

什么是全局管理?

全局管理是以用户为中心的综合性服务板块，包含用户与访问控制、工作空间与层级、审计日志、平台设置等基础服务模块。

- [用户与访问控制](#): 帮助用户安全管理资源的访问权限。您可以通过用户与访问控制创建、管理、删除用户/用户组，并灵活配置用户/用户组权限，来完成用户职能权限的划分。
- [工作空间与层级](#): 具有层级结构和访问权限控制的资源隔离单元。您可以按照企业开发环境、部门结构等设置层级结构，并控制哪些人对哪些资源具有访问权限。
- [审计日志](#): 提供资源的操作记录。通过操作记录您可以快速实现安全分析、资源变更、问题定位等。
- [平台设置](#): 通过平台安全策略、邮件服务器、外观定制等，实现用户信息的安全性和平台的个性化。

全局管理优势

- 综合性汇总式服务

将平台的基础服务模块汇聚在一起，减少菜单切换和功能分散带来的困扰，使平台管理员在一个菜单下就能够完成基础性平台设置和用户管理。

- 用户与租户相结合

通过扁平化用户、用户组管理和层级式租户（工作空间）协作，多维度多场景实现基于平台的访问控制以及基于资源的权限划分。

- 个性化定制

支持个性化定制平台外观，包括登录页定制和顶部导航栏定制，通过可视化页面轻轻松

松配置一个属于您的个性化平台。

- 简化操作，上手即用

预先为用户完成了绝大多数的平台设置，包括密码规则配置、会话超时策略等，简化用

户操作，实现上手即用。

操作步骤

1. 使用 DCE 平台管理员（Admin）或具有管理员权限的用户登录 DCE 平台
2. 进入 **用户与访问控制**，[创建用户并授权](#), [创建用户组并授权](#), [创建身份提供商](#)
3. 进入 **工作空间**，[创建层级（企业层级关系）](#), [创建工作空间（租户）](#)
4. 进入 **审计日志**，查看并[导出审计日志](#)
5. 进入 **平台设置**，设置安全策略、邮件服务器、外观定制、正版授权

[下载 DCE 5.0 安装 DCE 5.0 申请社区免费体验](#)

功能总览

本页说明全局管理的各项功能特性。

1. 用户管理

拥有用户帐号，是用户访问 DCE 平台的前提。[用户](#)由平台管理员 Admin 或者用户与访问控制管理员 IAM Owner 在 **全局管理 -> 用户与访问控制 -> 用户** 页面创建，或者通过 LDAP 对接而来。每位用户拥有独立的用户名和密码，通过给单个或者一组用户授予不同的权限，让不同的用户拥有不同资源的访问权限。

```
graph LR
```

```

admin([平台管理员或<br>用户与访问控制管理员]) --> |创建并管理用户|user[用户]
user --> user1[用户 A]
user --> user2[用户 B]
user --> user3[用户 C]

click user "https://docs.daocloud.io/ghippo/user-guide/access-control/user/"

classDef plain fill:#ddd,stroke:#fff,stroke-width:0px,color:#000;
classDef k8s fill:#326ce5,stroke:#fff,stroke-width:0px,color:#fff;
classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5;
class admin plain;
class user1,user2,user3 k8s;
class user cluster

```

2. 用户组管理

用户组是多个用户的集合。 用户可以通过加入用户组，实现继承用户组的角色权限。

通过用户组批量地给用户进行授权，可以更好地管理用户及其权限。

```
graph LR
```

```

admin([平台管理员或<br>用户与访问控制管理员])
admin --> user[创建用户]
admin --> group[创建用户组]
admin --> add[将用户加入用户组]

click user "https://docs.daocloud.io/ghippo/user-guide/access-control/user/"
click group "https://docs.daocloud.io/ghippo/user-guide/access-control/group/"
click add "https://docs.daocloud.io/ghippo/user-guide/access-control/group/#_5"

classDef plain fill:#ddd,stroke:#fff,stroke-width:0px,color:#000;
classDef k8s fill:#326ce5,stroke:#fff,stroke-width:0px,color:#fff;
classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5;
class admin plain;
class user,group,add cluster

```

3. 角色管理

一个角色对应一组权限。 权限决定了可以对资源执行的操作。向用户授予某个角色，

即授予该角色所包含的所有权限。您可以将不同模块的管理权限划分给不同的用户，

比如用户 A 管理容器管理模块，用户 B 管理应用工作台模块，共同管理可观测性模块。

角色

角色

4. 工作空间

[工作空间](#)用于管理资源，包含层级和工作空间两部分。

层级是资源层次结构中的节点，一个层级可以包含工作空间、其他层级或两者的组合。

可以将层级理解为有层次结构的部门、环境、供应商等多种概念。

工作空间可以理解为部门下的项目，管理员通过层级和工作空间映射企业中的层级关系。

虽然一个层级可以包含多个层级或工作空间，但是给定的层级或者工作空间只能有一个

父级。

工作空间

工作空间

5. 审计日志

[审计日志](#)完整地记录用户的各项操作行为，包括用户通过页面或 API 接口发起的操作

以及各服务内部自触发的操作。支持通过事件来源、资源类型、操作状态等多个维

度进行组合查询，支持审计日志导出。

6. 平台设置

[平台设置](#)包括账号安全设置、[外观定制](#)、[邮件服务器](#)等。当需要对账号的安全信息、

平台 logo、许可证授权、邮件服务器等平台级设置进行管理时，可以通过 **平台设**

置 进行操作，平台设置仅平台管理员具有管理权限。

graph LR

admin([平台管理员]) --> |管理|about[平台设置]

about --> password[用户密码等安全策略]

about --> appear[平台外观定制]

about --> mail[邮件服务器]

```
about --> license[正版授权]

click about "https://docs.daocloud.io/ghippo/user-guide/platform-setting/about/"
click password "https://docs.daocloud.io/ghippo/user-guide/password/"
click appear "https://docs.daocloud.io/ghippo/user-guide/platform-setting/appearance/"
click mail "https://docs.daocloud.io/ghippo/user-guide/platform-setting/mail-server/"
click license "https://docs.daocloud.io/dce/license0/"

classDef plain fill:#ddd,stroke:#fff,stroke-width:0px,color:#000;
classDef k8s fill:#326ce5,stroke:#fff,stroke-width:0px,color:#fff;
classDef cluster fill:#fff,stroke:#bbb,stroke-width:2px,color:#326ce5;
class admin plain;
class about,password,appear,mail,license cluster
```

功能清单

具体的功能清单如下所述。

1. 用户与访问控制

1. 用户

- 列表展示用户名、描述、创建时间、最近一次登录时间
- 列表支持通过用户名搜索用户
- 列表支持给用户快捷授权
- 列表支持将用户快捷加入用户组
- 列表支持批量删除用户
- 列表支持创建用户
- 详情支持编辑用户基本信息，包括邮箱、描述、启用/禁用等
- 详情支持记录用户授权信息，支持添加/移除权限
- 详情支持记录用户加入用户组信息，支持用户加入新的用户组，支持将用户从老的用户组移除
- 支持管理员帮助用户修改密码

- 支持管理员帮助用户创建访问密钥

2. 用户组

- 列表展示用户组名、组内用户数、描述、创建时间
- 列表支持通过用户组名搜索
- 列表支持给用户组快捷授权
- 列表支持给用户组快捷添加用户
- 列表支持批量删除用户组
- 列表支持创建用户组
- 详情支持编辑用户组基本信息，例如描述
- 详情支持记录用户组授权信息，支持添加/移除权限
- 详情支持记录用户组成员信息，支持给用户组添加新成员，支持将成员从用户组移除

3. 角色

- 列表展示系统角色名称、描述
- 详情记录角色授权信息，支持将角色授予用户/用户组，支持将用户/用户组从该角色移除
- 支持 Folder Admin、Folder Editor、Folder Viewer 三种预定义文件夹角色
- 支持 Workspace Admin、Workspace Editor、Workspace Viewer 三种预定义工作空间角色

4. 身份提供商

- 支持 LDAP、OIDC、OAuth 2.0 等协议对接外部用户

- LDAP 协议支持手动/自动同步外部用户
- LDAP 协议支持手动同步外部用户组
- OIDC 协议支持手动同步外部用户
- OAuth 2.0 协议支持手动同步外部用户

2. 工作空间与层级

1. 文件夹

- 支持树状结构展示文件夹和工作空间
- 支持通过文件夹名称和工作空间名称搜索
- 支持 5 级文件夹映射企业层级
- 支持为文件夹添加用户/用户组并授权
- 支持权限继承，子文件夹、工作空间能够继承用户/用户组在上级文件夹权限
- 支持移动文件夹。映射企业中的部门变动，移动后其下的子文件夹、工作空间及其资源/成员均会跟随该文件夹移动，权限的继承关系重新发生变化

2. 工作空间

- 支持为工作空间添加用户/用户组并授权
- 支持添加资源到工作空间 - 资源组，支持 6 种资源类型
- 支持权限继承，资源组中的资源能够继承用户/用户组在工作空间和上级文件夹的角色
- 支持移动工作空间。映射企业中的项目变动，移动后其下的资源/成员均会跟随该工作空间移动，权限的继承关系重新发生变化（开发

中)

- 支持添加资源到工作空间 - 共享资源，将一个集群资源共享给多个工作空间使用
- 支持针对每个共享的集群资源进行资源限额
- 支持通过 CPU Limit、CPU Request 、内存 Limit、内存 Request、存储请求总量 Request Storage、存储卷声明 PersistentVolumeClaim 六个维度进行资源限额

3. 审计日志

- 列表展示事件名称、资源类型、资源名称、状态、操作人、操作时间
- 列表支持查看审计日志详情
- 列表支持展示最近一天或自定义时间内的审计日志
- 列表支持通过状态、操作人、资源类型、资源名称搜索审计日志
- 列表支持 .csv 格式导出审计日志
- 默认采集全局管理的全部审计日志
- 默认保存 365 天内的审计日志
- 支持手动/自动两种方式清理全局管理的审计日志
- 支持开启/关闭收集 K8s 的审计日志
- 支持手动/自动两种方式清理 K8s 的审计日志

4. 平台设置

1. 安全策略

- 支持自定义密码规则
- 支持自定义密码策略

- 支持自定义会话超时策略，超出时长自动退出当前账号
- 支持自定义账号锁定策略，限制时间内多次登录失败，账号将被锁定
- 支持登录登出策略，开启后关闭浏览器的同时退出登录

2. [邮件服务器设置](#)：支持管理员配置邮件服务器，支持通过邮件方式找回用户密码，接收告警通知等

3. [外观定制](#)

- 支持自定义登录页，包括更换平台 LOGO、登录页图标、标签页图标等
- 支持一键还原登录页外观配置
- 支持自定义顶部导航栏，包括导航栏图标、标签页图标等
- 支持一键还原顶部导航栏外观配置

4. [正版授权](#)

- 列表展示许可证名称、所属模块、许可证级别、许可证状态过期时间
- 支持管理许可证，确保子模块在有效期内

5. [关于平台](#)

- 支持展示模块版本
- 支持展示平台使用的开源软件
- 支持展示技术团队风采

5. 个人中心

1. [安全设置](#)

- 支持用户更新登录密码
- 支持通过邮箱找回登录密码

2. [访问密钥](#): 支持每个用户创建独立的 API 密钥，支持进行 API 密钥过期设置，以确保系统安全
3. [语言设置](#): 支持多语言，支持简体中文、英文、自动检测您的浏览器首选语言
三种方式确定语言

常见术语

本节列出全局管理的常见术语。

IAM

IAM (Identity and access management) 是用户与访问控制模块的简称，该模块的管理员被称为 IAM Admin，拥有该模块的最高权限。被赋予 IAM Owner 的用户（用户组），将拥有用户与访问控制的全部且最高权限。

更多详细信息，参见[什么是 IAM](#)。

RBAC

RBAC (Role-based access control，基于角色的访问控制)，基本理念是将[角色](#)这个概念赋予用户，在用户与权限之间通过角色进行关联，实现灵活配置。RBAC 模型的三要素为：用户、角色、权限。使用 RBAC 机制授权 IAM 用户访问平台资源。

用户

[用户](#)是发起操作的主体，每个用户都有唯一的 ID，并被授予不同的角色。默认创建的 IAM 用户没有任何权限，需要将其加入用户组，授予角色或策略，才能让用户获得对应的权限。

用户以用户名登录 DCE，按照被授予的权限操作平台资源和服务。所以用户是资源归属的主体，对其拥有的资源具有相应权限。

用户可以在[个人中心](#)修改用户信息，设置密码、访问密钥和 UI 语言。

用户组

[用户组](#)是一个或多个用户的集合，IAM 可以通过用户组实现用户的授权。通常先创建一个 IAM 用户，加入某个用户组，该用户将继承这个用户组的权限。当一个用户加入多个用户组时，该用户将同时拥有多个用户组的权限。

角色

[角色](#)是连接用户与权限的桥梁，一个角色对应一组权限，不同角色具有不同的权限。向用户授予某角色，即授予该角色所包含的所有权限。全局管理中有两种角色：

- 预定义角色：由系统创建，用户只能使用不能修改，每个子模块都有一个管理员 Admin 角色。
- 自定义角色：用户自主创建、更新和删除，自定义角色中的权限由用户自己维护。同时因为全局管理汇聚了多个子模块，各子模块也拥有相应的管理员角色，例如：
 - IAM Owner：管理用户与访问控制，即管理用户/用户组以及授权
 - Workspace Admin：管理层级及工作空间的权限，仅此权限可以创建层级
 - Audit Admin：管理审计日志

权限

[权限](#)指是否允许用户对某种资源执行某种操作。为了降低使用门槛，DCE 采用 RBAC 模型将权限聚合成一个个角色，管理员只需要将角色授权给用户，该用户就一次性得到了该角色

下聚合的一组权限。

默认情况下，管理员创建的 IAM 用户没有任何角色权限，需要对其单独授予角色或将其加入用户组并给用户组授予角色，才能使得用户获得对应的角色权限，这一过程称为授权。授权后，用户就可以基于被授予的角色权限对平台资源进行操作。

授权

[授权](#)指将用户完成具体工作所需的权限授予用户，授权通过系统角色或自定义角色的权限生效。 用户获得具体的权限后，可以对资源或服务进行操作。

工作空间

通过[工作空间](#)协调全局管理和子模块的权限关系，解决资源聚合和映射层级关系。通常一个工作空间对应一个项目，可以为每个工作空间分配不同的资源，指派不同的用户和用户组。

工作空间

工作空间

层级

参见上图，为了满足企业内各个部门的分支划分，DCE 引入了[层级](#)的概念，通常层级对应着不同的部门，每个层级可以包含一个或多个工作空间。

资源

[资源 \(Resource\)](#)泛指 DCE 平台上通过各个子模块创建的资源，是完成授权的具体数据。通常资源描述一个或多个操作对象，每个子模块拥有其各自的资源和对应的资源定义详情，如集群、Namesapce、网关等。

资源的拥有者是主账号 Admin。 Admin 具有在各子模块创建/管理/删除资源的权限，普通用户在没有授权的情况下，不会自动拥有资源的访问权限，需要 Admin 进行授权。 工作空间支持跨子模块授权用户（用户组）对于资源的访问权限。

身份凭证

[身份凭证](#)是识别用户身份的依据，登录和使用资源时，需要有身份凭证才能通过系统的鉴权认证。 身份凭证包括密码和访问密钥，可以通过 IAM 管理自己以及下属 IAM 用户的身份凭证。

有关更多术语，请参阅本站收录的[云原生词汇大全](#)。

全局管理 Release Notes

本页列出全局管理各版本的 Release Notes，便于您了解各版本的演进路径和特性变化。

2025-01-31

v0.34.0

- **新增** 全局管理使用 cert-manager 来生成和管理 SSL 证书
- **新增** 如果用户被删除/禁用，将联动禁用 ssh key 和访问密钥
- **新增** 公有云 feature gate
- **新增** 公有云用户注册
- **新增** 公有云用户个人信息页面增加手机号信息
- **新增** 公有云双侧边栏(管理员侧边栏/用户侧边栏)

- **修复** 修复工作空间 vGPU 已分配配额计算问题
- **修复** 修复用户列表搜索用户名包含 _ 无返回的异常问题
- **修复** 修复部分审计日志资源名称为空的问题
- **修复** 修复审计日志用户名显示异常的问题

2024-11-30

v0.33.0

- **新增** 升级 Istio 到 1.22.3
- **新增** 增加 虚拟机模块 许可证
- **新增 全局管理 ->运营管理** 计量计费支持沐曦 GPU
- **修复** 修复审计日志 Create-User 对应的资源名称显示异常问题

2024-10-31

v0.32.0

- **新增** 实现 Sidecar 版本角色权限 SDK
- **优化** Workspace Editor 增加 云原生 AI 相关权限点
- **优化** session-limit 接口实现用户名密码加密
- **修复** 修复 user ldap filter 重新设置为空同步后相应字段没有清空的问题

2024-9-30

v0.31.0

- **新增** 创建用户时用户名支持加密
- **新增** 自定义角色支持容器管理的文件夹和工作空间角色，且支持其权限点映射为容器管理中的预定义角色
- **修复** license 到期时间比较长时的错误问题
- **修复** 工作空间解绑资源报错问题
- **修复 全局管理 ->运营管理** 中相同类型报表的查询方法不一致的问题

2024-8-31

v0.30.0

- **新增** 配置及展示系统信息
- **新增** SDK RegisterDeltaResourcePermissionHandler 方法
- **新增 全局管理 ->运营管理** 租户视角及相关权限
- **优化** 用 trivy 和 gosec 来扫描硬编码证书等问题
- **优化 全局管理 ->运营管理** 开发指南
- **修复 全局管理 ->运营管理** 命名空间下如没有 pod 则报表不会展示对应的命名空间信息问题

2024-7-31**v0.29.0**

功能

- **新增** LDAP 服务器支持 LDAPS
- **新增 全局管理 -> 平台设置 -> 关于平台** 中安装器版本

优化

- **优化** 登录和修改密码功能支持用户名密码加密
- **优化 全局管理 -> 工作空间与层级 -> 资源组** 中绑定资源权限提示

修复

- **修复** SMTP 服务器设置
- **修复** OEM 配置为 enable 时无法安装的问题
- **修复 全局管理 ->运营管理** 中导出报表英文列名不一致问题

2024-6-30**v0.28.0**

功能

- **新增** OAuth2 Identity Provider 通用插件
- **新增** LDAP 配置的 username、userLdpaFilter 字段

- **新增 全局管理 ->运营管理 -> 计费配置** 中安装 insight-agent 提示

优化

- **优化** 创建用户，用户名前后有空格时自动去除后保存
- **优化** 资源搜索的绑定机制
- **优化** SDK 方法 ListWorkspaceUsersByPermission()

修复

- **修复** 自定义角色授权用户后能看到所有工作空间问题
- **修复** 浏览器关闭后再打开，可观测性等页面会出现 RBAC Access Denied 页面问题
- **修复 全局管理 ->运营管理** 修复报表导出字段名称与页面不一致问题

2024-5-31

v0.27.0

- **新增** 平台设置中的登录页背景视频开关
- **新增** 个人中心中 查看/增加/更新/删除 ssh 证书信息 功能
- **新增** 创建用户和修改密码：用户名密码作为参数传给 API 时支持加密
- **新增** Mysql 的 MGR 模式
- **优化** Keycloak 组件至 22.0.4
- **修复** License 按照 GPU 授权后 GPU 数量统计值问题

2024-4-30**v0.26.0**

- **新增** 支持 license 按照 GPU 授权
- **新增** 审计日志添加汇总链接
- **优化** 审计日志 SDK 单元测试
- **修复** 审计日志 user-agent 字段错误问题
- **修复** 审计日志登录记录经常一条失败一条成功的问题
- **修复** **运营管理** 修改计费配置后点击取消，校验提示未去除问题
- **修复** **运营管理** 容器组报表 GPU 计费时长计算问题

2024-4-1**v0.25.1**

- **修复** 工作空间资源组绑定命名空间资源配额检查问题

2024-3-31**v0.25.0**

功能

- **新增** 一级导航栏可配置成根据权限显示
- **新增** Workspace/Folder Editor 不支持修改 工作空间/层级 名称
- **新增** Ghippo 支持 ARM 离线安装包

- **新增** SDK 提供一个根据 Workspace ID 查找 Alias 的方法 GetWorkspaceById()
- **新增 运营管理 报表管理 -> 容器组报表 增加 GPU 统计指标**
- **新增 运营管理 计量计费 -> 容器组计费 增加 GPU 计费**

修复

- **修复** 全局管理登录前端 css 报错问题
- **修复** 可观测性界面 Refresh Token API 可能无法更新 Token 的问题

2024-1-31

v0.24.0

功能

- **新增** 资源配额支持限制 GPU
- **新增** 支持 DCE 5.0 的登录用户才能打开 Insight 的 Grafana 等组件的界面

优化

- **优化** 触发 FoldersAuthz CR 时用 single flight 机制来限流
- **优化** 过滤掉重复授权的情况
- **优化** Workspace 授权信息不存到 FoldersAuthz CR 里
- **优化** ghippo-controller-manager 重启时全量更新一次 FoldersAuthz CR

修复

- **修复** 大量授权请求导致 FoldersAuthz CR 更新过于频繁把 k8s 打爆问题

2023-12-29**v0.23.0**

功能

- **新增** DCE 5.0 退出时对应的身份提供商页面也要自动登出功能
- **新增** 用户名支持下划线等特殊字符功能
- **新增** 带特殊字符的用户名禁止授权功能废除
- **新增** Folder 之间用户隔离模式
- **新增** 支持 Folder/Workspace 用户授权多个角色

优化

- **优化** 初始化的用户名密码 (admin/changeme) 需要从 ConfigMap 里改存到 Secret 里

修复

- **修复** 离线环境文档站跳转问题
- **修复** 安装器 0.13 版本升级失败问题
- **修复** 工作空间大规模测试性能问题

2023-12-05**v0.22.1**

- **修复** 导航栏中间件显示不全问题

2023-11-30**v0.22.0**

- **新增** 自动展开 active 导航栏
- **新增** 用户名支持下划线 SDK
- **新增** DCE 4.0 迁移到 DCE 5.0 方案验证支持
- **新增** **运营管理** 添加列表精准搜索
- **优化** 关于平台：产品版本子模块 kcoral、dowl、kcollie、virtnest 支持中文
- **修复** 同步过来的 ldap 用户添加用户组失败问题
- **修复** 运营管理中工作空间报表展示不全的问题

2023-11-01**v0.21.0**

功能

- **新增** 对接多个 AD/LDAP 功能
- **新增** 对接多个 OIDC 功能
- **新增** 区分用户来源于哪个 LDAP/OIDC 功能
- **新增** 权限依赖项不允许取消功能
- **新增** 后端定制一级导航栏功能
- **新增** 支持非中文语言下跳转至英文文档站

优化

- **优化** 界面无端报错 400 问题
- **优化** 运营管理中的报表空数据展示
- **优化** 运营管理中的集群计费列表节点数量跳转至节点计费列表的功能

修复

- **修复** Ghippo APIServer 启动时可能会死锁的问题
- **修复** 运营管理中的集群计费计算费用错误的问题

2023-09-04

v0.20.1

- **修复** 界面无端报错 400 问题

2023-08-30

v0.20.0

- **新增** 对接企业微信功能
- **新增** 密码支持 8-32 位
- **新增** 支持云边协同 Kant 对接 license
- **新增** 运营管理模块支持 postgres/kingbase 数据库
- **优化** 给部分没有主键的数据库表加上主键

2023-07-28**v0.19.0**

功能

- **新增** 支持对接项目组 IDP 登录免点击
- **新增** 资源组绑定 mesh / mesh-namespace
- **新增** 自定义角色权限点增加 tips
- **新增** 平台设置 -> 安全策略 -> 对单个用户的多重并发会话进行限制
- **新增** 平台设置 -> 安全策略 -> 能够对系统的最大并发会话连接数进行限制

优化

- **优化** Webhook URL 支持自定义参数

修复

- **修复** 填写邮件服务器时报错问题
- **修复** LDAP 用户加入用户组失败问题

2023-07-06**v0.18.1**

- **修复** 工作空间集群名称可能为空的问题
- **修复** Folder Admin 角色工作空间授权列表权限问题
- **修复** Ghippo 0.17 升级到 0.18 失败的问题

- **修复** 新增共享资源，选择集群类型 options 都相同的问题
- **修复** 资源组列表，网格类型资源不能解绑

2023-06-29

v0.18.0

功能

- **新增** 移动工作空间
- **新增** 接入管理支持 Webhook (创建/编辑/删除/列表/查看)
- **新增** 接入管理支持 Webhook 记录后台自动清理
- **新增** License 过期前提醒
- **新增** License 过期后功能不可用
- **新增** 用户在修改密码时需要输入旧密码
- **新增** 资源组支持绑定服务网格资源
- **新增** 支持外接 Postgres 数据库
- **新增** 运营管理报表支持更多类型的货币
- **新增** Istio-ingressgateway 和 Istiod 支持高可用

优化

- **优化** 后台配置的密码存入 secret

修复

- **修复** 审计日志时区问题

- **修复** 工作空间授权时角色列表不能显示问题

2023-06-02

v0.17.1

- **修复** 迁移文件错误导致迁移失败的问题
- **修复** 创建工作空间时，角色只能选择 Workspace Admin 的问题

2023-05-30

v0.17.0

功能

- **新增** 接入管理中接入客户端（创建/查看/列表/更新/删除）功能
- **新增** 接入管理中 Webhook API（创建/编辑/删除/列表/查看）功能
- **新增** 用户 CRUD/Login/Logout 事件触发 Webhook
- **新增** 接入管理权限点
- **新增** 用户名/用户组名支持 . 和 @
- **新增** 平台默认语言改成 自动检测浏览器首选项
- **新增** 审计日志支持下载为 Excel 和 CSV 格式
- **新增** 两种日志（系统和用户）在审计日志中分开显示
- **新增** 禁止移除 admin 用户的 admin 角色或编辑 admin 的权限
- **新增** 工作空间授权展示角色的权限详情
- **新增** 单元测试的覆盖率，只能够上升，不能够下降

- **新增** OpenAPI 支持版本，OpenAPI 文档支持弃用（其他 GProduct 模块必须同步更新到安装器 v0.8.0 里对应的版本）
- **新增** 运营管理：GMagpie 使用数据库配置规范（Helm 参数、DSN）对接安装器
- **新增** 运营管理中 Excel 和 CSV 为支持下载格式
- **新增** 运营管理 Workspace 报表
- **新增** 运营管理 Workspace 计费报表
- **新增** 运营管理容器组报表
- **新增** 运营管理容器组计费报表
- **新增** 运营管理 Namespace 计费报表

优化

- **优化** 角色列表默认排序
- **优化** 修改 Workspace Editor 权限点

修复

- **修复** LDAP 同步用户组的同步时间问题

2023-04-28

v0.16.1

- **修复** 死循环导致的 CPU 占用过高问题
- **修复** 审计日志导出不全的问题

2023-04-27**v0.16.0**

功能

- **新增** LDAP 支持 AD
- **新增** LDAP 支持更多参数
- **新增** 支持修改登录页背景图
- **新增** 支持首页和登录页下方备案信息修改 (更新/展示)
- **新增** 数据库连接 Helm 参数格式修改
- **新增** 接入管理：接入客户端 API (创建/查看/列表/更新/删除)
- **新增** 反代 URL 支持前面加 Path
- **新增** 审计日志 SDK 直接调用 AuditServer API
- **新增** 审计日志用户和系统两种审计分表存，分 API 存取
- **新增** 去除 API Server 访问外网的依赖

优化

- **优化** 创建自定义角色时，过滤不存在的权限
- **优化** 简化审计日志数据库分区功能

修复

- **修复** 自定义角色类型显示错误问题

2023-03-29**v0.15.0**

功能

- **新增** 自定义角色功能(创建/编辑/删除/查看/列表)
- **新增** 支持 GProduct 权限点及自定义角色功能对接
- **新增** 失联集群资源处理
- **新增** Ghippo OpenAPI 文档实现
- **新增** 给 GProduct 提供插入审计日志 SDK

修复

- **修复** 密钥过期后 token 依旧可以使用的问题
- **修复** 镜像仓库不受 license 控制问题

2023-02-27**v0.14.0**

功能

- **新增** 平台设置 -> 外观定制 -> 高级定制 -> 登录页定制及登录后页面定制
- **新增** 支持 keycloak quarkus 架构在双栈环境运行
- **新增** CI 过程中加入 Job 检测 Helm chart 包中是否有不合规的镜像和检测安装参数是否正确

- **新增** 在 Helm values 里设置开关，可以一键开关 audit 相关功能
- **新增** 审计日志支持记录 kpanda 页面操作

优化

- **优化** OpenAPI 调用方式

修复

- **修复** 部分错误的审计日志名称
- **修复** keycloak 启动探针 failureThreshold 值，提高启动成功率
- **修复** 绑定/解绑资源错误 i18n

2022-12-30

v0.13.2

- **修复** 界面缺少权限说明的英文文案
- **修复** 数据库表更新时，可能因为数据库编码导致报错的问题

2022-12-28

v0.13.0

功能

- **新增** DCE 5.0 一个国密网关，并用国密浏览器来访问 DCE 5.0
- **新增** Helm values 里的开关，可以一键开关 Istio Sidecar 功能

- **新增** 工作空间与层级中 Workspace and Folder Owner 角色
- **新增** 拥有 Workspace/Folder Admin 和 Kpanda Owner 权限的用户才可进行资源绑定的设置
- **新增** 对所用的库，进行开源 License 扫描
- **新增** 用户列表 状态 列
- **新增** 对内提供的离线安装文档
- **新增** SDK 单元测试达 65%
- **新增** 界面支持发送测试邮件和无账号密码邮件服务器
- **新增** 界面支持对不符合系统要求的用户名进行提示

优化

- **优化** Ghippo 鉴权代码优化（减少内存使用量）
- **优化** 前端界面低网络情况下预加载机制优化

修复

- **修复** OpenAPI cycle 为必填参数问题，修复后为可选参数

2022-11-30

v0.12.1

优化

- **优化** 通过 CI 自动构建纯离线包
- **优化** Ghippo 升级文档

2022-11-28

v0.12.0

功能

- **新增** 资源组里的 **模块** 改为 **来源**
- **新增** SDK 提供 Workspace 和 Resource 的绑定变化通知
- **新增** 完整对接 Insight metrics 和 otel tracing (加入 keycloak 和 db 链路)
- **新增** Keycloak 改成 Quarkus 架构
- **新增** Keycloak 镜像升级成 20.0.1 版本

优化

- **优化** 重构导出审计日志 http 接口为 gRPC stream 接口
- **优化** SDK 内存使用量优化，峰值减少 50%
- **优化** 审计日志部分代码优化
- **优化** e2e 的 kind 镜像切到 1.25
- **优化** 提高资源使用效率到 40% 以上

修复

- **修复** 强绑定集群问题
- **修复** 配置身份提供商界面选择 **Client secret sent as basic auth** 没有保存到 keycloak 里的问题

2022-11-01**v0.11.2**

修复

- **修复** 关闭资源组绑定集群功能
- **修复** 无工作空间时无法创建工作空间问题

2022-10-28**v0.11.0**

功能

- **新增** 给第三方应用提供接口在 Keycloak 创建 SSO 对接 Client
- **新增** 支持 Mysql8
- **新增** 对接 Insight (metrics, log, otel tracing)
- **新增** License 模块名支持 i18n
- **新增** 支持一个 License 中可以包含多个 GProduct
- **新增** 资源组新增绑定集群类型资源
- **新增** 资源组列表增加“模块”字段
- **新增** 资源组列表增加已绑定标识
- **新增** 资源绑定接口支持 Registry 资源种类。

优化

- **优化** 资源种类枚举
- **优化** GProduct license 是否需要灌入变量改为可配
- **优化** CICD 流程

修复

- **修复** 已经删除的集群依然存在问题
- **修复** keycloak jwks 变化后没有重置 Istio 缓存问题
- **修复** 用户组创建时间零值问题
- **修复** 访问密钥 **最后使用时间** 字段在未使用时返回空字符

2022-09-28

v0.10.0

功能

- **新增** 支持登录
- **新增** 支持忘记密码
- **新增** 支持站内信增删改查功能
- **新增** 支持 SMTP 设置邮件服务器
- **新增** 支持顶部导航栏获取查询
- **新增** 支持顶部导航栏更新
- **新增** 支持用户角色权限管理 CRUD

- **新增** 工作空间 -> 生命周期管理 (创建/编辑/删除/查看/列表)
- **新增** 工作空间 -> 层级关系管理 (绑定/列表)
- **新增** 工作空间 -> 工作空间与资源关系管理 (绑定/解绑/列表)
- **新增** 工作空间 -> 工作空间和角色和用户 (组) 关系管理 (绑定/解绑/列表) (API/SDK)
- **新增** 工作空间 -> 鉴权 (API/SDK)
- **新增** 工作空间 -> GProduct 资源名字注册
- **新增** 关于 -> 产品版本 (创建/编辑/删除/查看/列表)
- **新增** 关于 -> 开源软件 (列表/初始化)
- **新增** 关于 -> 技术团队 (列表/初始化)
- **新增** 许可证 -> 生命周期管理 (创建/编辑/删除/查看/列表)
- **新增** 许可证 -> 获取 ESN 序列号
- **新增** 许可证 -> 未灌入或错误情况处理
- **新增** 工作空间 -> 资源配额管理 (创建/编辑/删除/查看/列表/计算已分配)
- **新增** 工作空间 -> GProduct 资源配额注册
- **新增** 用户与访问控制 -> 鉴权 (APIServer/SDK)
- **新增** 审计日志 -> 展示 (查看/列表/清理设置/导出)
- **新增** 审计日志 -> 批量插入
- **新增** 身份提供商 -> 对接 LDAP -> 用户/用户组同步设置 (创建/编辑/删除/查看/同步)
- **新增** 平台设置 -> 安全策略 -> 密码策略设置
- **新增** 个人中心 -> 访问密钥(创建/编辑/删除/查看/列表)
- **新增** 审计日志 -> 全局管理操作插入审计日志
- **新增** 审计日志 -> 对接 Insight 来收集审计日志

- **新增** 平台设置 -> 安全策略 -> 账号锁定策略
- **新增** 平台设置 -> 安全策略 -> 浏览器关闭策略
- **新增** 身份提供商 -> 对接 IDP (OIDC 协议)
- **新增** 工作空间 -> 共享集群权限管理
- **新增** 工作空间 -> 共享集群配额管理 -> 存储
- **新增** 平台设置 -> 顶部导航外观定制 -> 重置功能
- **新增** 平台设置 -> 安全策略 -> 会话超时策略
- **新增** 审计日志 -> 自动清理功能
- **新增** 平台设置 -> 安全策略 -> 账号锁定策略
- **新增** 平台设置 -> 顶部导航外观定制 -> 还原功能
- **新增** 平台设置 -> 登录页外观定制 -> 还原功能
- **新增** 产品导航 -> 首页仅对 admin 用户展示
- **新增** 工作空间 -> 用户仅能查看有权限的工作空间与层级的树状结构
- **新增** Keycloak 高可用
- **新增** 邮件服务器配置 -> 支持 Insight 和应用工作台发送邮件
- **新增** 满足 Helm 规范，支持安装器和离线化
- **新增** 审计日志 -> 数据库自动创建和合并分区
- **新增** 支持 ARM64 架构
- **新增** 支持 https
- **新增** 登录 -> 背景 theme 支持动画
- **新增** 授权鉴权 -> 给前端提供当前登录用户的权限列表
- **新增** 关于 -> 软件版本 -> 模块支持中文名

- **新增** 添加整体双语文档站结构及主要内容

优化

- **优化** 授权鉴权 -> 提供一个 Job 来确保数据库和自定义资源的同步
- **优化** LDAP -> 配置错误检查
- **优化** 各功能操作反馈和提示语报错支持中英文
- **优化** 工作空间及层级 -> 删除前对是否存在子资源进行检查
- **优化** keycloak jvm 参数
- **优化** 通过 mockery 框架简化 mock

离线升级全局管理模块

本页说明[下载全局管理模块](#)后，应该如何安装或升级。

!!! info

下述命令或脚本内出现的 `ghippo` 字样是全局管理模块的内部开发代号。

从安装包中加载镜像

!!! info

前置条件：将离线包上传至目标节点

您可以根据下面两种方式之一加载镜像，当环境中存在镜像仓库时，建议选择 chart-syncer 同步镜像到镜像仓库，该方法更加高效便捷。

chart-syncer 同步镜像到镜像仓库

1. 创建 load-image.yaml

!!! note

该 YAML 文件中的各项参数均为必填项。您需要一个私有的镜像仓库，并修改相关配置。

==== “已安装 chart repo”

若当前环境已安装 chart repo, chart-syncer 也支持将 Chart 导出为 tgz 文件。

```
```yaml title="load-image.yaml"
source:
 intermediateBundlesPath: ghippo-offline # (1)!
target:
 containerRegistry: 10.16.10.111 # (2)!
 containerRepository: release.daocloud.io/ghippo # (3)!
repo:
 kind: HARBOR # (4)!
 url: http://10.16.10.111/chartrepo/release.daocloud.io # (5)!
 auth:
 username: "admin" # (6)!
 password: "Harbor12345" # (7)!
containers:
 auth:
 username: "admin" # (8)!
 password: "Harbor12345" # (9)!
```
```

```

1. 到执行 charts-syncer 命令的相对路径, 而不是此 YAML 文件和离线包之间的相对路径
2. 需更改为你的镜像仓库 url
3. 需更改为你的镜像仓库
4. 也可以是任何其他支持的 Helm Chart 仓库类别
5. 需更改为 chart repo url
6. 你的镜像仓库用户名
7. 你的镜像仓库密码
8. 你的镜像仓库用户名
9. 你的镜像仓库密码

### ==== “未安装 chart repo”

若当前环境未安装 chart repo, chart-syncer 也支持将 Chart 导出为 tgz 文件, 并存放在指定路径。

```
```yaml title="load-image.yaml"
source:
  intermediateBundlesPath: ghippo-offline # (1)!
target:
  containerRegistry: 10.16.10.111 # (2)!
  containerRepository: release.daocloud.io/ghippo # (3)!
repo:
```
```

```

```

kind: LOCAL
path: ./local-repo # (4)!
containers:
  auth:
    username: "admin" # (5)!
    password: "Harbor12345" # (6)!

```

```

1. 到执行 charts-syncer 命令的相对路径，而不是此 YAML 文件和离线包之间的相对路径
2. 需更改为你的镜像仓库 URL
3. 需更改为你的镜像仓库
4. Chart 本地路径
5. 你的镜像仓库用户名
6. 你的镜像仓库密码

## 2. 执行同步镜像命令。

```
charts-syncer sync --config load-image.yaml
```

## Docker 或 containerd 直接加载

解压并加载镜像文件。

### 1. 解压 tar 压缩包。

```
tar xvf ghippo.bundle.tar
```

解压成功后会得到几个文件：

- hints.yaml
- images.tar
- original-chart

### 2. 从本地加载镜像到 Docker 或 containerd。

```

==== "Docker"
```
shell
  docker load -i images.tar
```

==== "containerd"
```
shell
  ctr -n k8s.io image import images.tar
```

```

### !!! note

每个 node 都需要做 Docker 或 containerd 加载镜像操作，  
加载完成后需要 tag 镜像，保持 Registry、Repository 与安装时一致。

# 升级

升级注意事项：

==== “从 v0.11.x 升级到 ≥v0.12.0”

当从 v0.11.x (或更低版本) 升级到 v0.12.0 (或更高版本) 时，需要将 `__bak.yaml__` 中所有 `keycloak` key 修改为 `keycloakx` 。

修改前：

```
```yaml title="bak.yaml"
USER-SUPPLIED VALUES:
keycloak:
...
```

```

修改后：

```
```yaml title="bak.yaml"
USER-SUPPLIED VALUES:
keycloakx:
...
```

```

==== “从 v0.15.x 升级到 ≥v0.16.0”

当从 v0.15.x (或更低版本) 升级到 v0.16.0 (或更高版本) 时，需要修改数据库连接参数。

修改前：

```
```yaml title="bak.yaml"
USER-SUPPLIED VALUES:
global:
  database:
    host: 127.0.0.1
    port: 3306
    apiserver:
      dbname: ghippo
      password: passowrd
      user: ghippo
  keycloakx:
    dbname: keycloak
    password: passowrd
    user: keycloak
```

```

```

auditDatabase:
 auditserver:
 dbname: audit
 password: passowrd
 user: audit
 host: 127.0.0.1
 port: 3306
```

```

修改后：

```

```yaml title="bak.yaml"
USER-SUPPLIED VALUES:
global:
 storage:
 ghippo:
 - driver: mysql
 accessType: readwrite
 dsn: {global.database.apiserver.user}:{global.database.apiserver.password}@tcp({global.database.host}:{global.database.port})/{global.database.apiserver.dbname}?charset=utf8mb4&multiStatements=true&parseTime=true
 audit:
 - driver: mysql
 accessType: readwrite
 dsn: {global.auditDatabase.auditserver.user}:{global.auditDatabase.auditserver.password}@tcp({global.auditDatabase.host}:{global.auditDatabase.port})/{global.auditDatabase.auditserver.dbname}?charset=utf8mb4&multiStatements=true&parseTime=true
 keycloak:
 - driver: mysql
 accessType: readwrite
 dsn: {global.database.keycloakx.user}:{global.database.keycloakx.password}@tcp({global.database.host}:{global.database.port})/{global.database.keycloakx.dbname}?charset=utf8mb4
```

```

有两种升级方式。您可以根据前置操作，选择对应的升级方案：

==== “通过 Helm 仓库升级”

1. 检查全局管理 Helm 仓库是否存在。

```

```shell
helm repo list | grep ghippo
```

```

若返回结果为空或如下提示，则进行下一步；反之则跳过下一步。

```
```none
Error: no repositories to show
```

```

- 添加全局管理的 Helm 仓库。

```
```shell
helm repo add ghippo http://{harbor url}/chartrepo/{project}
```

```

- 更新全局管理的 Helm 仓库。

```
```shell
helm repo update ghippo # (1)!
```

```

- Helm 版本过低会导致失败，若失败，请尝试执行 `helm update repo`
- 选择您想安装的全局管理版本（建议安装最新版本）。

```
```shell
helm search repo ghippo/ghippo --versions
```

```

```
```none
NAME CHART VERSION APP VERSION DESCRIPTION
ghippo/ghippo 0.9.0 v0.9.0 A Helm chart for GHippo
...
```

```

- 备份 `--set` 参数。

在升级全局管理版本之前，建议您执行如下命令，备份老版本的 `--set` 参数。

```
```shell
helm get values ghippo -n ghippo-system -o yaml > bak.yaml
```

```

- 更新 Ghippo CRD:

```
```shell
helm pull ghippo/ghippo --version 0.9.0 && tar -zxf ghippo-0.9.0.tgz
kubectl apply -f ghippo/crds
```

```

```
```
```

1. 执行 `helm upgrade`。

升级前建议您覆盖 bak.yaml 中的 `global.imageRegistry` 字段为当前使用的镜像仓库地址。

```
```shell
export imageRegistry={你的镜像仓库}
```

```

```
```shell
helm upgrade ghippo ghippo/ghippo \
-n ghippo-system \
-f ./bak.yaml \
--set global.imageRegistry=$imageRegistry \
--version 0.9.0
```

```

#### ==== “通过 Chart 包升级”

1. 备份 `--set` 参数。

在升级全局管理版本之前，建议您执行如下命令，备份老版本的 `--set` 参数。

```
```shell
helm get values ghippo -n ghippo-system -o yaml > bak.yaml
```

```

1. 更新 Ghippo CRD:

```
```shell
kubectl apply -f ./crds
```

```

1. 执行 `helm upgrade`。

升级前建议您覆盖 bak.yaml 中的 `global.imageRegistry` 为当前使用的镜像仓库地址。

```
```shell
export imageRegistry={你的镜像仓库}
```

```

```
```shell
helm upgrade ghippo . \
-n ghippo-system \

```

```
-f ./bak.yaml \
--set global.imageRegistry=$imageRegistry
```

```

# 自定义 DCE 5.0 反向代理服务器地址

具体设置步骤如下：

1. 检查全局管理 helm 仓库是否存在。

```
helm repo list | grep ghippo
```

若返回结果为空或如下提示，则进行下一步；反之则跳过下一步。

```
Error: no repositories to show
```

2. 添加并且更新全局管理的 helm 仓库。

```
helm repo add ghippo http://{harbor url}/chartrepo/{project}
helm repo update ghippo
```

3. 设置环境变量，方便在下文中使用。

```
您的反向代理地址，例如 `export DCE_PROXY="https://demo-alpha.daocloud.io"`
export DCE_PROXY="https://domain:port"

helm --set 参数备份文件
export GHIPPO_VALUES_BAK="ghippo-values-bak.yaml"

获取当前 ghippo 的版本号
export GHIPPO_HELM_VERSION=$(helm get notes ghippo -n ghippo-system | grep "Chart Version" | awk -F ': ' '{ print $2 }')
```

4. 备份 -set 参数。

```
helm get values ghippo -n ghippo-system -o yaml > ${GHIPPO_VALUES_BAK}
```

5. 添加您的反向代理地址。

**!!! note**

- 如果可以，您可以使用 yq 命令：

```
```shell
yq -i ".global.reverseProxy = \"${DCE_PROXY}\\" ${GHIPPO_VALUES_BAK}
```

```

- 或者您可以使用 vim 命令编辑并保存：

```

```shell
vim ${GHIPPO_VALUES_BAK}

USER-SUPPLIED VALUES:
...
global:
...
reverseProxy: ${DCE_PROXY} # 只需要修改这一行
```

```

#### 6. 执行 helm upgrade 使配置生效。

```

helm upgrade ghippo ghippo/ghippo \
-n ghippo-system \
-f ${GHIPPO_VALUES_BAK} \
--version ${GHIPPO_HELM_VERSION}

```

#### 7. 使用 kubectl 重启全局管理 Pod，使配置生效。

```

kubectl rollout restart deploy/gippo-apiserver -n gippo-system
kubectl rollout restart statefulset/gippo-keycloak -n gippo-system

```

## 开启 Folder/WS 之间的隔离模式

具体设置步骤如下：

#### 1. 检查全局管理 helm 仓库是否存在。

```
helm repo list | grep ghippo
```

若返回结果为空或如下提示，则进行下一步；反之则跳过下一步。

```
Error: no repositories to show
```

#### 2. 添加并且更新全局管理的 helm 仓库。

```

helm repo add ghippo http://{harbor url}/chartrepo/{project}
helm repo update ghippo

```

#### 3. 设置环境变量，方便在下文中使用。

```

helm --set 参数备份文件
export GHIPPO_VALUES_BAK="gippo-values-bak.yaml"

获取当前 gippo 的版本号

```

```
export GHIPPO_HELM_VERSION=$(helm get notes ghippo -n ghippo-system | grep "Chart Version" | awk -F ': ' '{ print $2 }')
```

#### 4. 备份 -set 参数。

```
helm get values ghippo -n ghippo-system -o yaml > ${GHIPPO_VALUES_BAK}
```

#### 5. 打开 Folder/WS 之间的隔离模式开关。

**!!! note**

- 如果可以，您可以使用 `yq` 命令：

```
```shell
yq -i ".apiserver.userIsolationMode = \"Folder\"" ${GHIPPO_VALUES_BAK}
```

```

- 或者您可以使用 `vim` 命令编辑并保存：

```
```shell
vim ${GHIPPO_VALUES_BAK}

```

USER-SUPPLIED VALUES:

...

添加下面两行即可

```
apiserver:
  userIsolationMode: Folder
```

```

#### 6. 执行 `helm upgrade` 使配置生效。

```
helm upgrade ghippo ghippo/ghippo \
-n ghippo-system \
-f ${GHIPPO_VALUES_BAK} \
--version ${GHIPPO_HELM_VERSION}
```

#### 7. 使用 `kubectl` 重启全局管理 Pod，使配置生效。

```
kubectl rollout restart deploy/ghippo-apiserver -n ghippo-system
```

## 使用国密网关代理 DCE 5.0

参照以下步骤为 DCE 5.0 配置国密网关。

## 软件介绍

**Tengine:** Tengine 是由淘宝网发起的 Web 服务器项目。它在 Nginx 的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。比如支持 Tongsuo 插件，支持国密证书等。

**Tongsuo:** 铜锁/Tongsuo (原 BabaSSL) 是一个提供现代密码学算法和安全通信协议的开源基础密码库，为存储、网络、密钥管理、隐私计算等诸多业务场景提供底层的密码学基础能力，实现数据在传输、使用、存储等过程中的私密性、完整性和可认证性，为数据生命周期中的隐私和安全提供保护能力。

## 准备工作

一台安装了 Docker 的 Linux 主机，并且确保它能访问互联网。

## 编译和安装国密网关

下面介绍如何使用 Tengine 和 Tongsuo 构建国密网关。

!!! note

此配置仅供参考。

**FROM** docker.m.daocloud.io/debian: 11.3

```
Version
ENV TENGINE_VERSION="2.3.4" \
 TONGSUO_VERSION="8.3.2"

Install required system packages and dependencies
RUN apt update && \
 apt -y install \
 wget \
 gcc \
 make \
 libpcre3 \
 libpcre3-dev \
 zlib1g-dev \
 perl \
```

```

 && apt clean

Build tengine
RUN mkdir -p /tmp/pkg/cache/ && cd /tmp/pkg/cache/ \
 && wget https://github.com/alibaba/tengine/archive/refs/tags/${TENGINE_VERSION}.tar.gz -O tengine-${TENGINE_VERSION}.tar.gz \
 && tar zxvf tengine-${TENGINE_VERSION}.tar.gz \
 && wget https://github.com/Tongsuo-Project/Tongsuo/archive/refs/tags/${TONGSUO_VERSION}.tar.gz -O Tongsuo-${TONGSUO_VERSION}.tar.gz \
 && tar zxvf Tongsuo-${TONGSUO_VERSION}.tar.gz \
 && cd tengine-${TENGINE_VERSION} \
 && ./configure \
 --add-module=modules/ngx_openssl_ntls \
 --with-openssl=/tmp/pkg/cache/Tongsuo-${TONGSUO_VERSION} \
 --with-openssl-opt="--strict-warnings enable-ntls" \
 --with-http_ssl_module --with-stream \
 --with-stream_ssl_module --with-stream_sni \
 && make \
 && make install \
 && ln -s /usr/local/nginx/sbin/nginx /usr/sbin/ \
 && rm -rf /tmp/pkg/cache

EXPOSE 80 443
STOP SIGNAL SIGTERM
CMD ["nginx", "-g", "daemon off;"]
docker build -t tengine: 0.0.1 .

```

## 生成 SM2 和 RSA TLS 证书

下面介绍如何生成 SM2 和 RSA TLS 证书，并配置国密网关。

### SM2 TLS 证书

**!!! note**

此证书仅适用于测试环境。

您可以参考 [Tongsuo 官方文档](#) 使用 [OpenSSL 生成 SM2 证书](#)，或者访问[国密 SSL 实验室申请 SM2 证书](#)。

最终我们会得到以下文件：

```
- rw-r--r-- 1 root root 749 Dec 8 02:59 sm2.*.enc.crt.pem
- rw-r--r-- 1 root root 258 Dec 8 02:59 sm2.*.enc.key.pem
- rw-r--r-- 1 root root 749 Dec 8 02:59 sm2.*.sig.crt.pem
- rw-r--r-- 1 root root 258 Dec 8 02:59 sm2.*.sig.key.pem
```

## RSA TLS 证书

```
- rw-r--r-- 1 root root 216 Dec 8 03:21 rsa.*.crt.pem
- rw-r--r-- 1 root root 4096 Dec 8 02:59 rsa.*.key.pem
```

## 给国密网关配置 SM2 和 RSA TLS 证书

本文中使用的国密网关，支持 SM2 和 RSA 等 TLS 证书。双证书的优点是：当浏览器不支持 SM2 TLS 证书时，自动切换到 RSA TLS 证书。

更多详细配置，请参考 [Tongsuo 官方文档](#)。

我们进入 Tengine 容器内部：

```
进入 nginx 配置文件存放目录
cd /usr/local/nginx/conf

创建 cert 文件夹，用于存放 TLS 证书
mkdir cert

把 SM2、RSA TLS 证书拷贝到 `/usr/local/nginx/conf/cert` 目录下
cp sm2.*.enc.crt.pem sm2.*.enc.key.pem sm2.*.sig.crt.pem sm2.*.sig.key.pem /usr/local/nginx/conf/cert
cp rsa.*.crt.pem rsa.*.key.pem /usr/local/nginx/conf/cert

编辑 nginx.conf 配置
vim nginx.conf
...
server {
 listen 443 ssl;
 proxy_http_version 1.1;
 # 启用国密功能，使其支持 SM2 算法的 TLS 证书
 enable_ntls on;

 # RSA 证书
 # 如果您的浏览器不支持国密证书，那么您可以开启此选项，Tengine 会自动识别最终用户的
 # 浏览器，并使用 RSA 证书进行回退
```

```
ssl_certificate /usr/local/nginx/conf/cert/rsa.*.crt.pem;
ssl_certificate_key /usr/local/nginx/conf/cert/rsa.*.key.pem;

配置两对 SM2 证书，用于加密和签名
SM2 签名证书
ssl_sign_certificate /usr/local/nginx/conf/cert/sm2.*.sig.crt.pem;
ssl_sign_certificate_key /usr/local/nginx/conf/cert/sm2.*.sig.key.pem;
SM2 加密证书
ssl_enc_certificate /usr/local/nginx/conf/cert/sm2.*.enc.crt.pem;
ssl_enc_certificate_key /usr/local/nginx/conf/cert/sm2.*.enc.key.pem;
ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3;

location / {
 proxy_set_header Host $http_host;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header REMOTE-HOST $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 # 您需要将这里的地址修改为 Istio 入口网关的地址
 # 例如 proxy_pass https://istio-ingressgateway.istio-system.svc.cluster.local
 # 或者 proxy_pass https://demo-dev.daocloud.io
 proxy_pass https://istio-ingressgateway.istio-system.svc.cluster.local;
}
}
```

## 重新加载国密网关的配置

```
nginx -s reload
```

## 下一步

国密网关部署成功之后，[自定义 DCE 5.0 反向代理服务器地址](#)。

## 验证

您可以部署一个支持国密证书的 Web 浏览器。例如 [Samarium Browser](#)，然后通过 Tengine 访问 DCE5 UI 界面，验证国密证书是否生效。

# 登录

用户在使用一个新系统前，在这个系统中是没有任何数据的，系统也无法识别这个新用户。

为了标识用户身份、绑定用户数据，用户需要一个能唯一标识用户身份的帐号。

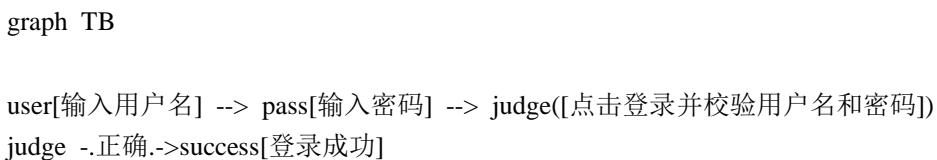
DCE 5.0 在 **用户与访问控制** 中通过管理员创建新用户的方式为用户分配一个附有一定权限的账号。该用户产生的所有行为都将关联到自己的帐号。

用户通过账号/密码进行登录，系统验证身份是否合法，如果验证合法，则用户成功登录。

#### !!! note

如果用户登录后 24 小时内无任何操作，将自动退出登录状态。如果登录的用户始终活跃，将持续处于登录状态。

用户登录的简单流程如下图。



```
classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000;
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:1px,color:#fff;
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;

class user,pass cluster;
class judge plain
class success,fail k8s
```

用户登录界面如下图。具体登录画面，请与实际产品为准。

登录界面

登录界面

# 什么是用户与访问控制

IAM (Identity and Access Management, 用户与访问控制) 是全局管理的一个重要模块，您可以通过用户与访问控制模块创建、管理和销毁用户（用户组），并使用系统角色和自定义角色控制其他用户使用 DCE 平台的权限。

## IAM 定义

### IAM 定义

## 优势

- 简洁流畅

企业内部的结构和角色可能非常复杂，项目、工作小组及授权的管理都在不断地变化。

用户与访问控制采用清晰整洁的页面，打通用户、用户组、角色之间的授权关系，以最短链路实现对用户（用户组）的授权。

- 适当的角色

用户与访问控制为每个子模块预定义了一个管理员角色，无需用户维护，您可以直接将

平台预定义的系统角色授权给用户，实现平台的模块化管理（细粒度权限请参阅[权限管理](#)。

- 企业级访问控制

当您希望本企业员工可以使用企业内部的认证系统登录 DCE 平台，而不需要在 DCE 平台创建对应的用户，您可以使用用户与访问控制的身份提供商功能，建立您所在企业与 DCE 的信任关系，通过联合认证使员工使用企业已有账号直接登录 DCE 平台，实现单点登录。

## 使用流程

有关访问控制的常规流程为：



## 用户

用户指的是由平台管理员 admin 或者用户与访问控制管理员 IAM Owner 在 **全局管理 ->**

**用户与访问控制 -> 用户** 页面创建的用户，或者通过 LDAP / OIDC 对接过来的用户。用户

名代表账号，用户通过用户名和密码登录 DaoCloud Enterprise 平台。

拥有一个用户账号是用户访问平台的前提。新建的用户默认没有任何权限，例如您需要给用

户赋予相应的角色权限，比如在 **用户列表** 或 **用户详情** 授予子模块的管理员权限。子模

块管理员拥有该子模块的最高权限，能够创建、管理、删除该模块的所有资源。如果用户

需要被授予具体资源的权限，比如某个资源的使用权限，请查看[资源授权说明](#)。

本页介绍用户的创建、授权、禁用、启用、删除等操作。

## 创建用户

前提：拥有平台管理员 Admin 权限或者用户与访问控制管理员 IAM Owner 权限。

1. 管理员进入 **用户与访问控制**，选择 **用户**，进入用户列表，点击右上方的 **创建用户**。

创建用户按钮

创建用户按钮

2. 在 **创建用户** 页面填写用户名和登录密码。如需一次性创建多个用户，可以点击 **创建用户** 后进行批量创建，一次性最多创建 5 个用户。根据您的实际情况确定是否设置用户在首次登录时重置密码。

设置用户名和密码

设置用户名和密码

3. 点击 **确定**，创建用户成功，返回用户列表页。

!!! note

此处设置的用户名和密码将用于登录平台。

## 为用户授予子模块管理员权限

前提：该用户已存在。

1. 管理员进入 **用户与访问控制**，选择 **用户**，进入用户列表，点击 **授权**。

授权菜单

授权菜单

2. 在 **授权** 页面勾选需要的角色权限（可多选）。

授权界面

授权界面

3. 点击 **确定** 完成为用户的授权。

**!!! note**

在用户列表中，点击某个用户，可以进入用户详情页面。

## 将用户加入用户组

1. 管理员进入 **用户与访问控制**，选择 **用户**，进入用户列表，点击 **！-> 加入用户组**。

加入用户组菜单

加入用户组菜单

2. 在 **加入用户组** 页面勾选需要加入的用户组（可多选）。若没有可选的用户组，点击 **创建用户组** 创建用户组，再返回该页面点击 **刷新** 按钮，显示刚创建的用户组。

加入用户组界面

加入用户组界面

3. 点击 **确定** 将用户加入用户组。

**!!! note**

用户会继承用户组的权限，可以在 **用户详情** 中查看该用户已加入的用户组。

## 启用/禁用用户

禁用用户后，该用户将无法再访问平台。与删除用户不同，禁用的用户可以根据需要再次启用，建议删除用户前先禁用，以确保没有关键服务在使用该用户创建的密钥。

1. 管理员进入 **用户与访问控制**，选择 **用户**，进入用户列表，点击一个用户名进入用户详情。

用户详情

用户详情

2. 点击右上方的 **编辑**，关闭状态按钮，使按钮置灰且处于未启用状态。

编辑

编辑

3. 点击 **确定** 完成禁用用户的操作。

## 忘记密码

前提：需要设置用户邮箱，有两种方式可以设置用户邮箱。

- 管理员在该用户详情页面，点击 **编辑**，在弹出框输入用户邮箱地址，点击 **确定** 完成邮箱设置。

编辑

编辑

- 用户还可以进入 **个人中心**，在 **安全设置** 页面设置邮箱地址。

个人中心

个人中心

如果用户登录时忘记密码，请参考[重置密码](#)。

## 删除用户

!!! warning

删除用户后，该用户将无法再通过任何方式访问平台资源，请谨慎删除。

在删除用户之前，请确保您的关键程序不再使用该用户创建的密钥。

如果您不确定，建议在删除前先禁用该用户。

如果您删除了一个用户，然后再创建一个同名的新用户，则新用户将被视为一个新的独立身份，它不会继承已删除用户的角色。

1. 管理员进入 **用户与访问控制**，选择 **用户**，进入用户列表，点击 **！-> 删除**。

删除用户菜单

删除用户菜单

2. 点击 **移除** 完成删除用户的操作。

删除用户确认

删除用户确认

# 用户组

用户组是用户的集合，用户可以通过加入用户组，继承用户组的角色权限。通过用户组批量地给用户进行授权，可以更好地管理用户及其权限。

## 适用场景

当用户权限发生变化时，只需将其移到相应的用户组下，不会对其他用户产生影响。

当用户组的权限发生变化时，只需修改用户组的角色权限，即可应用到组内的所有用户。

## 创建用户组

前提：拥有平台管理员 Admin 权限或者用户与访问控制管理员 IAM Owner 权限。

1. 管理员进入 **用户与访问控制**，选择 **用户组**，进入用户组列表，点击右上方的 **创建用户组**。

创建用户组

创建用户组

2. 在 **创建用户组** 页面填写用户组信息。

创建用户组

创建用户组

3. 点击 **确定**，创建用户组成功，返回用户组列表页面。列表中的第一行是新创建的用户组。

## 为用户组授权

前提：该用户组已存在。

1. 管理员进入 **用户与访问控制**，选择 **用户组**，进入用户组列表，点击 ... -> **授权**。

创建用户组按钮

创建用户组按钮

2. 在 **授权** 页面勾选需要的角色权限（可多选）。

创建用户组按钮

创建用户组按钮

3. 点击 **确定** 完成为用户组的授权。自动返回用户组列表，点击某个用户组，可以查看用户组被授予的权限。

创建用户组按钮

创建用户组按钮

## 给用户组添加用户

1. 管理员进入 **用户与访问控制**，选择 **用户组** 进入用户组列表，在某个用户组右侧，点击 ... -> **添加用户**。

添加用户

添加用户

2. 在 **添加用户** 页面点选需要添加的用户（可多选）。若没有可选的用户，点击 **前往创建新用户**，先前往创建用户，再返回该页面点击 **刷新** 按钮，显示刚创建的用户。

选择用户

选择用户

3. 点击 **确定** 完成给用户组添加用户。

!!! note

用户组中的用户会继承用户组的权限；可以在用户组详情中查看加入该组的用户。

## 删除用户组

说明：删除用户组，不会删除组内的用户，但组内用户将无法再继承该组的权限

1. 管理员进入 **用户与访问控制**，选择 **用户组** 进入用户组列表，在某个用户组右侧，点击 ... -> **删除**。

删除按钮

删除按钮

2. 点击 **移除** 删除用户组。

确认删除

确认删除

3. 返回用户组列表，屏幕上方将提示删除成功。

删除提示

删除提示

!!! note

说明：删除用户组，不会删除组内的用户，但组内用户将无法再继承该组的权限。

## 角色和权限管理

一个角色对应一组权限。权限决定了可以对资源执行的操作。向用户授予某角色，即授予该角色所包含的所有权限。

DCE 5.0 平台存在三种角色范围，能够灵活、有效地解决您在权限上的使用问题：

- [平台角色](#)

- [工作空间角色](#)
- [文件夹角色](#)

## 平台角色

平台角色是粗粒度权限，对平台上所有相关资源具有相应权限。通过平台角色可以赋予用户对所有集群、所有工作空间等的增删改查权限，而不能具体到某一个集群或某一个工作空间。DCE 5.0 提供了 5 个预置的、用户可直接使用的平台角色：

- Admin
- Kpanda Owner
- Workspace and Folder Owner
- IAM Owner
- Audit Owner

### 5 个预置平台角色

#### 5 个预置平台角色

同时，DCE 5.0 还支持用户创建自定义平台角色，可根据需要自定义角色内容。如创建一个平台角色，包含应用工作台的所有功能权限，由于应用工作台依赖于工作空间，因此平台会帮助用户默认勾选工作空间的查看权限，请不要手动取消勾选。若用户 A 被授予该 Workbench（应用工作台）角色，将自动拥有所有工作空间下的应用工作台相关功能的增删改查等权限。

#### 权限列表

#### 权限列表

## 平台角色授权方式

给平台角色授权共有三种方式：

- 在 **全局管理 -> 用户与访问控制 -> 用户** 的用户列表中，找到该用户，点击 ...，选

选择 **授权**，为该用户赋予平台角色权限。

点击授权

点击授权

- 在 **全局管理 -> 用户与访问控制 -> 用户组** 的用户组列表中创建用户组，将该用户加入用户组，并给用户组授权（具体操作为：在用户组列表找到该用户组，点击 ...，选择 **授权**，为该用户组赋予平台角色）。

点击授权

点击授权

- 在 **全局管理 -> 用户与访问控制 -> 角色** 的角色列表中，找到相应的平台角色，点击角色名称进入详情，点击 **关联成员** 按钮，选中该用户或用户所在的用户组，点击 **确定**。

关联成员按钮

关联成员按钮

## 工作空间角色

工作空间角色是细粒度角色，通过工作空间角色可以赋予用户某个工作空间的管理权限、查看权限或该工作空间应用工作台相关的权限等。获得该角色权限的用户只能管理该工作空间，而无法访问其他工作空间。DCE 5.0 提供了 3 个预置的、用户可直接使用的工作空间角色：

- Workspace Admin
- Workspace Editor
- Workspace Viewer

3 种工作空间预置角色

3 种工作空间预置角色

同时，DCE 5.0 还支持用户创建自定义工作空间角色，可根据需要自定义角色内容。如创建工作空间角色，包含应用工作台的所有功能权限，由于应用工作台依赖于工作空间，因此平台会帮助用户默认勾选工作空间的查看权限，请不要手动取消勾选。若用户 A 在工作空间 01 中被授予该角色，将拥有工作空间 01 下的应用工作台相关功能的增删改查权限。

!!! note

与平台角色不同，工作空间角色被创建后需要前往工作空间使用，被授权后用户仅在该工作空间下拥有该角色中的功能权限。

## 工作空间角色授权方式

在 **全局管理 -> 工作空间与层级** 列表中，找到该工作空间，点击 **添加授权**，为该用户赋予工作空间角色权限。

添加授权按钮

添加授权按钮

填写和选择

填写和选择

## 文件夹角色

文件夹角色的权限粒度介于平台角色与工作空间角色之间，通过文件夹角色可以赋予用户某个文件夹及其子文件夹和该文件夹下所有工作空间的管理权限、查看权限等，常适用于企业中的部门场景。比如用户 B 是一级部门的 Leader，通常用户 B 能够管理该一级部门、其下的所有二级部门和部门中的项目等，在此场景中给用户 B 授予一级文件夹的管理员权限，用户 B 也将拥有其下的二级文件夹和工作空间的相应权限。DCE 5.0 提供了 3 个预置的、用户可直接使用文件夹角色：

- Folder Admin
- Folder Editor
- Folder Viewer

### 3 种预置文件夹角色

#### 3 种预置文件夹角色

同时，DCE 5.0 还支持用户创建自定义文件夹角色，可根据需要自定义角色内容。如创建一个文件夹角色，包含应用工作台的所有功能权限。若用户 A 在文件夹 01 中被授予该角色，将拥有该文件夹下所有工作空间中应用工作台相关功能的增删改查权限。

#### !!! note

功能模块本身依赖的是工作空间，文件夹是工作空间上的进一步分组机制且具有权限继承能力，因此文件夹权限不光包含文件夹本身，还包括其下的子文件夹和工作空间。

## 文件夹角色授权方式

在 **全局管理 -> 工作空间与层级** 列表中，找到该文件夹，点击 **添加授权**，为该用户赋予文件夹角色权限。

添加授权按钮

添加授权按钮

填写和选择

填写和选择

## 系统角色

## 适用场景

DCE 5.0 提供了预置的系统角色，帮助用户简化角色权限的使用步骤。

#### !!! note

DCE 5.0 提供了三种类型的系统角色，分别为平台角色、工作空间角色和文件夹角色。

- 平台角色：对平台上所有相关资源具有相应权限，请前往用户/用户组授权。
- 工作空间角色：对某个工作空间具有相应权限，请前往具体工作空间授权。
- 文件夹角色：对某个文件夹、子文件夹及其工作空间下的资源具有相应权限，请前往具体文件夹授权。

## 平台角色

在用户与访问控制中预定义了 5 个系统角色，分别是：Admin、IAM Owner、Audit Owner、Kpanda Owner 和 Workspace and Folder Owner。这 5 个角色由系统创建，用户只能使用不能修改。角色对应的权限如下：

角色名称 角 所 角色权限

色 属

类 模

型 块

Admin 系 全 平台管理员，管理所有平台资源，代表平台的最高权限

统 部

角

色

IAM 系 用 用户与访问控制的管理员，拥有该服务下的所有权限，如管理用户/  
Owner

统 户 用户组及授权

角 与

色 访

问

控

制

角色名称 角 所 角色权限

色 属

类 模

型 块

Audit Owner 系 审 审计日志的管理员，拥有该服务下的所有权限，如设置审计日志策略，

统 计 导出审计日志

角 日

色 志

Kpanda Owner 系 容 容器管理的管理员，拥有该服务下的所有权限，如创建/接入集群，部

统 器 署应用，给用户/用户组授予集群/命名空间相关的权限

角 管

色 理

Workspac e and Folder Owner 系 工 工作空间与层级管理员，拥有该服务下的所有权限，如创建文件夹/

统 作 工作空间，给用户/用户组授权文件夹/工作空间的相关权限，在工作

角 空 空间下使用应用工作台、微服务引擎等功能

色 间

与

层

级

## 工作空间角色

在用户与访问控制中预定义了 3 个系统角色，分别是：Workspace Admin、Workspace Editor、

Workspace Viewer。这 3 个角色由系统创建，用户只能使用不能修改。角色对应的权限如下：

| 角色名称             | 角色类型 | 所属模块 | 角色权限      |
|------------------|------|------|-----------|
| Workspace Admin  | 系统角色 | 工作空间 | 工作空间的管理权限 |
| Workspace Editor | 系统角色 | 工作空间 | 工作空间的编辑权限 |
| Workspace Viewer | 系统角色 | 工作空间 | 工作空间的只读权限 |

## 文件夹角色

在用户与访问控制中预定义了 3 个系统角色，分别是：Folder Admin、Folder Editor、Folder Viewer。这 3 个角色由系统创建，用户只能使用不能修改。角色对应的权限如下：

| 角色名称          | 角色类<br>型 | 所属模<br>块 | 角色权限                 |
|---------------|----------|----------|----------------------|
| Folder Admin  | 系统角<br>色 | 工作空<br>间 | 文件夹及其下子文件夹、工作空间的管理权限 |
| Folder Editor | 系统角<br>色 | 工作空<br>间 | 文件夹及其下子文件夹、工作空间的编辑权限 |
| Folder Viewer | 系统角<br>色 | 工作空<br>间 | 文件夹及其下子文件夹、工作空间的只读权限 |

## 自定义角色

DCE 5.0 支持创建三种范围的自定义角色：

- **平台角色** 的权限对平台所有相关资源生效

- **工作空间角色** 的权限对该用户所在的工作空间下的资源生效
- **文件夹角色** 的权限对该用户所在的文件夹及其下的子文件夹和工作空间资源生效

## 创建平台角色

平台角色是粗粒度角色，能够对所选权限内的所有资源生效。如授权后用户可以拥有所有工作空间的查看权限、所有集群的编辑权限等，而不能针对某个工作空间或某个集群生效。平台角色创建完成后可以在用户/用户组列表中进行授权。

1. 从左侧导航栏点击 **全局管理** -> **用户与访问控制** -> **角色**，点击 **创建自定义角色**。

创建自定义角色

创建自定义角色

2. 输入名称、描述，选择 **平台角色**，勾选角色权限后点击 **确定**。

平台角色

平台角色

3. 返回角色列表，搜索刚创建的自定义角色，点击右侧的 **更多操作**，可以执行复制、编辑和删除等操作。

更多操作

更多操作

4. 平台角色创建成功后，可以去[用户/用户组](#)授权，为这个角色添加用户和用户组。

## 创建工作空间角色

工作空间角色是细粒度角色，针对某个工作空间生效。如在该角色中选择应用工作台的全部权限，给用户在某个工作空间下授予该角色后，该用户将仅能在该工作空间下使用应用工作

台相关的功能，而无法使用如微服务引擎、中间件等其他模块的能力。工作空间角色创建完成后，可以在工作空间与层级中选择工作空间后进行授权。

1. 从左侧导航栏点击 **全局管理** -> **用户与访问控制** -> **角色**，点击 **创建自定义角色**。

创建自定义角色

创建自定义角色

2. 输入名称、描述，选择 **工作空间角色**，勾选角色权限后点击 **确定**。

工作空间角色

工作空间角色

3. 返回角色列表，搜索刚创建的自定义角色，点击右侧的 **！**，可以执行复制、编辑和删除等操作。

更多操作

更多操作

4. 工作空间角色创建成功后，可以去[工作空间](#)授权，设定这个角色可以管理哪些工作空间。

## 创建文件夹角色

文件夹角色针对某个文件夹和该文件夹下的所有子文件夹及工作空间生效。如在该角色中选择全局管理-工作空间和应用工作台，给用户在某个文件夹下授予该角色后，该用户将能够在其下的所有工作空间中使用应用工作台的相关功能，而无法使用如微服务引擎、中间件等其他模块的能力。文件夹角色创建完成后，可以在工作空间与层级中选择文件夹后进行授权。

请注意：应用工作台、多云编排、镜像仓库、微服务引擎、服务网格和中间件均依赖于工作空间，因此在创建文件夹角色时大部分场景下需要用到工作空间，请注意在全局管理-工作空间下勾选。

1. 从左侧导航栏点击 **全局管理** -> **用户与访问控制** -> **角色**，点击 **创建自定义角色**。

创建自定义角色

创建自定义角色

2. 输入名称、描述，选择 **文件夹角色**，勾选角色权限后点击 **确定**。

文件夹角色

文件夹角色

3. 返回角色列表，搜索刚创建的自定义角色，点击右侧的 **！**，可以执行复制、编辑和删

除等操作。

更多操作

更多操作

4. 文件夹角色创建成功后，可以去[文件夹](#)授权，设定这个角色可以管理哪些文件夹。

## 身份提供商

全局管理支持基于 LDAP 和 OIDC 协议的单点登录，如果您的企业或组织已有自己的账号体系，同时希望管理组织内的成员使用 DCE 5.0 资源，您可以使用全局管理提供的身份提供商功能，而不必在您的 DCE 5.0 中为每一位组织成员创建用户名/密码。您可以向这些外部用户身份授予使用 DCE 5.0 资源的权限。

## 基本概念

- 身份提供商 (Identity Provider，简称 IdP)

负责收集和存储用户身份信息、用户名、密码等，在用户登录时负责认证用户的服务。

在企业与 DCE 5.0 进行身份认证的过程中，身份提供商指企业自身的身份提供商。

- 服务提供商（Service Provider，简称 SP）

服务提供商通过与身份提供商 IdP 建立信任关系，使用 IdP 提供的用户信息，为用户提供具体的服务。在企业与 DCE 5.0 进行身份认证的过程中，服务提供商指 DCE 5.0。

- LDAP

LDAP 指轻型目录访问协议（Lightweight Directory Access Protocol），常用于单点登录，即用户可以在多个服务中使用一个账号密码进行登录。全局管理支持 LDAP 进行身份认证，因此与 DCE 5.0 通过 LDAP 协议建立身份认证的企业 IdP 必须支持 LDAP 协议。关于 LDAP 的详细描述请参见：[欢迎使用 LDAP](#)。

- OIDC

OIDC 是 OpenID Connect 的简称，是一个基于 OAuth 2.0 协议的身份认证标准协议。全局管理支持使用 OIDC 协议进行身份认证，因此与 DCE 5.0 通过 OIDC 协议建立身份认证的企业 IdP 必须支持 OIDC 协议。关于 OIDC 的详细描述请参见：[欢迎使用 OpenID Connect](#)。

- OAuth 2.0

OAuth 2.0 是 Open Authorization 2.0 的简称，是一种开放授权协议，授权框架支持第三方应用程序以自己的名义获取访问权限。

## 功能特性

- 管理员无需重新创建 DCE 5.0 用户

使用身份提供商进行身份认证前，管理员需要在企业管理系统和 DCE 5.0 中分别为用户创建账号；使用身份提供商进行身份认证后，企业管理员只需要在企业管理系统中为用户创建账号，用户即可同时访问两个系统，降低了人员管理成本。

- 用户无需记住两套平台账号

使用身份提供商进行身份认证前，用户访问企业管理系统和 DCE 5.0 需要使用两个系

系统的账号登录；使用身份提供商进行身份认证后，用户在本企业管理系统中登录即可访问两个系统。

## LDAP

LDAP 英文全称为 Lightweight Directory Access Protocol，即轻型目录访问协议，这是一个开放的、中立的工业标准应用协议，通过 IP 协议提供访问控制和维护分布式信息的目录信息。

如果您的企业或组织已有自己的账号体系，同时您的企业用户管理系统支持 LDAP 协议，就可以使用全局管理提供的基于 LDAP 协议的身份提供商功能，而不必在 DCE 5.0 中为每一位组织成员创建用户名/密码。您可以向这些外部用户身份授予使用 DCE 5.0 资源的权限。在全局管理中，其操作步骤如下：

1. 使用具有 **admin** 角色的用户登录 DCE 5.0。点击左侧导航栏左下角的 **全局管理 -> 用户与访问控制**。

global  
global

2. 在左侧导航栏点击 **身份提供商**，点击 **创建身份提供商** 按钮。

The screenshot shows the 'Identity Provider' creation screen in the Global Management section of the DaoCloud Enterprise 5.0 web interface. The left sidebar has '全局管理' selected under '用户与访问控制'. The main area shows a table of existing identity providers:

| 身份提供商名称      | 状态  | 类型   |
|--------------|-----|------|
| test1        | 未启用 | LDAP |
| test-ldap-06 | 未启用 | LDAP |
| AD-test      | 启用  | AD   |
| ldap03       | 未启用 | LDAP |
| test1        | 未启用 | LDAP |
| test-ad02    | 未启用 | AD   |
| test1        | 未启用 | LDAP |
| henry-test   | 未启用 | LDAP |

A red box highlights the '创建身份提供商' button at the top right of the table.

## 身份提供商

3. 在 **LDAP** 页签中，填写以下字段后点击 **保存**，建立与身份提供商的信任关系及用户的映射关系。

ldap

| 字段          | 描述                                                                         |
|-------------|----------------------------------------------------------------------------|
| 类型 (Vendor) | 支持 LDAP (Lightweight Directory Access Protocol)<br>和 AD (Active Directory) |

| 字段                             | 描述                                                                                                                                                        |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 身份提供商名称 (UI display name)      | 用于区分不同的身份提供商                                                                                                                                              |
| 服务器 (Connection URL)           | LDAP 服务的地址和端口号, 如<br>ldap://10.6.165.2:30061                                                                                                              |
| 用户名 (Bind DN)                  | LDAP 管理员的 DN, Keycloak 将使用该 DN 来访问 LDAP 服务器                                                                                                               |
| 密码 (Bind credentials)          | LDAP 管理员的密码。该字段可以从 vault 中获取其值, 使用 \${vault.ID} 格式。                                                                                                       |
| 用户 DN (Users DN)               | 您的用户所在的 LDAP 树的完整 DN。此 DN 是 LDAP 用户的父级。例如, 假设您的典型用户的 DN 类似于“uid='john',ou=users,dc=example,dc=com”, 则可以是“ou=users,dc=example,dc=com”。                     |
| 用户对象类 (User object classes)    | LDAP 中用户的 LDAP objectClass 属性的所有值, 以逗号分隔。例如: “inetOrgPerson, organizationalPerson”。新创建的 Keycloak 用户将与所有这些对象类一起写入 LDAP, 并且只要现有 LDAP 用户记录包含所有这些对象类, 就会找到它们。 |
| 是否启用 TLS (Enable StartTLS)     | 启用后将加密 DCE 5.0 与 LDAP 的连接                                                                                                                                 |
| 预设权限 (Default permission)      | 同步后的用户/用户组默认没有任何权限                                                                                                                                        |
| 全名映射 (First/Last name mapping) | 对应 First name 和 Last Name                                                                                                                                 |
| 用户名映射 (User name mapping)      | 用户唯一的用户名                                                                                                                                                  |

| 字段                              | 描述                                                                                                                                                     |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 邮箱映射 (Mailbox mapping)          | 用户的邮箱                                                                                                                                                  |
| <b>高级配置</b>                     |                                                                                                                                                        |
| 字段                              | 描述                                                                                                                                                     |
| 是否启用 (Enable or not)            | 默认启用, 关闭后该 LDAP 配置不生效                                                                                                                                  |
| 自动同步用户 (Periodic full sync)     | 默认不启用, 启用后可配置同步周期, 如每小时同步一次                                                                                                                            |
| 数据同步模式 (Edit mode)              | 只读模式不会修改 LDAP 的源数据; 写入模式在平台编辑用户信息后, 数据将同步回 LDAP                                                                                                        |
| 读取超时 (Read timeout)             | 当 LDAP 数据量较大时, 调整该数值可以有效避免接口超时                                                                                                                         |
| 用户对象过滤器 (User LDAP filter)      | 用于过滤搜索用户的附加 LDAP 过滤器。如果您不需要额外的过滤器, 请将其留空。确保它以“(”开头, 并以“)”结尾。                                                                                           |
| 用户名属性 (Username LDAP attribute) | LDAP 属性的名称, 映射为 Keycloak 用户名。对于许多 LDAP 服务器供应商来说, 它可以是“uid”。对于 Active Directory, 它可以是“sAMAccountName”或“cn”。应为您想要从 LDAP 导入到 Keycloak 的所有 LDAP 用户记录填写该属性。 |
| RDN 属性 (RDN LDAP attribute)     | LDAP 属性名称, 作为典型用户 DN 的 RDN(顶级)                                                                                                                         |

| 字段                            | 描述                                                                                                                                                                                                     |
|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UUID 属性 (UUID LDAP attribute) | 级属性)。通常它与用户名 LDAP 属性相同,但这不是必需的。例如,对于 Active Directory, 当用户名属性可能是“sAMAccountName”时, 通常使用“cn”作为 RDN 属性。                                                                                                  |
|                               | LDAP 属性的名称,用作 LDAP 中对象的唯一对象标识符 (UUID)。对于许多 LDAP 服务器供应商来说, 它是“entryUUID”; 然而有些是不同的。例如,对于 Active Directory, 它应该是“objectGUID”。如果您的 LDAP 服务器不支持 UUID 概念, 您可以使用在树中的 LDAP 用户之间应该唯一的任何其他属性。例如“uid”或“entryDN”。 |

4. 在 **同步用户组** 页签中, 填写以下字段配置用户组的映射关系后, 再次点击 **保存**。

| 身份提供商 | 身份提供商 | 字段    | 描述                                                            | 举例值                          |
|-------|-------|-------|---------------------------------------------------------------|------------------------------|
|       |       | 基准 DN | 用户组在 LDAP 树状结构中的位置                                            | ou=groups,dc=example,dc=org* |
|       |       | 用户组对象 | 用户组的对象类, 如果需要更多类, 则用逗号分隔。                                     |                              |
|       |       | 过滤器   | 在典型的 LDAP 部署中, 通常是 “groupOfNames”, 系统已自动填入, 如需更改请直接编辑。* 表示所有。 |                              |

| 字段   | 描述 | 举例值  |
|------|----|------|
| 用户组名 | cn | 不可更改 |

**!!! note**

- 当您通过 LDAP 协议将企业用户管理系统与 DCE 5.0 建立信任关系后，可通过手动同步或自动同步的方式，将企业用户管理系统中的用户或用户组一次性同步至 DCE 5.0。
- 同步后管理员可对用户组/用户组进行批量授权，同时用户可通过在企业用户管理系统中的账号/密码登录 DCE 5.0。
- 有关实际操作教程，请参阅 [LDAP 操作演示视频](../../../../videos/ghippo.md#ldap)。

## 创建和管理 OIDC

OIDC (OpenID Connect) 是建立在 OAuth 2.0 基础上的一个身份层，是基于 OAuth2 协议的身份认证标准协议。

如果您的企业或组织已有自己的账号体系，同时您的企业用户管理系统支持 OIDC 协议，可以使用全局管理提供的基于 OIDC 协议的身份提供商功能，而不必在 DCE 5.0 中为每一位组织成员创建用户名/密码。您可以向这些外部用户身份授予使用 DCE 5.0 资源的权限。

具体操作步骤如下。

- 使用具有 `admin` 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理 -> 用户与访问控制**。



## 全局管理

2. 在左侧导航栏选择 **身份提供商**，点击 **OIDC** 页签 -> **创建身份提供商** 按钮。

The screenshot shows the '全局管理' (Global Management) section of the DaoCloud interface. On the left, there's a sidebar with various navigation items like '全局管理', '用户与访问控制', '身份提供商' (which is highlighted in blue), etc. The main area is titled '身份提供商' (Identity Provider). It has three tabs at the top: 'LDAP', 'OIDC' (which is selected and highlighted in blue), and 'OAuth 2.0'. Below the tabs is a table listing several identity providers:

| 提供商名称             | 状态  | 操作 |
|-------------------|-----|----|
| azure             | 启用  | ⋮  |
| 10-6-8-1          | 启用  | ⋮  |
| sadsaasd          | 启用  | ⋮  |
| demo-alpha        | 启用  | ⋮  |
| 测试                | 启用  | ⋮  |
| hyt-kangaroo-test | 启用  | ⋮  |
| test-oidc01       | 启用  | ⋮  |
| test-oidc-04      | 启用  | ⋮  |
| demo-test1        | 未启用 | ⋮  |

A blue button labeled '创建身份提供商' (Create Identity Provider) is located in the top right corner of the table area.

点击创建按钮

3. 填写表单字段，建立与身份提供商的信任关系后，点击 **确定**。

The screenshot shows the '创建身份提供商' (Create Identity Provider) dialog box. The left sidebar is identical to the previous screenshot, showing '全局管理' selected. The main form has the following fields:

- 提供商名称**:  (必填)
- 提供商 ID**:  (必填)
- 认证方式**:  (必填)
- 客户端ID**:  (必填)
- 客户端密钥**:  (必填)
- 客户端URL**:  (One-click generate)

Below the form, there's a summary of URLs generated by the provider:

- 登录URL**:  (用于单点登录的 URL)
- 获取 Token URL**:  (用于获取 Token 的 URL)
- 获取用户信息 URL**:  (用于获取身份提供商用户信息的 URL)
- 登出 URL**:  (用于单点登出的 URL)

At the bottom right of the dialog are two buttons: '取消' (Cancel) and a blue '确定' (Confirm) button.

填写表单

| 字段      | 描述                                                                                                             |
|---------|----------------------------------------------------------------------------------------------------------------|
| 提供商名称   | 显示在登录页上，是身份提供商的入口                                                                                              |
| 认证方式    | 客户端身份验证方法。如果 JWT 使用私钥签名，请下拉选择 <b>JWT signed with private key</b> 。具体参阅 <a href="#">Client Authentication</a> 。 |
| 客户端 ID  | 客户端的 ID                                                                                                        |
| 客户端密钥   | 客户端密码                                                                                                          |
| 客户端 URL | 可通过身份提供商 well-known 接口一键获取登录 URL、Token URL、用户信息 URL 和登出 URL                                                    |
| 自动关联    | 开启后当身份提供商用户名/邮箱与 DCE 5.0 用户名/邮箱重复时将自动使二者关联                                                                     |

**!!! note**

- 当用户通过企业用户管理系统完成第一次登录 DCE 5.0 后，用户信息才会被同步至 DCE 5.0 的 用户与访问控制 -> 用户列表。
- 初次登录的用户不会被赋予任何默认权限，需要有管理员给其赋权（管理员可以是平台管理员、子模块管理员或资源管理员）。
- 有关实际操作教程，请参阅 [OIDC 操作演示视频](../../../../videos/ghippo.md#oidc)，也可参考 [Azure OpenID Connect (OIDC) 接入流程](https://learn.microsoft.com/zh-cn/azure/active-directory/develop/v2-protocols-oidc)。

## 用户身份认证交互流程

用户身份认证的交互流程为：

oidc  
oidc

- 使用浏览器发起单点登录 DCE 5.0 的请求。
- DCE 5.0 根据登录链接中携带的信息，查找 **全局管理** -> **用户与访问控制** -> **身份提供**

商 中对应的配置信息， 构建 OIDC 授权 Request，发送给浏览器。

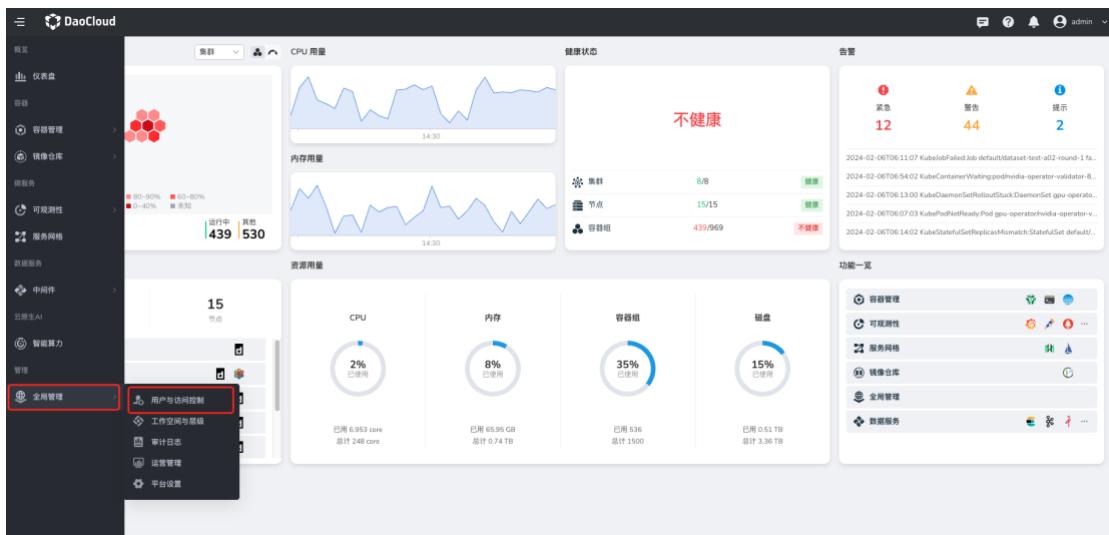
3. 浏览器收到请求后，转发 OIDC 授权 Request 给企业 IdP。
4. 在企业 IdP 的登录页面中输入用户名和密码，企业 IdP 对提供的身份信息进行验证，并构建携带用户信息的 ID Token，向浏览器发送 OIDC 授权 Response。
5. 浏览器响应后转发 OIDC 授权 Response 给 DCE 5.0。
6. DCE 5.0 从 OIDC 授权 Response 中取出 ID Token，并根据已配置的身份转换规则映射到具体的用户列表，颁发 Token。
7. 完成单点登录，访问 DCE 5.0。

## OAuth 2.0 - 企业微信

如果您的企业或组织中的成员均管理在企业微信中，您可以使用全局管理提供的基于 OAuth 2.0 协议的身份提供商功能，而不必在 DCE 5.0 中为每一位组织成员创建用户名/密码。您可以向这些外部用户身份授予使用 DCE 5.0 资源的权限。

### 操作步骤

1. 使用具有 Admin 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理 -> 用户与访问控制**。



## 用户与访问控制

2. 在左侧导航栏选择 **身份提供商**，点击 **OAuth2.0** 页签。填写表单字段，建立与企业微信的信任关系后，点击 **保存**。



## Oauth2.0

## 企业微信中对应的字段

### !!! note

对接前需要在企业微信管理后台中创建自建应用，参阅[如何创建自建应用链接]([https://open.work.weixin.qq.com/help2/pc/16892?person\\_id=1&searchData=](https://open.work.weixin.qq.com/help2/pc/16892?person_id=1&searchData=))。

| 字段    | 描述       |
|-------|----------|
| 企业 ID | 企业微信的 ID |

|              |              |
|--------------|--------------|
| Agent ID     | 自建应用的 ID     |
| ClientSecret | 自建应用的 Secret |

企业微信 ID:

The screenshot shows the 'My Company' section of the DaoCloud interface. It includes fields for company logo, name, address, contact, and industry. A prominent red box highlights the 'Enterprise ID' input field at the bottom.

## Oauth2.0

Agent ID 和 ClientSecret:

This screenshot shows the 'Application Management' section where 'AgentId' and 'Secret' are highlighted with red boxes. Other visible fields include 'Visible Scope' and application settings like 'Responsible Person' and 'Home Page'.

agent

# 接入管理

当两个或两个以上平台相互对接或嵌入时，通常需要进行用户体系打通。在用户打通过程中，**接入管理** 主要提供 SSO 接入能力，当您需要将 DCE 5.0 作为用户源接入客户系统时，您可以通过 **接入管理** 创建 SSO 接入来实现。

## 创建 SSO 接入

前提：拥有平台管理员 Admin 权限或者用户与访问控制管理员 IAM Owner 权限。

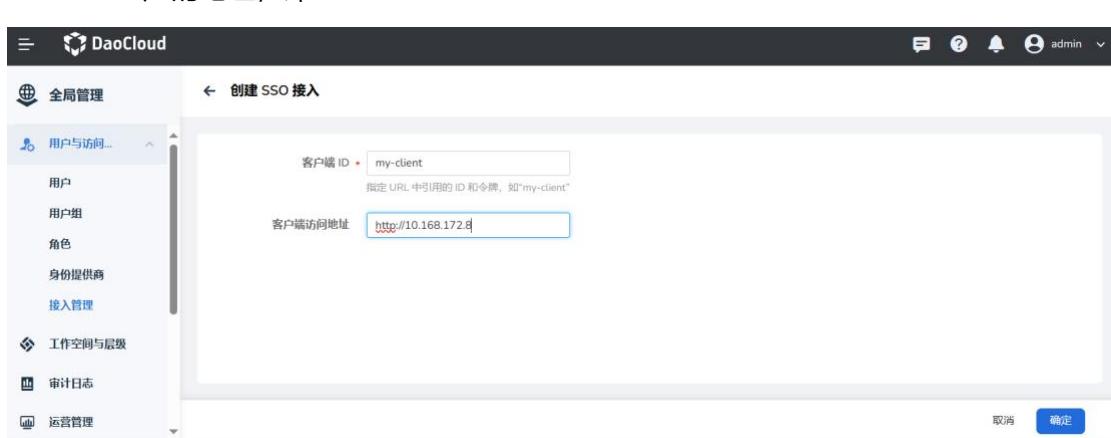
1. 管理员进入 **用户与访问控制**，选择 **接入管理**，进入接入管理列表，点击右上方的 **创建 SSO 接入**。

创建 SSO 接入按钮

创建 SSO 接入按钮

2. 在 **创建 SSO 接入** 页面填写客户端 ID。

- 客户端 ID：对应 client 名称
- 客户端访问地址：用户完成登录并通过身份验证后，认证服务器用来重定向用户的地址，即 Callback URL



## 创建 SSO 接入

3. 创建 SSO 接入成功后，在 **接入管理** 管理列表，点击刚创建的客户端 ID 进入详情，复制客户端 ID、密钥和单点登录 URL 信息，填写至客户系统完成用户体系打通。

!!! note

realm 名称为 ghippo。

[接入管理详情](#)

[接入管理详情](#)

# Webhook 消息通知

DCE 5.0 在接入客户的系统后，可以创建 Webhook，在用户创建/更新/删除/登录/登出之时发送消息通知。

Webhook 是一种用于实现实时事件通知的机制。它允许一个应用程序将数据或事件推送到另一个应用程序，而无需轮询或持续查询。通过配置 Webhook，您可以指定在某个事件发生时，由目标应用程序接收并处理通知。

Webhook 的工作原理如下：

1. 源应用程序（DCE 5.0）执行某个特定操作或事件。
2. 源应用程序将相关数据和信息打包成 HTTP 请求，并将其发送到目标应用程序指定的 URL（例如企业微信群机器人）。
3. 目标应用程序接收到请求后，根据其中的数据和信息进行相应的处理。

通过使用 Webhook，您可以实现以下功能：

- 实时通知：当某个特定事件发生时，通过 Webhook 及时通知其他应用程序。
- 自动化处理：目标应用程序可以根据收到的 Webhook 请求自动触发事先定义好的操作，无需手动干预。

- 数据同步：通过 Webhook 将数据从一个应用程序传递到另一个应用程序，实现数据的同步更新。

常见的应用场景包括：

- 版本控制系统（例如 GitHub、GitLab）中，当代码仓库发生变动时，自动触发构建和部署操作。
- 电子商务平台中，当订单状态发生变化时，向物流系统发送更新通知。
- 聊天机器人平台中，当接收到用户消息时，通过 Webhook 将消息推送到目标服务器进行处理。

## 配置步骤

DCE 5.0 图形化配置 Webhook 的操作步骤如下：

1. 在 **全局管理 -> 用户与访问控制 -> 接入管理**，创建一个客户端 ID。

oem in  
oem in

2. 点击某个客户端 ID，进入详情页，点击 **创建 Webhook** 按钮。

button  
button

3. 在弹窗中填入字段信息后点击 **确定**。

- 对象：目前仅支持 **用户** 对象
- 行为：用户创建/更新/删除/登录/登录时发送 Webhook 消息
- URL：接收消息的地址
- Method：视情况选择适用的方法，例如企业微信推荐使用 POST 方法
- 高级配置：可以用 Json 编写消息体。如果是企业微信群，请参阅[群机器人配置说明](#)

### [配置说明](#)

fill

```
fill
```

4. 屏幕提示 Webhook 创建成功。

```
success
```

```
success
```

5. 现在去试着创建一个用户。

```
create
```

```
create
```

6. 用户创建成功，可以看到企业微信群收到了一条消息。

```
message
```

```
message
```

## 高级配置示例

### 系统默认的消息体

DCE 5.0 预先定义了一些变量，您可以根据自己情况在消息体中使用这些变量。

```
{
 "id": "{$.ID$$}",
 "email": "{$.Email$$}",
 "username": "{$.Name$$}",
 "last_name": "{$.LastName$$}",
 "first_name": "{$.FirstName$$}",
 "created_at": "{$.CreatedAt$$}",
 "enabled": "{$.Enabled$$}"
}
```

### 企业微信群机器人的 Message Body

```
{
 "msgtype": "text",
 "text": {
 "content": "{$.Name$$} hello world"
 }
}
```

## 参考文档

- [OEM OUT 文档](#)

- [OEM IN 文档](#)

# 工作空间与层级

**工作空间与层级** 是一个具有层级的资源隔离和资源分组特性，主要解决资源统一授权、资源分组以及资源限额问题。

## 层级结构

### 层级结构

**工作空间与层级** 有两个概念：工作空间和文件夹。

## 工作空间

工作空间可通过 **授权** 、 **资源组** 和 **共享资源** 来管理资源，使用户（用户组）之间能够共享工作空间中的资源。

## 工作空间

### 工作空间

- 资源

资源处于资源管理模块层级结构的最低层级，资源包括 Cluster、Namespace、Pipeline、网关等。所有这些资源的父级只能是工作空间，而工作空间作为资源容器是一种资源分组单位。

- 工作空间

工作空间通常代指一个项目或环境，每个工作空间的资源相对于其他工作空间中的资源时逻辑隔离的。您可以通过工作空间中的授权，授予用户（用户组）同一组资源的不同访问权限。

从层次结构的底层算起，工作空间位于第一层，且包含资源。除共享资源外，所有资源有且仅有一个父项。所有工作空间也有且仅有一个父级文件夹。

资源通过工作空间进行分组，而工作空间中存在两种分组模式，分别是 **资源组** 和 **共享资源**。

- 资源组

一个资源只能加入一个资源组，资源组与工作空间一一对应。资源被加入到资源组后，Workspace Admin 将获得资源的管理权限，相当于该资源的所有者。

- 共享资源

而对于共享资源来说，多个工作空间可以共享同一个或者多个资源。资源的所有者，可以选择将自己拥有的资源共享给工作空间使用，一般共享时资源所有者会限制被共享工作空间能够使用的资源额度。资源被共享后，Workspace Admin 仅具有资源限额下的资源使用权，无法管理资源或者调整工作空间能够使用的资源量。

同时共享资源对于资源本身也具有一定的要求，只有 Cluster (集群) 资源可以被共享。

Cluster Admin 能够将 Cluster 资源分享给不同的工作空间使用，并且限制工作空间在此 Cluster 上的使用额度。

Workspace Admin 在资源限额内能够创建多个 Namespace，但是 Namespace 的资源额度总和不能超过 Cluster 在该工作空间的资源限额。对于 Kubernetes 资源，当前能够分享的资源类型仅有 Cluster。

## 文件夹

文件夹可用于构建企业业务层级关系。

- 文件夹是在工作空间基础之上的进一步分组机制，具有层级结构。一个文件夹可以包

含工作空间、其他文件夹或两者的组合，能够形成树状的组织关系。

- 借助文件夹您可以映射企业业务层级关系，按照部门对工作空间进行分组。 文件夹不直接与资源挂钩，而是通过工作空间间接实现资源分组。
- 文件夹有且仅有一个父级文件夹，而根文件夹是层次结构的最高层级。 根文件夹没有父级，文件夹和工作空间均挂靠到根文件夹下。

另外，用户（用户组）在文件夹中能够通过层级结构继承来自父项的权限。 用户在层次结构中的权限来自当前层级的权限以及继承其父项权限的组合结果，权限之间是加合关系不存在互斥。

## 创建/删除工作空间

工作空间是一种资源范畴，代表一种资源层级关系。 工作空间可以包含集群、命名空间、注册中心等资源。 通常一个工作空间对应一个项目，可以为每个工作空间分配不同的资源，指派不同的用户和用户组。

参照以下步骤创建一个工作空间。

1. 使用 admin/folder admin 角色的用户登录 DCE 5.0，点击左侧导航栏底部的 **全局管理** -> **工作空间与层级**。

全局管理

全局管理

2. 点击右上角的 **创建工作空间** 按钮。

创建工作空间

创建工作空间

3. 填写工作空间名称、所属文件夹等信息后，点击 **确定**，完成创建工作空间。

确定

确定

!!! tip

创建成功后工作空间名称将显示在左侧的树状结构中，以不同的图标表示文件夹和工作空间。

![文件夹与工作空间](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/ws04.png)

!!! note

选中某一个工作空间或文件夹，点击右侧的  可以进行编辑或删除。

- 当该工作空间下资源组、共享资源中存在资源时，该工作空间无法被删除，需要将所有资源解绑后再删除。
- 当微服务引擎模块在该工作空间下存在接入注册中心资源时，该工作空间无法被删除，需要将所有接入注册中心移除后再删除工作空间。
- 当镜像仓库模块在该工作空间下存在镜像空间或集成仓库时，该工作空间无法被删除，需要将镜像空间解绑，将仓库集成删除后再删除工作空间。

## 工作空间权限说明

工作空间具有权限映射和资源隔离能力，能够将用户/用户组在工作空间的权限映射到其下的资源上。若用户/用户组在工作空间是 `Workspace Admin` 角色，同时工作空间-资源组中绑定了资源 `Namespace`，则映射后该用户/用户组将成为 `Namespace Admin`。

!!! note

工作空间的权限映射能力不会作用到共享资源上，因为共享是将集群的使用权限共享给多个工作空间，而不是将管理权限受让给工作空间，因此不会实现权限继承和角色映射。

## 应用场景

通过将资源绑定到不同的工作空间能够实现资源隔离。因此借助权限映射、资源隔离和共享资源能力能够将资源灵活分配给各个工作空间（租户）。

通常适用于以下两个场景：

- 集群一对一

| 普通集群  | 部门/租户 (工作空间) | 用途    |
|-------|--------------|-------|
| 集群 01 | A            | 管理和使用 |
| 集群 02 | B            | 管理和使用 |

- 集群一对多

| 集群    | 部门/租户 (工作空间) | 资源限额      |
|-------|--------------|-----------|
| 集群 01 | A            | 100 核 CPU |
|       | B            | 50 核 CPU  |

## 权限说明

操作对象 | 操作 | Workspace Admin | Workspace Editor | Workspace Viewer |

:----|:-----|:-----|:-----|:-----|

本身 | 查看 | ✓ | ✓ | ✓ |

- | 授权 | ✓ | X | X |

- | 修改别名 | ✓ | ✓ | X |

资源组 | 查看 | ✓ | ✓ | ✓ |

- | 资源绑定 | ✓ | X | X |

- | 解除绑定 | ✓ | X | X |

共享资源 | 查看 | ✓ | ✓ | ✓ |

- | 新增共享 | ✓ | X | X |

- | 解除共享 | ✓ | X | X |

- | 资源限额 | ✓ | X | X |

- | 使用共享资源 1 | ✓ | X | X |

# 创建/删除文件夹

文件夹具有权限映射能力，能够将用户/用户组在本文件夹的权限映射到其下的子文件夹、工作空间以及资源上。

参照以下步骤创建一个文件夹。

1. 使用 admin/folder admin 角色的用户登录 DCE 5.0，点击左侧导航栏底部的 **全局管理** -> **工作空间与层级**。

全局管理

全局管理

2. 点击右上角的 **创建文件夹** 按钮。

创建文件夹

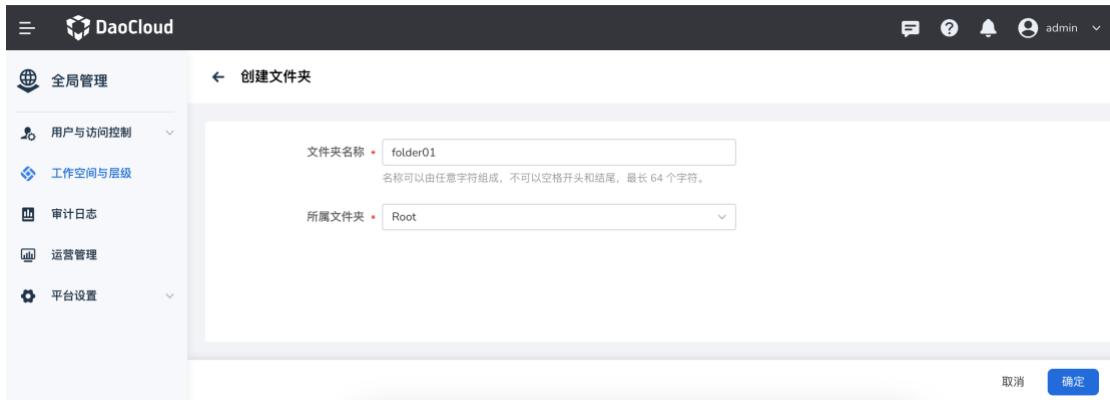
创建文件夹

---

1 授权用户可前往应用工作台、微服务引擎、中间件、多云编排、服务网格等模块使用工作空间中的资源。有关 Workspace Admin、Workspace Editor、Workspace Viewer 角色在各产品模块的操作范围，请查阅各模块的权限说明：

- [应用工作台权限说明](#)
- [服务网格权限说明](#)
- [中间件权限说明](#)
- [微服务引擎权限说明](#)
- [容器管理权限说明](#)

### 3. 填写文件夹名称，选择上一级文件夹后，点击 确定



**确定**

**!!! tip**

创建成功后文件夹名称将显示在左侧的树状结构中，以不同的图标表示工作空间和文件夹。

![工作空间和文件夹](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/ws04.png)

**!!! note**

选中某一个文件夹或文件夹，点击右侧的 **\_!\_** 可以进行编辑或删除。

- 当该文件夹下资源组、共享资源中存在资源时，该文件夹无法被删除，需要将所有资源解绑后再删除。

- 当微服务引擎模块在该文件夹下存在接入注册中心资源时，该文件夹无法被删除，需要将所有接入注册中心移除后再删除文件夹。

## 文件夹权限说明

文件夹具有权限映射能力，能够将用户/用户组在本文件夹的权限映射到其下的子文件夹、

工作空间以及资源上。

若用户/用户组在本文件夹是 **Folder Admin** 角色，映射到子文件夹仍为 **Folder Admin** 角色，

映射到其下的工作空间则为 **Workspace Admin**； 若在 **工作空间与层级 -> 资源组** 中绑定

了 **Namespace**，则映射后该用户/用户组同时还是 **Namespace Admin**。

**!!! note**

文件夹的权限映射能力不会作用到共享资源上，因为共享是将集群的使用权限共享给多个工作空间，而不是将管理权限受让给工作空间，因此不会实现权限继承和角色映射。

## 应用场景

文件夹具有层级能力，因此将文件夹对应于企业中的部门/供应商/项目等层级时，

- 若用户/用户组在一级部门具有管理权限（Admin），其下的二级、三级、四级部门或项目同样具有管理权限；
- 若用户/用户组在一级部门具有使用权限（Editor），其下的二级、三级、四级部门或项目同样具有使用权限；
- 若用户/用户组在一级部门具有只读权限（Viewer），其下的二级、三级、四级部门或项目同样具有只读权限。

| 对象       | 操作   | Folder Admin | Folder Editor | Folder Viewer |
|----------|------|--------------|---------------|---------------|
| 对文件夹本身   | 查看   | ✓            | ✓             | ✓             |
|          | 授权   | ✓            | ✗             | ✗             |
|          | 修改别名 | ✓            | ✗             | ✗             |
| 对子文件夹    |      |              |               |               |
| 对子文件夹    | 创建   | ✓            | ✗             | ✗             |
|          | 查看   | ✓            | ✓             | ✓             |
|          | 授权   | ✓            | ✗             | ✗             |
|          | 修改别名 | ✓            | ✗             | ✗             |
| 对其下的工作空间 |      |              |               |               |
| 对其下的工作空间 | 创建   | ✓            | ✗             | ✗             |
|          | 查看   | ✓            | ✓             | ✓             |
|          | 授权   | ✓            | ✗             | ✗             |

| 对象              | 操作  | Folder Admin | Folder Editor | Folder Viewer |
|-----------------|-----|--------------|---------------|---------------|
|                 |     | ✓            | ✗             | ✗             |
| 名               |     |              |               |               |
| 对其下的工作空间 - 资源组  | 查看  | ✓            | ✓             | ✓             |
|                 | 资源绑 | ✓            | ✗             | ✗             |
| 定               |     |              |               |               |
| 对其下的工作空间 - 共享资源 | 解除绑 | ✓            | ✗             | ✗             |
|                 | 新增共 | ✓            | ✗             | ✗             |
| 享               |     |              |               |               |
| 共享              | 解除共 | ✓            | ✗             | ✗             |
|                 | 资源限 | ✓            | ✗             | ✗             |
| 额               |     |              |               |               |

## 资源配置 (Quota)

共享资源并非意味着被共享者可以无限制地使用被共享的资源。 Admin、Kpanda Owner 和 Workspace Admin 可以通过共享资源中的 **资源配置** 功能限制某个用户的最大使用额度。若不限制，则表示可以无限制使用。

- CPU 请求 (Core)
- CPU 限制 (Core)

- 内存请求 (MB)
- 内存限制 (MB)
- 存储请求总量 (GB)
- 存储卷声明 (个)
- GPU 类型、规格、数量 (包括但不限于 Nvidia、Ascend、Iluvatar 等 GPU 卡类型)

一个资源 (集群) 可以被多个 workspace 共享，一个 workspace 也可以同时使用多个共享集群中的资源。

## 资源组和共享资源

共享资源和资源组中的集群资源均来自[容器管理](#)，但是集群绑定和共享给同一个 workspace 将会产生两种截然不同的效果。

### 1. 绑定资源

使 workspace 中的用户/用户组具有该集群的全部管理和使用权限，Workspace Admin 将被映射为 Cluster Admin。 Workspace Admin 能够进入[容器管理模块](#)管理该集群。

#### 资源组

##### 资源组

###### !!! note

当前容器管理模块暂无 Cluster Editor 和 Cluster Viewer 角色，因此 Workspace Editor、Workspace Viewer 还无法映射。

### 2. 新增共享资源

使 workspace 中的用户/用户组具有该集群资源的使用权限，这些资源可以在[创建命名空间 \(Namespace\)](#)时使用。

#### 共享资源

##### 共享资源

与资源组不同，将集群共享到工作空间时，用户在工作空间的角色不会映射到资源上，

因此 Workspace Admin 不会被映射为 Cluster admin。

本节展示 3 个与资源配置有关的场景。

## 创建命名空间

创建命名空间时会涉及到资源配置。

1. 在工作空间 ws01 新增一个共享集群。

新增共享集群

新增共享集群

2. 在应用工作台选择工作空间 ws01 和共享集群，创建命名空间 ns01。

创建命名空间

创建命名空间

- 若在共享集群中未设置资源配置，则创建命名空间时可不设置资源配置。
- 若在共享集群中已设置资源配置（例如 CPU 请求 = 100 core），则创建命名空间时 CPU 请求 ≤ 100 core。

## 命名空间绑定到工作空间

前提：工作空间 ws01 已新增共享集群，操作者为 Workspace Admin + Kpanda Owner 或 Admin 角色。

以下两种绑定方式的效果相同。

- 在容器管理中将创建的命名空间 ns01 绑定到 ws01

绑定到工作空间

### 绑定到工作空间

- 若在共享集群未设置资源配额，则命名空间 ns01 无论是否已设置资源配额，均可成功绑定。
  - 若在共享集群已设置资源配额 **CPU 请求 = 100 core**，则命名空间 ns01 必须满足 **CPU 请求 ≤ 100 core** 才能绑定成功。
- 在全局管理中，将命名空间 ns01 绑定到 ws01

### 绑定到工作空间

#### 绑定到工作空间

- 若在共享集群未设置资源配额，则命名空间 ns01 无论是否已设置资源配额，均可成功绑定。
- 若在共享集群已设置资源配额 **CPU 请求 = 100 core**，则命名空间 ns01 必须满足 **CPU 请求 ≤ 100 core** 才能绑定成功。

## 从工作空间解绑命名空间

以下两种解绑方式的效果相同。

- 在容器管理中将命名空间 ns01 从工作空间 ws01 解绑
- ### 绑定到工作空间
- #### 绑定到工作空间
- 若在共享集群中未设置资源配额，则命名空间 ns01 无论是否已设置资源配额，解绑后均不会对资源配额产生影响。
  - 若在共享集群已设置资源配额 **CPU 请求 = 100 core**，命名空间 ns01 也设置了资源配额，则解绑后将释放相应的资源额度。
- 上海道客网络科技有限公司
- 9

- 在全局管理中将命名空间 ns01 从工作空间 ws01 解绑

绑定到工作空间

绑定到工作空间

- 若在共享集群未设置资源配额，则命名空间 ns01 无论是否已设置资源配额，解绑后均不会对资源配额产生影响。
- 若在共享集群已设置资源配额 **CPU 请求 = 100 core**，命名空间 ns01 也设置了资源配额，则解绑后将释放相应的资源额度。

## 资源组与共享资源的区别

资源组与共享资源均支持绑定集群，但使用上存在很大区别。

### 使用场景区别

- 资源组绑定集群：资源组绑定集群通常被用来批量授权。资源组绑定集群后，工作空间管理员将被映射为集群管理员，能够管理并使用集群资源。
- 共享资源绑定集群：资源共享绑定集群通常被用来做资源限额。典型的场景是平台管理员将集群分配给一级供应商后，再由一级供应商分配给二级供应商并对二级供应商进行资源限额。

diff

diff

说明：在该场景中，需要平台管理员对二级供应商进行资源限制，暂时还不支持一级供应商限制二级供应商的集群额度。

## 集群额度的使用区别

- 资源组绑定集群：工作空间的管理员将被映射为该集群的管理员，相当于在容器管理-权限管理中被授予 Cluster Admin 角色，能够无限制支配该集群资源，管理节点等重要内容，且资源组不能够被资源限额。
- 共享资源绑定资源：工作空间管理员仅能够使用集群中的额度在应用工作台创建命名空间，不具备集群的管理权限。若对该工作空间限制额度，则工作空间管理仅能够在额度范围内创建并使用命名空间。

## 资源类型的区别

- 资源组：能够绑定集群、集群-命名空间、多云、多云-命名空间、网格、网格-命名空间
- 共享资源：仅能够绑定集群

## 资源组与共享资源的相同点

在资源组/共享资源绑定集群后都可以前往应用工作台创建命名空间，创建后命名空间将自动绑定到工作空间。

## 资源绑定权限说明

假如用户小明（“小明”代表任何有资源绑定需求的用户）已经具备了 [Workspace Admin 角色](#) 或已通过[自定义角色](#)授权，同时自定义角色中包含[工作空间的“资源绑定”权限](#)，希望将某个集群或者某个命名空间绑定到其所在的工作空间中。

要将集群/命名空间资源绑定到工作空间，不仅需要该[工作空间的“资源绑定”权限](#)，还需要

Cluster Admin 的资源权限。

## 给小明授权

- 使用平台 Admin 角色，在 **工作空间 -> 授权** 页面给小明授予 Workspace Admin 角色。

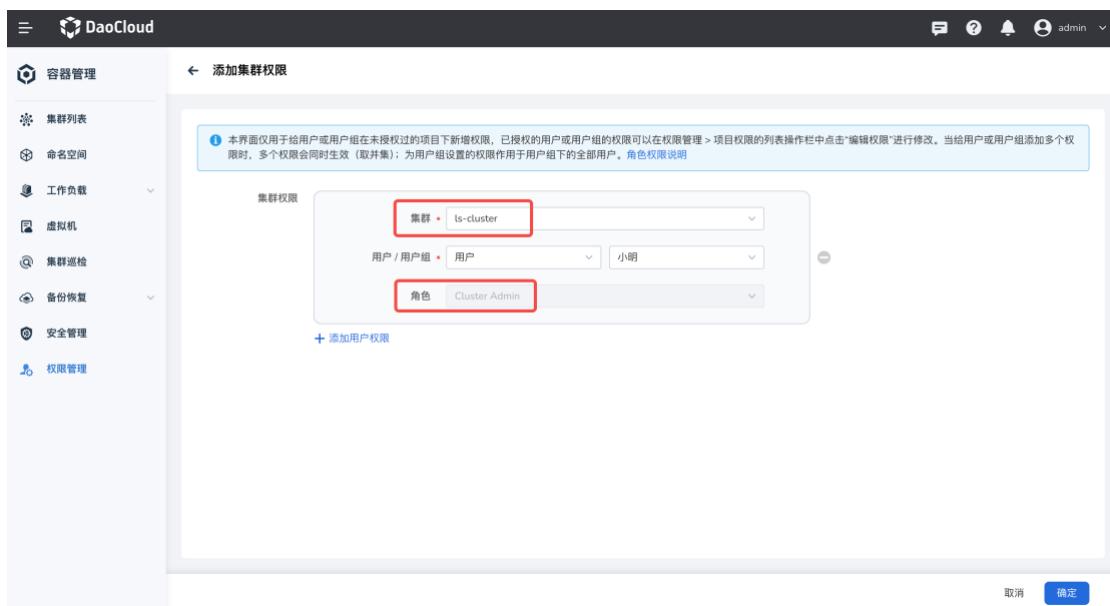
The screenshot shows the 'Authorization' page for workspace 'ws01'. On the left, there's a sidebar with '全局管理' (Global Management) and '工作空间与层级' (Workspaces and Hierarchies). Under '工作空间与层级', '审计日志' (Audit Log), '运营管理' (Operations Management), and '平台设置' (Platform Settings) are listed. The main area has a title '授权 | ws01 (工作空间)'. Below it, a note says: '本界面仅用于给用户或用户组在未授权过的工作空间下新增授权, 已授权的用户或用户组的权限可以在授权列表中进行修改。为用户组设置的权限作用于用户组下的全部用户。'. A '添加授权' (Add Authorization) button is present. A red box highlights the '角色' (Role) dropdown, which is set to 'Workspace Admin'. There are also '用户/用户组' (User/User Group) and '用户' (User) dropdowns, and a '查看权限说明' (View Permission Description) link. At the bottom right are '取消' (Cancel) and '确定' (Confirm) buttons.

## 资源绑定

- 然后在 **容器管理 -> 权限管理** 页面，通过 **添加授权** 将小明授权为 Cluster Admin。

The screenshot shows the '权限管理' (Permission Management) page under '容器管理'. On the left, there's a sidebar with '容器管理' and '集群列表' (Cluster List). Under '集群列表', '命名空间' (Namespaces), '工作负载' (Workloads), '虚拟机' (Virtual Machines), '集群巡检' (Cluster Audit), '备份恢复' (Backup Recovery), and '安全管理' (Security Management) are listed. A red box highlights the '权限管理' (Permission Management) link. The main area has a title '权限管理'. It shows a table with columns '集群' (Cluster), '用户 / 用户组' (User / User Group), and '角色' (Role). The table contains three rows: 'fuguo-poc' with 'Cluster Admin' role assigned to 'tanghai-gpu', 'lrf' with 'Cluster Admin' role assigned to 'tanghai-gpu', and 'yunnan-test' with 'Cluster Admin' role assigned to 'tanghai-gpu'. At the top right of the table, there's a '添加授权' (Add Authorization) button, which is highlighted with a red box. Other buttons include settings and delete icons.

## 集群授权 1



## 集群授权 2

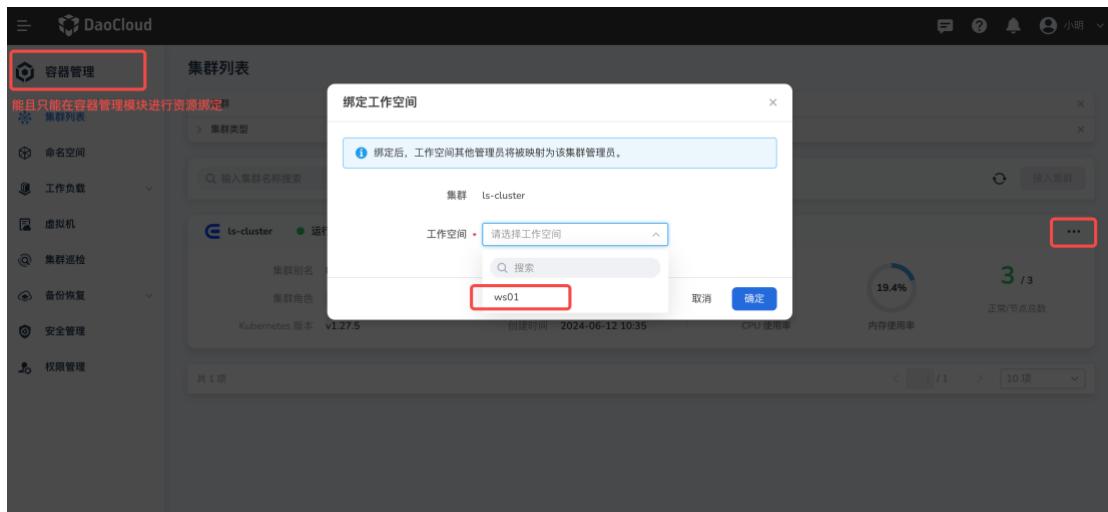
## 绑定到工作空间

使用小明的账号登录 DCE 5.0，在 **容器管理** -> **集群列表** 页面，通过 **绑定工作空间** 功能，

小明可以将指定集群绑定到自己的工作空间中。

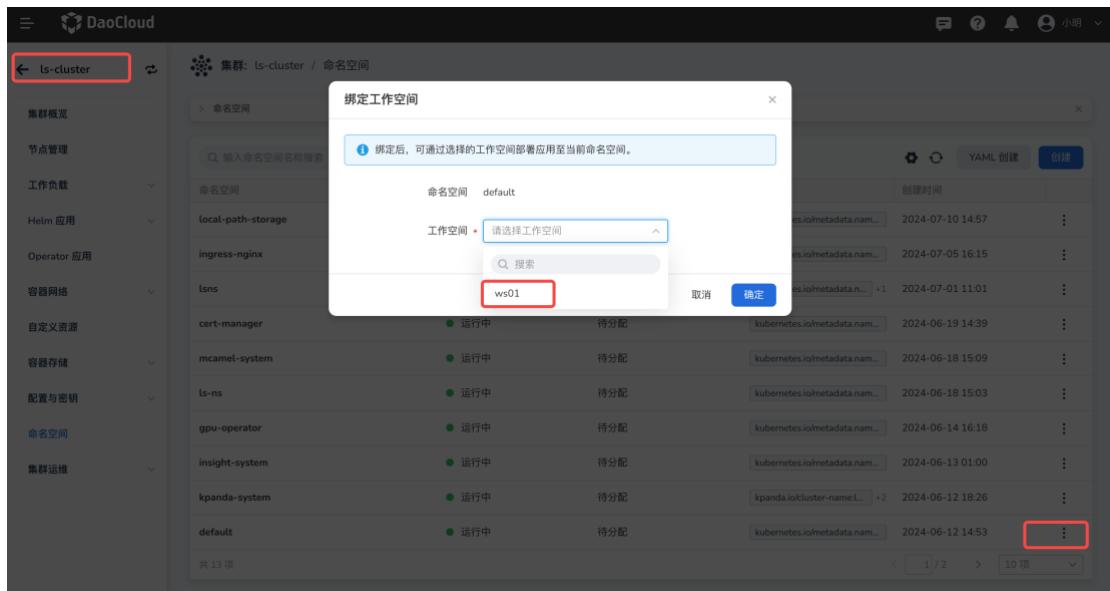
### !!! note

小明能且只能在[容器管理模块](../../../../kafka/intro/index.md)将集群或者该集群下的命名空间绑定到某个工作空间，无法在全局管理模块完成此操作。



## cluster 绑定

绑定命名空间到工作空间也至少需要 **Workspace Admin + Cluster Admin** 权限。



ns 绑定

## 采集 K8s 审计日志

- 生成 K8s 审计日志：K8s 本身生成的审计日志，开启该功能后，会在指定目录下生成 K8s 审计日志的日志文件
- 采集 K8s 审计日志：通过 insight-agent 采集上述 'K8s 审计日志' 的日志文件，'采集 K8s 审计日志' 的前提条件是：
  - 集群生成了 'K8s 审计日志'
  - 日志输出开关已打开
  - 日志采集开关已打开

## DCE 5.0 安装完成时状态

- 社区版安装管理集群过程中未操作 K8s 审计日志开关
- 商业版管理集群的 K8s 审计日志开关默认开启

- 如需设置成默认关闭，可[修改安装器 clusterConfig.yaml](#) 来配置 (logPath 设置为空 "")
- 管理集群的采集 K8s 审计日志开关默认关闭
  - 默认设置不支持配置

## 管理集群采集 K8s 审计日志开关

### 商业版安装环境

#### 确认是否开启了 K8s 审计日志

执行以下命令查看 `/var/log/kubernetes/audit` 目录下是否有审计日志生成。若有，则表示 K8s 审计日志成功开启。

```
ls /var/log/kubernetes/audit
```

若未开启，请参考[生成 K8s 审计日志](#)。

### 开启采集 K8s 审计日志流程

#### 1. 添加 chartmuseum 到 helm repo 中

```
helm repo add chartmuseum http://10.5.14.30:8081
```

这条命令中的 IP 需要修改为火种节点的 IP 地址。

#### !!! note

使用自建 Harbor 仓库的情况下，请修改第一步中的 chart repo 地址为自建仓库的 insight-agent chart 地址。

#### 2. 保存当前 insight-agent helm value

```
helm get values insight-agent -n insight-system -o yaml > insight-agent-values-bak.yaml
```

#### 3. 获取当前版本号 \${insight\_version\_code}

```
insight_version_code=`helm list -n insight-system |grep insight-agent | awk {'print $10'}`
```

#### 4. 更新 helm value 配置

```
helm upgrade --install --create-namespace --version ${insight_version_code} --cleanup-on-fail
 insight-agent chartmuseum/insight-agent -n insight-system -f insight-agent-values-bak.yaml
 --set global.exporters.auditLog.kubeAudit.enabled=true
```

### 5. 重启 insight-system 下的所有 fluentBit pod

```
fluent_pod=`kubectl get pod -n insight-system | grep insight-agent-fluent-bit | awk {'print
$1'} | xargs`
kubectl delete pod ${fluent_pod} -n insight-system
```

## 关闭采集 K8s 审计日志

其余步骤和开启采集 K8s 审计日志一致，仅需修改上一节中第 4 步：更新 helm value 配置。

```
helm upgrade --install --create-namespace --version ${insight_version_code} --cleanup-on-fail insight-agent chartmuseum/insight-agent -n insight-system -f insight-agent-values-bak.yaml --set global.exporters.auditLog.kubeAudit.enabled=false
```

## 社区版在线安装环境

### !!! note

若在 Kind 集群中安装社区版 DCE5.0 社区版，需要在 Kind 容器内进行以下操作。

## 确认开启 K8s 审计日志

执行以下命令查看 `/var/log/kubernetes/audit` 目录下是否有审计日志生成，若有，则表示 K8s 审计日志成功开启。

```
ls /var/log/kubernetes/audit
```

若未开启，请参考[文档的开启关闭 K8s 审计日志](#)。

## 开启采集 K8s 审计日志流程

### 1. 保存当前 value

```
helm get values insight-agent -n insight-system -o yaml > insight-agent-values-bak.yaml
```

### 2. 获取当前版本号 \${insight\_version\_code}，然后更新配置

```
insight_version_code=`helm list -n insight-system |grep insight-agent | awk {'print $10'}`
```

### 3. 更新 helm value 配置

```
helm upgrade --install --create-namespace --version ${insight_version_code} --cleanup-on-fail
 insight-agent insight-release/insight-agent -n insight-system -f insight-agent-values-bak.yaml
 --set global.exporters.auditLog.kubeAudit.enabled=true
```

如果因为版本未找到而升级失败，请检查命令中使用的 helm repo 是否有这个版本。

若没有，请尝试更新 helm repo 后重试。

```
helm repo update insight-release
```

### 4. 重启 insight-system 下的所有 fluentBit pod

```
fluent_pod=`kubectl get pod -n insight-system | grep insight-agent-fluent-bit | awk {'print
 $1'} | xargs`

 kubectl delete pod ${fluent_pod} -n insight-system
```

## 关闭采集 K8s 审计日志

其余步骤和开启采集 K8s 审计日志一致，仅需修改上一节中第 3 步：更新 helm value 配置

```
helm upgrade --install --create-namespace --version ${insight_version_code} --cleanup-on-fail
 insight-agent insight-release/insight-agent -n insight-system -f insight-agent-values-bak.yaml
 --set global.exporters.auditLog.kubeAudit.enabled=false
```

## 工作集群开关

各工作集群开关独立，按需开启。

## 创建集群时打开采集审计日志步骤

采集 K8s 审计日志功能默认为关闭状态。若需要开启，可以按照如下步骤：

默认关闭

默认关闭

开启审计日志

## 开启审计日志

将该按钮设置为启用状态，开启采集 K8s 审计日志功能。

通过 DCE 5.0 创建工作集群时，确认该集群的 K8s 审计日志选择 ‘true’，这样创建出来的  
工作集群 K8s 审计日志是开启的。

审计日志开启

审计日志开启

等待集群创建成功后，该工作集群的 K8s 审计日志将被采集。

## 接入的集群和创建完成后开关步骤

### 确认开启 K8s 审计日志

执行以下命令查看 `/var/log/kubernetes/audit` 目录下是否有审计日志生成，若有，则表示  
K8s 审计日志成功开启。

```
ls /var/log/kubernetes/audit
```

若未开启，请参考[文档的开启关闭 K8s 审计日志](#)

### 开启采集 K8s 审计日志

采集 K8s 审计日志功能默认为关闭状态，若需要开启，可以按照如下步骤：

1. 选中已接入并且需要开启采集 K8s 审计日志功能的集群

选中集群

选中集群

2. 进入 helm 应用管理页面，更新 insight-agent 配置（若未安装 insight-agent，可以[安  
装 insight-agent](#)）

进入 Helm 应用页面

进入 Helm 应用页面

3.开启/关闭采集 K8s 审计日志按钮

开启/关闭按钮

开启/关闭按钮

4.接入集群的情况下开关后仍需要重启 fluent-bit pod 才能生效

重启

重启

## 生成 K8s 审计日志

默认 Kubernetes 集群不会生成审计日志信息。通过以下配置，可以开启 Kubernetes 的审计日志功能。

!!! note

公有云环境中可能无法控制 Kubernetes 审计日志输出及输出路径。

1.准备审计日志的 Policy 文件

2.配置 API 服务器，开启审计日志

3.重启并验证

## 准备审计日志 Policy 文件

??? note “点击查看审计日志 Policy YAML 文件”

```
```yaml title="policy.yaml"
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
# The following requests were manually identified as high-volume and low-risk,
# so drop them.
```

```
- level: None
  users: ["system:kube-proxy"]
  verbs: ["watch"]
  resources:
    - group: "" # core
      resources: ["endpoints", "services", "services/status"]
- level: None
  # Ingress controller reads `configmaps/ingress-uid` through the unsecured port.
  # TODO(#46983): Change this to the ingress controller service account.
  users: ["system:unsecured"]
  namespaces: ["kube-system"]
  verbs: ["get"]
  resources:
    - group: "" # core
      resources: ["configmaps"]
- level: None
  users: ["kubelet"] # legacy kubelet identity
  verbs: ["get"]
  resources:
    - group: "" # core
      resources: ["nodes", "nodes/status"]
- level: None
  userGroups: ["system:nodes"]
  verbs: ["get"]
  resources:
    - group: "" # core
      resources: ["nodes", "nodes/status"]
- level: None
  users:
    - system:kube-controller-manager
    - system:kube-scheduler
    - system:serviceaccount:kube-system:endpoint-controller
      verbs: ["get", "update"]
      namespaces: ["kube-system"]
      resources:
        - group: "" # core
          resources: ["endpoints"]
- level: None
  users: ["system:apiserver"]
  verbs: ["get"]
  resources:
    - group: "" # core
      resources: ["namespaces", "namespaces/status", "namespaces/finalize"]
# Don't log HPA fetching metrics.
```

```
- level: None
users:
  - system:kube-controller-manager
    verbs: ["get", "list"]
    resources:
      - group: "metrics.k8s.io"
# Don't log these read-only URLs.
- level: None
  nonResourceURLs:
    - /healthz*
    - /version
    - /swagger*
# Don't log events requests.
- level: None
  resources:
    - group: "" # core
      resources: ["events"]
# Secrets, ConfigMaps, TokenRequest and TokenReviews can contain sensitive & binary data,
# so only log at the Metadata level.
- level: Metadata
  resources:
    - group: "" # core
      resources: ["secrets", "configmaps", "serviceaccounts/token"]
    - group: authentication.k8s.io
      resources: ["tokenreviews"]
    omitStages:
      - "RequestReceived"
# Get responses can be large; skip them.
- level: Request
  verbs: ["get", "list", "watch"]
  resources:
    - group: "" # core
    - group: "admissionregistration.k8s.io"
    - group: "apiextensions.k8s.io"
    - group: "apiregistration.k8s.io"
    - group: "apps"
    - group: "authentication.k8s.io"
    - group: "authorization.k8s.io"
    - group: "autoscaling"
    - group: "batch"
    - group: "certificates.k8s.io"
    - group: "extensions"
    - group: "metrics.k8s.io"
    - group: "networking.k8s.io"
```

```

    - group: "policy"
    - group: "rbac.authorization.k8s.io"
    - group: "settings.k8s.io"
    - group: "storage.k8s.io"
      omitStages:
        - "RequestReceived"

# Default level for known APIs
- level: RequestResponse
  resources:
    - group: "" # core
    - group: "admissionregistration.k8s.io"
    - group: "apiextensions.k8s.io"
    - group: "apiregistration.k8s.io"
    - group: "apps"
    - group: "authentication.k8s.io"
    - group: "authorization.k8s.io"
    - group: "autoscaling"
    - group: "batch"
    - group: "certificates.k8s.io"
    - group: "extensions"
    - group: "metrics.k8s.io"
    - group: "networking.k8s.io"
    - group: "policy"
    - group: "rbac.authorization.k8s.io"
    - group: "settings.k8s.io"
    - group: "storage.k8s.io"
      omitStages:
        - "RequestReceived"

# Default level for all other requests.
- level: Metadata
  omitStages:
    - "RequestReceived"
```

```

将以上审计日志文件放到 `/etc/kubernetes/audit-policy/` 文件夹下，并取名为 `apiserver-audit-policy.yaml`。

## 配置 API 服务器

打开 API 服务器的配置文件 `kube-apiserver.yaml`，一般会在 `/etc/kubernetes/manifests/` 文件夹下，并添加以下配置信息：

这一步操作前请备份 `kube-apiserver.yaml`，并且备份的文件不能放在 `/etc/kubernetes/manifests/` 下，建议放在 `/etc/kubernetes/tmp`。

1. 在 `spec.containers.command` 下添加命令：

```
--audit-log-maxage=30
--audit-log-maxbackup=10
--audit-log-maxsize=100
--audit-log-path=/var/log/audit/kube-apiserver-audit.log
--audit-policy-file=/etc/kubernetes/audit-policy/apiserver-audit-policy.yaml
```

2. 在 `spec.containers.volumeMounts` 下添加：

- `mountPath: /var/log/audit`  
`name: audit-logs`
- `mountPath: /etc/kubernetes/audit-policy`  
`name: audit-policy`

3. 在 `spec.volumes` 下添加：

- `hostPath:`
  - `path: /var/log/kubernetes/audit`
  - `type: ""`
  - `name: audit-logs`
- `hostPath:`
  - `path: /etc/kubernetes/audit-policy`
  - `type: ""`
  - `name: audit-policy`

## 测试并验证

稍等一会，API 服务器会自动重启，执行以下命令查看 `/var/log/kubernetes/audit` 目录下是否有审计日志生成，若有，则表示 K8s 审计日志成功开启。

```
ls /var/log/kubernetes/audit
```

如果想关闭，去掉 `spec.containers.command` 中的相关命令即可。

# 审计日志

审计日志帮助您监控并记录每个用户的活动，提供了与安全相关的、按时间顺序排列的记录

的收集、存储和查询功能。 借助审计日志服务，您可以持续监控并保留用户在全局管理模块的使用行为，包括但不限于创建用户、用户登录/登出、用户授权以及与 Kubernetes 相关的用户操作行为。

## 功能特性

审计日志功能具有以下特点：

- 开箱即用：在安装使用该平台时，审计日志功能将会被默认启用，自动记录与用户相关的各种行为，如创建用户、授权、登录/登出等。默认可以在平台内查看 365 天的用户行为。
- 安全分析：审计日志会对用户操作进行详细的记录并提供导出功能，通过这些事件您可以判断账号是否存在风险。
- 实时记录：迅速收集操作事件，用户操作后可在审计日志列表进行追溯，随时发现可疑行为。
- 方便可靠：审计日志支持手动清理和自动清理两种方式，可根据您的存储大小配置清理策略。

## 查看审计日志

1. 使用具有 `admin` 或 `Audit Owner` 角色的用户登录 DCE 5.0。

[登录 DCE 5.0](#)

[登录 DCE 5.0](#)

2. 在左侧导航栏底部，点击 **全局管理 -> 审计日志**。

[审计日志](#)

审计日志

## 用户操作

在 **用户操作** 页签中，可以按时间范围，也可以通过模糊搜索、精确搜索来查找用户操作事件。

点击某个事件最右侧的 ，可以查看事件详情。

用户审计日志

用户审计日志

事件详情如下图所示。

用户事件详情

用户事件详情

点击右上角的 **导出** 按钮，可以按 CSV 和 Excel 格式导出当前所选时间范围内的用户操作日志。

导出

导出

## 系统操作

在 **系统操作** 页签中，可以按时间范围，也可以通过模糊搜索、精确搜索来查找系统操作事件。

同样点击某个事件最右侧的 ，可以查看事件详情。

系统事件详情

系统事件详情

点击右上角的 **导出** 按钮，可以按 CSV 和 Excel 格式导出当前所选时间范围内的系统操作日志。

导出

导出

## 设置

在 **设置** 页签中，您可以清理用户操作和系统操作的审计日志。

清理

清理

可以手动清理，建议清理前先导出并保存。也可以设置日志的最长保存时间实现自动清理。

**!!! note**

审计日志中与 Kubernetes 相关的日志记录由可观测性模块提供，为减轻审计日志的存储压力，全局管理默认不采集 Kubernetes 相关日志。

如需记录请参阅开启 [K8s 审计日志](./open-k8s-audit.md)。开启后的清理功能与全局管理的清理功能一致，但互不影响。

## 审计日志获取源 IP

审计日志源 IP 在系统和网络管理中扮演着关键角色，它有助于追踪活动、维护安全、解决问题并确保系统合规性。但是获取源 IP 会带来一定的性能损耗，所以在 DCE 5.0 中审计日志并不总是开启的，在不同的安装模式下，审计日志源 IP 的默认开启情况不同，并且开启的方式不同。下面会根据安装模式分别介绍审计日志源 IP 的默认开启情况以及如何开启。

**!!! note**

开启审计日志会修改 istio-ingressgateway 的副本数，带来一定的性能损耗。

开启审计日志需要关闭 kube-proxy 的负载均衡以及拓扑感知路由，会对集群性能产生一定的影响。

开启审计日志后，访问 IP 所对应的节点上必须保证存在 istio-ingressgateway，若因为节点健康

或其他问题导致 `istio-ingressgateway` 发生漂移，需要手动调度回该节点，否则会影响 DCE 5.0 的正常使用。

## 判断安装模式的方法

```
kubectl get pod -n metallb-system
```

在集群中执行上面的命令，若返回结果如下，则表示该集群为非 MetalLB 安装模式

```
No resources found in metallb-system namespace.
```

## NodePort 安装模式

该模式安装下，审计日志源 IP 默认是关闭的，开启步骤如下：

### 1. 设置 `istio-ingressgateway` 的 HPA 的最小副本数为控制面节点数

```
count=$(kubectl get nodes --selector=node-role.kubernetes.io/control-plane | wc -l)
count=$((count-1))
```

```
kubectl patch hpa istio-ingressgateway -n istio-system -p '{"spec":{"minReplicas":$count}}'
```

### 2. 修改 `istio-ingressgateway` 的 service 的 `externalTrafficPolicy` 和 `internalTrafficPolicy`

值为 “Local”

```
kubectl patch svc istio-ingressgateway -n istio-system -p '{"spec":{"externalTrafficPolicy":"Local","internalTrafficPolicy":"Local"}}'
```

## MetalLB 安装模式

该模式下安装完成后，会默认获取审计日志源 IP，若需要了解更多，请参考 [MetalLB 源 IP](#)。

## 应用工作台审计项汇总

| 事件名称               | 资源类型        | 备注 |
|--------------------|-------------|----|
| 创建原生应用：            | Application |    |
| Create-Application |             |    |

| 事件名称                                    | 资源类型                             | 备注              |
|-----------------------------------------|----------------------------------|-----------------|
| 更新原生应用:                                 | Application                      | 编辑 yaml、创建版本快照、 |
| Update-Application                      |                                  | 回滚              |
| 删除原生应用:                                 | Application                      |                 |
| Delete-Application                      |                                  |                 |
| 创建 OAM 应用:                              | OAMApplication                   |                 |
| Create-OAMApplication                   |                                  |                 |
| 更新 OAM 应用:                              | OAMApplication                   | 编辑 yaml         |
| Update-OAMApplication                   |                                  |                 |
| 添加 OAM 应用组件:                            | OAMApplicationComponent          | 添加组件            |
| Add-OAMApplicationCompon<br>ent         |                                  |                 |
| 更新 OAM 组件运维特征:                          | OAMApplicationComponentTr<br>ait | 更新应用组件运维特征      |
| Update-OAMApplicationComp<br>onentTrait |                                  |                 |
| 删除 OAM 应用:                              | OAMApplication                   |                 |
| Delete-OAMApplication                   |                                  |                 |
| 创建 HELM 应用:                             | HelmApplication                  |                 |
| Create-HelmApplication                  |                                  |                 |
| 创建 OLM 应用:                              | OLMApplication                   |                 |
| Create-OLMApplication                   |                                  |                 |
| 更新 OLM 应用:                              | OLMApplication                   |                 |
| Update-OLMApplication                   |                                  |                 |
| 创建 OLM 应用:                              | OLMApplication                   |                 |
| Create-OLMApplication                   |                                  |                 |
| 创建命名空间:                                 | Namespace                        |                 |
| Create-Namespace                        |                                  |                 |
| 更新命名空间配额:                               | NamespaceQuota                   |                 |
| Update-NamespaceQuota                   |                                  |                 |

| 事件名称                      | 资源类型               | 备注                                    |
|---------------------------|--------------------|---------------------------------------|
| 删除命名空间:                   | Namespace          |                                       |
| Delete-Namespace          |                    |                                       |
| 创建流水线: Create-Pipeline    | Pipeline           |                                       |
| 更新流水线: Update-Pipeline    | Pipeline           | 包含所有的更新操作(编辑 jenkinsfile、编辑配置、编辑图形化、) |
| 运行流水线: Run-Pipeline       | Pipeline           | 立即运行 c 奥做                             |
| 重新运行: ReRun-Pipeline      | Pipeline           | 重新执行操作                                |
| 终止流水线: Abort-Pipeline     | Pipeline           | 终止操作+审批步骤的终止操作                        |
| 审批流水线:                    | Pipeline           | 审批通过流水线                               |
| Approval-Pipeline         |                    |                                       |
| 删除流水线: Delete-Pipeline    | Pipeline           |                                       |
| 创建流水线凭证:                  | PipelineCredential |                                       |
| Create-PipelineCredential |                    |                                       |
| 删除流水线凭证:                  | PipelineCredential |                                       |
| Delete-PipelineCredential |                    |                                       |
| 创建灰度发布任务:                 | GrayscaleTask      | 是否要区分是蓝绿还是金丝雀                         |
| Create-GrayscaleTask      |                    |                                       |
| 更新灰度发布任务:                 | GrayscaleTask      | 更新发布任务、更新版本、编辑 yaml、更新实例数             |
| Update-GrayscaleTask      |                    |                                       |
| 发布灰度发布任务任务:               | GrayscaleTask      |                                       |
| Upgrade-GrayscaleTask     |                    |                                       |

| 事件名称                       | 资源类型              | 备注                  |
|----------------------------|-------------------|---------------------|
| 终止灰度发布任务:                  | GrayscaleTask     |                     |
| Abort-GrayscaleTask        |                   |                     |
| 回滚灰度发布任务:                  | GrayscaleTask     |                     |
| Undo-GrayscaleTask         |                   |                     |
| 删除灰度发布任务:                  | GrayscaleTask     |                     |
| Delete-GrayscaleTask       |                   |                     |
| 创建 GitOps 应用:              | GitOpsApplication |                     |
| Create-GitOpsApplication   |                   |                     |
| 更新 GitOps 应用:              | GitOpsApplication |                     |
| Update-GitOpsApplication   |                   |                     |
| 同步 GitOps 应用:              | GitOpsApplication |                     |
| Sync-GitOpsApplication     |                   |                     |
| 删除 GitOps 应用:              | GitOpsApplication |                     |
| Delete-GitOpsApplication   |                   |                     |
| 导入 GitOps 仓库:              | GitOpsRepository  |                     |
| Import-GitOpsRepository    |                   |                     |
| 删除 GitOps 仓库:              | GitOpsRepository  |                     |
| Delete-GitOpsRepository    |                   |                     |
| 工具链集成:                     | Toolchain         |                     |
| Integrated-Toolchain       |                   |                     |
| 解除集成工具链:                   | Toolchain         |                     |
| Delete-Toolchain           |                   |                     |
| 绑定工具链项目:                   | ToolchainProject  | jira、gitlab 支持管理员视角 |
| Bind-ToolchainProject      |                   | 下 sonarqube 也支持     |
| 解除绑定工具链工具链项                | ToolchainProject  | jira、gitlab 支持管理员视角 |
| 目: Unbind-ToolchainProject |                   | 下 sonarqube 也支持     |

# 容器管理审计项汇总

| 事件名称                              | 资源类型        |
|-----------------------------------|-------------|
| 创建集群: Create-Cluster              | Cluster     |
| 卸载集群: Delete-Cluster              | Cluster     |
| 接入集群: Integrate-Cluster           | Cluster     |
| 解除接入的集群: Remove-Cluster           | Cluster     |
| 集群升级: Upgrade-Cluster             | Cluster     |
| 集群接入节点: Integrate-Node            | Node        |
| 集群节点移除: Remove-Node               | Node        |
| 集群节点 GPU 模式切换: Update-NodeGPUMode | NodeGPUMode |
| helm 仓库创建: Create-HelmRepo        | HelmRepo    |
| helm 应用部署: Create-HelmApp         | HelmApp     |
| helm 应用删除: Delete-HelmApp         | HelmApp     |
| 创建无状态负载: Create-Deployment        | Deployment  |
| 删除无状态负载: Delete-Deployment        | Deployment  |
| 创建守护进程: Create-DaemonSet          | DaemonSet   |
| 删除守护进程: Delete-DaemonSet          | DaemonSet   |
| 创建有状态负载: Create-StatefulSet       | StatefulSet |
| 删除有状态负载: Delete-StatefulSet       | StatefulSet |
| 创建任务: Create-Job                  | Job         |
| 删除任务: Delete-Job                  | Job         |

| 事件名称                                 | 资源类型                  |
|--------------------------------------|-----------------------|
| 创建定时任务：Create-CronJob                | CronJob               |
| 删除定时任务：Delete-CronJob                | CronJob               |
| 删除容器组：Delete-Pod                     | Pod                   |
| 创建服务：Create-Service                  | Service               |
| 删除服务：Delete-Service                  | Service               |
| 创建路由：Create-Ingress                  | Ingress               |
| 删除路由：Delete-Ingress                  | Ingress               |
| 创建存储池：Create-StorageClass            | StorageClass          |
| 删除存储池：Delete-StorageClass            | StorageClass          |
| 创建数据卷：Create-PersistentVolume        | PersistentVolume      |
| 删除数据卷：Delete-PersistentVolume        | PersistentVolume      |
| 创建数据卷声明：Create-PersistentVolumeClaim | PersistentVolumeClaim |
| 删除数据卷声明：Delete-PersistentVolumeClaim | PersistentVolumeClaim |
| 删除副本集：Delete-ReplicaSet              | ReplicaSet            |
| ns 绑定工作空间：BindResourceTo-Workspace   | Workspace             |
| ns 解绑工作空间：UnBindResource-Workspace   | Workspace             |
| 集群绑定工作空间：BindResourceTo-Workspace    | Workspace             |
| 集群解绑工作空间：UnBindResource-Workspace    | Workspace             |
| 打开控制台：Create-CloudShell              | CloudShell            |
| 关闭控制台：Delete-CloudShell              | CloudShell            |

# 多云编排审计项汇总

| 事件名称                    | 资源类型       | 备注 |
|-------------------------|------------|----|
| 添加多云实例:                 | Instance   |    |
| Create-Instance         |            |    |
| 删除多云实例:                 | Instance   |    |
| Delete-Instance         |            |    |
| 接入集群: Integrate-Cluster | Cluster    |    |
| 移除集群: Remove-Cluster    | Cluster    |    |
| 创建无状态负载:                | Deployment |    |
| Create-Deployment       |            |    |
| 更新无状态负载:                | Deployment |    |
| Update-Deployment       |            |    |
| 删除无状态负载:                | Deployment |    |
| Delete-Deployment       |            |    |
| 创建任务: Create-Job        | Job        |    |
| 更新任务: Update-Job        | Job        |    |
| 删除任务: Delete-Job        | Job        |    |
| 创建定时任务:                 | CronJob    |    |
| Create-CronJob          |            |    |
| 更新定时任务:                 | CronJob    |    |
| Update-CronJob          |            |    |
| 删除定时任务:                 | CronJob    |    |
| Delete-CronJob          |            |    |
| 创建多云自定义资源:              | CRD        |    |
| Create-CRD              |            |    |

| 事件名称                         | 资源类型                  | 备注 |
|------------------------------|-----------------------|----|
| 更新多云自定义资源:                   | CRD                   |    |
| Update-CRD                   | CRD                   |    |
| 删除多云自定义资源:                   | CRD                   |    |
| Delete-CRD                   | CRD                   |    |
| 创建多云服务:                      | Multicloud Services   |    |
| Create-Multicloud Services   | Multicloud Services   |    |
| 更新多云服务:                      | Multicloud Services   |    |
| Update-Multicloud Services   | Multicloud Services   |    |
| 删除多云服务:                      | Multicloud Services   |    |
| Delete-Multicloud Services   | Multicloud Services   |    |
| 创建多云路由:                      | Multicloud Ingress    |    |
| Create-Multicloud Ingress    | Multicloud Ingress    |    |
| 更新多云路由:                      | Multicloud Ingress    |    |
| Update-Multicloud Ingress    | Multicloud Ingress    |    |
| 删除多云路由:                      | Multicloud Ingress    |    |
| Delete-Multicloud Ingress    | Multicloud Ingress    |    |
| 创建多云命名空间:                    | Multicloud Namespace  |    |
| Create-Multicloud Namespace  | Multicloud Namespace  |    |
| 更新多云命名空间:                    | Multicloud Namespace  |    |
| Update-Multicloud Namespace  | Multicloud Namespace  |    |
| Namespace                    | Multicloud Namespace  |    |
| 删除多云命名空间:                    | Multicloud Namespace  |    |
| Delete-Multicloud Namespace  | Multicloud Namespace  |    |
| 创建多云配置项:                     | Multicloud ConfigMaps |    |
| Create-Multicloud ConfigMaps | Multicloud ConfigMaps |    |
| 更新多云配置项:                     | Multicloud ConfigMaps |    |
| Update-Multicloud ConfigMaps | Multicloud ConfigMaps |    |

| 事件名称                         | 资源类型                  | 备注 |
|------------------------------|-----------------------|----|
| 删除多云配置项:                     | Multicloud ConfigMaps |    |
| Delete-Multicloud ConfigMaps |                       |    |
| 创建多云密钥:                      | Multicloud Secret     |    |
| Create-Multicloud Secret     |                       |    |
| 更新多云密钥:                      | Multicloud Secret     |    |
| Update-Multicloud Secret     |                       |    |
| 删除多云密钥:                      | Multicloud Secret     |    |
| Delete-Multicloud Secret     |                       |    |
| 创建部署策略:                      | Propagation Policies  |    |
| Create-Propagation Policies  |                       |    |
| 更新部署策略:                      | Propagation Policies  |    |
| Update-Propagation Policies  |                       |    |
| 删除部署策略:                      | Propagation Policies  |    |
| Delete-Propagation Policies  |                       |    |
| 创建差异化策略:                     | Override Policies     |    |
| Create-Override Policies     |                       |    |
| 更新差异化策略:                     | Override Policies     |    |
| Update-Override Policies     |                       |    |
| 删除差异化策略:                     | Override Policies     |    |
| Delete-Override Policies     |                       |    |

## 镜像仓库审计项汇总

| 事件名称                          | 资源类型      | 备注 |
|-------------------------------|-----------|----|
| 镜像删除: Delete-Image            | Image     |    |
| artifact 删除: Delete-Artifacts | Artifacts |    |

| 事件名称                                     | 资源类型                              | 备注                 |
|------------------------------------------|-----------------------------------|--------------------|
| 创建回收规则:                                  | ReclaimRule                       | 创建和删除是同一个接口,因      |
| Update-ReclaimRule                       |                                   | 此均会被记录为            |
|                                          |                                   | Update-ReclaimRule |
| 删除回收规则:                                  | ReclaimRule                       |                    |
| Update-ReclaimRule                       |                                   |                    |
| 手动运行回收规则:                                | ReclaimRule                       |                    |
| Manual-ReclaimRule                       |                                   |                    |
| 创建仓库集成:                                  | IntegratedRegistryinWorkspac<br>e |                    |
| Create-IntegratedRegistryinW<br>orkspace |                                   |                    |
| 删除仓库集成:                                  | IntegratedRegistryinWorkspac<br>e |                    |
| Delete-IntegratedRegistryinW<br>orkspace |                                   |                    |
| 更新仓库集成:                                  | IntegratedRegistryinWorkspac<br>e |                    |
| Update-IntegratedRegistryinW<br>orkspace |                                   |                    |
| 创建仓库集成:                                  | IntegratedRegistrybyAdmin         |                    |
| Create-IntegratedRegistrybyAd<br>min     |                                   |                    |
| 删除仓库集成:                                  | IntegratedRegistrybyAdmin         |                    |
| Delete-IntegratedRegistrybyAd<br>min     |                                   |                    |
| 更新仓库集成:                                  | IntegratedRegistrybyAdmin         |                    |
| Update-IntegratedRegistrybyA<br>dmin     |                                   |                    |
| 创建托管 harbor:                             | Harbor                            |                    |
| Create-Harbor                            |                                   |                    |
| 删除托管 Harbor:                             | Harbor                            |                    |
| Delete-Harbor                            |                                   |                    |

| 事件名称          | 资源类型   | 备注 |
|---------------|--------|----|
| 更新托管 Harbor:  | Harbor |    |
| Update-Harbor |        |    |

## 虚拟机审计项汇总

| 事件名称                            | 资源类型        | 备注 |
|---------------------------------|-------------|----|
| 重启虚拟机: Restart-VMs              | VM          |    |
| 虚拟机转换为模板: ConvertToTemplate-VMs | VM          |    |
| 编辑虚拟机: Edit-VMs                 | VM          |    |
| 更新虚拟机: Update-VMs               | VM          |    |
| 快照恢复: Restore-VMs               | VM          |    |
| 开机虚拟机: Power on-VMs             | VM          |    |
| 实时迁移: LiveMigrate-VMs           | VM          |    |
| 删除虚拟机: Delete-VMs               | VM          |    |
| 删除虚拟机模板: Delete-VM Template     | VM Template |    |
| 创建虚拟机: Create-VMs               | VM          |    |
| 创建快照: CreateSnapshot-VMs        | VM          |    |
| 关机虚拟机: Power off-VMs            | VM          |    |
| 克隆虚拟机: Clone-VMs                | VM          |    |

# 可观测性审计项汇总

| 事件名称                | 资源类型         | 备注 |
|---------------------|--------------|----|
| 创建拨测任务：             | ProbeJob     |    |
| Create-ProbeJob     | ProbeJob     |    |
| 编辑拨测任务：             | ProbeJob     |    |
| Update-ProbeJob     | ProbeJob     |    |
| 删除拨测任务：             | ProbeJob     |    |
| Delete-ProbeJob     | ProbeJob     |    |
| 创建告警策略：             | AlertPolicy  |    |
| Create-AlertPolicy  | AlertPolicy  |    |
| 编辑告警策略：             | AlertPolicy  |    |
| Update-AlertPolicy  | AlertPolicy  |    |
| 删除告警策略：             | AlertPolicy  |    |
| Delete-AlertPolicy  | AlertPolicy  |    |
| 导入告警策略：             | AlertPolicy  |    |
| Import-AlertPolicy  | AlertPolicy  |    |
| 在告警策略中添加规则：         | AlertRule    |    |
| Create-AlertRule    | AlertRule    |    |
| 在告警策略中编辑规则：         | AlertRule    |    |
| Update-AlertRule    | AlertRule    |    |
| 在告警策略中删除规则：         | AlertRule    |    |
| Delete-AlertRule    | AlertRule    |    |
| 创建告警模板：             | RuleTemplate |    |
| Create-RuleTemplate | RuleTemplate |    |
| 编辑告警模板：             | RuleTemplate |    |
| Update-RuleTemplate | RuleTemplate |    |
| 删除告警模板：             | RuleTemplate |    |

| 事件名称                    | 资源类型                    | 备注 |
|-------------------------|-------------------------|----|
| Delete-RuleTemplate     |                         |    |
| 创建邮箱组: Create-email     | email                   |    |
| 编辑邮箱组: Update-email     | email                   |    |
| 删除邮箱组: Delete-Receiver  | Receiver                |    |
| 创建钉钉机器人:                | dingtalk                |    |
| Create-dingtalk         |                         |    |
| 编辑钉钉机器人:                | dingtalk                |    |
| Update-dingtalk         |                         |    |
| 删除钉钉机器人:                | Receiver                |    |
| Delete-Receiver         |                         |    |
| 创建企微机器人:                | wecom                   |    |
| Create-wecom            |                         |    |
| 编辑企微机器人:                | wecom                   |    |
| Update-wecom            |                         |    |
| 删除企微机器人:                | Receiver                |    |
| Delete-Receiver         |                         |    |
| 创建 Webhook:             | webhook                 |    |
| Create-webhook          |                         |    |
| 编辑 Webhook:             | webhook                 |    |
| Update-webhook          |                         |    |
| 删除 Webhook:             | Receiver                |    |
| Delete-Receiver         |                         |    |
| 创建 SMS: Create-sms      | sms                     |    |
| 编辑 SMS: Update-sms      | sms                     |    |
| 删除 SMS: Delete-Receiver | Receiver                |    |
| 创建 SMS 服务器:             | aliyun, tencent, custom |    |

| 事件名称                               | 资源类型                    | 备注 |
|------------------------------------|-------------------------|----|
| Create-aliyun(或者: tencent, custom) |                         |    |
| 编辑 SMS 服务器:                        | aliyun, tencent, custom |    |
| Update-aliyun(或者: tencent, custom) |                         |    |
| 删除 SMS 服务器:                        | SMSserver               |    |
| Delete-SMSserver                   |                         |    |
| 创建消息模板:                            | MessageTemplate         |    |
| Create-MessageTemplate             |                         |    |
| 编辑消息模板:                            | MessageTemplate         |    |
| Update-MessageTemplate             |                         |    |
| 删除消息模板:                            | MessageTemplate         |    |
| Delete-MessageTemplate             |                         |    |
| 创建告警静默:                            | AlertSilence            |    |
| Create-AlertSilence                |                         |    |
| 编辑告警静默:                            | AlertSilence            |    |
| Update-AlertSilence                |                         |    |
| 删除告警静默:                            | AlertSilence            |    |
| Delete-AlertSilence                |                         |    |
| 创建告警抑制规则:                          | AlertInhibition         |    |
| Create-AlertInhibition             |                         |    |
| 编辑告警抑制规则:                          | AlertInhibition         |    |
| Update-AlertInhibition             |                         |    |
| 删除告警抑制规则:                          | AlertInhibition         |    |
| Delete-AlertInhibition             |                         |    |
| 更新系统配置:                            | SystemSettings          |    |
| Update-SystemSettings              |                         |    |

# 微服务引擎审计项汇总

| 事件名称                       | 资源类型       | 备注 |
|----------------------------|------------|----|
| 创建网关: Create-Gateway       | Gateway    |    |
| 更新网关: Update-Gateway       | Gateway    |    |
| 集群中网关列表:                   | Gateway    |    |
| ListClusterGateway-Gateway |            |    |
| 删除网关: Delete-Gateway       | Gateway    |    |
| API 上下线:                   | GatewayAPI |    |
| UpdateAPIStatus-GatewayAPI |            |    |
| API 调用测试:                  | GatewayAPI |    |
| DebugGatewayAPI-GatewayAP  |            |    |
|                            |            |    |
| 创建 API:                    | GatewayAPI |    |
| CreateGatewayAPI-GatewayAP |            |    |
|                            |            |    |
| 导入 API 检查:                 | GatewayAPI |    |
| ImportAPICheck-GatewayAPI  |            |    |
| 导入 API:                    | GatewayAPI |    |
| ImportAPI-GatewayAPI       |            |    |
| 更新 API 高级策略:               | GatewayAPI |    |
| UpdateGatewayAPIAdvancedP  |            |    |
| olicy-GatewayAPI           |            |    |
| 更新 API:                    | GatewayAPI |    |
| UpdateGatewayAPI-GatewayA  |            |    |
| PI                         |            |    |
| 批量更新 API 状态:               | GatewayAPI |    |
| BatchOperationAPI-GatewayA |            |    |
| PI                         |            |    |

| 事件名称                               | 资源类型           | 备注 |
|------------------------------------|----------------|----|
| 删除 API:                            | GatewayAPI     |    |
| DeleteAPI-GatewayAPI               |                |    |
| 创建网关接入服务:                          | GatewayService |    |
| CreateService-GatewayService       |                |    |
| 更新网关服务策略:                          | GatewayService |    |
| UpdateServicePolicy-GatewayService |                |    |
| 更新网关接入服务:                          | GatewayService |    |
| UpdateService-GatewayService       |                |    |
| 删除网关服务:                            | GatewayService |    |
| DeleteService-GatewayService       |                |    |
| 创建流量泳道:                            | Lane           |    |
| CreateLane-Lane                    |                |    |
| 创建泳道引流规则:                          | Lane           |    |
| CreateLaneDrainageRule-Lane        |                |    |
| 更新流量泳道状态:                          | Lane           |    |
| ActionLane-Lane                    |                |    |
| 更新泳道引流规则状态:                        | Lane           |    |
| UpdateLaneDrainageRuleStatus-Lane  |                |    |
| 更新泳道引流规则:                          | Lane           |    |
| UpdateLaneDrainageRule-Lane        |                |    |
| 删除流量泳道:                            | Lane           |    |
| DeleteLane-Lane                    |                |    |
| 删除泳道引流规则:                          | Lane           |    |
| DeleteLaneDrainageRule-Lane        |                |    |
| 添加流量泳道服务:                          | Lane           |    |

| 事件名称                          | 资源类型 | 备注 |
|-------------------------------|------|----|
| AddLaneService-Lane           | Lane |    |
| 移除流量泳道服务：                     |      |    |
| DeleteLaneService-Lane        | Mesh |    |
| 创建云原生微服务的服务治理插件：              |      |    |
| CreateServiceIstioPlugin-Mesh | Mesh |    |
| 创建云原生微服务的服务治理端口超时规则：          |      |    |
| CreateTimeout-Mesh            | Mesh |    |
| 创建云原生微服务的服务治理端口故障注入：          |      |    |
| CreateFault-Mesh              | Mesh |    |
| 创建云原生微服务的服务治理端口离群检测：          |      |    |
| CreateOutlierDetection-Mesh   | Mesh |    |
| 创建云原生微服务的服务治理端口熔断规则：          |      |    |
| CreateConnectionPool-Mesh     | Mesh |    |
| 创建云原生微服务的服务治理端口重试规则：          |      |    |
| CreateRetry-Mesh              | Mesh |    |
| 创建云原生微服务的服务治理端口重写规则：          |      |    |
| CreateRewrite-Mesh            | Mesh |    |
| 对云原生微服务的服务治理                  |      |    |

| 事件名称                                                  | 资源类型 | 备注 |
|-------------------------------------------------------|------|----|
| 插件排序：                                                 |      |    |
| SortServiceIstioPlugin-Mesh<br>更新云原生微服务的服务治理插件：       | Mesh |    |
| UpdateServiceIstioPlugin-Mesh<br>更新云原生微服务的服务治理端口超时规则： | Mesh |    |
| UpdateTimeout-Mesh<br>更新云原生微服务的服务治理端口负载均衡：            | Mesh |    |
| UpdateLb-Mesh<br>更新云原生微服务的服务治理端口故障注入：                 | Mesh |    |
| UpdateFault-Mesh<br>更新云原生微服务的服务治理端口离群检测：              | Mesh |    |
| UpdateOutlierDetection-Mesh<br>更新云原生微服务的服务治理端口熔断规则：   | Mesh |    |
| UpdateConnectionPool-Mesh<br>更新云原生微服务的服务治理端口重试规则：     | Mesh |    |
| UpdateRetry-Mesh<br>更新云原生微服务的服务治理                     | Mesh |    |

| 事件名称                                          | 资源类型 | 备注 |
|-----------------------------------------------|------|----|
| <b>理端口重写规则：</b>                               |      |    |
| UpdateRewrite-Mesh<br>将网格服务导入到云原生微            | Mesh |    |
| <b>服务中： ExportService-Mesh</b>                |      |    |
| 删除云原生微服务的服务治                                  | Mesh |    |
| <b>理插件：</b>                                   |      |    |
| DeleteServiceIstioPlugin-Mesh<br>删除云原生微服务的服务治 | Mesh |    |
| <b>理端口超时规则：</b>                               |      |    |
| DeleteTimeout-Mesh<br>删除云原生微服务的服务治            | Mesh |    |
| <b>理端口故障注入：</b>                               |      |    |
| DeleteFault-Mesh<br>删除云原生微服务的服务治              | Mesh |    |
| <b>理端口离群检测：</b>                               |      |    |
| DeleteOutlierDetection-Mesh<br>删除云原生微服务的服务治   | Mesh |    |
| <b>理端口熔断规则：</b>                               |      |    |
| DeleteConnectionPool-Mesh<br>删除云原生微服务的服务治     | Mesh |    |
| <b>理端口重试规则：</b>                               |      |    |
| DeleteRetry-Mesh<br>删除云原生微服务的服务治              | Mesh |    |
| <b>理端口重写规则：</b>                               |      |    |
| DeleteRewrite-Mesh                            |      |    |

| 事件名称                                    | 资源类型  | 备注 |
|-----------------------------------------|-------|----|
| 为云原生微服务的服务全局限流端口绑定限流规则：                 | Mesh  |    |
| CreateServiceIstioPluginRLSRules-Mesh   |       |    |
| 为云原生微服务的服务全局限流端口更新限流规则：                 | Mesh  |    |
| UpdateServiceIstioPluginRLSRules-Mesh   |       |    |
| 为云原生微服务的服务全局限流端口删除限流规则：                 | Mesh  |    |
| DeleteServiceIstioPluginRLSRules-Mesh   |       |    |
| 移除云原生微服务的服务：                            | Mesh  |    |
| RemoveService-Mesh                      |       |    |
| 按 metadata 查询托管 Nacos 服务的实例列表：          | Nacos |    |
| QueryServiceInstance-Nacos              |       |    |
| 创建托管 Nacos 命名空间中 的配置：CreateConfig-Nacos | Nacos |    |
| CreateConfig-Nacos                      |       |    |
| 创建托管 Nacos 命名空间中 服务的 API 信息：            | Nacos |    |
| CreateServiceAPI-Nacos                  |       |    |
| 创建托管 Nacos 命名空间：CreateNamespace-Nacos   | Nacos |    |
| CreateNamespace-Nacos                   |       |    |
| 创建托管 Nacos：                             | Nacos |    |
| Create-Nacos                            |       |    |

| 事件名称                                            | 资源类型  | 备注 |
|-------------------------------------------------|-------|----|
| 更新托管 Nacos 命名空间中<br>的服务: UpdateService-Nacos    | Nacos |    |
| 更新托管 Nacos 命名空间中<br>的配置: UpdateConfig-Nacos     | Nacos |    |
| 更新托管 Nacos 命名空间中<br>服务的 API 信息:                 | Nacos |    |
| UpdateServiceAPI-Nacos<br>更新托管 Nacos 命名空间中      | Nacos |    |
| 服务的实例详情:                                        |       |    |
| UpdateServiceInstance-Nacos<br>更新托管 Nacos 命名空间: | Nacos |    |
| UpdateNamespace-Nacos<br>更新托管 Nacos:            | Nacos |    |
| Update-Nacos<br>回滚托管 Nacos 命名空间中                | Nacos |    |
| 的配置: RollbackConfig-Nacos                       |       |    |
| 删除托管 Nacos 命名空间中                                | Nacos |    |
| 的配置: DeleteConfig-Nacos                         |       |    |
| 删除托管 Nacos 命名空间中                                | Nacos |    |
| 配置的灰度配置:                                        |       |    |
| DeleteBetaConfig-Nacos<br>删除托管 Nacos 命名空间:      | Nacos |    |
| DeleteNamespace-Nacos<br>删除托管 Nacos:            | Nacos |    |

| 事件名称 | 资源类型 | 备注 |
|------|------|----|
|------|------|----|

Delete-Nacos  
托管 Nacos 命名空间中的服

Nacos

务可观测信息：

GetServiceInsight-Nacos  
托管 Nacos 命名空间中服务

Nacos

的实例可观测信息：

GetServiceInstanceInsight-Nac  
os  
修改托管 Nacos 用户密码：

Nacos

UpdateUserPassword-Nacos  
更新托管 Nacos 的插件详

Plugin

情：Update-Plugin

创建接入注册中心服务的 Registry

API 文档：

CreateServiceAPI-Registry  
创建接入注册中心： Registry

Create-Registry  
更新接入注册中心服务的 Registry

API 文档：

UpdateServiceAPI-Registry  
更新接入注册中心服务的实 Registry

例：UpdateInstance-Registry

更新接入注册中心： Registry  
Update-Registry  
接入注册中心服务的可观测 Registry

| 事件名称 | 资源类型 | 备注 |
|------|------|----|
|------|------|----|

数据：

GetServiceInsight-Registry  
接入注册中心服务的实例可

Registry

观测信息：

GetInstanceInsight-Registry  
接入注册中心可用性检测：

Registry

Ping-Registry  
移除接入注册中心：

Registry

Delete-Registry  
创建/更新服务中的 Sentinel

Sentinel

Token 服务器：

CreateOrUpdateTokenServer-S  
entinel  
创建服务中的 Sentinel 流控

Sentinel

规则：CreateFlowRule-Sentinel

创建服务中的 Sentinel 热点

Sentinel

规则：

CreateParamFlowRule-Sentine  
l  
创建服务中的 Sentinel 熔断

Sentinel

规则：

CreateDegradeRule-Sentinel  
创建服务中的 Sentinel 授权

Sentinel

规则：

CreateAuthorityRule-Sentinel  
创建服务中的 Sentinel 系统

Sentinel

| 事件名称 | 资源类型 | 备注 |
|------|------|----|
|------|------|----|

规则：

CreateSystemRule-Sentinel  
更新服务中的 Sentinel 流控

Sentinel

规则：

UpdateFlowRule-Sentinel  
更新服务中的 Sentinel 热点

Sentinel

规则：

UpdateParamFlowRule-Sentinel  
更新服务中的 Sentinel 熔断

Sentinel

规则：

UpdateDegradeRule-Sentinel  
更新服务中的 Sentinel 授权

Sentinel

规则：

UpdateAuthorityRule-Sentinel  
更新服务中的 Sentinel 系统

Sentinel

规则：

UpdateSystemRule-Sentinel  
删除服务中的 Sentinel 集群

Sentinel

流控：

DeleteClusterFlow-Sentinel  
删除服务中的 Sentinel 流控

Sentinel

规则：DeleteFlowRule-Sentinel

删除服务中的 Sentinel 热点

Sentinel

规则：

| 事件名称 | 资源类型 | 备注 |
|------|------|----|
|------|------|----|

DeleteParamFlowRule-Sentinel  
|  
删除服务中的 Sentinel 熔断

规则：

DeleteDegradeRule-Sentinel  
删除服务中的 Sentinel 授权

规则：

DeleteAuthorityRule-Sentinel  
删除服务中的 Sentinel 系统

规则：

DeleteSystemRule-Sentinel  
移除服务实例中的 Sentinel

治理详情：

DeleteInsGovern-Sentinel  
创建插件：

CreatePlugin-SkoalaPluginTemplate  
更新插件状态：

UpdatePluginStatus-SkoalaPlugin  
更新插件：

UpdatePlugin-SkoalaPluginTemplate  
删除插件：

DeletePlugin-SkoalaPluginTemplate  
创建插件模板：

CreatePluginTemplate-SkoalaPluginTemplate  
上海道客网络科技有限公司

| 事件名称                                      | 资源类型                 | 备注 |
|-------------------------------------------|----------------------|----|
| 创建插件:                                     | SkoalaPluginTemplate |    |
| CreatePlugin-SkoalaPluginTemplate         |                      |    |
| 更新插件模板:                                   | SkoalaPluginTemplate |    |
| UpdatePluginTemplate-SkoalaPluginTemplate |                      |    |
| 更新插件:                                     | SkoalaPluginTemplate |    |
| UpdatePlugin-SkoalaPluginTemplate         |                      |    |
| 删除插件模板:                                   | SkoalaPluginTemplate |    |
| DeletePluginTemplate-SkoalaPluginTemplate |                      |    |
| 删除插件:                                     | SkoalaPluginTemplate |    |
| DeletePlugin-SkoalaPluginTemplate         |                      |    |
| 创建域名:                                     | Virtualhost          |    |
| CreateVirtualhost-Virtualhost             |                      |    |
| 更新域名:                                     | Virtualhost          |    |
| UpdateVirtualhost-Virtualhost             |                      |    |
| 删除域名:                                     | Virtualhost          |    |
| DeleteVirtualhost-Virtualhost             |                      |    |

## 服务网格审计项汇总

| 事件名称                | 资源类型         | 备注 |
|---------------------|--------------|----|
| 创建网格:               | MeshInstance |    |
| create-MeshInstance |              |    |
| 删除网格:               | MeshInstance |    |
| delete-MeshInstance |              |    |

| 事件名称                                              | 资源类型            | 备注 |
|---------------------------------------------------|-----------------|----|
| 接入集群: Add-Cluster                                 | cluster         |    |
| 移除集群: Remove-Cluster                              | cluster         |    |
| 命名空间边车注入启用:<br>InjectSidecarTo-Namespace          | Namespace       |    |
| 命名空间边车注入禁用:<br>ForbiddenInjectSidecarTo-Namespace | Namespace       |    |
| 工作负载边车注入启用:<br>InjectSidecarTo-Workload           | workload        |    |
| 工作负载边车注入禁用:<br>ForbiddenInjectSidecarTo-Workload  | workload        |    |
| 创建网格网关:<br>create-MeshGateway                     | MeshGateway     |    |
| 删除网格网关:<br>delete-MeshGateway                     | MeshGateway     |    |
| 启用多云: Enable-Multicloud                           | Multicloud      |    |
| 关闭多云: Close-Multicloud                            | Multicloud      |    |
| 启用互联:<br>EnableInterconnection                    | MulticloudGroup |    |
| 移出互联:<br>DisableInterconnection                   | MulticloudGroup |    |

## 中间件审计项汇总

| 事件名称         | 资源类型 | 备注 |
|--------------|------|----|
| 上海道客网络科技有限公司 |      | 1  |

| 事件名称                                     | 资源类型                  | 备注 |
|------------------------------------------|-----------------------|----|
| 创建 es: create-ElasticsearchInstance      | ElasticsearchInstance |    |
| 创建 kafka: create-KafkaInstance           | KafkaInstance         |    |
| 创建 MinIO: create-MinIOInstance           | MinIOInstance         |    |
| 创建 PostgreSQL: create-PostgreSQLInstance | PostgreSQLInstance    |    |
| 创建 RabbitMQ: create-RabbitMQInstance     | RabbitMQInstance      |    |
| 创建数据库: create-MySQLInstance              | MySQLInstance         |    |
| 创建数据库: create-RedisInstance              | RedisInstance         |    |
| 删除 es: delete-ElasticsearchInstance      | ElasticsearchInstance |    |
| 删除 kafka: delete-KafkaInstance           | KafkaInstance         |    |
| 删除 MinIO: delete-MinIOInstance           | MinIOInstance         |    |
| 删除 PostgreSQL: delete-PostgreSQLInstance | PostgreSQLInstance    |    |
| 删除 RabbitMQ: delete-RabbitMQInstance     | RabbitMQInstance      |    |
| 删除数据库: delete-MySQLInstance              | MySQLInstance         |    |
| 删除数据库: delete-RedisInstance              | RedisInstance         |    |
| 更新备份设置: update-MySQLBackup               | MySQLBackup           |    |
| 更新备份设置: update-RedisBackup               | RedisBackup           |    |
| 更新实例配置: update-ElasticsearchInstance     | ElasticsearchInstance |    |
| 更新实例配置: update-KafkaInstance             | KafkaInstance         |    |
| 更新实例配置: update-MinIOInstance             | MinIOInstance         |    |
| 更新实例配置: update-MySQLInstance             | MySQLInstance         |    |
| 更新实例配置: update-PostgreSQLInstance        | PostgreSQLInstance    |    |

| 事件名称                            | 资源类型             | 备注 |
|---------------------------------|------------------|----|
| 更新实例配置: update-RabbitMQInstance | RabbitMQInstance |    |
| 更新实例配置: update-RedisInstance    | RedisInstance    |    |

## 全局管理审计项汇总

| 事件名称                     | 资源类型    | 备注 |
|--------------------------|---------|----|
| 修改用户 email:              | Account |    |
| UpdateEmail-Account      |         |    |
| 修改用户密码:                  | Account |    |
| UpdatePassword-Account   |         |    |
| 创建 sk:                   | Account |    |
| CreateAccessKeys-Account |         |    |
| 修改 sk:                   | Account |    |
| UpdateAccessKeys-Account |         |    |
| 删除 sk:                   | Account |    |
| DeleteAccessKeys-Account |         |    |
| 创建用户: Create-User        | User    |    |
| 删除用户: Delete-User        | User    |    |
| 更新用户信息: Update-User      | User    |    |
| 更新用户角色:                  | User    |    |
| UpdateRoles-User         |         |    |
| 设置用户密码:                  | User    |    |
| UpdatePassword-User      |         |    |
| 创建用户密钥:                  | User    |    |
| CreateAccessKeys-User    |         |    |

| 事件名称                  | 资源类型  | 备注 |
|-----------------------|-------|----|
| 更新用户密钥:               | User  |    |
| UpdateAccessKeys-User |       |    |
| 删除用户密钥:               | User  |    |
| DeleteAccessKeys-User |       |    |
| 角色关联用户:               | User  |    |
| UpdateRoles-User      |       |    |
| 登录: Login-User        | User  |    |
| 退出: Logout-User       | User  |    |
| 创建用户组: Create-Group   | Group |    |
| 删除用户组: Delete-Group   | Group |    |
| 更新用户组: Update-Group   | Group |    |
| 添加用户至用户组:             | Group |    |
| AddUserTo-Group       |       |    |
| 从用户组删除用户:             | Group |    |
| RemoveUserFrom-Group  |       |    |
| 更新用户组角色:              | Group |    |
| UpdateRoles-Group     |       |    |
| 创建自定义角色:              | Role  |    |
| Create-CustomRole     |       |    |
| 更新自定义角色:              | Role  |    |
| Update-CustomRole     |       |    |
| 删除自定义角色:              | Role  |    |
| Delete-CustomRole     |       |    |
| 创建 Ldap : Create-LADP | LDAP  |    |
| 更新 Ldap: Update-LADP  | LDAP  |    |

| 事件名称                                           | 资源类型           | 备注                          |
|------------------------------------------------|----------------|-----------------------------|
| 删除 Ldap : Delete-LADP                          | LADP           | OIDC 没有走 APIserver 审计不<br>到 |
| 设置密码策略：<br>UpdatePassword-SecurityPolicy       | SecurityPolicy |                             |
| 设置会话超时：<br>UpdateSessionTimeout-SecurityPolicy | SecurityPolicy |                             |
| 设置账号锁定：<br>UpdateAccountLockout-SecurityPolicy | SecurityPolicy |                             |
| 设置自动登出：<br>UpdateLogout-SecurityPolicy         | SecurityPolicy |                             |
| 邮件服务器设置<br>MailServer-SecurityPolicy           | SecurityPolicy |                             |
| 外观定制<br>CustomAppearance-SecurityPolicy        | SecurityPolicy |                             |
| 正版授权<br>OfficialAuthz-SecurityPolicy           | SecurityPolicy |                             |
| 创建工作空间：<br>Create-Workspace                    | Workspace      |                             |
| 删除工作空间：<br>Delete-Workspace                    | Workspace      |                             |
| 绑定资源：<br>BindResourceTo-Workspace              | Workspace      |                             |
| 解绑资源：<br>UnBindResource-Workspace              | Workspace      |                             |

| 事件名称                            | 资源类型      | 备注 |
|---------------------------------|-----------|----|
| 绑定共享资源:                         | Workspace |    |
| BindShared-Workspace            | Workspace |    |
| 设置资源配额:                         | Workspace |    |
| SetQuota-Workspace              | Workspace |    |
| 工作空间授权:                         | Workspace |    |
| Authorize-Workspace             | Workspace |    |
| 删除授权                            | Workspace |    |
| DeAuthorize-Workspace           | Workspace |    |
| 编辑授权                            | Workspace |    |
| UpdateDeAuthorize-Workspac<br>e | Workspace |    |
| 更新工作空间                          | Workspace |    |
| Update-Workspace                | Folder    |    |
| 创建文件夹: Create-Folder            | Folder    |    |
| 删除文件夹: Delete-Folder            | Folder    |    |
| 编辑文件夹授权:                        | Folder    |    |
| UpdateAuthorize-Folder          | Folder    |    |
| 更新文件夹: Update-Folder            | Folder    |    |
| 新增文件夹授权:                        | Folder    |    |
| Authorize-Folder                | Folder    |    |
| 删除文件夹授权:                        | Folder    |    |
| DeAuthorize-Folder              | Audit     |    |
| 设置审计日志自动清理:                     | Audit     |    |
| AutoCleanup-Audit               | Audit     |    |
| 手动清理审计日志:                       | Audit     |    |
| ManualCleanup-Audit             | Audit     |    |
| 导出审计日志: Export-Audit            | Audit     |    |

# 运营管理

运营管理通过可视化的方式，为您展示平台上统计时间范围内集群、节点、命名空间、容器组、工作空间等维度的 CPU/内存/存储/GPU 的使用总量和使用率等信息。以及通过使用量、使用时间及单价等信息，自动计算出的平台消费信息。该模块默认开启所有报表统计，同时也支持平台管理员对单个报表进行手动开启或关闭，开启/关闭后将在最长 20 分钟内，平台开始/停止采集报表数据，往期已采集到的数据还将正常展示。运营管理数据最多可在平台上保留 365 天，超过保留时间的统计数据将被自动删除。您也可以通过 CSV 或 Excel 方式下载报表后进行进一步的统计和分析。

运营管理仅对标准版及以上版本开放，社区版暂不支持。

你需要先[安装或升级运营管理模块](#)，然后就可以体验报表管理和计费计量。

## 报表管理

报表管理通过 CPU 利用率、内存利用率、存储利用率、GPU 算力利用率、GPU 显存利用率 5 个维度，对集群、节点、容器组、工作空间、命名空间 5 种资源进行数据统计。同时联动审计和告警模块，支持对审计数据和告警数据进行统计管理。共计支持 7 种类型报表。

## 计量计费

计量计费针对平台上的集群、节点、容器组、命名空间和工作空间 5 种资源进行计费统计。

根据不同资源中 CPU、内存、存储和 GPU 的使用量，以及用户手动配置的价格和货币单位自动计算出每种资源在统计时间的消费情况，根据所选时间跨度不同，可快速计算出该跨度内的实际消费情况，如月度、季度、年度等。

# 报表管理

报表管理以可视化的方式，展示了集群、节点、容器组（Pod）、工作空间、命名空间、审计及告警维度的统计数据，为平台的计费及使用情况的调优提供了可靠的基础数据。

## 功能特性

- 支持查询自定义时间范围的统计数据
- 支持以 CSV 和 Excel 两种格式导出报表
- 支持开启/关闭单个报表，开启/关闭后，平台将在 20 分钟内开始/停止采集数据，往期已经采集到的数据还将正常显示
- 支持展示 CPU 使用率、内存使用率、存储使用率和 GPU 显存使用率的最大、最小和平均值

## 报表维度

目前支持以下几种报表：

- 集群报表：展示某段时间内所有集群的 CPU 使用率、内存使用率、存储使用率和 GPU 显存使用率的最大、最小和平均值，以及该段时间内集群下的节点数量，可通过点击节点数量快捷进入节点报表，并查看该段时间内该集群下的节点使用情况。
- 节点报表：展示某段时间内所有节点的 CPU 使用率、内存使用率、存储使用率和 GPU 显存使用率的最大、最小和平均值，以及节点的 IP、类型和所属集群。
- 容器组报表：展示某段时间内所有容器组的 CPU 使用率、内存使用率、存储使用率和 GPU 显存使用率的最大、最小和平均值，以及容器组的所属命名空间、所属集群和

所属工作空间。

- **工作空间报表：**展示某段时间内所有工作空间的 CPU 使用率、内存使用率、存储使用率和 GPU 显存使用率的最大、最小和平均值，以及命名空间数量和容器组数量，可通过点击命名空间数量快捷进入命名空间报表，并查看该段时间内该工作空间下命名空间的使用情况；同样的方式可查看该段时间下该工作空间下的容器组的使用情况。
- **命名空间报表：**展示某段时间内所有命名空间的 CPU 使用率、内存使用率、存储使用率和 GPU 显存使用率的最大、最小和平均值，以及容器组数量、所属集群、所属工作空间，可通过点击容器组数量快捷进入容器组报表，并查看该段时间内该命名空间下的容器组的使用情况。
- **审计报表：**分为用户操作和资源操作两个报表。用户操作报表主要统计单个用户在一段时间内的操作次数，以及成功和失败的次数； 资源操作报表主要统计所有用户对某种类型资源的操作次数。
- **告警报表：**展示某段时间内所有节点的告警数量，以及致命、严重、告警分别产生的次数。

## 操作步骤

1. 使用具有 **Admin** 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理 -> 运营**

**管理**。

**报表管理**

**报表管理**

2. 进入运营管理后切换不同菜单可查看集群、节点、容器组等报表。

| 集群名称                    | 节点数量 | CPU 使用率 (%) |       |       | 内存使用率 (%) |       |       | 存储使用率 |       |
|-------------------------|------|-------------|-------|-------|-----------|-------|-------|-------|-------|
|                         |      | 最大          | 最小    | 平均    | 最大        | 最小    | 平均    | 最大    | 最小    |
| peter-centos7-single    | 1    | 10.02       | 8.89  | 9.34  | 48.88     | 44.82 | 46.26 | 40.33 | 38.82 |
| skoala-dev              | 3    | 17.58       | 14.38 | 15.23 | 41.54     | 38.82 | 40.19 | 79.14 | 78.27 |
| virtual-machine-cluster | 3    | 7.93        | 6.83  | 7.02  | 18.39     | 18.19 | 18.28 | 51.95 | 50.36 |
| yiting-gpu3             | 1    | 5.64        | 4.99  | 5.21  | 11.69     | 10.62 | 10.8  | 58.59 | 57.97 |

## 报表

# 离线升级运营管理模块

本页说明[下载运营管理模块](#)后，应该如何安装或升级。

### !!! info

下述命令或脚本内出现的 `gmagpie` 字样是运营管理模块的内部开发代号。

## 从安装包中加载镜像

### !!! info

前置条件：将离线包上传至目标节点

您可以根据下面两种方式之一加载镜像，当环境中存在镜像仓库时，建议选择 chart-syncer

同步镜像到镜像仓库，该方法更加高效便捷。

## chart-syncer 同步镜像到镜像仓库

### 1. 创建 load-image.yaml

#### !!! note

该 YAML 文件中的各项参数均为必填项。您需要一个私有的镜像仓库，并修改相关配置。

==== “已安装 chart repo”

```
```yaml title="load-image.yaml"
source:
  intermediateBundlesPath: gmagpie-offline # (1)!
target:
  containerRegistry: 10.16.10.111 # (2)!
  containerRepository: release.daocloud.io/gmagpie # (3)!
repo:
  kind: HARBOR # (4)!
  url: http://10.16.10.111/chartrepo/release.daocloud.io # (5)!
  auth:
    username: "admin" # (6)!
    password: "Harbor12345" # (7)!
containers:
  auth:
    username: "admin" # (8)!
    password: "Harbor12345" # (9)!

```

```

1. 到执行 charts-syncer 命令的相对路径，而不是此 YAML 文件和离线包之间的相对路径
2. 需更改为你的镜像仓库 url
3. 需更改为你的镜像仓库
4. 也可以是任何其他支持的 Helm Chart 仓库类别
5. 需更改为 chart repo url
6. 你的镜像仓库用户名
7. 你的镜像仓库密码
8. 你的镜像仓库用户名
9. 你的镜像仓库密码

#### ==== “未安装 chart repo”

若当前环境未安装 chart repo，chart-syncer 也支持将 chart 导出为 tgz 文件，并存放在指定路径。

```
```yaml title="load-image.yaml"
source:
  intermediateBundlesPath: gmagpie-offline # (1)!
target:
  containerRegistry: 10.16.10.111 # (2)!
  containerRepository: release.daocloud.io/gmagpie # (3)!
repo:
  kind: LOCAL
  path: ./local-repo # (4)!
containers:
  auth:
```

```

username: "admin" # (5)!
password: "Harbor12345" # (6)!

```

```

1. 到执行 charts-syncer 命令的相对路径，而不是此 YAML 文件和离线包之间的相对路径
2. 需更改为你的镜像仓库 url
3. 需更改为你的镜像仓库
4. chart 本地路径
5. 你的镜像仓库用户名
6. 你的镜像仓库密码

## 2. 执行同步镜像命令。

```
charts-syncer sync --config load-image.yaml
```

若有 x509 认证失败，可使用参数 --insecure

```
charts-syncer sync --config load-image.yaml --insecure
```

## Docker 或 containerd 直接加载

解压并加载镜像文件。

### 1. 解压 tar 压缩包。

```
tar xvf gmagpie.bundle.tar
```

解压成功后会得到 3 个文件：

- hints.yaml
- images.tar
- original-chart

### 2. 从本地加载镜像到 Docker 或 containerd。

```

==== "Docker"
```
shell
docker load -i images.tar
```

==== "containerd"
```
shell
ctr -n k8s.io image import images.tar
```

```

#### !!! note

每个节点都需要执行 Docker 或 containerd 加载镜像操作，  
加载完成后需要 tag 镜像，保持 Registry、Repository 与安装时一致。

# 升级

有两种升级方式。您可以根据前置操作，选择对应的升级方案：

## !!! note

当从 v0.1.x（或更低版本）升级到 v0.2.0（或更高版本）时，需要修改数据库连接参数。

数据库连接参数的修改示例：

```
```yaml title="bak.yaml"
USER-SUPPLIED VALUES:
global:
  database:
    host: 127.0.0.1
    port: 3306
    dbname: gmagpie
    password: passowrd
    user: gmagpie
```

```

修改为：

```
```yaml title="bak.yaml"
USER-SUPPLIED VALUES:
global:
  storage:
    gmagpie:
      - driver: mysql
        accessType: readwrite
        dsn: {global.database.apiserver.user}:{global.database.apiserver.password}@tcp({global.database.host}:{global.database.port})/{global.database.apiserver.dbname}?charset=utf8mb4&multiStatements=true&parseTime=true
```

```

## ==== “通过 helm repo 升级”

1. 检查运营管理 Helm 仓库是否存在。

```
```shell
helm repo list | grep gmagpie
```

```

若返回结果为空或如下提示，则进行下一步；反之则跳过下一步。

```
```none

```

```
Error: no repositories to show
```
```

1. 添加运营管理的 Helm 仓库。

```
```shell
helm repo add gmagpie http://{harbor url}/chartrepo/{project}
```
```

1. 更新运营管理的 Helm 仓库。

```
```shell
helm repo update gmagpie # (1)!
```
```

1. Helm 版本过低会导致失败，若失败，请尝试执行 helm update repo
1. 选择您想安装的运营管理版本（建议安装最新版本）。

```
```shell
helm search repo gmagpie/gmagpie --versions
```
```

```
```none
[root@master ~]# helm search repo gmagpie/gmagpie --versions
NAME          CHART VERSION APP VERSION DESCRIPTION
gmagpie/gmagpie 0.3.0      v0.3.0       A Helm chart for GHippo
...
```
```

1. 备份 `--set` 参数。

在升级运营管理版本之前，建议您执行如下命令，备份老版本的 `--set` 参数。

```
```shell
helm get values gmagpie -n gmagpie-system -o yaml > bak.yaml
```
```

1. 执行 `helm upgrade`。

升级前建议您覆盖 bak.yaml 中的 \_\_global.imageRegistry\_\_ 字段为当前使用的镜像仓库地址。

```
```shell
```

```

export imageRegistry={你的镜像仓库}
```
```
shell
helm upgrade gmagpie gmagpie/gmagpie \
-n gmagpie-system \
-f ./bak.yaml \
--set global.imageRegistry=$imageRegistry \
--version 0.3.0
```
```

```

==== “通过 Chart 包升级”

1. 备份 `--set` 参数。

在升级运营管理版本之前，建议您执行如下命令，备份老版本的 `--set` 参数。

```

```
shell
helm get values gmagpie -n gmagpie-system -o yaml > bak.yaml
```
```

```

1. 执行 `helm upgrade`。

升级前建议您覆盖 bak.yaml 中的 \_\_global.imageRegistry\_\_ 为当前使用的镜像仓库地址。

```

```
shell
export imageRegistry={你的镜像仓库}
```
```
shell
helm upgrade gmagpie . \
-n gmagpie-system \
-f ./bak.yaml \
--set global.imageRegistry=$imageRegistry
```
```

```

计量计费

计量计费在报表的基础上，对资源的使用数据做了进一步的计费处理。支持用户手动设置

CPU、内存、存储、GPU 的单价以及货币单位等，设置后系统将自动统计出集群、节点、容器组、命名空间、工作空间在一段时间内的花费情况，时间段用户可自由调整，可按照周、

月、季度、年筛选调整后导出 Excel 或 Csv 格式的计费报表。

计费规则及生效时间

- 计费规则：默认按照请求值和使用量的最大值计费。
- 生效时间：次日生效，以次日凌晨时获取的单价和数量计算当天产生的费用。

功能特性

- 支持自定义设置 CPU、内存、存储以及 GPU 的计费单位，以及货币单位。
- 支持查询自定义时间范围的统计数据，根据所选时间段自动计算出该时间段内的计费情况。
- 支持以 CSV 和 Excel 两种格式导出计费报表。
- 支持开启/关闭单个计费报表，开启/关闭后，平台将在 20 分钟内开始/停止采集数据，往期已经采集到的数据还将正常显示。
- 支持对 CPU、内存总量、存储、GPU、总计等计费数据的选择性展示。

报表维度

目前支持以下几种报表：

- 集群计费报表：展示某段时间内全部集群的 CPU、内存总量、存储、GPU、总计等计费情况，以及该段时间内该集群下的节点数量，可通过点击节点数量快捷进入节点计费报表，并查看该段时间内该集群下的节点计费情况。
- 节点计费报表：展示某段时间内全部节点的 CPU、内存总量、存储、GPU、总计等计费情况，以及节点的 IP、类型和所属集群。
- 容器组报表：展示某段时间内全部容器组的 CPU、内存总量、存储、GPU、总计等计费

情况，以及容器组的所属命名空间、所属集群和所属工作空间。

- 工作空间计费报表：展示某段时间内全部工作空间的 CPU、内存总量、存储、GPU、总计等计费情况，以及命名空间数量和容器组数量，可通过点击命名空间数量快捷进入命名空间计费报表，并查看该段时间内该工作空间下命名空间的计费情况；同样 的方式可查看该段时间内该工作空间下的容器组的计费情况。
- 命名空间计费报表：某段时间内全部命名空间的 CPU、内存总量、存储、GPU、总计等 计费情况，以及容器组数量、所属集群、所属工作空间，可通过点击容器组数量快 捷进入容器组计费报表，并查看该段时间内该命名空间下的容器组的计费情况。

操作步骤

1. 使用具有 `admin` 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理 -> 运营 管理**。

集群名称	节点数量	CPU 使用率 (%)			内存使用率 (%)			存储使用率	
		最大	最小	平均	最大	最小	平均	最大	最小
peter-centos7-single	1	9.8	8.7	9.14	47.46	44.04	45.46	40.31	38.84
skoala-dev	3	16.14	14.38	15.1	40.06	38.51	39.13	77.61	76.87
virtual-machine-cluster	3	7.21	6.3	6.52	15.97	15.86	15.92	46.7	46.48
yiting-gpu3	1	5.59	4.95	5.15	11.1	10.42	10.56	57.99	57.34

报表管理

2. 进入 **运营管理** 后切换不同菜单可查看集群、节点、容器组等计费报表。

安全策略

DCE 5.0 在图形界面上提供了基于密码和访问控制的安全策略。

密码策略

- 新密码不能与最近的历史密码相同。
- 密码过期后，系统强制要求修改密码。
- 密码不能与用户名相同。
- 密码不能和用户的邮箱地址相同。
- 自定义密码规则。
- 自定义密码最小长度。

访问控制策略

- 会话超时策略：用户在 x 小时内没有操作，退出当前账号。
- 账号锁定策略：限制时间内多次登录失败，账号将被锁定。
- 限制单个账号的最大并发会话连接数：为了防止单个用户占用过多的服务器资源，或者为了防止恶意用户通过创建大量并发连接来攻击系统，通过设定一个上限，系统可以确保每个用户只能建立一定数量的并发连接，从而保护服务器的稳定性和安全性。
注意：单个账号达到最大并发数后，必须手动登出其他设备才能登录新设备，可能会带来操作不便，请谨慎开启。（关闭页面不属于登出行行为，需手动单击“退出登录”）
- 限制系统的最大并发会话连接数：对整个系统（而非单个账号）在同一时间内能够处理

或允许的最大会话连接数量进行限制，从而保护系统资源不被过度消耗，确保系统的稳定性和性能

- 关闭浏览器的同时退出登录：开启后，通过浏览器打开新的标签页将导致用户信息丢失，需要重新登录，请谨慎操作。
- 双因子认证：（Two-Factor Authentication，简称 2FA）是一种安全验证过程，用于增强用户账户的安全性和可靠性。在这个过程中，提供用户名密码和动态口令两种不同的认证因素来证明自己的身份，从而增加攻击者访问用户设备和在线账户的难度。

进入全局管理后，在左侧导航栏点击 **平台设置 -> 安全策略**，即可设置密码策略和访问控制策略。



安全策略

The screenshot shows the 'Access Control' tab selected under 'Security Policies'. It includes sections for session timeout policies (e.g., 'User in 24 hours'), account lockout policies (e.g., 'Limiting multiple failed logins within a time period'), and multi-session policies. There are also sections for 'Logout/Logout Policy' and 'Two-factor Authentication'. Each section has a 'Cancel' and 'Save' button.

访问控制

双因子认证

开启双因子认证后，用户在登录时需要进行二次验证。

第一次登录

1. 开启双因子认证后，用户第一次登录时需要在手机等终端设备安装支持 2FA 动态口令生成的应用，如 Google Authenticator 等。

16:10

4G 86%

 Google Authenticator

搜索...



ghippo: DaoCloud双因子认证

上海

970 4

输入设置密钥

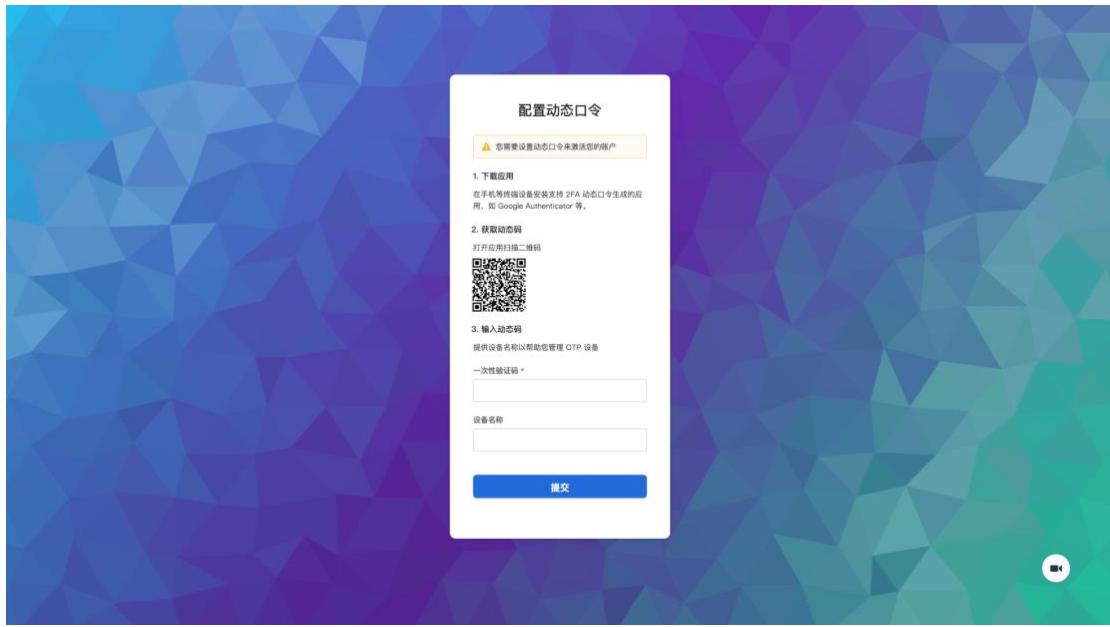


扫描二维码



2fa

2. 打开应用扫描二维码

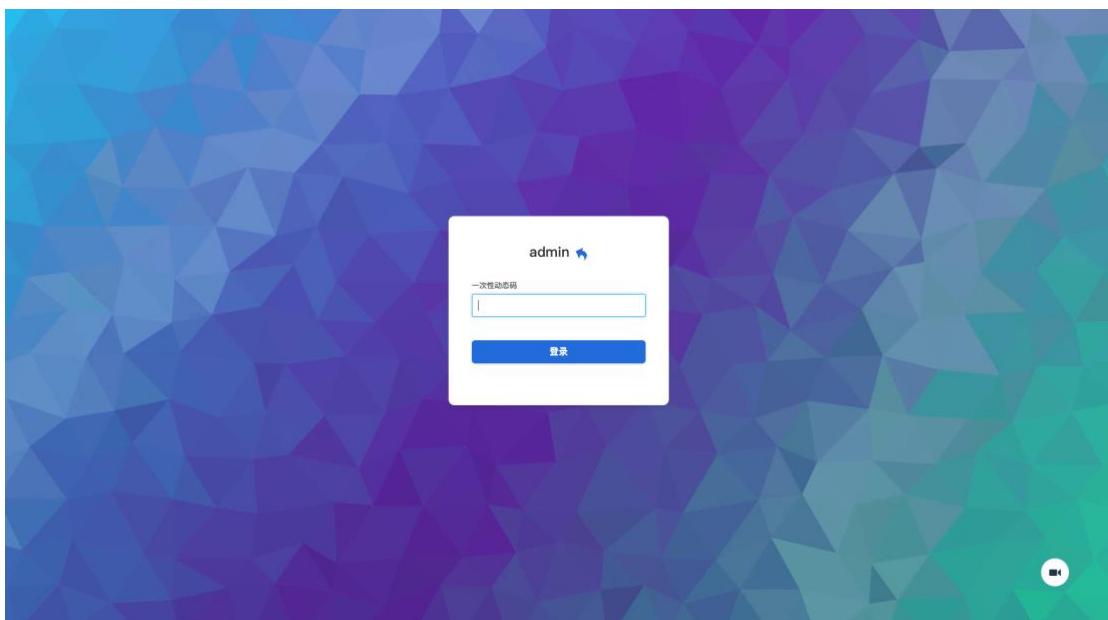


2fa

3. 输入动态码，点击 **提交** 后登录成功。

第二次登录

进入登录页面，输入用户名密码后，打开 Google Authenticator 等动态口令生成应用，输入动态口令，点击登录后登录成功。



2fa

邮件服务器

DCE 5.0 会在用户忘记密码时，向用户发送电子邮件以验证电子邮件地址，确保用户是本人操作。要使 DCE 5.0 能够发送电子邮件，需要先提供您的邮件服务器地址。具体操作步骤如下：

1. 使用具有 **admin** 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理**。

全局管理

全局管理

2. 点击 **平台设置**，选择 **邮件服务器设置**。

邮件服务器

邮件服务器

填写以下字段配置邮件服务器：

字段	描述	举例值
SMTP 服务器	能够提供邮件服务的 SMTP 服务器地址	smtp.163.com
地址		
SMTP 服务器	发送邮件的端口	25
端口		
用户名	SMTP 用户的名称	test@163.com
密码	SMTP 账号的密码	123456
发件人邮箱	发件人的邮箱地址	test@163.com
使用 SSL 安全连接	SSL 可以用于加密邮件，从而提高通过邮件传输的信息的安全性，通常需为邮件服务器配置证书	不开启

3. 配置完成后点击 **保存**，点击 **测试邮件服务器**。

测试

测试

4. 屏幕右上角出现成功发送邮件的提示，则表示邮件服务器被成功设置。

成功

成功

常见问题

问：邮件服务器设置后用户仍无法找回密码是什么原因？

答：用户可能未设置邮箱或者设置了错误的邮箱地址；此时可以让 admin 角色的用户在 **全局管理 -> 用户与访问控制** 中通过用户名找到该用户，并在用户详情中为该用户设置新的登录密码。

如果邮件服务器没有连通，请检查邮件服务器地址、用户名及密码是否正确。

外观定制

在 DCE 5.0 中，可通过 **外观定制** 更换登录界面、顶部导航栏以及底部版权和备案信息，帮助用户更好地辨识产品。

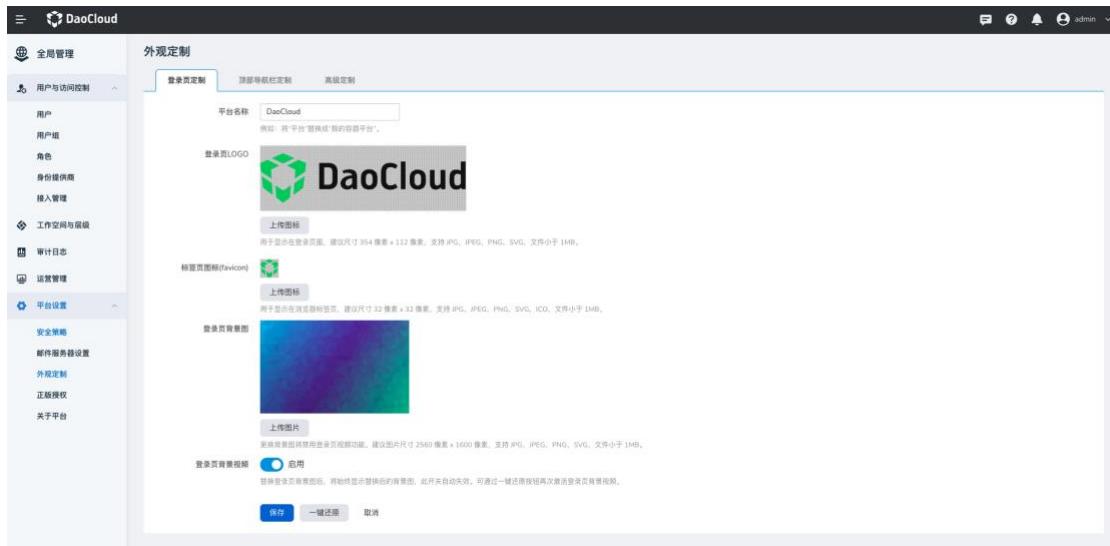
定制说明

1. 使用具有 **admin** 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理 -> 平台设置**。

全局管理

全局管理

2. 选择 **外观定制** ,在 **登录页定制** 页签中, 修改登录页的图标和文字后, 点击 **保存** 。



外观定制

3. 退出登录, 在登录页刷新后可看到配置后的效果

登录页

登录页

4. 点击 **顶部导航栏定制** 页签, 修改导航栏的图标和文字后, 点击 **保存** 。

顶部导航栏

顶部导航栏

5. 点击 **高级定制** , 可以用 CSS 样式设置登录页、导航栏、底部版权及备案信息。

高级定制

高级定制

高级定制

高级定制能够通过 CSS 样式来修改整个容器平台的颜色、字体间隔、字号等。 您需要熟悉

CSS 语法。删除黑色输入框的内容，可恢复到默认状态，当然也可以点击 **一键还原** 按钮。

登录页定制的 CSS 样例：

```
.test {
    width: 12px;
}

#kc-login {
    /* color: red!important; */
}
```

登录后页面定制的 CSS 样例：

```
.dao-icon.dao-iconfont.icon-service-global.dao-nav__head-icon {
    color: red!important;
}

.gippo-header-logo {
    background-color: green!important;
}

.gippo-header {
    background-color: rgb(128, 115, 0)!important;
}

.gippo-header-nav-main {
    background-color: rgb(0, 19, 128)!important;
}

.gippo-header-sub-nav-main .dao-popper-inner {
    background-color: rgb(231, 82, 13) !important;
}
```

Footer（页面底部的版权、备案等信息）定制示例

```
<div class="footer-content">
    <span class="footer-item">Copyright © DaoCloud 道客网络科技有限公司保留所有权利</span>
    <a class="footer-item" href="https://beian.miit.gov.cn/" target="_blank" rel="noopener noreferrer">沪 ICP 备 14048409 号 - 1</a>
    <a class="footer-item" href="https://beian.miit.gov.cn/" target="_blank" rel="noopener noreferrer">沪 ICP 备 14048409 号 - 2</a>
</div>
<div class="footer-content">
    

<a class="footer-item" href="http://www.beian.gov.cn/portal/registerSystemInfo">沪公网安备  
**12345678912345 号</a>**

</div>

<style>

```
.footer-content {
 display: flex;
 flex-wrap: wrap;
 align-items: center;
 justify-content: center;
}
.footer-content + .footer-content {
 margin-top: 8px;
}
.login-pf .footer-item {
 color: white;
}
.footer-item {
 color: var(--dao-gray-010);
 text-decoration: none;
}
.footer-item + .footer-item {
 margin-left: 8px;
}
.gongan-icon {
```

```

width: 18px;
height: 18px;
margin-right: 4px;
}
</style>
!!! note
如果想要恢复默认设置,可以点击 一键还原。请注意,一键还原后将丢弃所有自定义设置。

```

# 关于平台

**关于平台** 主要呈现平台各个子模块当前更新的版本，声明了平台使用的各个开源软件，并以动画视频的方式致谢了平台的技术团队。

查看步骤：

1. 使用具有 **Admin** 角色的用户登录 DCE 5.0。点击左侧导航栏底部的 **全局管理**。

## 全局管理

### 全局管理

2. 点击 **平台设置**，选择 **关于平台**，查看产品版本、开源软件声明和技术团队。

The screenshot shows the 'Global Management' dashboard. On the left, there is a sidebar with various management sections like 'User and Access Control', 'Space and Hierarchical Management', 'Audit Log', 'Operations Management', and 'Platform Settings'. Under 'Platform Settings', the 'About Platform' option is selected. The main content area is titled 'About Platform' and contains a table showing product versions for different modules. The table has two columns: 'Module Name' and 'Current Version'.

| 模块名称          | 当前版本                     |
|---------------|--------------------------|
| 应用工作台         | v0.20-dev-5b00b66c       |
| 虚拟机           | v0.7.0-dev1-8-gc6838aea  |
| 集群巡检          | v0.7.0-dev-1d6fb7ca      |
| 安全管理          | v0.8.0-dev-04da1954      |
| 容器管理          | v0.26.0-dev-aaa1f388     |
| 多云编排          | v0.17.0-dev-760a801a     |
| 镜像仓库          | v0.15.0-dev-d0ae76be     |
| 可观测性          | v0.24.0-rc3-24-g98e51c9b |
| 微服务引擎         | v0.34.0-1-g8d225df       |
| 服务网格          | v0.24-dev-9105bace       |
| RocketMQ 消息队列 | 0.0.1-237-g9b77bc6       |
| MongoDB 数据库   | 0.0.1-319-g10a9e09       |

### 关于平台

#### License 声明

## license 声明

license 声明

**技术团队**

技术团队

技术团队

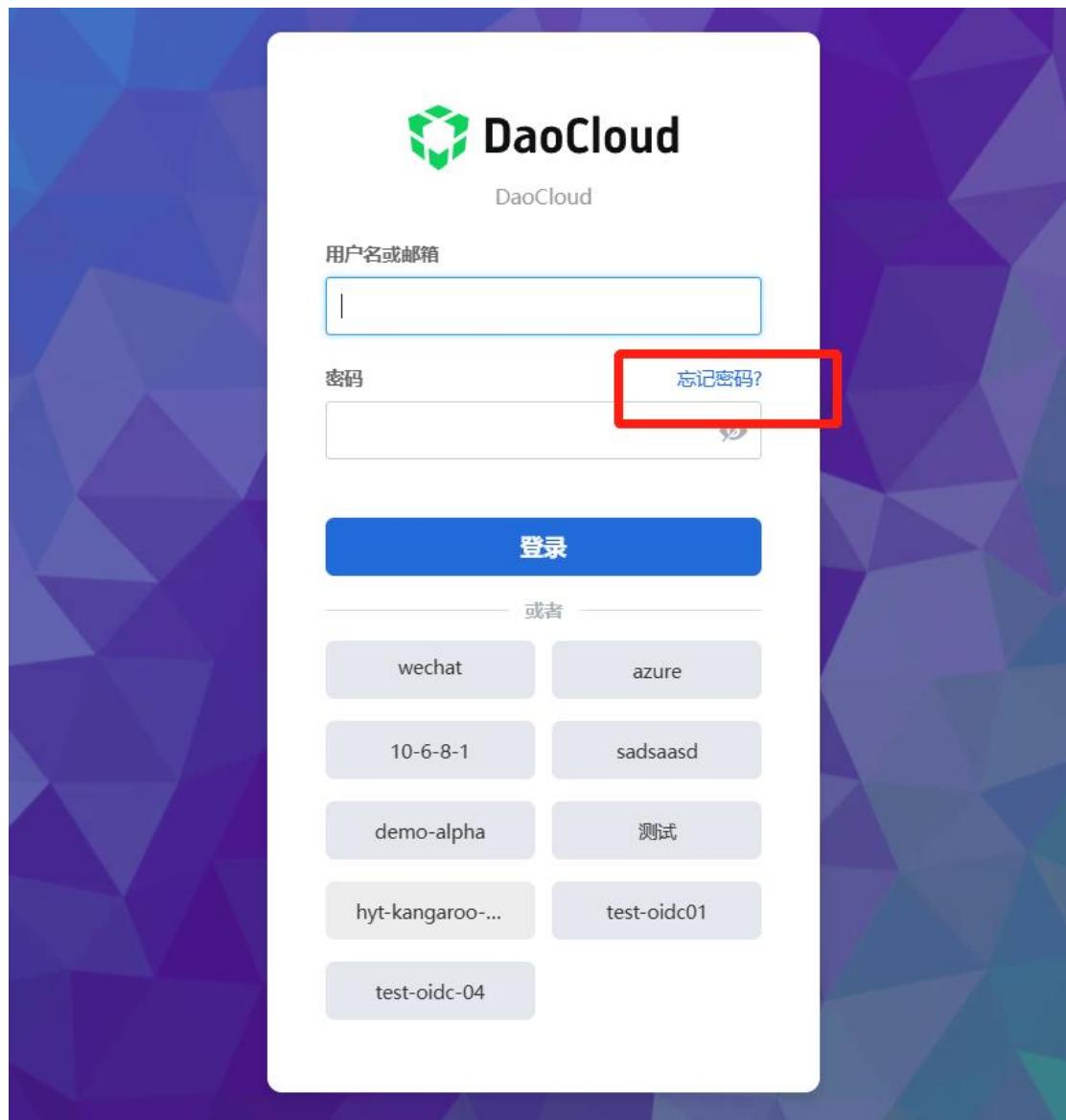
# 密码重置

如果您忘记密码，可以按本页面说明重置密码。

## 重置密码步骤

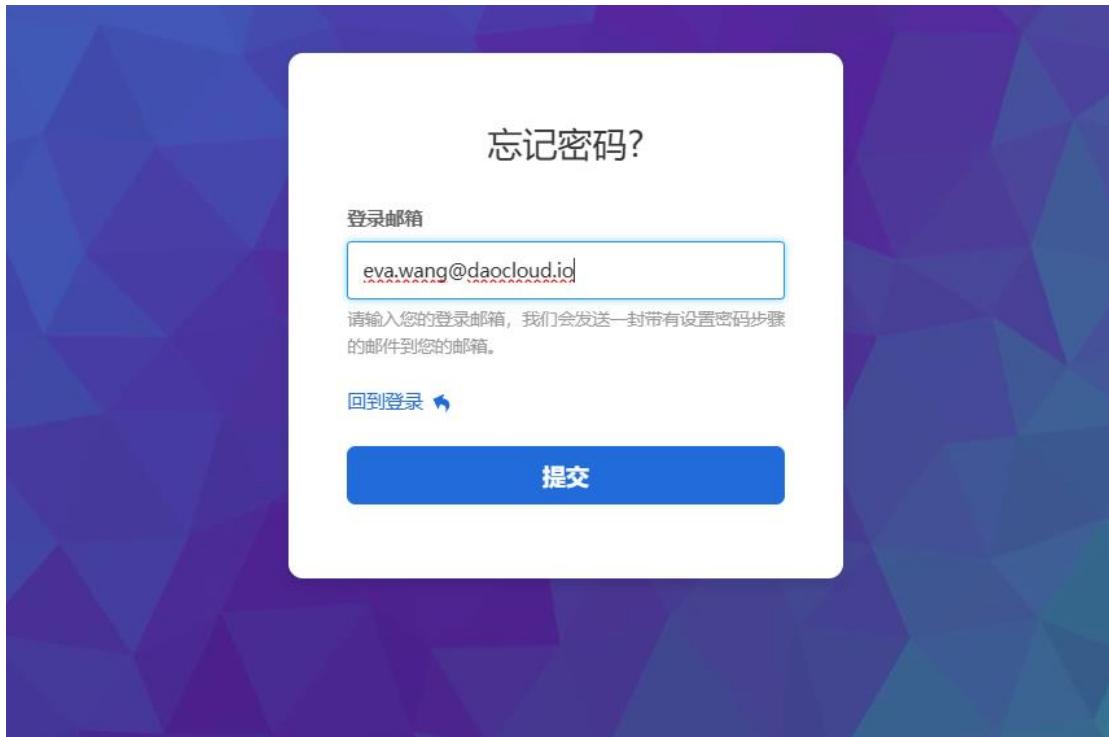
管理员最初创建一个用户时，会为其设置用户名和密码。 该用户登录后，在 **个人中心** 填写邮箱并修改密码。 若该用户未设置邮箱，则只能联系管理员进行密码重置。

1. 如果用户忘记了密码，可以在登录界面点击 **忘记密码**。



登录界面

2. 输入登录邮箱，点击 **提交**。



输入登录邮箱

3. 在邮箱中找到密码重置邮件，点击下方链接进行密码重置，链接时效 5 分钟。

hexiaodai123@163.com  
To: Eva Wang  
Tue 2024-07-09 9:57

有用户要求修改账户 Ghippo 的密码如是本人操作，请点击下面链接进行重置。  
[这个链接会在 5 分钟后过期](https://demo-dev.daocloud.io/auth/realms/ghippo/login-actions/action-token?key=eyJhbGciOiJUzI1NiIsInR5cClgiAiSlDUiwi2IkliA6ICjMyWzIMGE2YS0vYmU5LTrmMDUtOTfjOS1mMGM4MzgxYWl1NDMifQeyJleHAiQiE3MjA0OTA1MjMsImhdCl6MTcyMDQ5MDIyMywianRpjjoYfjM2U3ZGMtNDUwNC00ZTjlTg1ZTEiZDNmNTBkODC1ZGJmlwiwaXNzIjoiaHR0cHM6Ly9kZW1vLWRldi5kYW9jbG91ZC5pb9hdXR0l3JlYWxty9naGlwcG8iLChdWoijodHRwczoVl2RlbW8tZGV2LmRh2NsB3VklmlvL2F1dGgycmVhbG1zLdoaX8wbyslnN1yl6jBlNmESMzkzLTU1ODYtNDU4MC05NGI2lTE0jI4OWYzOTk4YiislnR5cCl6jnJlc2V0LNyZWRlbnRpYWxzlwYXpwjjoIX19pbnRlc5hbC1naGlwcG8iLCJub25jZSI6ImixYzNIN2RjLTQ1MDQtNGUyY04NWUxLWQzzJuwZDg3NWRiZlslmVtbCl6lmV2YS53YW5nQGRhb2Nsb3VklmlvIwYXNpZC16lmY5YzQ2Nzg5LTThzZWQtnDhjZS04YTAzLTkwMglyMmU3NGZhZC5wT25lWUplMEZBRSSjMny1N2MxNy0xMzQSLTRkMDUTODJmZS04NWQyNzg2OTk2YzciLChc2IkjoiZjijNDY3ODktOGFIzC00OGNIThhMDMtOTAwYjlyZTc0ZmFkln8Pbk7ZSmUwRkFFLmMyZjU3YfE3lTEzNDktNGQwNS04MmzIlIg1ZD3ODY5OTjNy9.PWteLshNpD5Yi-dqPNSDXyHTvKjbWpMH-i_zIMzPucs&execution=9ef80c59-ec81-47ba-ab4d-8da0795c5a82&client_id=_internal-ghippo&tab_id=pOnKYje0FAE)  
 如果您不想重置您的密码，请忽略这条消息，密码不会改变。

[Reply](#) [Forward](#)

点击重置链接

4. 在手机等终端设备安装支持 2FA 动态口令生成的应用（如 Google Authenticator），按照页面提示配置动态口令以激活账户，点击 **提交**。



## 配置动态口令

5. 设置新密码，点击 **提交**。设置新密码的要求与创建用户时的密码规则一致。

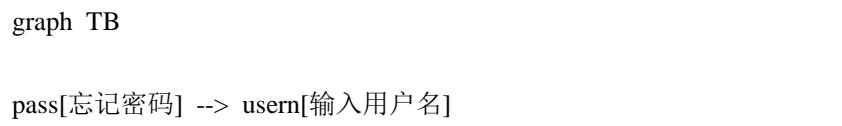


更新密码

6. 修改密码成功，直接跳转首页。

## 重置密码流程

整个密码重置的流程示意图如下。



judge1 -. 正确 .-> judge2[判断是否绑定邮箱]

judge1 -. 错误 .-> tip1[提示用户名不正确]

```
judge2 -.已绑定邮箱.-> send[发送重置邮件]
judge2 -.未绑定邮箱.-> tip2[提示未绑定邮箱
联系管理员重置密码]

send --> click[点击邮件中的链接] --> config[配置动态口令] --> reset[重置密码]
- -> success[成功重置密码]

classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000;
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:1px,color:#fff;
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;

class pass,usern,button,tip1,send,tip2,send,click,config,reset,success plain;
class judge1,judge2 k8s
```

# 安全设置

功能说明：用于填写邮箱地址和修改登录密码。

- 邮箱：当管理员配置邮箱服务器地址之后，用户能够通过登录页的忘记密码按钮，填写该处的邮箱地址以找回密码。
- 密码：用于登录平台的密码，建议定期修改密码。

具体操作步骤如下：

1. 点击右上角的用户名位置，选择 **个人中心**。

个人中心

个人中心

2. 点击 **安全设置** 页签。填写您的邮箱地址或修改登录密码。

安全设置

安全设置

# 访问密钥

访问密钥（Access Key）可用于访问开放 API 和持续发布，用户可在个人中心参照以下步骤

获取密钥并访问 API。

## 获取密钥

登录 DCE 5.0，在右上角的下拉菜单中找到 **个人中心**，可以在 **访问密钥** 页面管理账号的访问密钥。

```
ak list
```

```
ak list
```

```
created a key
```

```
created a key
```

```
!!! info
```

访问密钥信息仅显示一次。如果您忘记了访问密钥信息，您需要重新创建新的访问密钥。

## 使用密钥访问 API

在访问 DCE 5.0 openAPI 时，在请求中加上请求头 `Authorization: Bearer ${token}` 以标识访问者的身份，其中 `${token}` 是上一步中获取到的密钥，具体接口信息参见 [OpenAPI 接口文档](#)。

### 请求示例

```
curl -X GET -H 'Authorization: Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6IkRKVjlBTHRBLXZ4MmtQUC1TQnVGS0dCSWe1cnBfdkxiQVVqM2U3RVByWnMiLCJ0eXAiOiJKV1QifQ.eyJleHAiOjE2NjE0MTU5NjksImlhdCI6MTY2MDgxMTE2OSwiaXNzIjoiZ2hpcHBvLmlvIiwick3ViIjoiZjdjOGIxZjUtMTc2MS00NjYwLTg2MWQtOWI3MmI0MzMjNGViIiwickHJIzZmVycmVkX3VzZXJuYW1lIjoiYWRtaW4iLCJncm91cHMiOltdfQ.RsUcrAYkQQ7C6BxMOrdD3qbBRUt0VVxynIGeq4wyIgye6R8Ma4cjxG5CbU1WyiHKpvIKJDbeFQHro2euQyVde3ygA672ozkwLTnx3Tu-_mB1BubvWCBsDdUjhCQfT39rk6EQozMjb-1X1sbLwzkfzKMls-oxkjagI_RFrYITVPwT3Oaw-qOyulRSw7Dxd7jb0vINPq84vmlQIsI3UuTZSNO5BCgHpubcWwBss-Aon_DmYA-Et_-QtmPBA3k8E2hzDSzc7eqK0I68P25r9rwQ3DeKwD1dbRyndqWORRnz8TLEXSiCFXdZT2oiMrcJtO188Ph4eLGut1-4PzKhwgrQ' https://demo-dev.daocloud.io/apis/ghippo.io/v1alpha1/users?page=1&pageSize=10 -k
```

## 请求结果

```
{
 "items": [
 {
 "id": "a7cf010-ebbe-4601-987f-d098d9ef766e",
 "name": "a",
 "email": "",
 "description": "",
 "firstname": "",
 "lastname": "",
 "source": "locale",
 "enabled": true,
 "createdAt": "1660632794800",
 "updatedAt": "0",
 "lastLoginAt": ""
 }
,
 "pagination": {
 "page": 1,
 "pageSize": 10,
 "total": 1
 }
}
```

# 配置 SSH 公钥

本文说明如何配置 SSH 公钥。

## 步骤 1：查看已存在的 SSH 密钥

在生成新的 SSH 密钥前，请先确认是否需要使用本地已生成的 SSH 密钥，SSH 密钥对一般存放在本地用户的根目录下。Linux、Mac 请直接使用以下命令查看已存在的公钥，Windows 用户在 WSL (需要 Windows 10 或以上) 或 Git Bash 下使用以下命令查看已生成的公钥。

- **ED25519 算法：**

```
cat ~/.ssh/id_ed25519.pub
```

- RSA 算法：

```
cat ~/.ssh/id_rsa.pub
```

如果返回一长串以 ssh-ed25519 或 ssh-rsa 开头的字符串，说明已存在本地公钥，您可以跳过步骤 2 生成 SSH 密钥，直接操作步骤 3。

## 步骤 2：生成 SSH 密钥

若步骤 1 未返回指定的内容字符串，表示本地暂无可用 SSH 密钥，需要生成新的 SSH 密钥，请按如下步骤操作：

1. 访问终端（Windows 请使用 [WSL](#) 或 [Git Bash](#)），运行 ssh-keygen -t。
2. 输入密钥算法类型和可选的注释。

注释会出现在 .pub 文件中，一般可使用邮箱作为注释内容。

- 基于 ED25519 算法，生成密钥对命令如下：

```
ssh-keygen -t ed25519 -C "<注释内容>"
```

- 基于 RSA 算法，生成密钥对命令如下：

```
ssh-keygen -t rsa -C "<注释内容>"
```

3. 点击回车，选择 SSH 密钥生成路径。

以 ED25519 算法为例，默认路径如下：

```
Generating public/private ed25519 key pair.
```

```
Enter file in which to save the key (/home/user/.ssh/id_ed25519):
```

密钥默认生成路径：/home/user/.ssh/id\_ed25519，公钥与之对应为：

```
/home/user/.ssh/id_ed25519.pub。
```

4. 设置一个密钥口令。

```
Enter passphrase (empty for no passphrase):
```

```
Enter same passphrase again:
```

口令默认为空，您可以选择使用口令保护私钥文件。如果您不想在每次使用 SSH 协议

访问仓库时，都要输入用于保护私钥文件的口令，可以在创建密钥时，输入空口令。

5. 点击回车，完成密钥对创建。

## 步骤 3：拷贝公钥

除了在命令行打印出已生成的公钥信息手动复制外，可以使用命令拷贝公钥到粘贴板下，请参考操作系统使用以下命令进行拷贝。

- Windows (在 [WSL](#) 或 [Git Bash](#) 下) :

```
cat ~/.ssh/id_ed25519.pub | clip
```

- Mac:

```
tr -d '\n' < ~/.ssh/id_ed25519.pub | pbcopy
```

- GNU/Linux (requires xclip):

```
xclip -sel clip < ~/.ssh/id_ed25519.pub
```

## 步骤 4：在 DCE 5.0 平台上设置公钥

1. 登录 DCE 5.0 UI 页面，在页面右上角选择 **个人中心 -> SSH 公钥**。

2. 添加生成的 SSH 公钥信息。

1. SSH 公钥内容。

2. 公钥标题：支持自定义公钥名称，用于区分管理。

3. 过期时间：设置公钥过期时间，到期后公钥将自动失效，不可使用；如果不设置，则永久有效。

## 语言设置

本节说明如何设置界面语言。目前支持中文、English 两个语言。

语言设置是平台提供多语言服务的入口，平台默认显示为中文，用户可根据需要选择英语或

自动检测浏览器语言首选项的方式来切换平台语言。每个用户的多语言服务是相互独立的，

切换后不会影响其他用户。

平台提供三种切换语言方式：中文、英语-English、自动检测您的浏览器语言首选项。

操作步骤如下。

1. 使用您的用户名/密码登录 DCE 5.0。点击左侧导航栏底部的 **全局管理**。

全局管理

全局管理

2. 点击右上角的用户名位置，选择 **个人中心**。

个人中心

个人中心

3. 点击 **语言设置** 页签。

语言设置

语言设置

4. 切换语言选项。

切换语言

切换语言

## 容器管理权限说明

容器管理模块使用以下角色：

- Admin / Kpanda Owner
- [Cluster Admin](#)
- [NS Admin](#)
- [NS Editor](#)
- [NS Viewer](#)

**!!! note**

- 有关权限的更多信息，请参阅[容器管理权限体系说明](../../kpanda/user-guide/permissions/permission-brief.md)。
- 有关角色的创建、管理和删除，请参阅[角色和权限管理](./user-guide/access-control/role.md)。
- Cluster Admin，NS Admin，NS Editor，NS Viewer 的权限仅在当前的集群或命名空间内生效。

各角色所具备的权限如下：

```
preapreens[准备命名空间] - -> judge([命名空间是否与绑定到其他工作空间]) judge -.未绑定->nstows[将命名空间绑定到工作空间] -> wsperm[管理工作空间访问权限] judge -.已绑定.->createns[创建新的命名空间]
```

```
classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000; classDef k8s fill:#326ce5,stroke:#fff,stroke-width:1px,color:#fff; classDef cluster fill:#fff,stroke:#bbb,stroke-width:1px,color:#326ce5;
class preparews,preapreens,createns,nstows,wsperm cluster; class judge plain
click preparews "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-to-ns/#_3" click preapreens "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-to-ns/#_4" click nstows "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-to-ns/#_5" click wsperm "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-to-ns/#_6" click createns "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-to-ns/#_4"
```

**!!! tip**

一个命名空间只能被一个工作空间绑定。

**## 准备工作空间**

工作空间是为了满足多租户的使用场景，基于集群、集群命名空间、网格、网格命名空间、多云、多云命名空间等多种资源形成相互隔离的资源环境。

工作空间可以映射为项目、租户、企业、供应商等多种概念。

1. 使用 admin/folder admin 角色的用户登录 DCE 5.0，点击左侧导航栏底部的 全局管理。

![全局管理](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/ws01.png)

1. 点击左侧导航栏的 工作空间与层级，点击右上角的 创建工作空间 按钮。

![创建工作空间](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/ws02.png)

1. 填写工作空间名称、所属文件夹等信息后，点击 确定，完成创建工作空间。

![确定](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/ws03.png)

提示：若平台中已存在创建好的命名空间，点击某个工作空间，在 资源组 页签下，点击 绑定资源，可以直接绑定命名空间。

![弹出菜单绑定](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/across02.png)

## ## 准备命名空间

命名空间是更小的资源隔离单元，将其绑定到工作空间后，工作空间的成员就可以进行管理和使用。

参照以下步骤准备一个还未绑定到任何工作空间的命名空间。

1. 点击左侧导航栏底部的 容器管理。

![容器管理](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/crd00.png)

1. 点击目标集群的名称，进入 集群详情。

![集群详情](https://docs.daocloud.io/daocloud-docs-images/docs/kpanda/images/crd01.png)

1. 在左侧导航栏点击 命名空间，进入命名空间管理页面，点击页面右侧的 创建 按钮。

![创建命名空间](https://docs.daocloud.io/daocloud-docs-images/docs/kpanda/images/ns01.png)

1. 填写命名空间的名称，配置工作空间和标签（可选设置），然后点击 确定。

!!! info

工作空间主要用于划分资源组并为用户（用户组）授予对该资源的不同访问权限。有关工作空间的详细说明，可参考[工作空间与层级](./user-guide/workspace/workspace.md)。

![填写](https://docs.daocloud.io/daocloud-docs-images/docs/kpanda/images/ns02.png)

1. 点击 确定，完成命名空间的创建。在命名空间列表右侧，点击 ⋮，可以从弹出菜单中选择 绑定工作空间。

![确定和绑定](https://docs.daocloud.io/daocloud-docs-images/docs/kpanda/images/ns03.png)

## ## 将命名空间绑定到工作空间

除了在命名空间列表中绑定外，也可以返回 全局管理，按照以下步骤绑定工作空间。

- 依次点击 全局管理 -> 工作空间与层级 -> 资源组，点击某个工作空间名称后，点击 绑定资源 按钮。

![绑定资源](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/bind01.png)

- 选中要绑定的工作空间（可多选），点击 确定 完成绑定。

![确定](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/bind02.png)

## 为工作空间添加成员并授权

- 在 工作空间与层级 -> 授权 中，点击某个工作空间名称后，点击 添加授权 按钮。

![添加授权](https://docs.daocloud.io/daocloud-docs-images/docs/ghippo/images/wsauth01.png)

- 选择要授权的 用户/用户组、角色 后，点击 确定 完成授权。

![确定](docs/zh/docs/images/wsauth02.png)

# 将集群分配给多个工作空间（租户）

集群资源通常由运维人员进行管理。在分配资源分配时，他们需要创建命名空间来隔离资源，并设置资源配额。

这种方式有个弊端，如果企业的业务量很大，手动分配资源需要较大的工作量，而想要灵活调配资源额度也有不小难度。

DCE 为此引入了工作空间的概念。工作空间通过共享资源可以提供更高维度的资源限额能力，实现工作空间（租户）在资源限额下自助式创建 Kubernetes 命名空间的能力。

举例而言，如果想要让几个部门共享不同的集群。

|           | Cluster01（普通） | Cluster02（高可用） |  |
|-----------|---------------|----------------|--|
| 部门（工作空间）A | 50 quota      | 10 quota       |  |
| 部门（工作空间）B | 100 quota     | 20 quota       |  |

可以参照以下流程将集群分享给多个部门/工作空间/租户：

```
```mermaid
graph TB
    preparews[准备工作空间] --> preparecs[准备集群]
```

```
- -> share[将集群共享到工作空间]
- -> judge([判断工作空间剩余额度])
judge -.大于剩余额度.->modifyns[修改命名空间额度]
judge -.小于剩余额度.->createns[创建命名空间]

classDef plain fill: #ddd,stroke:#fff,stroke-width:1px,color:#000;
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:1px,color:#fff;
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;

class preparews,preparecs,share, cluster;
class judge plain
class modifyns,createns k8s

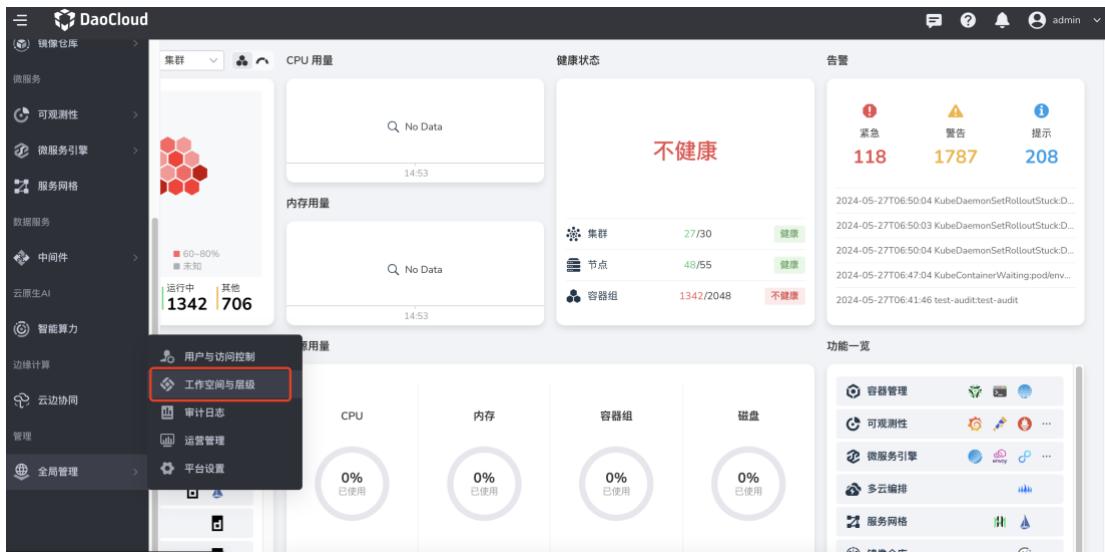
click preparews "https://docs.daocloud.io/ghippo/user-guide/workspace/cluster-for-multiws/#_2"
click preparecs "https://docs.daocloud.io/ghippo/user-guide/workspace/cluster-for-multiws/#_3"
click share "https://docs.daocloud.io/ghippo/user-guide/workspace/cluster-for-multiws/#_4"
click createns "https://docs.daocloud.io/amamba/user-guide/namespace/namespace/#_3"
click modifyns "https://docs.daocloud.io/amamba/user-guide/namespace/namespace/#_4"
```

准备一个工作空间

工作空间是为了满足多租户的使用场景，基于集群、集群命名空间、网格、网格命名空间、多云、多云命名空间等多种资源形成相互隔离的资源环境，工作空间可以映射为项目、租户、企业、供应商等多种概念。

1. 使用 admin/folder admin 角色的用户登录 DCE 5.0，点击左侧导航栏底部的 **全局管理**

-> **工作空间与层级**。



全局管理

2. 点击右上角的 **创建工作空间** 按钮。

创建工作空间

创建工作空间

3. 填写工作空间名称、所属文件夹等信息后，点击 **确定**，完成创建工作空间。

确定

确定

准备一个集群

工作空间是为了满足多租户的使用场景，基于集群、集群命名空间、网格、网格命名空间、多云、多云命名空间等多种资源形成相互隔离的资源环境，工作空间可以映射为项目、租户、企业、供应商等多种概念。

参照以下步骤准备一个集群。

1. 点击左侧导航栏底部的 **容器管理**，选择 **集群列表**。

容器管理

容器管理

2. 点击 **创建集群** [创建一个集群](#)，或点击 **接入集群** [接入一个集群](#)。

在工作空间添加集群

返回 **全局管理**，为工作空间添加集群。

1. 依次点击 **全局管理** -> **工作空间与层级** -> **共享资源**，点击某个工作空间名称后，点击 **新增共享资源** 按钮。

新增资源

新增资源

2. 选择集群，填写资源限额后，点击 **确定**。

新增资源

新增资源

下一步：将集群资源分配给多个工作空间后，用户可以前往 **应用工作台** 在这些工作空间下[创建命名空间并部署应用](#)。

文件夹最佳实践

文件夹代表一个组织机构（例如一个部门），是资源层次结构中的一个节点。

一个文件夹可以包含工作空间、子文件夹或两者的组合。它提供了身份管理、多层级和权限映射能力，能够将用户/用户组在文件夹中的角色映射到其下的子文件夹、工作空间和资源上。因此借助于文件夹，企业管理者能够集中管控所有资源。

1. 构建企业层级关系

首先要按照现有的企业层级结构，构建与企业相同的文件夹层级。DCE 支持 5 级文

件夹，可以根据企业实际情况自由组合，将文件夹和工作空间映射为企业中的部门、项目、供应商等实体。

文件夹不直接与资源挂钩，而是通过工作空间间接实现资源分组。

层级结构

层级结构

2. 用户身份管理

文件夹提供了 Folder Admin、Folder Editor、Folder Viewer 三种角色。[查看角色权限](#)，可通过[授权](#)给同一文件夹中的用户/用户组授予不同的角色。

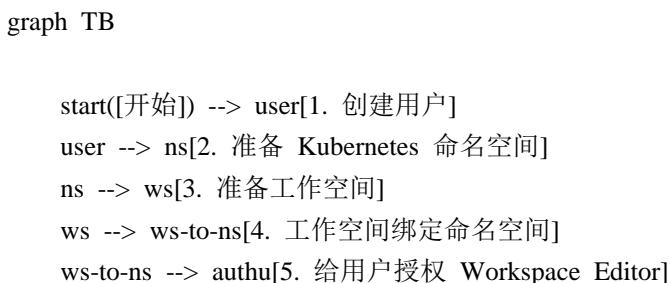
3. 角色权限映射

- 企业管理者：在根文件夹授予 Folder Admin 角色。他将拥有所有部门、项目及其资源的管理权限。
- 部门管理者：在各个子文件夹、工作空间单独授予管理权限。
- 项目成员：在工作空间、资源层级单独授予管理权限。

普通用户授权规划

普通用户是指能够使用 DCE 大部分产品模块及功能（管理功能除外），对权限范围内的资源有一定的操作权限，能够独立使用资源部署应用。

对这类用户的授权及资源规划流程如下图所示。



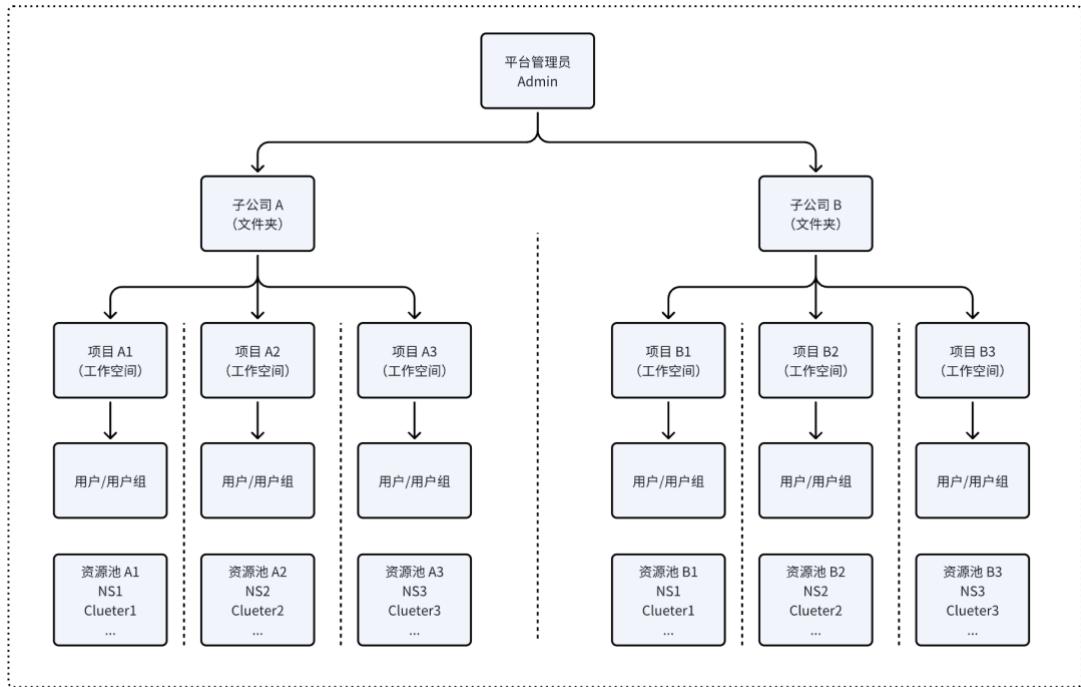
```
authu --> complete([结束])  
  
click user "https://docs.daocloud.io/ghippo/user-guide/access-control/user/"  
click ns "https://docs.daocloud.io/kpanda/user-guide/namespaces/createns/"  
click ws "https://docs.daocloud.io/ghippo/user-guide/workspace/workspace/"  
click ws-to-ns "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-to-ns/"  
click authu "https://docs.daocloud.io/ghippo/user-guide/workspace/ws-permission/"  
  
classDef plain fill: #ddd,stroke:#fff,stroke-width:4px,color:#000;  
classDef k8s fill: #326ce5,stroke:#fff,stroke-width:4px,color:#fff;  
classDef cluster fill: #fff,stroke:#bbb,stroke-width:1px,color:#326ce5;  
class user,ns,ws,ws-to-ns,authu cluster;  
class start,complete plain;
```

授权后普通用户在各模块的权限为：

- [应用工作台](#)
- [微服务引擎](#)
- [服务网格](#) (需要准备网格/网格命名空间并绑定到工作空间)
- [可观测性](#)
- [数据服务](#)

超大型企业的架构管理

伴随业务的持续扩张，公司规模不断壮大，子公司、分公司纷纷设立，有的子公司还进一步设立孙公司，原先的大部门也逐渐细分成多个小部门，从而使得组织结构的层级日益增多。这种组织结构的变化，也对 IT 治理架构产生了影响。



architecture

具体操作步骤如下：

1. 开启 Folder/WS 之间的隔离模式

请参考[开启 Folder/WS 之间的隔离模式](#)。

2. 按照实际情况规划企业架构

在多层级组织架构下，建议将二级文件夹作为隔离单元，进行“子公司”之间的用户/用

户组/资源之间的隔离。 隔离后“子公司”之间的用户/用户组/资源互不可见。

The screenshot shows the DaoCloud Enterprise 5.0 web interface. On the left, there's a sidebar with navigation items like '全局管理', '用户与访问控制', '用户', '用户组', '角色', etc. The main area is titled '工作空间与层级' (Space and Hierarchy). It shows a tree structure with 'Root', '子公司1', '子公司2' (which is expanded to show '研发部门', '产品部门', '测试部门'), and '子公司3'. To the right of this tree, there's a '授权' (Authorization) panel. This panel has a search bar for '请输入授权主体搜索' (Search for authorization subject), a checkbox for '授权主体' (Authorization subject), a list item '小明' (Xiaoming) with a 'Folder Admin' role assigned, and a button '添加授权' (Add Authorization). Below this panel, it says '共 1 项' (1 item). At the top of the page, there's a header with the DaoCloud logo and a user profile for 'admin'.

WS

3. 创建用户/打通用户体系

由主平台管理员 Admin 在平台统一[创建用户](#)或通过 LDAP/OIDC/OAuth2.0 等[身份提供商](#)将用户统一一对接到 DCE 5.0。

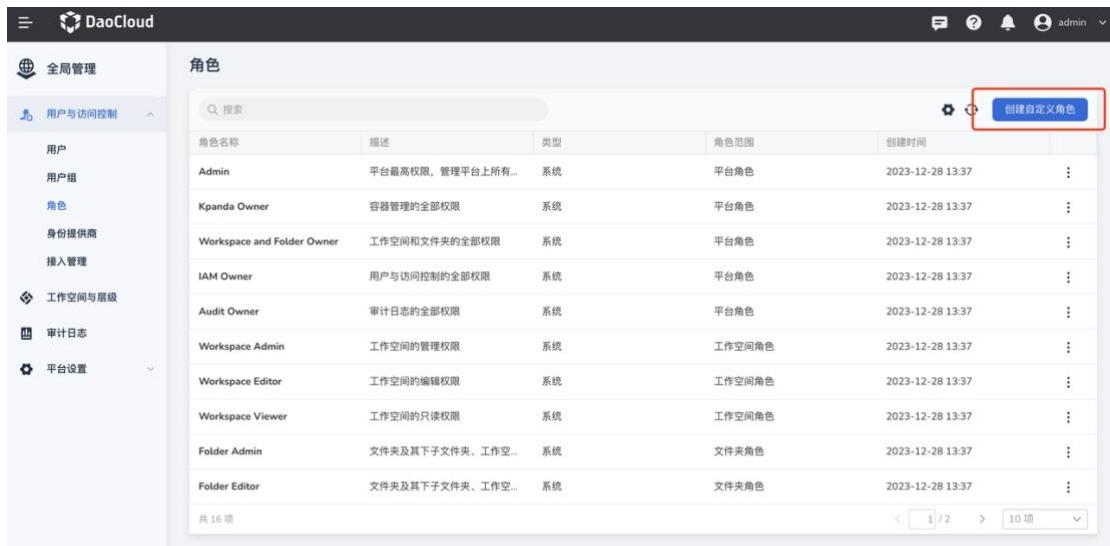
4. 创建文件夹角色

在 Folder/WS 的隔离模式下, 需要平台管理员 Admin 通过[授权](#)首先将用户邀请到各个子公司, “子公司管理员 (Folder Admin)”才能够对这些用户进行管理, 如二次授权或者编辑权限。建议简化平台管理员 Admin 的管理工作, 创建一个无实际权限的角色来辅助平台管理员 Admin 实现通过“授权”将用户邀请到子公司的操作。而子公司用户的实际权限下放到各个子公司管理员 (Folder Admin) 自行管理。

!!! note

资源绑定权限点单独使用不生效, 因此符合上述通过“授权”将用户邀请到子公司的操作, 再由子公司管理员 Folder Admin 自行管理的要求。

以下演示如何创建资源绑定 **无实际权限的角色**, 即 minirole。



角色名称	描述	类型	角色范围	创建时间
Admin	平台最高权限, 管理平台上所有...	系统	平台角色	2023-12-28 13:37
Kpanda Owner	容器管理的全部权限	系统	平台角色	2023-12-28 13:37
Workspace and Folder Owner	工作空间和文件夹的全部权限	系统	平台角色	2023-12-28 13:37
IAM Owner	用户与访问控制的全部权限	系统	平台角色	2023-12-28 13:37
Audit Owner	审计日志的全部权限	系统	平台角色	2023-12-28 13:37
Workspace Admin	工作空间的管理权限	系统	工作空间角色	2023-12-28 13:37
Workspace Editor	工作空间的编辑权限	系统	工作空间角色	2023-12-28 13:37
Workspace Viewer	工作空间的只读权限	系统	工作空间角色	2023-12-28 13:37
Folder Admin	文件夹及其下子文件夹、工作空...	系统	文件夹角色	2023-12-28 13:37
Folder Editor	文件夹及其下子文件夹、工作空...	系统	文件夹角色	2023-12-28 13:37

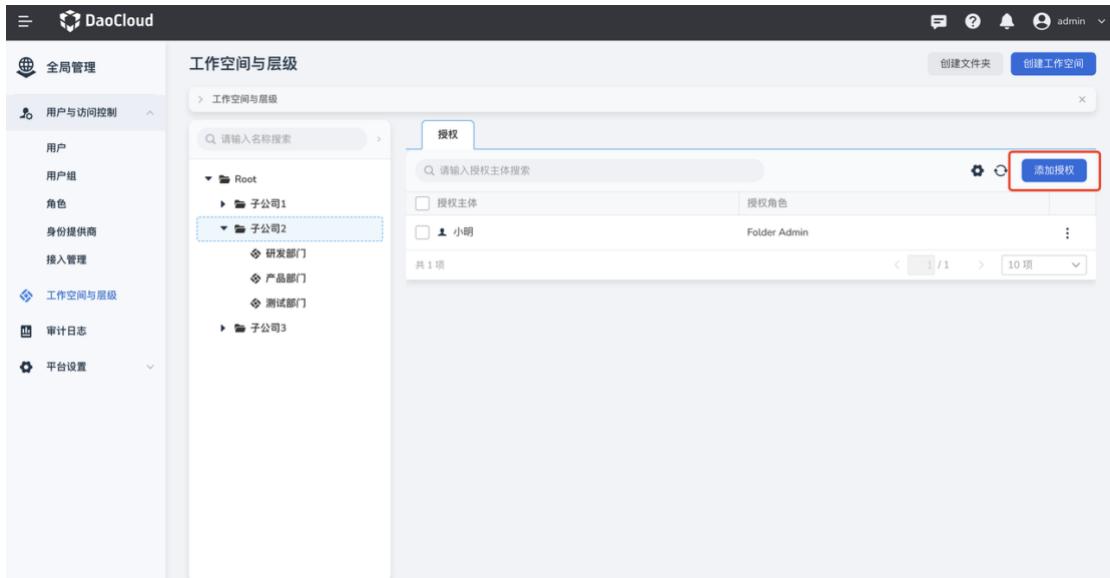
role1



role2

5. 给用户授权

平台管理员通过“授权”将用户按照实际情况邀请到各个子公司，并任命子公司管理员。

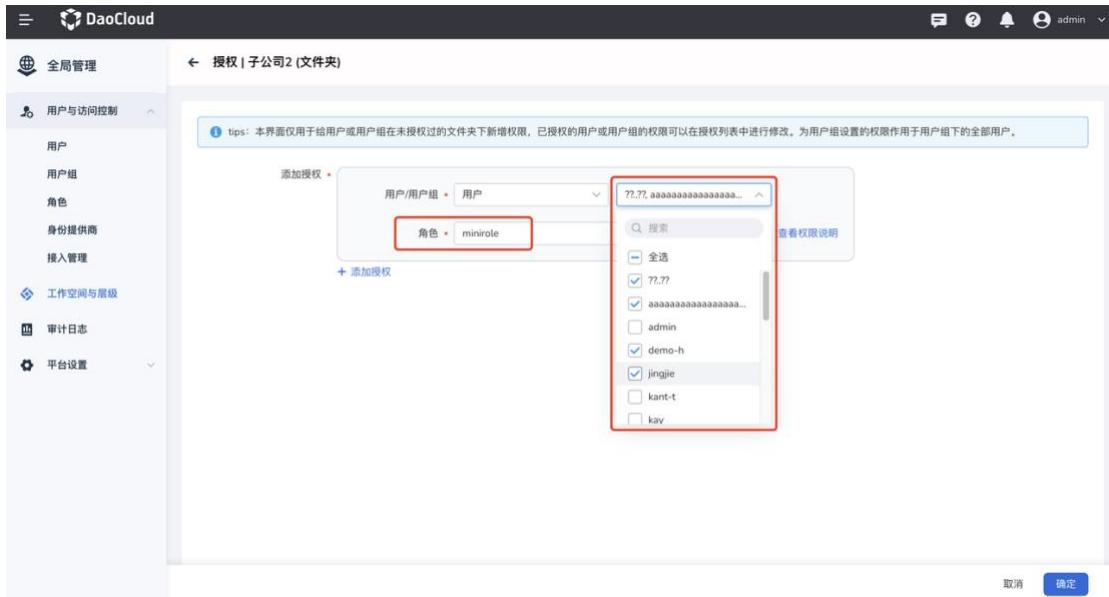


folerauth

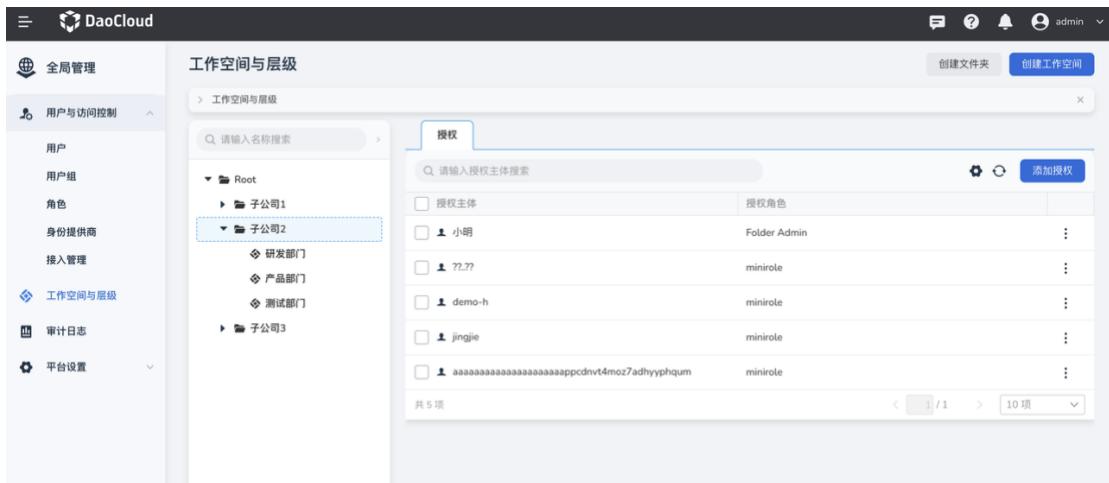
将子公司普通用户授权为“minirole”(1)，将子公司管理员授权为 Folder Admin。

```
{ .annotate }
```

1. 即第 4 步（上一步）中创建的 **无实际权限的角色**



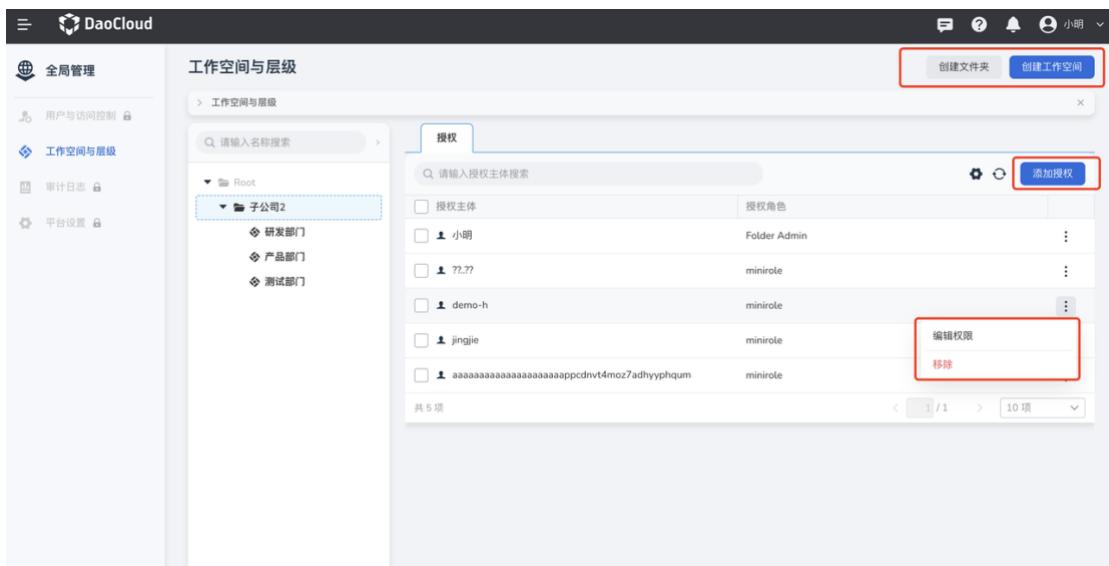
folerauth1



folerauth2

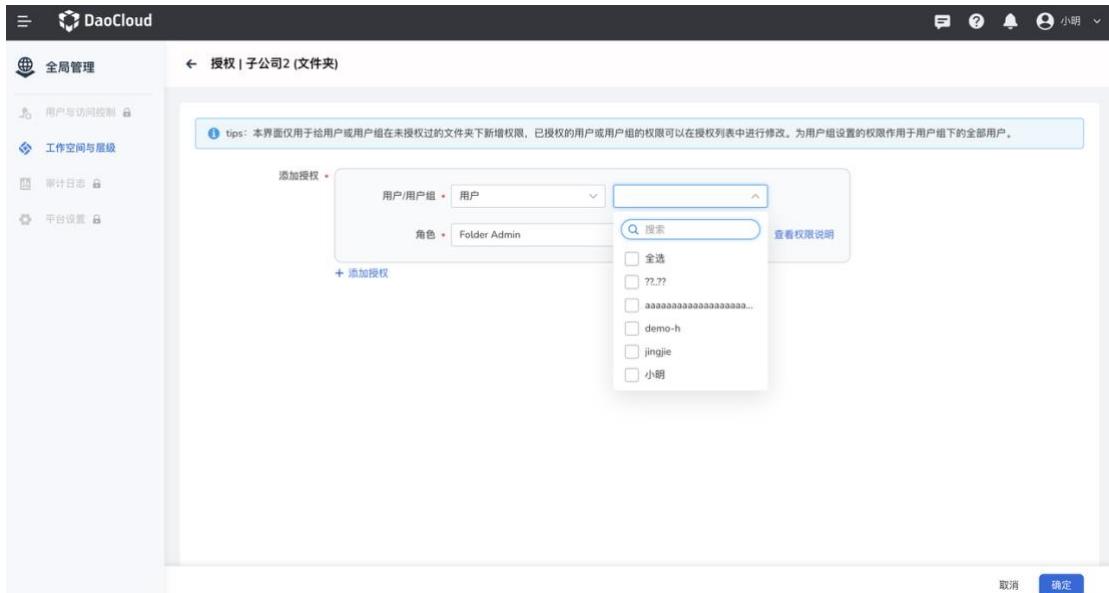
6. 子公司管理员自行管理用户/用户组

子公司管理员 Folder Admin 登录平台后只能看到自己所在的“子公司 2”，并能够通过创建文件夹、创建工作空间调整架构，通过添加授权/编辑权限为子公司 2 中的用户赋予其他权限。



folerauth3

在添加授权时，子公司管理员 Folder Admin 只能看到被平台管理员通过“授权”邀请进来的用户，而不能看到平台上的所有用户，从而实现 Folder/WS 之间的用户隔离，用户组同理（平台管理员视角能够看到并授权平台上所有的用户和用户组）。



folerauth4

!!! note

超大型企业与大/中/小型企业的主要区别在于 Folder 和工作空间中用户/用户组之间是否可见。

超大型企业里子公司与子公司之间用户/用户组不可见 + 权限隔离；

大/中/小型企业部门之间的用户相互可见 + 权限隔离。

GProduct 如何对接全局管理

GProduct 是 DCE 5.0 中除全局管理外的所有其他模块的统称，这些模块需要与全局管理对接后才能加入到 DCE 5.0 中。

对接什么

- [对接导航栏](#)

入口统一放在左侧导航栏。

- [接入路由和 AuthN](#)

统一 IP 或域名，将路由入口统一走全局管理的 Istio Gateway。

- 统一登录 / 统一 AuthN 认证

登录统一使用全局管理 (Keycloak) 登录页，API authn token 验证使用 Istio Gateway。

GProduct 对接全局管理后不需要关注如何实现登录和认证。

视频演示和 PDF

将 DCE 5.0 集成到客户系统 (OEM OUT) , 参阅 [OEM OUT 文档](#)。

将客户系统集成到 DCE 5.0 (OEM IN) , 参阅 [OEM IN 文档](#)。

[查阅和下载 GProduct PDF](#)

对接导航栏

以容器管理 (开发代号 kpanda) 为例，对接到导航栏。

对接后的预期效果如图：

对接效果

对接效果

对接方法

参照以下步骤对接 GProduct：

1. 通过 GProductNavigator CR 将容器管理的各项功能项注册到导航栏菜单。

```
apiVersion: ghippo.io/v1alpha1
kind: GProductNavigator
metadata:
  name: kpanda
spec:
  gproduct: kpanda
  name: 容器管理
  localizedName:
    zh-CN: 容器管理
    en-US: Container Management
  url: /kpanda
  category: 容器 # (1)
  iconUrl: /kpanda/nav-icon.png
  order: 10 # (2)
  menus:
    - name: 备份管理
      localizedName:
        zh-CN: 备份管理
        en-US: Backup Management
      iconUrl: /kpanda/bkup-icon.png
      url: /kpanda/backup
```

1. 当前只支持概览、工作台、容器、微服务、数据服务、管理，六选一
2. 数字越大排在越上面

全局管理的导航栏 **category** 配置在 ConfigMap，暂时不能以注册方式增加，需要联系全局管理团队来添加。

2. kpanda 前端作为微前端接入到 DCE 5.0 父应用 Anakin 中

前端使用 [qiankun](#) 来接入子应用 UI，可以参考[快速上手](#)。

在注册 GProductNavigator CR 后，接口会生成对应的注册信息，供前端父应用注册使用。

例如 kpanda 就会生成以下注册信息：

```
{
  "id": "kpanda",
  "title": "容器管理",
  "url": "/kpanda",
  "uiAssetsUrl": "/ui/kpanda/", // 结尾的/是必须的
  "needImportLicense": false
},
```

以上注册信息与 qiankun 子应用信息字段的对应关系是：

```
{
  name: id,
  entry: uiAssetsUrl,
  container: '#container',
  activeRule: url,
  loader,
  props: globalProps,
}
```

container 和 loader 由前端父应用提供，子应用无需关心。props 会提供一个包含用户基本信息、子产品注册信息等的 pinia store。

qiankun 启动时会使用如下参数：

```
start({
  sandbox: {
    experimentalStyleIsolation: true,
  },
  // 去除子应用中的favicon 防止在Firefox 中覆盖父应用的favicon
  getTemplate: (template) => template.replaceAll(/<link\s* rel="[\w\s]*icon[\w\s]*"\s*(\s*ef=".*?")?\s*/?>/g, ""),
});
```

请参阅前端团队出具的 [GProduct 对接 demo tar 包](#)。

接入路由和登录认证

接入后统一登录和密码验证，效果如下图：

接入效果

接入效果

各个 GProduct 模块的 API bear token 验证都走 Istio Gateway。

接入后的路由映射图如下：

接入效果

接入效果

接入方法

以 kpanda 为例注册 GProductProxy CR。

GProductProxy CR 示例, 包含路由和登录认证

```
# spec.proxies: 后写的路由不能是先写的路由子集, 反之可以
# spec.proxies.match.uri.prefix: 如果是后端 api, 建议在 prefix 末尾添加 "/" 表述这段 path 结束(特殊需求可以不用加)
# spec.proxies.match.uri: 支持 prefix 和 exact 模式; Prefix 和 Exact 只能 2 选 1; Prefix 优先级大于 Exact
```

```
apiVersion: ghippo.io/v1alpha1
kind: GProductProxy
metadata:
  name: kpanda # (1)
spec:
  gproduct: kpanda # (2)
  proxies:
    - labels:
        kind: UIEntry
      match:
        uri:
          prefix: /kpanda # (3)
      rewrite:
        uri: /index.html
      destination:
        host: ghippo-anakin.ghippo-system.svc.cluster.local
        port: 80
      authnCheck: false # (4)
    - labels:
```

```

kind: UIAssets
match:
  uri:
    prefix: /ui/kpanda/ # (5)
destination:
  host: kpanda-ui.kpanda-system.svc.cluster.local
  port: 80
  authnCheck: false
- match:
  uri:
    prefix: /apis/kpanda.io/v1/a
destination:
  host: kpanda-service.kpanda-system.svc.cluster.local
  port: 80
  authnCheck: false
- match:
  uri:
    prefix: /apis/kpanda.io/v1 # (6)
destination:
  host: kpanda-service.kpanda-system.svc.cluster.local
  port: 80
  authnCheck: true

```

1. cluster 级别 CRD

2. 需要用小写指定 GProduct 名字

3. 还可支持 exact

4. 是否需要 istio-gateway 给该条路由 API 作 AuthN Token 认证, false 为跳过认证

5. UIAssets 建议末尾添加 / 表示结束 (不然前端可能会出现问题)

6. 后写的路由不能是先写的路由的子集, 反之可以

如何将客户系统集成到 DCE 5.0 (OEM IN)

OEM IN 是指合作伙伴的平台作为子模块嵌入 DCE 5.0, 出现在 DCE 5.0 一级导航栏。用户通过 DCE 5.0 进行登录和统一管理。实现 OEM IN 共分为 5 步, 分别是:

1. 统一域名

2. 打通用户体系

3. 对接导航栏

4. 定制外观

5. 打通权限体系 (可选)

具体操作演示请参见 [OEM IN 最佳实践视频教程](#)。

!!! note

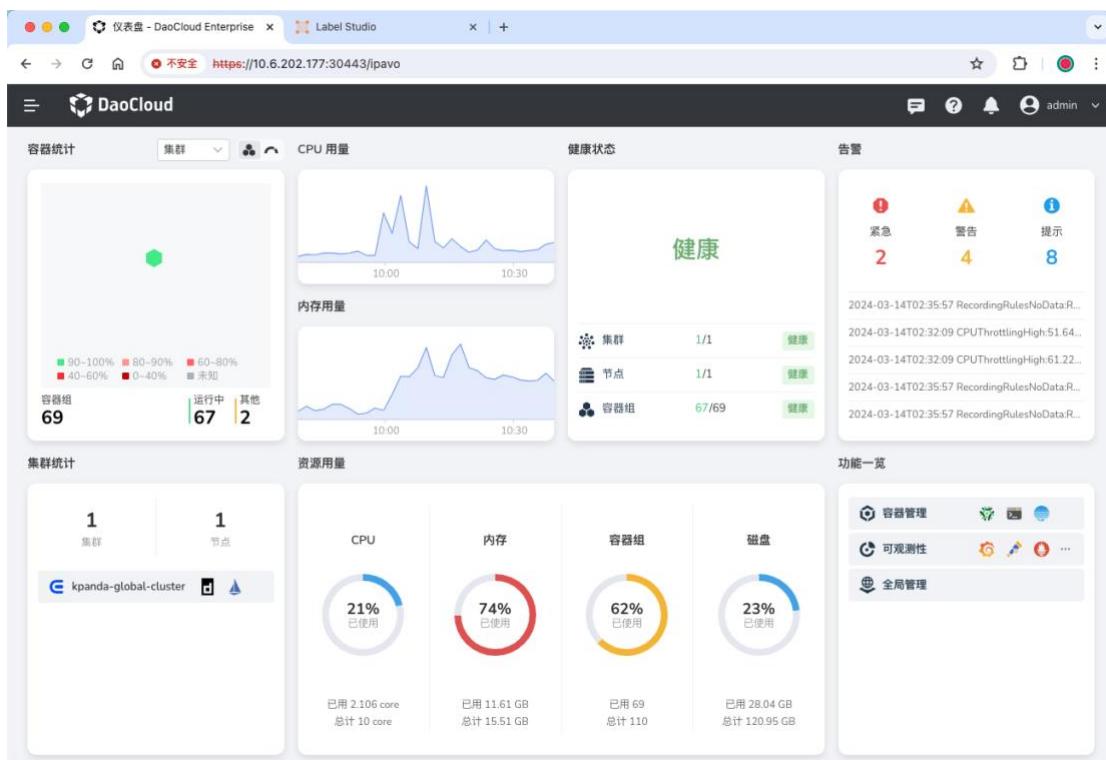
以下使用开源软件 Label Studio 来做嵌套演示。实际场景需要自己解决客户系统的问题：

例如客户系统需要自己添加一个 Subpath，用于区分哪些是 DCE 5.0 的服务，哪些是客户系统的服务。

环境准备

1. 部署 DCE 5.0 环境：

<https://10.6.202.177:30443> 作为 DCE 5.0 的环境。



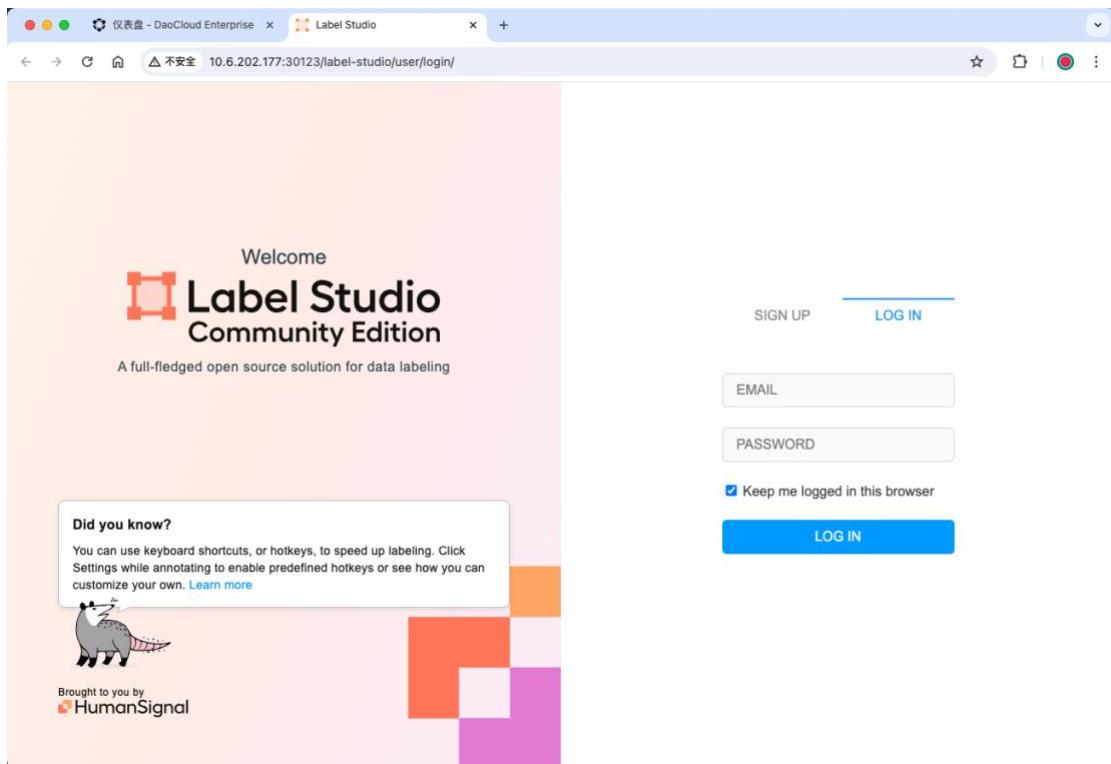
DCE 5.0

2. 部署客户系统环境：

<http://10.6.202.177:30123> 作为客户系统

应用过程中对客户系统的操作请根据实际情况进行调整。

3. 规划客户系统的 Subpath 路径: <http://10.6.202.177:30123/label-studio> (建议使用辨识度高的名称作为 Subpath, 不能与主 DCE 5.0 的 HTTP router 发生冲突)。请确保用户通过 <http://10.6.202.177:30123/label-studio> 能够正常访问客户系统。



Label Studio

统一域名和端口

1. SSH 登录到 DCE 5.0 服务器。

```
ssh root@10.6.202.177
```

2. 使用 vim 命令创建和修改 `label-studio.yaml` 文件

```
vim label-studio.yaml
```

```
```yaml
title="label-studio.yaml"
apiVersion: networking.istio.io/v1beta1
kind: ServiceEntry
metadata:
 name: label-studio
 namespace: ghippo-system
spec:
 exportTo:
 - "*"
 hosts:
 - label-studio.svc.external
 ports:
 - name: http
 port: 30123
 targetPort: 30123
```
# 添加虚拟端口
```

- number: 80 name: http protocol: HTTP location: MESH_EXTERNAL resolution: STATIC endpoints: # 改为客户系统的域名（或 IP）
- address: 10.6.202.177 ports: # 改为客户系统的端口号 http: 30123 -
 - apiVersion: networking.istio.io/v1alpha3 kind: VirtualService metadata: # 修改为客户系统的名字 name: label-studio namespace: ghippo-system spec:
 - exportTo:
 - "*" hosts:
 - "*" gateways:
 - ghippo-gateway http:
 - match:
 - uri: exact: /label-studio # 修改为客户系统在 DCE5.0 Web UI 入口中

的路由地址
 - uri: prefix: /label-studio/ # 修改为客户系统在 DCE5.0 Web UI 入口中

的路由地址 route:
 - destination: # 修改为上文 ServiceEntry 中的 spec.hosts 的值 host:

label-studio.svc.external port: # 修改为上文 ServiceEntry 中的

spec.ports 的值 number: 80 — apiVersion: security.istio.io/v1beta1

kind: AuthorizationPolicy metadata: # 修改为客户系统的名字 name:

label-studio namespace: istio-system spec: action: ALLOW selector:

matchLabels: app: istio-ingressgateway rules:
 - from:
 - source: requestPrincipals:

-*'
 - to:
 - operation: paths:

-/label-studio # 修 改 为 VirtualService 中 的

spec.http.match.uri.prefix 的值
 - /label-studio/* # 修 改 为 VirtualService 中 的

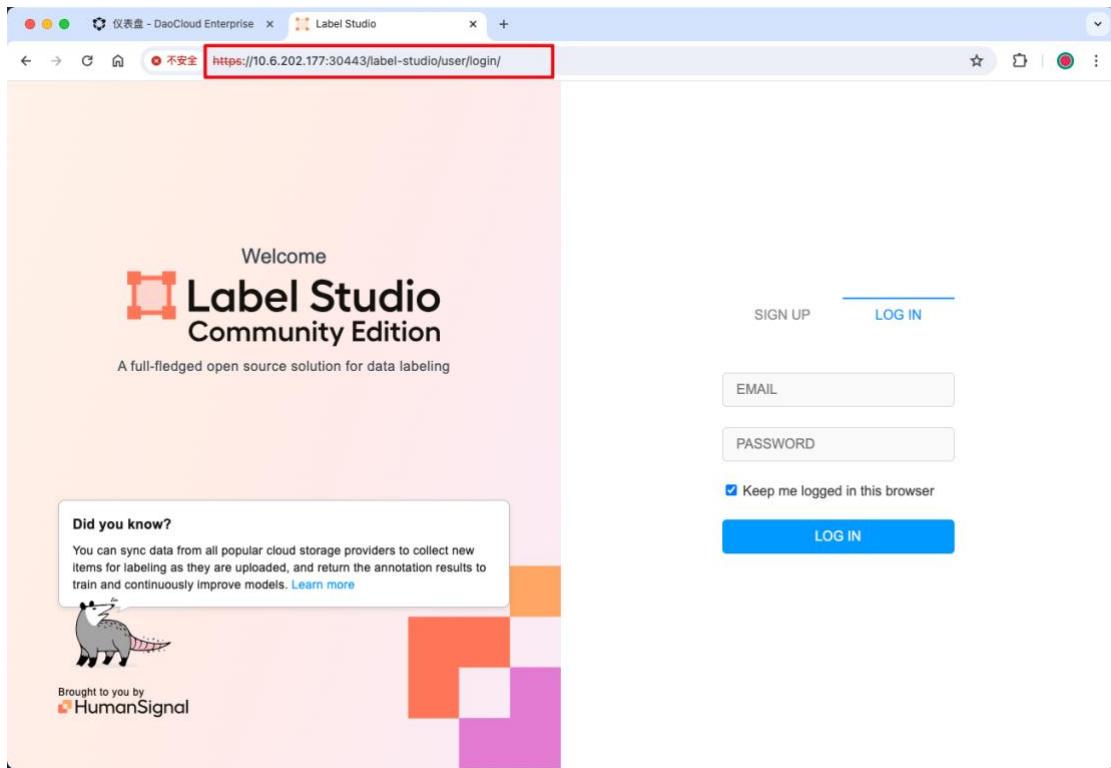
spec.http.match.uri.prefix 的值（注意，末尾需要添加 “*”）

...

3. 使用 kubectl 命令应用 `label-studio.yaml` :

```
kubectl apply -f label-studio.yaml
```

4. 验证 Label Studio UI 的 IP 和 端口是否一致:



Label Studio

打通用户体系

将客户系统与 DCE 5.0 平台通过 OIDC/OAUTH 等协议对接，使用户登录 DCE 5.0 平台后进入客户系统时无需再次登录。

!!! note

接入 SSO 需要 DCE 5.0 环境配置 SSL 证书，并使用 https 进行访问。

- 在两套 DCE 5.0 的场景下，可以在 DCE 5.0 中通过 **全局管理 -> 用户与访问控制 -> 接入管理** 创建 SSO 接入。

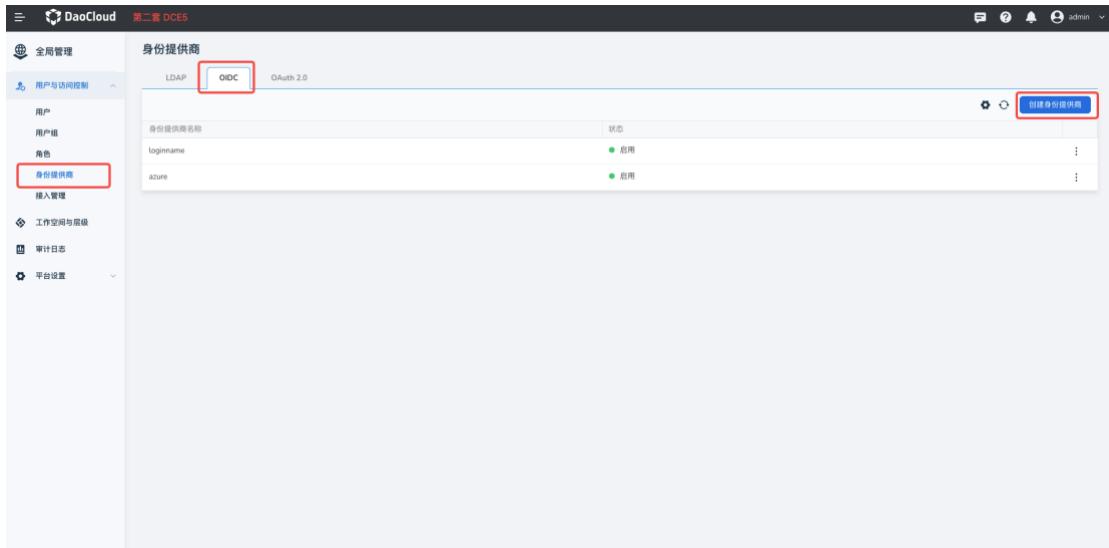
这里使用两套 DCE 5.0 相互对接来进行演示。涵盖将 DCE5 作为用户源登录客户平台，和将客户平台作为用户源登录 DCE5 平台两种场景。

2. DCE 5.0 作为用户源，登录客户平台：首先将第一套 DCE 5.0 作为用户源，实现对接后第一套 DCE 5.0 中的用户可以通过 OIDC 直接登录第二套 DCE 5.0，而无需在第二套中再次创建用户。在第一套 DCE 5.0 中通过 **全局管理 -> 用户与访问控制 -> 接入管理** 创建 SSO 接入。

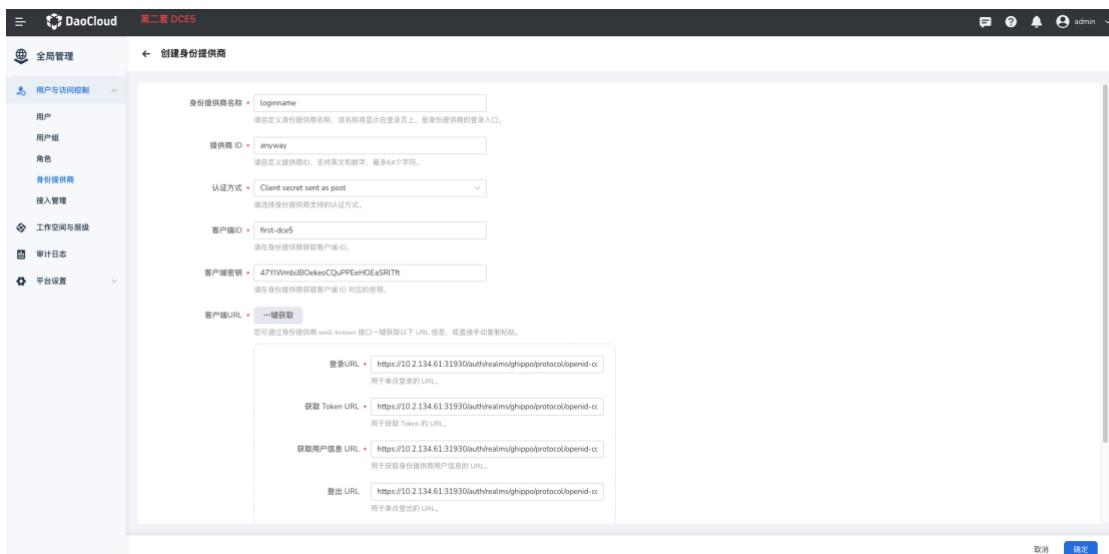
接入管理列表

接入管理列表

3. 客户平台作为用户源，登录 DCE 5.0：将第一套 DCE5 中生成的客户端 ID、客户端密钥、单点登录 URL 等填写到第二套 DCE 5.0 **全局管理 -> 用户与访问控制 -> 身份提供商** -> **OIDC** 中，完成用户对接。对接后，第一套 DCE 5.0 中的用户可以通过 OIDC 直接登录第二套 DCE 5.0，而无需在第二套中再次创建用户。

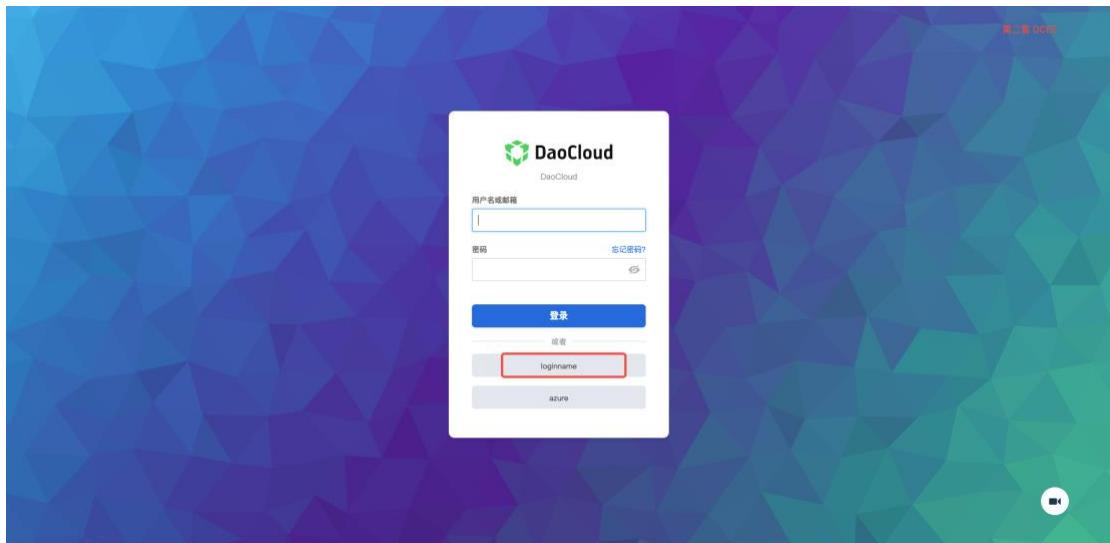


oidc1



oidc2

4. 对接完成后，第二套 DCE5 登录页面将出现 OIDC 选项，首次登录时选择通过 OIDC 登录（自定义名称，这里是名称是 loginname），后续将直接进入无需再次选择。



登录页

!!! note

使用两套 DCE 5.0，表明客户只要支持 OIDC 协议，无论是 DCE 5.0 作为用户源，还是“客户平台”作为用户源，两种场景都支持。

对接导航栏

参考文档下方的 tar 包来实现一个空壳的前端子应用，把客户系统以 iframe 的形式放进该空壳应用里。

1. 下载 gproduct-demo-main.tar.gz 文件，打开 src/App-iframe.vue 文件，修改其中的 src

属性值（即进入客户系统的地址）：

- 绝对地址：src="https://10.6.202.177:30443/label-studio" (DCE 5.0 地址 + Subpath)
- 相对地址：src="../external-anyproduct/insight"

```
```html      title="App-iframe.vue"           src="https://daocloud.io"      title="demo"
 class="iframe-container"
```
```

```

2. 删除 src 文件夹下的 App.vue 和 main.ts 文件，同时将：

- App-iframe.vue 重命名为 App.vue
- main-iframe.ts 重命名为 main.ts

3. 按照 `readme` 步骤构建镜像（注意：执行最后一步前需要将 `demo.yaml` 中的镜像地址

替换成构建出的镜像地址）

```
yaml title="demo.yaml" kind: Namespace apiVersion: v1 metadata: name: gproduct-demo --- apiVersion: apps/v1 kind: Deployment metadata: name: gproduct-demo namespace: gproduct-demo labels: app: gproduct-demo spec: selector: matchLabels: app: gproduct-demo template: metadata: name: gproduct-demo labels: app: gproduct-demo spec: containers: - name: gproduct-demo image: release.daocloud.io/gproduct-demo # 修改这个镜像地址 ports: - containerPort: 80 --- apiVersion: v1 kind: Service ...
```

对接完成后，将在 DCE 5.0 的一级导航栏出现 **客户系统**，点击可进入客户系统。

**客户系统**

**客户系统**

## 定制外观

### !!! note

DCE 5.0 支持通过写 CSS 的方式来实现外观定制。实际应用中客户系统如何实现外观定制需要根据实际情况处理。

登录客户系统，通过 **全局管理** -> **平台设置** -> **外观定制** 可以自定义平台背景颜色、logo、名称等，具体操作请参照[外观定制](#)。

## 打通权限体系（可选）

### 方案思路一：

定制化团队可实现一定制模块，DCE 5 将每一次的用户登录事件通过 Webhook 的方式通知到定制模块，定制模块可自行调用 AnyProduct 和 DCE 5.0 的 [OpenAPI](#) 将该用户的权限信息同步。

### 方案思路二：

通过 Webhook 方式, 将每一次的授权变化都通知到 AnyProduct (如有需求, 后续可实现)。

## AnyProduct 使用 DCE 5.0 的其他能力(可选)

操作方法为调用 DCE 5.0 [OpenAPI](#)。

## 参考资料

- 参考 [OEM OUT 文档](#)
- 参阅 [gProduct-demo-main 对接 tar 包](#)

## 如何将 DCE 5.0 集成到客户系统 (OEM OUT)

OEM OUT 是指将 DCE 5.0 作为子模块接入其他产品, 出现在其他产品的菜单中。 用户登录其他产品后可直接跳转至 DCE 5.0 无需二次登录。实现 OEM OUT 共分为 5 步, 分别是:

- [统一域名](#)
- [打通用户体系](#)
- [对接导航栏](#)
- [定制外观](#)
- [打通权限体系\(可选\)](#)

具体操作演示请参见 [OEM OUT 最佳实践视频教程](#)。

## 统一域名

1. 部署 DCE 5.0 (假设部署完的访问地址为 `https://10.6.8.2:30343/`)
2. 客户系统和 DCE 5.0 前可以放一个 nginx 反代来实现同域访问, / 路由到客户系统,

**/dce5 (subpath)** 路由到 DCE 5.0 系统, `vi /etc/nginx/conf.d/default.conf` 示例如下:

```
server {
 listen 80;
 server_name localhost;

 location /dce5/ {
 proxy_pass https://10.6.8.2:30343/;
 proxy_http_version 1.1;
 proxy_read_timeout 300s; # 如需要使用 kpanda cloudtty 功能需要这行, 否则可以去掉
 proxy_send_timeout 300s; # 如需要使用 kpanda cloudtty 功能需要这行, 否则可以去掉

 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_set_header Upgrade $http_upgrade; # 如需要使用 kpanda cloudtty 功能需要这行, 否则可以去掉
 proxy_set_header Connection $connection_upgrade; # 如需要使用 kpanda cloudtty 功能需要这行, 否则可以去掉
 }

 location / {
 proxy_pass https://10.6.165.50:30443/; # 假设这是客户系统地址(如意云)
 proxy_http_version 1.1;

 proxy_set_header Host $host;
 proxy_set_header X-Real-IP $remote_addr;
 proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
 }
}
```

3. 假设 nginx 入口地址为 10.6.165.50, 按[自定义 DCE 5.0 反向代理服务器地址](#)把 DCE\_PROXY 反代设为 http://10.6.165.50/dce5。确保能够通过 http://10.6.165.50/dce5 访问 DCE 5.0。客户系统也需要进行反代设置, 需要根据不同平台的情况进行处理。

## 反向代理

### 反向代理

## 打通用户体系

将客户系统与 DCE 5.0 平台通过 OIDC/OAUTH 等协议对接，使用户登录客户系统后进入 DCE 5.0 时无需再次登录。在拿到客户系统的 OIDC 信息后填入 **全局管理 -> 用户与访问控制 -> 身份提供商** 中。

身份提供商

身份提供商

对接完成后，DCE 5.0 登录页面将出现 OIDC（自定义）选项，首次从客户系统进入 DCE 5.0 时选择通过 OIDC 登录，后续将直接进入 DCE 5.0 无需再次选择。

登录页

登录页

## 对接导航栏

对接导航栏是指 DCE 5.0 出现在客户系统的菜单中，用户点击相应的菜单名称能够直接进入 DCE 5.0。因此对接导航栏依赖于客户系统，不同平台需要按照具体情况处理。

## 定制外观

通过 **全局管理 -> 平台设置 -> 外观定制** 可以自定义平台背景颜色、logo、名称等，具体操作请参照[外观定制](#)。

## 打通权限体系（可选）

打通权限较为复杂，如有需求请联系全局管理团队。

## 参考

- [OEM IN 文档](#)

# 定制 DCE 5.0 对接外部身份提供商 (IdP)

身份提供商 (IdP, Identity Provider)：当 DCE 5.0 需要使用客户系统作为用户源， 使用客户系统登录界面来进行登录认证时，该客户系统被称为 DCE 5 的身份提供商

## 适用场景

如果客户对 Ghippo 登录 IdP 有高度定制需求，例如支持企业微信、微信等其他社会组织登录需求，请根据本文档实施。

## 支持版本

Ghippo 0.15.0 及以上版本。

## 具体方法

### 自定义 ghippo keycloak plugin

#### 1. 定制 plugin

参考 [keycloak 官方文档](#)和 [keycloak 自定义 IdP](#) 进行开发。

#### 2. 构建镜像

```
FROM scratch
FROM scratch

plugin
COPY ./xxx-jar-with-dependencies.jar /plugins/
```

**!!! note**

如果需要两个定制化 IdP，需要复制两个 jar 包。

## 部署 Ghippo keycloak plugin 步骤

1. 把 [Ghippo 升级到 0.15.0 或以上](#)。您也可以直接安装部署 Ghippo 0.15.0 版本，但需要把以下信息手动记录下来。

```
helm -n ghippo-system get values ghippo -o yaml
apiserver:
 image:
 repository: release.daocloud.io/ghippo-ci/ghippo-apiserver
 tag: v0.4.2-test-3-gaba5ec2
controllermanager:
 image:
 repository: release.daocloud.io/ghippo-ci/ghippo-apiserver
 tag: v0.4.2-test-3-gaba5ec2
global:
 database:
 builtIn: true
 reverseProxy: http://192.168.31.10:32628
```

2. 升级成功后，手工跑一个安装命令，--set 里设的参数值从上述保存的内容里得到，并且外加几个参数值：

- global.idpPlugin.enabled：是否启用定制 plugin，默认已关闭
- global.idpPlugin.image.repository：初始化自定义 plugin 的 initContainer 用的 image 地址
- global.idpPlugin.image.tag：初始化自定义 plugin 的 initContainer 用的 image tag
- global.idpPlugin.path：自定义 plugin 的目录文件在上述 image 里所在的位置

具体示例如下：

```
helm upgrade \
 ghippo \
 ghippo-release/ghippo \
 --version v0.4.2-test-3-gaba5ec2 \
 -n ghippo-system \
```

```
--set apiserver.image.repository=release.daocloud.io/ghippo-ci/ghippo-apiserver \
--set apiserver.image.tag=v0.4.2-test-3-gaba5ec2 \
--set controllermanager.image.repository=release.daocloud.io/ghippo-ci/ghippo-apiserve
r \
--set controllermanager.image.tag=v0.4.2-test-3-gaba5ec2 \
--set global.reverseProxy=http://192.168.31.10:32628 \
--set global.database.builtIn=true \
--set global.idpPlugin.enabled=true \
--set global.idpPlugin.image.repository=chenyang-idp \
--set global.idpPlugin.image.tag=v0.0.1 \
--set global.idpPlugin.path=/plugins/.
```

3. 在 keycloak 管理页面选择所要使用的插件。

## Keycloak 自定义 IdP

要求: keycloak >= v20

已知问题 keycloak >= v21, 删除了旧版 theme 的支持, 可能会在 v22 修复。参见 [Issue #15344](#)。

此次 demo 使用 Keycloak v20.0.5。

## 基于 source 开发

### 配置环境

参照 [keycloak/building.md](#) 配置环境。

参照 [keycloak/README.md](#) 运行以下命令:

```
cd quarkus
mvn -f/pom.xml clean install -DskipTestsuite -DskipExamples -DskipTests
```

## 从 IDE 运行

从 IDE 运行

从 IDE 运行

## 添加 service 代码

### 如果可从 keycloak 继承部分功能

在目录 `services/src/main/java/org/keycloak/broker` 下添加文件：

文件名需要是 `xxxProvider.java` 和 `xxxProviderFactory.java`

java

java

查看 [xxxProviderFactory.java](#) 示例。

留意 `PROVIDER_ID = "oauth"`；这个变量，后面定义 html 会用到。

查看 [xxxProvider.java](#) 示例。

### 如果不能从 keycloak 继承功能

参考下图中的三个文件编写你的代码：

none heritance

none heritance

添加 `xxxProviderFactory` 到 resource service

在

`services/src/main/resources/META-INF/services/org.keycloak.broker.provider.IdentityProvider`

`Factory` 添加 `xxxProviderFactory`，这样刚刚编写的能工作了：

running

running

添加 html 文件

复

制

`themes/src/main/resources/theme/base/admin/resources/partials/realm-identity-provider-oi`

`dc.html` 文件到（改名为 `realm-identity-provider-oauth.html`，还记得上文中需要留意的变

量                          吗                          )

**themes/src/main/resources/theme/base/admin/resources/partials/realm-identity-provider-o  
auth.html**

到此所有的文件都添加完成了，开始调试功能。

## 打包成 jar 作为插件运行

新建一个 java 项目，并将上面的代码复制到项目中，如下所示：

pom  
pom

参见 [pom.xml](#)。

运行 `mvn clean package`，打包完成得到 `xxx-jar-with-dependencies.jar` 文件。

下载 [keycloak Release 20.0.5](#) zip 包并解压。

release  
release

将 `xxx-jar-with-dependencies.jar` 复制到 `keycloak-20.0.5/providers` 目录中。

运行以下命令查看功能是否完整：

`bin/kc.sh start-dev`

## 自定义导航栏

当前自定义导航栏需要通过手动创建导航栏的 YAML，并 apply 到集群中。

## 导航栏分类

### 导航栏分类

#### 导航栏分类

若需要新增或重新排序导航栏分类可以通过新增、修改 category YAML 实现。

category 的 YAML 示例如下：

```
apiVersion: ghippo.io/v1alpha1
kind: NavigatorCategory
metadata:
 name: management-custom # (1)!
spec:
 name: Management # (2)!
 isCustom: true # (3)!
 localizedName: # (4)!
 zh-CN: 管理
 en-US: Management
 order: 100 # (5)!
```

1. 命名规则：由小写的“spec.name”与“-custom”而成

2. 若是用于修改 category

3. 该字段必须为 true

4. 定义分类的中英文名称

5. 排序，数字越大，越靠上

编写好 YAML 文件后，通过执行如下命令后，刷新页面即可看到新增、修改的导航栏分类。

```
kubectl apply -f xxx.yaml
```

## 导航栏菜单

### 导航栏菜单

#### 导航栏菜单

若需要新增或重新排序导航栏菜单可以通过新增 navigator YAML 实现。

##### !!! note

若需要编辑已存在的导航栏菜单（非用户自己新增的 custom 菜单），需要令新增 custom 菜单 gproduct 字段与需要覆盖的菜单的 gproduct 相同，新的导航栏菜单会将 menus 中 name 相同的部分执行覆盖，name 不同的地方做新增操作。

# 一级菜单

使用以下 YAML，作为产品插入到某个导航栏分类下：

```
apiVersion: ghippo.io/v1alpha1
kind: GProductNavigator
metadata:
 name: gmagpie-custom # (1)!
spec:
 name: Operations Management
 iconUrl: ./ui/gmagpie/gmagpie.svg
 localizedName: # (2)!
 zh-CN: 运营管理
 en-US: Operations Management
 url: ./gmagpie
 category: management # (3)!
 menus: # (4)!
 - name: Access Control
 iconUrl: ./ui/ghippo/menus/access-control.svg
 localizedName:
 zh-CN: 用户与访问控制
 en-US: Access Control
 url: ./ghippo/users
 order: 50 # (5)!
 - name: Workspace
 iconUrl: ./ui/ghippo/menus/workspace-folder.svg
 localizedName:
 zh-CN: 工作空间与层级
 en-US: Workspace and Folder
 url: ./ghippo/workspaces
 order: 40
 - name: Audit Log
 iconUrl: ./ui/ghippo/menus/audit-logs.svg
 localizedName:
 zh-CN: 审计日志
 en-US: Audit Log
 url: ./ghippo/audit
 order: 30
 - name: Settings
 iconUrl: ./ui/ghippo/menus/setting.svg
 localizedName:
 zh-CN: 平台设置
 en-US: Settings
 url: ./ghippo/settings
```

```

order: 10
gproduct: gmagpie # (6)!
visible: true # (7)!
isCustom: true # (8)!
order: 20 # (9)!
target: blank # (10)!
```

1. 命名规则：由小写的“spec.gproduct”与“-custom”而成
2. 定义菜单的中英文名称
3. 与 parentGProduct 二选一，用于区分一级菜单还是二级菜单，与 NavigatorCategory 的 spec.name 字段对应来完成匹配
4. 二级菜单
5. 排序，数字越小，越靠上
6. 定义菜单的标志，用于和 parentGProduct 字段联动，实现父子关系。
7. 设置该菜单是否可见，默认为 true
8. 该字段必须为 true
9. 排序，数字越大，越靠上
10. 新开标签页

## 二级菜单

作为子产品插入到某个一级菜单的二级菜单中

```

apiVersion: ghippo.io/v1alpha1
kind: GProductNavigator
metadata:
 name: gmagpie-custom # (1)!
spec:
 name: Operations Management
 iconUrl: ./ui/gmagpie/gmagpie.svg
 localizedName: # (2)!
 zh-CN: 运营管理
 en-US: Operations Management
 url: ./gmagpie
```

```

parentGProduct: ghippo # (3)!
gproduct: gmagpie # (4)!
visible: true # (5)!
isCustom: true # (6)!
order: 20 # (7)!

```

1. 命名规则：由小写的“spec.gproduct”与“-custom”而成
2. 定义菜单的中英文名称
3. 与 category 二选一，用于区分一级菜单还是二级菜单，若添加该字段，则会忽视掉 menus 字段，并将该菜单作为二级菜单插入到与 gproduct 为 ghippo 的一级菜单中
4. 定义菜单的标志，用于和 parentGProduct 字段联动，实现父子关系
5. 设置该菜单是否可见，默认为 true
6. 该字段必须为 true
7. 排序，数字越大，越靠上

## 导航栏菜单根据权限显示/隐藏

在现有的权限体系下，全局管理可以根据用户的权限控制导航栏的菜单是否展示，但是由于容器管理的授权信息未同步到全局管理，导致全局管理无法准确判断容器管理菜单是否需要展示。

本文通过配置实现了：将容器管理及可观测性的菜单在 **全局管理无法判断的部分，默认不显示**，通过 **白名单** 授权的方式，实现菜单的隐藏与显示（通过容器管理页面授权的集群或命名空间权限，全局管理均无法感知和判断）。

例如：A 用户在容器管理是 cluster A 的 Cluster Admin 角色，这种情况下全局管理无法判断是否有权限展示容器管理菜单。通过本文档配置后，用户 A 默认不可见容器管理菜单，需要 **显式地在全局管理授权** 才可以看到容器管理菜单。

## 前提条件

已开启基于权限显示/隐藏菜单的功能，开启方法如下：

- 新安装的环境，使用 helm install 时增加 --set global.navigatorVisibleDependency=true 参数
- 已有环境， helm get values ghippo -n ghippo-system -o yaml 备份 values，随后修改 bak.yaml 并添加 global.navigatorVisibleDependency: true

```
apiserver:
 userIsolationMode: Folder
global:
 navigatorVisibleDependency: true
 reverseProxy: http://10.6.152.106:31596
storage:
 audit:
 - driver: mysql
 dsn: root:password@tcp(10.6.152.106:3306)/audit?charset=utf8mb4&multiStatements=true&parseTime=true
 ghippo:
 - driver: mysql
 dsn: root:password@tcp(10.6.152.106:3306)/ghippo?charset=utf8mb4&multiStatements=true&parseTime=true
 keycloak:
 - driver: mysql
 dsn: root:password@tcp(10.6.152.106:3306)/keycloak?charset=utf8mb4&multiStatements=true&parseTime=true
```

### 开启菜单隐藏

再使用以下命令升级全局管理：

```
helm upgrade ghippo ghippo-release/ghippo \
-n ghippo-system \
-f ./bak.yaml \
--version ${version}
```

## 配置导航栏

在 kpanda-global-cluster 中 apply 如下 YAML：

```
apiVersion: ghippo.io/v1alpha1
kind: GProductNavigator
metadata:
 name: kpanda-menus-custom
spec:
 category: container
```

```
gproduct: kpanda
iconUrl: ./ui/kpanda/kpanda.svg
isCustom: true
localizedName:
 en-US: Container Management
 zh-CN: 容器管理
menus:
 - iconUrl: "
 isCustom: true
 localizedName:
 en-US: Clusters
 zh-CN: 集群列表
 name: Clusters
 order: 80
 url: ./kpanda/clusters
 visible: true
 visibleDependency:
 permissions:
 - kpanda.cluster./*
 - kpanda.menu.get
 - iconUrl: "
 isCustom: true
 localizedName:
 en-US: Namespaces
 zh-CN: 命名空间
 name: Namespaces
 order: 70
 url: ./kpanda/namespaces
 visible: true
 visibleDependency:
 permissions:
 - kpanda.cluster./*
 - kpanda.menu.get
 - iconUrl: "
 isCustom: true
 localizedName:
 en-US: Workloads
 zh-CN: 工作负载
 name: Workloads
 order: 60
 url: ./kpanda/workloads/deployments
 visible: true
 visibleDependency:
 permissions:
```

```
- kpanda.cluster.*
- kpanda.menu.get
- iconUrl: "
 isCustom: true
 localizedName:
 en-US: Permissions
 zh-CN: 权限管理
 name: Permissions
 order: 10
 url: ./kpanda/rbac/content/cluster
 visible: true
 visibleDependency:
 permissions:
 - kpanda.cluster.*
 - kpanda.menu.get
 name: 容器管理
 order: 50
 url: ./kpanda/clusters
 visible: true

- --
apiVersion: ghippo.io/v1alpha1
kind: GProductNavigator
metadata:
 name: insight-menus-custom
spec:
 category: microservice
 gproduct: insight
 iconUrl: ./ui/insight/logo.svg
 isCustom: true
 localizedName:
 en-US: Insight
 zh-CN: 可观测性
 menus:
 - iconUrl: "
 isCustom: true
 localizedName:
 en-US: Overview
 zh-CN: 概览
 name: Overview
 order: 9
 url: ./insight/overview
 visible: true
 visibleDependency:
```

```
permissions:
 - kpanda.cluster./*
 - kpanda.menu.get
- iconUrl: ""
 isCustom: true
 localizedName:
 en-US: Dashboard
 zh-CN: 仪表盘
 name: Dashboard
 order: 8
 url: ./insight/dashboard
 visible: true
visibleDependency:
 permissions:
 - kpanda.cluster./*
 - kpanda.menu.get
- iconUrl: ""
 isCustom: true
 localizedName:
 en-US: Infrastructure
 zh-CN: 基础设施
 name: Infrastructure
 order: 7
 url: ./insight/clusters
 visible: true
visibleDependency:
 permissions:
 - kpanda.cluster./*
 - kpanda.menu.get
- iconUrl: ""
 isCustom: true
 localizedName:
 en-US: Metrics
 zh-CN: 指标
 name: Metrics
 order: 6
 url: ./insight/metric/basic
 visible: true
visibleDependency:
 permissions:
 - kpanda.cluster./*
 - kpanda.menu.get
- iconUrl: ""
 isCustom: true
```

```
localizedName:
 en-US: Logs
 zh-CN: 日志
name: Logs
order: 5
url: ./insight/logs
visible: true
visibleDependency:
 permissions:
 - kpanda.cluster.*
 - kpanda.menu.get
- iconUrl: "
 isCustom: true
 localizedName:
 en-US: Trace Tracking
 zh-CN: 链路追踪
 name: Trace Tracking
 order: 4
 url: ./insight/topology
 visible: true
 visibleDependency:
 permissions:
 - kpanda.cluster.*
 - kpanda.menu.get
- iconUrl: "
 isCustom: true
 localizedName:
 en-US: Alerts
 zh-CN: 告警
 name: Alerts
 order: 3
 url: ./insight/alerts/active/metrics
 visible: true
 visibleDependency:
 permissions:
 - kpanda.cluster.*
 - kpanda.menu.get
- iconUrl: "
 isCustom: true
 localizedName:
 en-US: Collect Management
 zh-CN: 采集管理
 name: Collect Management
 order: 2
```

```
url: ./insight/agents
visible: true
visibleDependency:
 permissions:
 - kpanda.cluster.*
 - kpanda.menu.get
- iconUrl: ""
isCustom: true
localizedName:
 en-US: System Management
 zh-CN: 系统管理
name: System Management
order: 1
url: ./insight/system-components
visible: true
visibleDependency:
 permissions:
 - kpanda.cluster.*
 - kpanda.menu.get
name: 可观测性
order: 30
url: ./insight
visible: true
```

```
--
apiVersion: ghippo.io/v1alpha1
kind: GProductResourcePermissions
metadata:
 name: kpanda
spec:
 actions:
 - localizedName:
 en-US: Create
 zh-CN: 创建
 name: create
 - localizedName:
 en-US: Delete
 zh-CN: 删除
 name: delete
 - localizedName:
 en-US: Update
 zh-CN: 编辑
 name: update
 - localizedName:
```

```

en-US: Get
zh-CN: 查看

name: get
- localizedName:
 en-US: Admin
 zh-CN: 管理
 name: admin

authScopes:
- resourcePermissions:
 - actions:
 - name: get
 - dependPermissions:
 - action: get
 name: create
 - dependPermissions:
 - action: get
 name: update
 - dependPermissions:
 - action: get
 name: delete
 resourceType: cluster
- actions:
 - name: get
 resourceType: menu

scope: platform
- resourcePermissions:
 - actions:
 - name: admin
 tips:
 - en-US: >-
 If the workspace is bound to a cluster, it will be assigned
 the Cluster Admin role upon authorization.
 zh-CN: 若工作空间绑定了集群，授权后还将被映射为对应集群的 Cluster
Admin 角色
 resourceType: cluster
 - actions:
 - name: get
 tips:
 - en-US: >-
 If the workspace is bound to a namespace, it will be
 assigned the NS View role upon authorization.
 zh-CN: 若工作空间绑定了命名空间，授权后还将被映射为对应命名空间的
NS View 角色
 - name: update

```

**tips:**

## - en-US: &gt;-

If the workspace is bound to a namespace, it will be assigned the NS Edit role upon authorization.

**zh-CN:** 若工作空间绑定了命名空间, 授权后还将被映射为对应命名空间的 NS Edit 角色

## - name: admin

**tips:**

## - en-US: &gt;-

If the workspace is bound to a namespace, it will be assigned the NS Admin role upon authorization.

**zh-CN:** 若工作空间绑定了命名空间, 授权后还将被映射为对应命名空间的 NS Admin 角色

**resourceType:** namespace

**scope:** workspace

**gproduct:** kpanda

**resourceTypes:**

- **localizedName:**

en-US: Cluster Management

zh-CN: 集群管理

**name:** cluster

- **localizedName:**

en-US: Menu

zh-CN: 菜单

**name:** menu

- **localizedName:**

en-US: Namespace Management

zh-CN: 命名空间

**name:** namespace

## 通过自定义角色实现上述效果

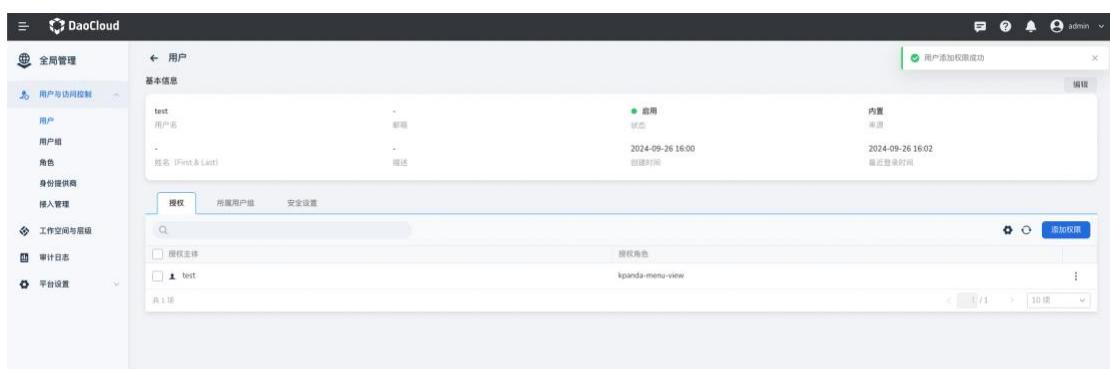
**!!! note**

仅容器管理模块的菜单需要单独配置菜单权限, 其他模块会根据用户的权限自动显示/隐藏

创建一个自定义角色, 包含的权限点为容器管理的菜单查看权限, 后续授权给需要查看容器管理菜单的用户。

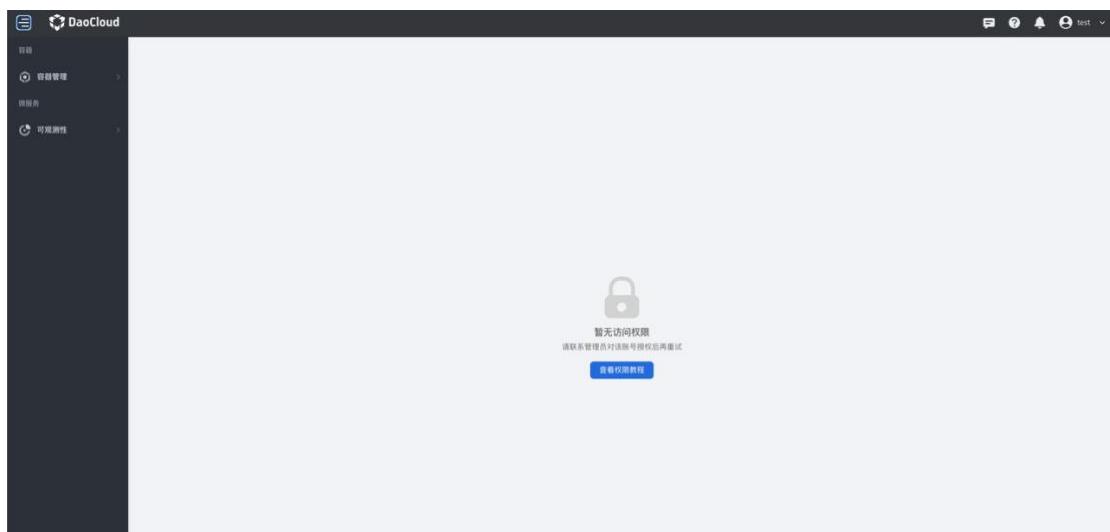


## 授权



## 查看权限点

效果如下，可以看到容器管理和可观测性的导航栏菜单：



## 验证结果

# 重启集群（虚拟机）istio-ingressgateway 无法启动？

报错提示如下图：

可能原因：RequestAuthentication CR 的 jwtsUri 地址无法访问，导致 istiod 无法下发配置给 istio-ingressgateway（Istio 1.15 可以规避这个 bug：<https://github.com/istio/istio/pull/39341/>）

解决方法：

1. 备份 RequestAuthentication ghippo CR。

```
kubectl get RequestAuthentication ghippo -n istio-system -o yaml > ghippo-ra.yaml
```

2. 删除 RequestAuthentication ghippo CR。

```
kubectl delete RequestAuthentication ghippo -n istio-system
```

3. 重启 Istio。

```
kubectl rollout restart deploy/istiod -n istio-system
```

```
kubectl rollout restart deploy/istio-ingressgateway -n istio-system
```

4. 重新 apply RequestAuthentication ghippo CR。

```
kubectl apply -f ghippo-ra.yaml
```

!!! note

apply RequestAuthentication ghippo CR 之前，请确保 ghippo-apiserver 和 ghippo-keycloak 已经正常启动。

# 登录无限循环，报错 401 或 403

出现这个问题原因为：gippo-keycloak 连接的 Mysql 数据库出现故障，导致 **OIDC Public keys** 被重置

在全局管理 0.11.1 及以上版本，您可以参照以下步骤，使用 **helm** 更新全局管理配置文件

即可恢复正常。

```
更新 helm 仓库
helm repo update ghippo

备份 ghippo 参数
helm get values ghippo -n ghippo-system -o yaml > ghippo-values-bak.yaml

获取当前部署的 ghippo 版本号
version=$(helm get notes ghippo -n ghippo-system | grep "Chart Version" | awk -F ':' '{ print $2 }')

执行更新操作，使配置文件生效
helm upgrade ghippo ghippo/ghippo \
- n ghippo-system \
- f ./gippo-values-bak.yaml \
- --version ${version}
```

## Keycloak 无法启动

\*[Ghippo]: DCE 5.0 全局管理的开发代号

### 常见故障

#### 故障表现

MySQL 已就绪，无报错。在安装全局管理后 keycloak 无法启动 (> 10 次)。

img  
img

#### 检查项

- 如果数据库是 MySQL，检查 keycloak database 编码是否是 UTF8。
- 检查从 keycloak 到数据库的网络，检查数据库资源是否充足，包括但不限于资源限制、

存储空间、物理机资源。

## 解决步骤

- img  
img
1. 检查 MySQL 资源占用是否到达 limit 限制
  2. 检查 MySQL 中 database keycloak table 的数量是不是 95 (Keycloak 不同版本数据库数量可能会不一样, 可以与同版本的开发或测试环境的 Keycloak 数据库数量进行比较), 如数量少了, 则说明数据库表初始化有问题 (查询表数量命令提示为: show tables;)
  3. 删除 keycloak database 并创建, 提示 CREATE DATABASE IF NOT EXISTS keycloak CHARACTER SET utf8
  4. 重启 Keycloak Pod 解决问题

## CPU does not support x86-64-v2

### 故障表现

keycloak 无法正常启动, keycloak pod 运行状态为 CrashLoopBackOff 并且 keycloak 的 log 出现如下图所示的信息

```
[root@demo-online-master1 ~]# k get pods -n ghippo-system
NAME READY STATUS RESTARTS AGE
ghippo-anakin-9b5f7789f-wbkk8 2/2 Running 0 9m34s
ghippo-apiserver-79cd876647-j55qt 2/2 Running 0 8h
ghippo-apiserver-cb574d499-rqrcf 0/2 Init:1/3 0 9m35s
ghippo-auditserver-7dc9c47b45-b2d68 2/2 Running 0 9m35s
ghippo-controller-manager-649bd6c756-k8s6m 2/2 Running 0 9m34s
ghippo-keycloakx-0 1/2 CrashLoopBackOff 1 (10s ago) 26s
ghippo-ui-59b767cccc-gmvms 2/2 Running 0 9m34s
[root@demo-online-master1 ~]# k logs -f ghippo-keycloakx-0 -n ghippo-system
Fatal glibc error: CPU does not support x86-64-v2
```

img.png

## 检查项

运行下面的检查脚本，查询当前节点 cpu 的 x86-64 架构的特征级别

```
cat <<"EOF" > detect-cpu.sh
#!/bin/sh -eu

flags=$(cat /proc/cpuinfo | grep flags | head -n 1 | cut -d: -f2)

supports_v2='awk "/cx16/&&/lahf/&&/popcnt/&&/sse4_1/&&/sse4_2/&&/ssse3/ {found=1} END
{exit !found}"'
supports_v3='awk "/avx/&&/avx2/&&/bmi1/&&/bmi2/&&/f16c/&&/fma/&&/abm/&&/movbe/&&/x
save/ {found=1} END {exit !found}"'
supports_v4='awk "/avx512f/&&/avx512bw/&&/avx512cd/&&/avx512dq/&&/avx512vl/ {found=1}
END {exit !found}"'

echo "$flags" | eval $supports_v2 || exit 2 && echo "CPU supports x86-64-v2"
echo "$flags" | eval $supports_v3 || exit 3 && echo "CPU supports x86-64-v3"
echo "$flags" | eval $supports_v4 || exit 4 && echo "CPU supports x86-64-v4"
EOF

chmod +x detect-cpu.sh
sh detect-cpu.sh
```

执行下面命令查看当前 cpu 的特性，如果输出中包含 sse4\_2，则表示你的处理器支持 SSE

4.2。

```
lscpu | grep sse4_2
```

## 解决方法

需要升级你的虚拟机或物理机 CPU 以支持 x86-64-v2 及以上，确保 x86 CPU 指令集支持 sse4.2，如何升级需要你咨询虚拟机平台提供商或着物理机提供商。

详见：<https://github.com/keycloak/keycloak/issues/17290>

# 单独升级全局管理时升级失败

若升级失败时包含如下信息，可以参考[离线升级](#)中的更新 ghippo crd 步骤完成 crd 安装

ensure CRDs are installed first