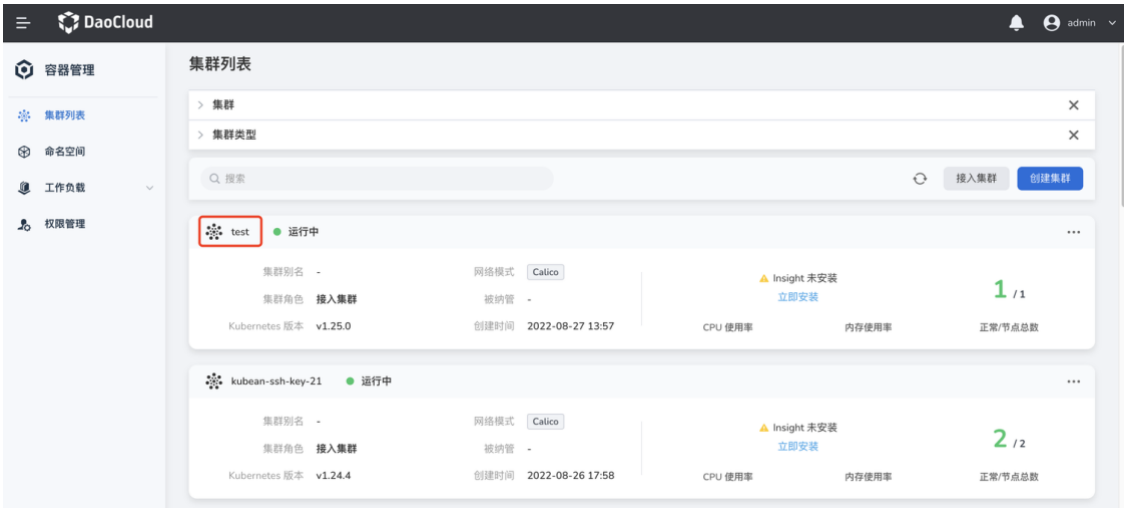
3. 此时页面下方将以控制台的形式展示详细的运行日志。



删除 Helm 应用

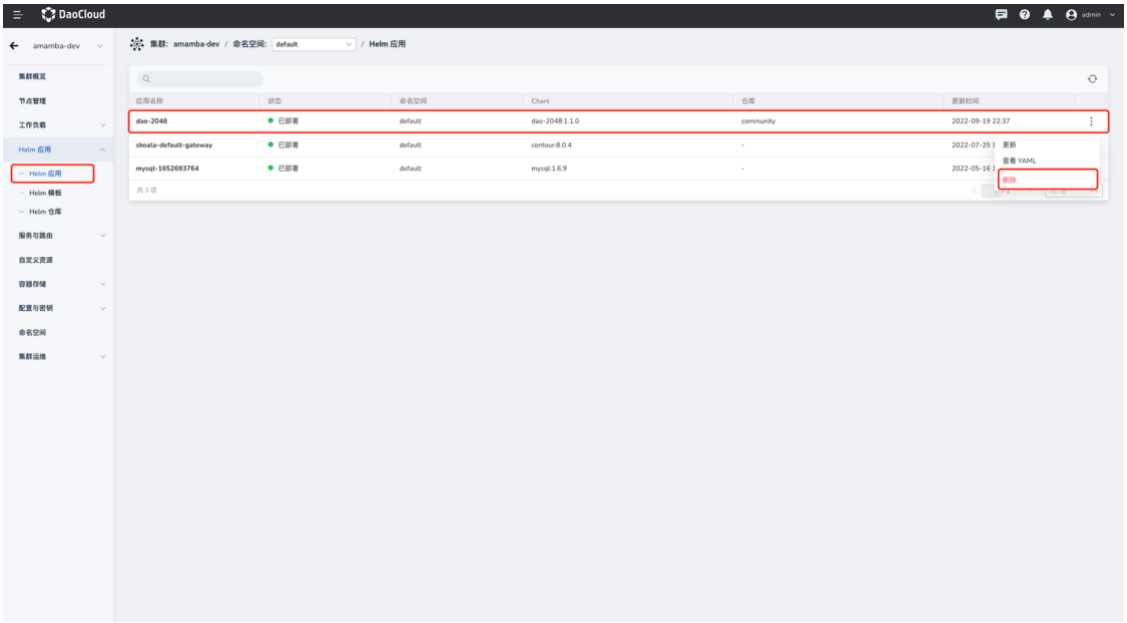
参照以下步骤删除 Helm 应用。

1. 找到待删除的 Helm 应用所在的集群，点击集群名称，进入 **集群详情** 。

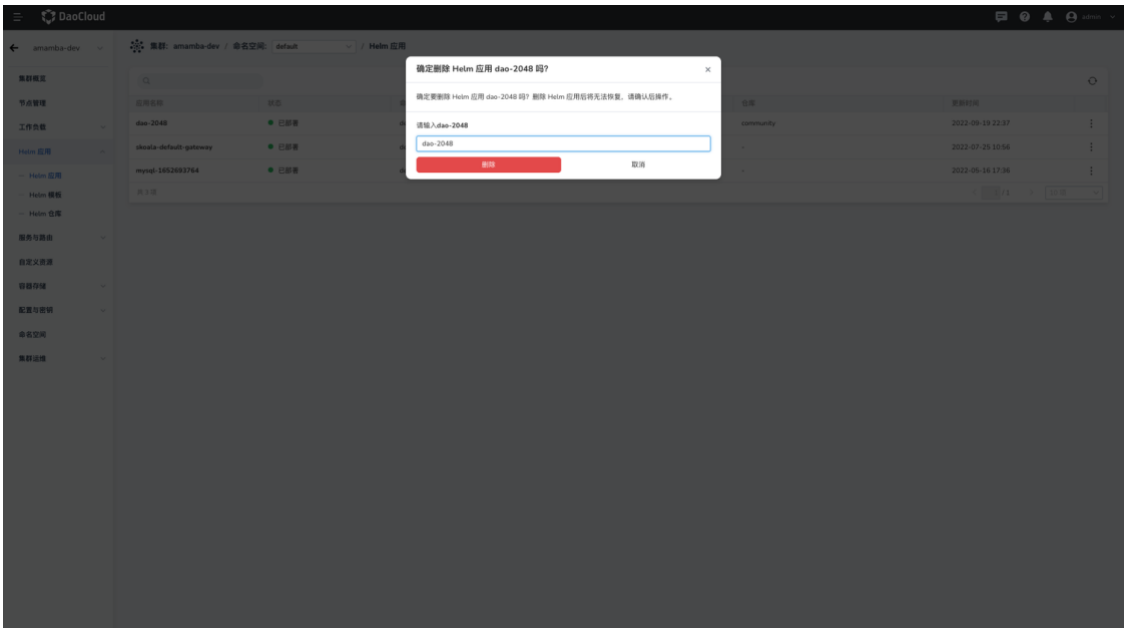


2. 在左侧导航栏，点击 **Helm 应用** ，进入 Helm 应用列表页面。

   在 Helm 应用列表页选择您需要删除的 Helm 应用，点击列表右侧的 ⋮ 操作按钮，在下拉选择中选择 **删除** 。



3. 在弹窗内输入 Helm 应用的名称进行确认，然后点击 **删除** 按钮。


DaoCloud
确定删除 Helm 应用 dao-2048 吗?
确定要删除 Helm 应用 dao-2048 吗? 删除 Helm 应用后将无法恢复，请确认后操作。
请输入 dao-2048
dao-2048
删除
取消

第 5 章　速查与排障

把散落在各节里的入口、命令与失败处理收在一处，方便现场查阅。本节不引入新操作，只做归拢；每条都注明了出处章节，需要看截图与上下文时按编号回查。

5.1　入口与命令速查

同一件事往往有三四个入口，先查表再动手：

要做什么	到哪里做	命令 / 字段
浏览有哪些模板	集群详情 → Helm 应用 → Helm 模板	—
接入、更新、删除仓库	集群详情 → Helm 应用 → Helm 仓库	仓库地址、认证方式、跳过 TLS 验证
手动同步仓库	Helm 仓库列表 → ⋮ → 同步仓库	—
开启仓库自动刷新	集群运维 → 集群设置 → 高级配置	Helm 仓库自动刷新
客户端推送 Chart	可访问仓库的节点	`helm cm-push ${charts-dir} ${HELM_REPO_URL}`
页面上传 Chart	Harbor 项目 → Helm Charts → 上传	仅 Harbor
同步 Chart 到内置 addon 仓库	可运行 charts-syncer 的节点	`charts-syncer sync --config <配置文件>`
导入离线包（单架构 / 升级）	安装节点	`dce5-installer cluster-create -c … -m … -d -j13`
导入离线包（多架构融合）	安装节点	`dce5-installer import-addon -c … --addon-path=…`
回看 Helm 操作记录	集群运维 → 最近操作 → Helm 操作	—
调整 Helm 任务资源配额	集群详情 → 集群运维 → 集群设置 → 高级设置	Helm 任务资源配额

内置 4 个仓库的定位也一并列出，接入第三方仓库前先确认是否已有覆盖：

仓库	定位	示例
partner	生态合作伙伴提供的优质特色 Chart	tidb
system	系统核心组件及高级功能的依赖 Chart	Insight（insight-agent 为集群监控所必需）
addon	业务场景中常见的 Chart	cert-manager
community	Kubernetes 社区较热门的开源组件 Chart	Istio

5.2 常见失败与处理

下表按「现象 → 原因 → 处理」列出各节提到过的失败场景，出处列可回查上下文：

现象	原因	处理	出处
应用更新耗时很长	Helm 任务资源配额不足	在集群设置的高级设置里调大 Helm 任务资源配额，更新完成后改回默认值	4.1
推送成功但模板目录里看不到	仓库未同步	手动点「同步仓库」，或开启 Helm 仓库自动刷新	3.1
添加 https 仓库失败	证书不受信任	勾选「跳过 TLS 验证」，或为仓库配置可信证书	2.1
拉取私有仓库的模板失败	认证方式或权限不足	选择用户名/密码认证，确认账号对该仓库有读权限	2.1
融合时报镜像空间不存在	融合包与导入包的镜像空间不一致	融合前先创建该镜像空间（例如 localhost）	3.3
融合失败后重试仍失败	上次的中间产物未清理	执行 `rm -rf addon-offline-target-package` 后重试	3.3
导入中途报 no space left	离线包需解压并 load 镜像，空间不足	预留充足的磁盘空间后重新导入	3.3
升级后多架构变成单架构	升级目标低于 0.12.0，charts-syncer 未检查镜像是否已存在	升级到 0.12.0 及以上版本	3.3

现象	原因	处理	出处
应用无法通过界面更新	该应用不是通过界面安装的	界面更新只对界面安装的应用生效，命令行安装的应用需改用命令行维护	4.1

卸载前先确认作用对象

删除仓库只影响目录，不影响已安装的应用；卸载应用才会真正删除它管理的 Kubernetes 资源。两个删除动作在界面上都需要输入名称二次确认，确认前请核对清楚当前操作的是仓库还是应用。