CLI 工具

ACP CLI (ac)

使用 ACP CLI 入门

配置 ACP CLI

ac 和 kubec

管理 CLI 配置文件

使用插件扩展 ACP CLI

AC CLI 开发

AC CLI 管理员命令参考

升级集群

violet CLI

violet CLI

Web 终端

Web 终端

ACP CLI (ac)

使用 **ACP CLI** 入门 | 配置 **ACP CLI** | **ac** 和 **kubec**

管理 **CLI** 配置文件 | 使用插件扩展 **ACP CLI** | **AC CLI** 开发

AC CLI 管理员命令参考 | 升级集群

使用 ACP CLI 入门

目录

关于 ACP CLI
安装
从二进制文件安装
在 Linux 上安装 ACP CLI
在 macOS 上安装 ACP CLI
在 Windows 上安装 ACP CLI
初始步骤
登录 ACP 平台
交互式登录
使用参数登录
使用环境变量登录
快速配置管理
查看当前状态
切换集群
切换命名空间
基础资源操作
管理多个环境
您的第一个应用
创建一个简单的 Pod
查看应用状态

清理
获取帮助
内置帮助系统
通用帮助
命令特定帮助
资源文档
登出

关于 **ACP CLI**

通过 ACP CLI（ac），您可以从终端管理 ACP 平台和集群。ACP CLI 提供类似 kubectl 的体验，针对 ACP 的集中式、基于代理的多集群架构进行了优化。

ACP CLI 适用于以下场景：

- 通过统一界面管理 ACP 平台和多个集群
- 直接操作项目源代码，编写 ACP 平台操作脚本并自动化工作流
- 在带宽受限且无法使用 Web 控制台时管理项目
- 管理不同 ACP 环境（生产、预发布、开发）中的应用

安装

从二进制文件安装

您可以通过下载适合您操作系统的二进制文件来安装 ACP CLI（ac）。

请按照以下步骤下载正确的软件包：

1. 在浏览器中打开 Alauda Cloud 下载页面↗。
2. 选择 **CLI Tools** 进入 CLI 下载页面。
3. 找到 **ACP CLI (ac)** 部分。

4. 下载与您的操作系统和 CPU 架构匹配的二进制文件（例如，`ac-linux-amd64`）。

在 Linux 上安装 ACP CLI

1. 按上述步骤完成下载 Linux 版本二进制文件（例如，`ac-linux-amd64` 或 `ac-linux-arm64`）。
2. 赋予二进制文件可执行权限：

   ```
   chmod +x ac-linux-amd64
   ```

   将 `ac-linux-amd64` 替换为您下载的文件名。
3. 将二进制文件移动到 PATH 中并重命名为 `ac`：

   ```
   sudo mv ac-linux-amd64 /usr/local/bin/ac
   ```

   如果下载了其他版本，请相应调整文件名。
4. 验证安装：

   ```
   ac version
   ```

在 macOS 上安装 ACP CLI

1. 按上述步骤完成下载 macOS 版本二进制文件（例如，`ac-darwin-amd64` 或 `ac-darwin-arm64`）。
2. 赋予二进制文件可执行权限：

   ```
   chmod +x ac-darwin-amd64
   ```

   将 `ac-darwin-amd64` 替换为您下载的文件名。
3. 将二进制文件移动到 PATH 中并重命名为 `ac`：

   ```
   sudo mv ac-darwin-amd64 /usr/local/bin/ac
   ```

   如果下载了其他版本，请相应调整文件名。

4. 验证安装：

```
ac version
```

在 **Windows** 上安装 **ACP CLI**

1. 按上述步骤完成下载 Windows 版本二进制文件（例如，`ac-windows-amd64.exe`）。
2. 将 `ac-windows-amd64.exe` 移动到 PATH 中的某个目录，并根据需要重命名为 `ac.exe`。如果不想重命名，请确保该文件所在目录已包含在 PATH 中。
3. 验证安装：

```
ac version
```

初始步骤

登录 **ACP** 平台

`ac login` 命令是连接 ACP 平台的入口。它负责身份验证并自动配置对所有可用集群的访问。

交互式登录

为了获得最简单的体验，运行 `ac login` 不带参数，并按照交互提示操作：

```
$ ac login
Platform URL: https://prod.acp.com
Session name: prod
Username: user@example.com
Password: [hidden]
✔ Login successful. Welcome, user@example.com!

Your kubeconfig has been configured for the 'prod' platform.
+ Default context 'prod/global' has been created and activated.

To switch clusters within this session, use:
  ac config use-cluster <cluster_name>

To switch between platforms, use:
  ac config get-sessions          # Discover all configured sessions
  ac config use-session <name>    # Switch to different platform
```

使用参数登录

您也可以直接提供参数登录：

```
ac login https://prod.acp.com --name prod --username user@example.com
```

使用环境变量登录

用于自动化和脚本时，可以使用环境变量：

```
export AC_LOGIN_PLATFORM_URL=https://prod.acp.com
export AC_LOGIN_SESSION=prod
export AC_LOGIN_USERNAME=user@example.com
export AC_LOGIN_PASSWORD=your-password
ac login
```

快速配置管理

登录后，ACP CLI 提供便捷的命令以支持日常操作：

查看当前状态

使用 `ac namespace` 查看当前操作上下文：

```
$ ac namespace
You are currently in namespace "default" (no namespace set in context).

Context:   prod/global
Cluster:   acp:prod:global
Server:    https://acp.prod.example.com/kubernetes/global/
```

切换集群

在当前会话中切换集群：

```
$ ac config use-cluster workload-a
Switched to context "prod/workload-a".

$ ac config use-cluster global
Switched to context "prod/global".
```

切换命名空间

更改当前使用的命名空间：

```
$ ac namespace my-app-dev
Now using namespace "my-app-dev" in context "prod/global".
```

基础资源操作

使用标准 kubectl 命令管理资源：

```
# 列出当前命名空间中的 pods
$ ac get pods

# 查看指定 pod 的详细信息
$ ac describe pod my-pod

# 获取所有命名空间中的服务
$ ac get services --all-namespaces

# 应用配置文件
$ ac apply -f deployment.yaml
```

管理多个环境

对于同时使用多个 ACP 平台的用户：

列出所有已配置的会话：

```
$ ac config get-sessions
CURRENT   SESSION    PLATFORM                              USER
CLUSTERS
*         prod       https://acp.prod.example.com          user@example.c
om      3
          staging    https://staging.acp.example.com       user@example.c
om      2
```

切换平台：

```
$ ac config use-session staging
Switched to session "staging".
Context "staging/global" activated.
```

您的第一个应用

让我们创建并查看一个简单的应用，以验证一切正常：

创建一个简单的 **Pod**

1. 创建一个基础 Pod 配置：

```
cat > test-pod.yaml << EOF
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
  labels:
    app: test
spec:
  containers:
  - name: nginx
    image: nginx:1.20
    ports:
    - containerPort: 80
EOF
```

2. 应用该配置：

```
$ ac apply -f test-pod.yaml
pod/test-pod created
```

查看应用状态

1. 列出 pods 以查看您的应用：

```
$ ac get pods
NAME       READY   STATUS    RESTARTS   AGE
test-pod   1/1     Running   0          30s
```

2. 获取 pod 的详细信息：

```
$ ac describe pod test-pod
```

3. 查看 pod 日志：

```
$ ac logs test-pod
```

清理

完成后删除测试 pod：

```
ac delete -f test-pod.yaml
```

获取帮助

内置帮助系统

ACP CLI 在多个层级提供全面的帮助：

通用帮助

获取所有可用命令的概览：

```
ac help
```

命令特定帮助

获取任意命令的详细帮助：

```
ac login --help
ac config --help
ac get --help
```

资源文档

获取 Kubernetes 资源的信息：

```
ac explain pod
ac explain deployment
ac explain service
```

登出

完成工作或需要切换凭据时，使用登出命令：

```
$ ac logout
✔ Successfully logged out from 'prod' platform.

All session configurations have been removed.
To reconnect, run: ac login https://prod.acp.com --name prod
```

登出命令将：

- 从本地配置中移除身份验证令牌
- 清理该会话的所有集群和上下文条目
- 撤销 ACP 中当前使用的令牌
- 确保不会留下孤立的配置项

配置 ACP CLI

目录

Shell 补全
为 Bash 启用 Tab 补全
前提条件
操作步骤
为 Zsh 启用 Tab 补全
前提条件
操作步骤
使用 ACP CLI 访问 kubeconfig
前提条件
操作步骤
多集群配置处理
安全注意事项

Shell 补全

您可以为 Bash 或 Zsh shell 启用 Tab 补全功能。

为 Bash 启用 Tab 补全

安装 ACP CLI (ac) 后，您可以启用 Tab 补全，在按下 Tab 键时自动完成 ac 命令或建议选项。以下操作步骤演示如何为 Bash shell 启用 Tab 补全。

前提条件

- 您必须已安装 ACP CLI (ac)。
- 您必须已安装 bash-completion 包。

操作步骤

1. 将 Bash 补全代码保存到文件：

```
$ ac completion bash > ac_bash_completion
```

2. 将该文件复制到 `/etc/bash_completion.d/` 目录：

```
$ sudo cp ac_bash_completion /etc/bash_completion.d/
```

   您也可以将文件保存到本地目录，并从您的 .bashrc 文件中 source 它。

打开新终端后，Tab 补全即被启用。

为 **Zsh** 启用 **Tab** 补全

安装 ACP CLI (ac) 后，您可以启用 Tab 补全，在按下 Tab 键时自动完成 ac 命令或建议选项。以下操作步骤演示如何为 Zsh shell 启用 Tab 补全。

前提条件

您必须已安装 ACP CLI (ac)。

操作步骤

要将 ac 的 Tab 补全添加到您的 .zshrc 文件，请运行以下命令：

```
cat >>~/.zshrc<<EOF
autoload -Uz compinit
compinit
if [[ $commands[ac] ]]; then
  source <(ac completion zsh)
  compdef _ac ac
fi
EOF
```

打开新终端后，Tab 补全即被启用。

使用 ACP CLI 访问 kubeconfig

您可以使用 ACP CLI (ac) 登录 ACP 平台，并获取用于从命令行访问集群的 kubeconfig 文件。与传统的单集群 kubeconfig 导出不同，ac login 通过平台发现创建了一个全面的多集群配置。

前提条件

您拥有 ACP 平台端点的访问权限和有效的身份验证凭证。

操作步骤

1. 运行以下命令登录您的 ACP 平台：

   ```
   $ ac login <platform-url> --name <session-name>
   ```

   - `<platform-url>`：ACP 平台的基础 URL（例如 https://acp.prod.example.com↗）
   - `<session-name>`：用户自定义的该平台连接的友好名称（例如 "prod"、"staging"）

2. 登录过程会自动：

   - 与 ACP 平台进行身份验证
   - 发现平台中所有可访问的集群
   - 为所有集群创建带有 ACP 特定元数据的 kubeconfig 条目
   - 设置指向 global 集群的默认上下文

3. 若要导出配置到单独文件，运行：

```
$ ac config view --raw > kubeconfig
```

4. 设置 KUBECONFIG 环境变量指向导出的文件：

```
$ export KUBECONFIG=./kubeconfig
```

5. 使用 ac 与您的 ACP 集群交互：

```
$ ac get nodes
```

多集群配置处理

ACP CLI 登录过程创建了一个包含以下内容的综合 kubeconfig 结构：

- 多个集群条目：平台中每个可访问集群对应一个条目
- 会话元数据：平台 URL、会话名称和集群描述存储在扩展字段中
- 统一身份验证：单个用户凭证条目适用于平台中所有集群
- 智能命名：使用 `acp:<session>:<cluster>` 格式的无冲突命名规范

安全注意事项

重要：导出的 kubeconfig 文件包含可访问您的 ACP 平台集群的身份验证令牌。

- 请安全存储该文件，并设置适当的文件权限
- 切勿将 kubeconfig 文件提交到版本控制系统
- 注意令牌的过期和刷新要求
- 对不同环境（prod、staging、dev）使用不同的会话名称以保持清晰分隔

如果您计划跨会话或机器重复使用导出的 kubeconfig 文件，请确保其安全存储，并定期使用 `ac config sync` 同步以保持集群列表的最新状态。

ac 和 kubectl 命令的使用

Kubernetes 命令行界面（CLI）kubectl 可用于对 Kubernetes 集群执行命令。由于 ACP 是一个兼容 Kubernetes 的平台，您可以使用随 ACP CLI 一起提供的支持的 kubectl 二进制文件，或者通过使用 ac 二进制文件获得扩展功能。

目录

ac 二进制文件
ACP 平台集成
智能资源路由
资源路由示例
额外命令
kubectl 二进制文件

ac 二进制文件

ac 二进制文件提供与 kubectl 二进制文件相同的功能，但扩展了对 ACP 平台附加功能的原生支持，包括：

ACP 平台集成

ACP CLI 内置支持 ACP 的集中式、基于代理的多集群架构：

- 平台认证 - 内置登录命令，用于与 ACP 平台的安全认证
- 会话管理 - 支持多平台会话管理的命令，如 `ac login`、`ac config use-session` 和 `ac logout`
- 增强配置 - 额外命令如 `ac config use-cluster`，使得在 ACP 多集群环境中操作更便捷

智能资源路由

ACP CLI 会自动将平台级资源类型（如 `User` 和 `Project`）路由到 global 集群，因为这些资源仅存在于平台级别。这使您可以在任何集群上下文中访问它们，无需手动切换。所有其他资源则在当前集群上下文中正常工作。

资源路由示例

```
# 当前上下文指向业务集群
$ ac config current-context
prod/workload-a

# 用户请求全局资源 - ACP CLI 自动路由到 global 集群
$ ac get projects
(i) Note: Targeting global cluster for this command only, as 'projects' i
s a global resource.
NAME          STATUS   AGE
project-a     Active   32d
project-b     Active   18d

# 用户请求业务资源 - 在当前集群上操作
$ ac get pods
NAME                     READY   STATUS    RESTARTS   AGE
my-app-7d4f8c9b6-xyz123  1/1     Running   0          2h
```

额外命令

ACP CLI 包含简化 ACP 平台工作流程的额外命令：

- `ac login` - 认证 ACP 平台并配置多集群访问
- `ac logout` - 结束平台会话并清理配置
- `ac config get-sessions` - 列出所有配置的 ACP 平台会话

- `ac config use-session <session_name>` - 在 ACP 平台间切换
- `ac config use-cluster <cluster_name>` - 在当前会话中切换集群
- `ac namespace` - 增强的命名空间管理，显示平台上下文
- `ac config sync` - 同步配置与平台状态

kubectl 二进制文件

kubectl 二进制文件旨在支持来自标准 Kubernetes 环境的新 ACP CLI 用户的现有工作流程和脚本，或偏好使用 kubectl CLI 的用户。现有的 kubectl 用户可以继续使用该二进制文件与 Kubernetes 原语交互，无需对 ACP 平台进行任何更改。

有关 kubectl 的更多信息，请参见 kubectl documentation ↗。

管理 CLI 配置文件

CLI 配置文件允许您为 ACP CLI（ac）配置不同的配置文件或上下文。一个上下文包含与昵称关联的用户认证信息和 ACP 平台服务器信息。

目录

便捷的配置管理

平台和会话管理

`ac login` - 认证并配置对 ACP 平台的访问

`ac logout` - 结束平台会话并清理配置

`ac config get-sessions` - 列出所有已配置的 ACP 平台会话

`ac config use-session <session_name>` - 切换 ACP 平台

日常操作

`ac config use-cluster <cluster_name>` - 在当前会话中切换集群

`ac namespace` - 查看当前状态并切换命名空间

`ac config sync` - 同步平台配置

了解 ACP CLI 配置结构

ACP CLI 增强的 kubeconfig 结构

元数据结构与组织

基于元数据的识别

命名规范

手动配置 CLI 配置文件

标准配置命令

手动操作示例

加载与合并规则

便捷的配置管理

ACP CLI 提供了增强命令，使配置管理比传统的 kubeconfig 操作更加简单。这些命令专为无缝支持 ACP 的多集群环境而设计。

平台和会话管理

`ac login` - 认证并配置对 ACP 平台的访问

`ac login` 命令是建立与 ACP 平台连接的主要入口。它负责用户认证并自动配置所有必要的 kubeconfig 条目。

```
# 交互式登录 ACP 平台
ac login https://prod.acp.com --name prod

# 指定集群和命名空间登录
ac login https://prod.acp.com --name prod --cluster workload-a --namespac
e my-app

# 使用环境变量登录（适用于自动化）
AC_LOGIN_PLATFORM_URL=https://prod.acp.com AC_LOGIN_SESSION=prod AC_LOGIN
_USERNAME=user AC_LOGIN_PASSWORD=secret ac login
```

登录流程：

1. 对 ACP 平台进行认证
2. 发现平台内所有可访问的集群
3. 在 kubeconfig 中创建集群和用户条目
4. 创建并激活上下文：
   - 如果指定了 `--cluster`：为该特定集群创建上下文
   - 如果指定了 `--namespace`：在上下文中设置命名空间

- 如果未指定集群：默认为 global 集群
- 上下文名称格式为：`<session_name>/<cluster_name>`

`ac logout` - 结束平台会话并清理配置

```
# 登出当前平台会话
ac logout

# 登出指定会话
ac logout --session prod
```

登出命令会移除所有与会话相关的配置条目，包括集群、用户和上下文。

`ac config get-sessions` - 列出所有已配置的 ACP 平台会话

```
ac config get-sessions
```

示例输出：

```
CURRENT   SESSION    PLATFORM                              USER
CLUSTERS
*         prod       https://acp.prod.example.com          user@example.c
om      3
          staging    https://staging.acp.example.com       user@example.c
om      2
          dev        https://dev.acp.example.com           dev-user@examp
le.com  1
```

该命令显示：

- **CURRENT**：指示当前上下文是否属于该会话（用 `*` 标记）
- **SESSION**：会话名称（登录时用户定义）
- **PLATFORM**：基础平台 URL
- **USER**：该会话的认证用户名
- **CLUSTERS**：该会话中可用集群数量

`ac config use-session <session_name>` - 切换 ACP 平台

```
# 切换到 staging 平台（默认 global 集群）
ac config use-session staging

# 切换到会话中的特定集群
ac config use-session prod --cluster workload-a

# 指定命名空间切换
ac config use-session staging --cluster workload-b --namespace my-app
```

该命令根据您的会话和集群需求智能选择或创建合适的上下文。

日常操作

`ac config use-cluster <cluster_name>` - 在当前会话中切换集群

```
# 切换到当前会话中的 workload 集群
ac config use-cluster workload-a

# 创建带特定命名空间的新上下文
ac config use-cluster workload-b --namespace my-app
```

该命令查找或创建当前平台会话中指定集群的上下文。

`ac namespace` - 查看当前状态并切换命名空间

查看当前状态：

```
ac namespace
```

示例输出：

```
You are currently in namespace "my-app-dev".

Context:   prod/workload-a
Cluster:   acp:prod:workload-a
Server:    https://acp.prod.example.com/kubernetes/workload-a/
Platform:  https://acp.prod.example.com/
Session:   prod
```

切换命名空间：

```
ac namespace my-app-dev
```

`ac config sync` - 同步平台配置

```
# 同步当前平台会话
ac config sync

# 同步指定会话
ac config sync --session prod

# 同步所有会话
ac config sync --all
```

sync 命令会使用 ACP 平台的最新信息刷新您的配置，添加新集群并更新凭证。

了解 ACP CLI 配置结构

ACP CLI 将所有配置信息存储在标准的 `~/.kube/config` 文件中，确保与 kubectl 及其他 Kubernetes 工具完全兼容，同时增加 ACP 特有的增强功能。

ACP CLI 增强的 kubeconfig 结构

ACP CLI 在标准 kubeconfig 格式基础上扩展了 ACP 特定的元数据，以增强平台集成：

```
apiVersion: v1
clusters:
- cluster:
    server: https://acp.prod.example.com/kubernetes/global/
    extensions:
    - name: acp.io/v1
      extension:
        isGlobal: true
        platformUrl: https://acp.prod.example.com
        sessionName: prod
        clusterName: global
        description: global cluster
        note: This cluster item is managed by ac CLI, to avoid unexpected
behavior, do not edit this item.
  name: acp:prod:global
- cluster:
    server: https://acp.prod.example.com/kubernetes/workload-a/
    extensions:
    - name: acp.io/v1
      extension:
        isGlobal: false
        platformUrl: https://acp.prod.example.com
        sessionName: prod
        clusterName: workload-a
        description: business cluster for team alpha
        note: This cluster item is managed by ac CLI, to avoid unexpected
behavior, do not edit this item.
  name: acp:prod:workload-a
contexts:
- context:
    cluster: acp:prod:global
    namespace: default
    user: acp:prod:user
  name: prod/global
- context:
    cluster: acp:prod:workload-a
    namespace: my-app
    user: acp:prod:user
  name: prod/workload-a
current-context: prod/global
kind: Config
preferences: {}
users:
```

```
- name: acp:prod:user
  user:
    token: <TOKEN>
    extensions:
    - name: acp.io/v1
      extension:
        platformUrl: https://acp.prod.example.com
        sessionName: prod
        username: user@example.com
        note: This user item is managed by ac CLI, to avoid unexpected be
havior, do not edit this item.
```

元数据结构与组织

ACP CLI 使用扩展元数据来组织和识别配置条目：

基于元数据的识别

- 平台识别：使用 `platformUrl` 标识所属平台
- 会话关联：使用 `sessionName` 将相关的集群、用户和上下文分组
- **global** 集群检测：使用 `isGlobal` 字段识别管理集群
- 用户凭证定位：在用户扩展中匹配 `sessionName` 和 `platformUrl`

命名规范

ACP CLI 创建新条目时遵循一致的命名规范：

- 集群条目：`acp:<session_name>:<cluster_name>`（例如 `acp:prod:global`）
- 用户条目：`acp:<session_name>:user`（例如 `acp:prod:user`）
- 上下文条目：`<session_name>/<cluster_name>`（例如 `prod/global`）

> **NOTE**
>
> `acp:` 前缀确保 ACP CLI 管理的条目不会与现有 kubeconfig 条目冲突。 用户可以手动重命名这些条目——ACP CLI 通过元数据识别条目，而非名称。

手动配置 CLI 配置文件

对于需要精确控制配置的高级用户，ACP CLI 支持所有标准的 kubectl config 命令，以便手动管理 kubeconfig。

TIP

大多数用户应使用上述便捷命令。

手动配置命令适用于高级场景：

- 自定义上下文命名 - 创建不遵循 ACP CLI 命名规范的上下文
- 非 **ACP** 环境 - 在 ACP 会话之外管理传统的 kubectl 上下文
- 复杂多上下文场景 - 需要精确控制上下文的高级工作流
- 排查配置问题 - 调试或修复配置问题

标准配置命令

ACP CLI 完全兼容 kubectl config 子命令：

子命令	用法
`set-cluster`	在 CLI 配置文件中设置集群条目
`set-context`	在 CLI 配置文件中设置上下文条目
`use-context`	使用指定的上下文名称设置当前上下文
`set`	在 CLI 配置文件中设置单个值
`unset`	在 CLI 配置文件中取消设置单个值
`view`	显示当前合并的 CLI 配置

手动操作示例

创建自定义上下文：

```
# 创建自定义命名的上下文
ac config set-context my-custom-context --cluster=acp:prod:workload-a --n
amespace=my-app

# 切换到自定义上下文
ac config use-context my-custom-context
```

查看当前配置：

```
# 显示合并后的配置
ac config view

# 显示指定文件的配置
ac config view --config=/path/to/config
```

更新上下文命名空间：

```
# 设置当前上下文的命名空间
ac config set-context `ac config current-context` --namespace=my-namespac
e
```

加载与合并规则

在执行 CLI 操作时，配置文件的加载和合并遵循以下规则：

- CLI 配置文件从您的工作站获取，遵循以下层级和合并规则：
  - 如果设置了 `--config` 选项，则仅加载该文件。该标志只设置一次，不进行合并。
  - 如果设置了 `$KUBECONFIG` 环境变量，则使用它。该变量可以是路径列表，路径会合并在一起。修改值时，修改定义该段落的文件。创建值时，创建于第一个存在的文件。如果链中无文件存在，则创建列表中的最后一个文件。
  - 否则，使用 `~/.kube/config` 文件，不进行合并。
- 使用的上下文根据以下流程的第一个匹配项确定：

  - `--context` 选项的值
  - CLI 配置文件中的 `current-context` 值
  - 此阶段允许为空值
- 使用的用户和集群确定。此时可能有或没有上下文；它们基于以下流程的第一个匹配项构建，分别针对用户和集群运行一次：
  - 用户名的 `--user` 和集群名的 `--cluster` 选项值
  - 如果存在 `--context` 选项，则使用该上下文的值
  - 此阶段允许为空值
- 实际使用的集群信息确定。此时可能有或没有集群信息。集群信息的每个部分基于以下流程的第一个匹配项构建：
  - 以下命令行选项的值：`--server`、`--api-version`、`--certificate-authority`、`--insecure-skip-tls-verify`
  - 如果存在集群信息且该属性有值，则使用它
  - 如果没有服务器地址，则报错
- 实际使用的用户信息确定。用户构建规则与集群相同，但每个用户只能有一种认证方式；认证方式冲突会导致操作失败。命令行选项优先于配置文件值。有效的命令行选项包括：
  - `--auth-path`
  - `--client-certificate`
  - `--client-key`
  - `--token`
- 对于仍缺失的信息，使用默认值并提示输入额外信息。

使用插件扩展 ACP CLI

您可以编写并安装插件，以扩展默认的 ac 命令，从而使您能够通过 ACP CLI 和 ACP 平台集成执行新的、更复杂的任务。

目录

编写 CLI 插件
- 创建简单插件
- 插件开发要求
- 其他资源

安装和使用 CLI 插件
- 前提条件
- 安装操作步骤

编写 CLI 插件

您可以使用任何支持编写命令行命令的编程语言或脚本为 ACP CLI（ac）编写插件。请注意，插件不能覆盖已有的 ac 命令。

创建简单插件

此操作步骤创建一个简单的 Bash 插件，当执行 `ac foo` 命令时，会在终端打印一条消息。

操作步骤

1. 创建一个名为 `ac-foo` 的文件。命名插件文件时，请注意以下几点：
   - 文件必须以 `ac-` 或 `kubectl-` 开头，才能被识别为插件
   - 文件名决定调用该插件的命令。例如，文件名为 `ac-foo-bar` 的插件可以通过命令 `ac foo bar` 调用
   - 如果希望命令中包含连字符，也可以使用下划线。例如，文件名为 `ac-foo_bar` 的插件可以通过命令 `ac foo-bar` 调用
2. 将以下内容添加到该文件中：

```
#!/bin/bash

# 可选参数处理
if [[ "$1" == "version" ]]; then
    echo "1.0.0"
    exit 0
fi

# 可选参数处理
if [[ "$1" == "config" ]]; then
    echo $KUBECONFIG
    exit 0
fi

echo "I am a plugin named ac-foo"
```

   安装此插件后，可以通过 `ac foo` 命令调用它。

插件开发要求

- 编程语言：使用任何支持命令行接口的编程语言或脚本
- 命名规范：插件文件必须遵循 `ac-<plugin-name>` 或 `kubectl-<plugin-name>` 的命名模式
- 可执行权限：插件文件必须具有可执行权限
- 命令覆盖：插件不能覆盖已有的 ACP CLI 命令
- 参数处理：插件应适当处理标准命令行参数和标志

其他资源

- 查看 kubectl 插件开发指南，了解实现模式和最佳实践
- 使用 Go 语言的 CLI 运行时工具进行插件开发
- 设计与集群资源交互的插件时，考虑 ACP 平台集成

安装和使用 **CLI** 插件

编写自定义 ACP CLI 插件后，必须先安装插件才能使用。

前提条件

- 已安装 ACP CLI（ac）
- 拥有以 `ac-` 或 `kubectl-` 开头的 CLI 插件文件

安装操作步骤

1. 如有必要，更新插件文件为可执行：

   ```
   chmod +x <plugin_file>
   ```

2. 将文件放置在 PATH 中的任意位置，例如 `/usr/local/bin/`：

   ```
   sudo mv <plugin_file> /usr/local/bin/
   ```

3. 运行 `ac plugin list`，确保插件已列出：

   ```
   ac plugin list
   ```

   示例输出

```
The following compatible plugins are available:

/usr/local/bin/<plugin_file>
```

如果插件未列出，请确认文件以 `ac-` 或 `kubectl-` 开头，具有可执行权限，且位于 PATH 中。

4. 调用插件引入的新命令或选项。
   例如，如果您构建并安装了 `ac-ns` 插件，可以使用以下命令查看当前命名空间：

```
ac ns
```

请注意，调用插件的命令取决于插件文件名。例如，文件名为 `ac-foo-bar` 的插件通过命令 `ac foo bar` 调用。

AC CLI 开发人员命令参考

本参考文档提供 AC CLI 开发人员命令的说明和示例命令。有关管理员命令，请参阅 AC CLI 管理员命令参考。

运行 `ac help` 可列出所有命令，或运行 `ac <command> --help` 以获取特定命令的更多详细信息。

目录

ac annotate
  示例用法
ac api-resources
  示例用法
ac api-versions
  示例用法
ac apply
  示例用法
ac apply edit-last-applied
  示例用法
ac apply set-last-applied
  示例用法
ac apply view-last-applied
  示例用法
ac attach

示例用法

ac auth

ac auth can-i

示例用法

ac auth reconcile

示例用法

ac auth whoami

示例用法

ac autoscale

示例用法

ac cluster-info

示例用法

ac cluster-info dump

示例用法

ac completion

示例用法

ac config

ac config current-context

示例用法

ac config delete-cluster

示例用法

ac config delete-context

示例用法

ac config delete-user

示例用法

ac config get-clusters

示例用法

ac config get-contexts

示例用法

ac config get-sessions

示例用法

ac config get-users

示例用法

ac config rename-context

示例用法

ac config set

示例用法

ac config set-cluster

示例用法

ac config set-context

示例用法

ac config set-credentials

示例用法

ac config sync

示例用法

ac config unset

示例用法

ac config use-cluster

示例用法

ac config use-context

示例用法

ac config use-session

示例用法

ac config view

示例用法

ac cp

示例用法

ac create

示例用法

ac create clusterrole

示例用法

ac create clusterrolebinding

示例用法

ac create configmap

示例用法

ac create cronjob

示例用法

ac create deployment

示例用法

ac create ingress

示例用法

ac create job

示例用法

ac create namespace

示例用法

ac create poddisruptionbudget

示例用法

ac create priorityclass

示例用法

ac create quota

示例用法

ac create role

示例用法

ac create rolebinding

示例用法

ac create secret

ac create secret docker-registry

示例用法

ac create secret generic

示例用法

ac create secret tls

示例用法

ac create service

ac create service clusterip

示例用法

ac create service externalname

示例用法

ac create service loadbalancer

示例用法

ac create service nodeport

示例用法

ac create serviceaccount

示例用法

ac create token

示例用法

ac delete

示例用法

ac describe

示例用法

ac diff

示例用法

ac edit

示例用法

ac events

示例用法

ac exec

示例用法

ac explain

示例用法

ac expose

示例用法

ac get

示例用法

ac kustomize

示例用法

ac label

示例用法

ac login

示例用法

ac logout

示例用法

ac logs

示例用法

ac namespace

示例用法

ac patch

示例用法

ac plugin

示例用法

ac plugin list

示例用法

ac port-forward

示例用法

ac process

示例用法

ac proxy

示例用法

ac replace

示例用法

ac rollout

示例用法

ac rollout history

示例用法

ac rollout pause

示例用法

ac rollout restart

示例用法

ac rollout resume

示例用法

ac rollout status

示例用法

ac rollout undo

示例用法

ac run

示例用法

ac scale

示例用法

ac set

ac set env

示例用法

ac set image

示例用法

ac set resources

示例用法

ac set selector

示例用法

ac set serviceaccount

示例用法

ac set subject

示例用法

ac version

示例用法

ac wait

示例用法

ac whoami

示例用法

ac annotate

更新资源上的注解

示例用法

```
# Update pod 'foo' with the annotation 'description' and the value 'my fr
ontend'
# If the same annotation is set multiple times, only the last value will
be applied
ac annotate pods foo description='my frontend'

# Update a pod identified by type and name in "pod.json"
ac annotate -f pod.json description='my frontend'

# Update pod 'foo' with the annotation 'description' and the value 'my fr
ontend running nginx', overwriting any existing value
ac annotate --overwrite pods foo description='my frontend running nginx'

# Update all pods in the namespace
ac annotate pods --all description='my frontend running nginx'

# Update pod 'foo' only if the resource is unchanged from version 1
ac annotate pods foo description='my frontend running nginx' --resource-v
ersion=1

# Update pod 'foo' by removing an annotation named 'description' if it ex
ists
# Does not require the --overwrite flag
ac annotate pods foo description-
```

ac api-resources

打印服务器上支持的 API 资源

示例用法

```
# Print the supported API resources
ac api-resources

# Print the supported API resources with more information
ac api-resources -o wide

# Print the supported API resources sorted by a column
ac api-resources --sort-by=name

# Print the supported namespaced resources
ac api-resources --namespaced=true

# Print the supported non-namespaced resources
ac api-resources --namespaced=false

# Print the supported API resources with a specific APIGroup
ac api-resources --api-group=rbac.authorization.k8s.io
```

ac api-versions

以“group/version”格式打印服务器上支持的 API 版本

示例用法

```
# 打印支持的 API 版本
ac api-versions
```

ac apply

通过文件名或 stdin 将配置应用到资源

示例用法

```
# Apply the configuration in pod.json to a pod
ac apply -f ./pod.json

# Apply resources from a directory containing kustomization.yaml - e.g. d
ir/kustomization.yaml
ac apply -k dir/

# Apply the JSON passed into stdin to a pod
cat pod.json | ac apply -f -

# Apply the configuration from all files that end with '.json'
ac apply -f '*.json'

# Note: --prune is still in Alpha
# Apply the configuration in manifest.yaml that matches label app=nginx a
nd delete all other resources that are not in the file and match label ap
p=nginx
ac apply --prune -f manifest.yaml -l app=nginx

# Apply the configuration in manifest.yaml and delete all the other confi
g maps that are not in the file
ac apply --prune -f manifest.yaml --all --prune-allowlist=core/v1/ConfigM
ap
```

ac apply edit-last-applied

编辑资源/对象最近的 last-applied-configuration 注解

示例用法

```
# Edit the last-applied-configuration annotations by type/name in YAML
ac apply edit-last-applied deployment/nginx

# Edit the last-applied-configuration annotations by file in JSON
ac apply edit-last-applied -f deploy.yaml -o json
```

ac apply set-last-applied

将活动对象上的 last-applied-configuration 注解设置为与文件内容匹配

示例用法

```
# Set the last-applied-configuration of a resource to match the contents
of a file
ac apply set-last-applied -f deploy.yaml

# Execute set-last-applied against each configuration file in a directory
ac apply set-last-applied -f path/

# Set the last-applied-configuration of a resource to match the contents
of a file; will create the annotation if it does not already exist
ac apply set-last-applied -f deploy.yaml --create-annotation=true
```

ac apply view-last-applied

查看资源/对象最近的 last-applied-configuration 注解

示例用法

```
# View the last-applied-configuration annotations by type/name in YAML
ac apply view-last-applied deployment/nginx

# View the last-applied-configuration annotations by file in JSON
ac apply view-last-applied -f deploy.yaml -o json
```

ac attach

附加到正在运行的容器

示例用法

```
# Get output from running pod mypod; use the 'ac.kubernetes.io/default-co
ntainer' annotation
# for selecting the container to be attached or the first container in th
e pod will be chosen
ac attach mypod

# Get output from ruby-container from pod mypod
ac attach mypod -c ruby-container

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container fr
om pod mypod
# and sends stdout/stderr from 'bash' back to the client
ac attach mypod -c ruby-container -i -t

# Get output from the first pod of a replica set named nginx
ac attach rs/nginx
```

ac auth

检查授权

ac auth can-i

检查某项操作是否被允许

示例用法

```
# Check to see if I can create pods in any namespace
ac auth can-i create pods --all-namespaces

# Check to see if I can list deployments in my current namespace
ac auth can-i list deployments.apps

# Check to see if service account "foo" of namespace "dev" can list pods
in the namespace "prod"
# You must be allowed to use impersonation for the global option "--as"
ac auth can-i list pods --as=system:serviceaccount:dev:foo -n prod

# Check to see if I can do everything in my current namespace ("*" means
all)
ac auth can-i '*' '*'

# Check to see if I can get the job named "bar" in namespace "foo"
ac auth can-i list jobs.batch/bar -n foo

# Check to see if I can read pod logs
ac auth can-i get pods --subresource=log

# Check to see if I can access the URL /logs/
ac auth can-i get /logs/

# Check to see if I can approve certificates.k8s.io
ac auth can-i approve certificates.k8s.io

# List all allowed actions in namespace "foo"
ac auth can-i --list --namespace=foo
```

ac auth reconcile

协调 RBAC role、role binding、cluster role 和 cluster role binding 对象的规则

示例用法

```
# Reconcile RBAC resources from a file
ac auth reconcile -f my-rbac-rules.yaml
```

ac auth whoami

实验性：检查自身主体属性

示例用法

```
# Get your subject attributes
ac auth whoami

# Get your subject attributes in JSON format
ac auth whoami -o json
```

ac autoscale

自动扩缩 deployment、replica set、stateful set 或 replication controller

示例用法

```
# Auto scale a deployment "foo", with the number of pods between 2 and 1
0, no target CPU utilization specified so a default autoscaling policy wi
ll be used
ac autoscale deployment foo --min=2 --max=10

# Auto scale a replication controller "foo", with the number of pods betw
een 1 and 5, target CPU utilization at 80%
ac autoscale rc foo --max=5 --cpu-percent=80
```

ac cluster-info

显示集群信息

示例用法

```
# Print the address of the control plane and cluster services
ac cluster-info
```

ac cluster-info dump

转储用于调试和诊断的相关信息

示例用法

```
# Dump current cluster state to stdout
ac cluster-info dump

# Dump current cluster state to /path/to/cluster-state
ac cluster-info dump --output-directory=/path/to/cluster-state

# Dump all namespaces to stdout
ac cluster-info dump --all-namespaces

# Dump a set of namespaces to /path/to/cluster-state
ac cluster-info dump --namespaces default,kube-system --output-directory
=/path/to/cluster-state
```

ac completion

为指定 shell（bash、zsh、fish 或 powershell）输出 shell 补全代码

示例用法

```
# Installing bash completion on macOS using homebrew
## If running Bash 3.2 included with macOS
brew install bash-completion
## or, if running Bash 4.1+
brew install bash-completion@2
## If ac is installed via homebrew, this should start working immediately
## If you've installed via other means, you may need add the completion t
o your completion directory
ac completion bash > $(brew --prefix)/etc/bash_completion.d/ac


# Installing bash completion on Linux
## If bash-completion is not installed on Linux, install the 'bash-comple
tion' package
## via your distribution's package manager.
## Load the ac completion code for bash into the current shell
source <(ac completion bash)
## Write bash completion code to a file and source it from .bash_profile
ac completion bash > ~/.kube/completion.bash.inc
printf "
# ac shell completion
source '$HOME/.kube/completion.bash.inc'
" >> $HOME/.bash_profile
source $HOME/.bash_profile

# Load the ac completion code for zsh[1] into the current shell
source <(ac completion zsh)
# Set the ac completion code for zsh[1] to autoload on startup
ac completion zsh > "${fpath[1]}/_ac"


# Load the ac completion code for fish[2] into the current shell
ac completion fish | source
# To load completions for each session, execute once:
ac completion fish > ~/.config/fish/completions/ac.fish

# Load the ac completion code for powershell into the current shell
ac completion powershell | Out-String | Invoke-Expression
# Set ac completion code for powershell to run on startup
## Save completion code to a script and execute in the profile
ac completion powershell > $HOME\.kube\completion.ps1
Add-Content $PROFILE "$HOME\.kube\completion.ps1"
## Execute completion code in the profile
```

```
Add-Content $PROFILE "if (Get-Command ac -ErrorAction SilentlyContinue) {
ac completion powershell | Out-String | Invoke-Expression
}"
## Add completion code directly to the $PROFILE script
ac completion powershell >> $PROFILE
```

ac config

修改 kubeconfig 文件

ac config current-context

显示当前上下文

示例用法

```
# Display the current-context
ac config current-context
```

ac config delete-cluster

从 kubeconfig 中删除指定的集群

示例用法

```
# Delete the minikube cluster
ac config delete-cluster minikube
```

ac config delete-context

从 kubeconfig 中删除指定的上下文

示例用法

```
# Delete the context for the minikube cluster
ac config delete-context minikube
```

ac config delete-user

从 kubeconfig 中删除指定的用户

示例用法

```
# Delete the minikube user
ac config delete-user minikube
```

ac config get-clusters

显示 kubeconfig 中定义的集群

示例用法

```
# List the clusters that ac knows about
ac config get-clusters
```

ac config get-contexts

描述一个或多个上下文

示例用法

```
# List all the contexts in your kubeconfig file
ac config get-contexts

# Describe one context in your kubeconfig file
ac config get-contexts my-context
```

ac config get-sessions

列出所有已配置的 ACP platform 会话

示例用法

```
# List all configured ACP sessions
ac config get-sessions
```

ac config get-users

显示 kubeconfig 中定义的用户

示例用法

```
# List the users that ac knows about
ac config get-users
```

ac config rename-context

重命名 kubeconfig 文件中的上下文

示例用法

```
# Rename the context 'old-name' to 'new-name' in your kubeconfig file
ac config rename-context old-name new-name
```

ac config set

在 kubeconfig 文件中设置单个值

示例用法

```
# Set the server field on the my-cluster cluster to https://1.2.3.4
ac config set clusters.my-cluster.server https://1.2.3.4

# Set the certificate-authority-data field on the my-cluster cluster
ac config set clusters.my-cluster.certificate-authority-data $(echo "cert
_data_here" | base64 -i -)

# Set the cluster field in the my-context context to my-cluster
ac config set contexts.my-context.cluster my-cluster

# Set the client-key-data field in the cluster-admin user using --set-raw
-bytes option
ac config set users.cluster-admin.client-key-data cert_data_here --set-ra
w-bytes=true
```

ac config set-cluster

在 kubeconfig 中设置集群条目

示例用法

```
# Set only the server field on the e2e cluster entry without touching oth
er values
ac config set-cluster e2e --server=https://1.2.3.4

# Embed certificate authority data for the e2e cluster entry
ac config set-cluster e2e --embed-certs --certificate-authority=~/.kube/e
2e/kubernetes.ca.crt

# Disable cert checking for the e2e cluster entry
ac config set-cluster e2e --insecure-skip-tls-verify=true

# Set the custom TLS server name to use for validation for the e2e cluste
r entry
ac config set-cluster e2e --tls-server-name=my-cluster-name

# Set the proxy URL for the e2e cluster entry
ac config set-cluster e2e --proxy-url=https://1.2.3.4
```

ac config set-context

在 kubeconfig 中设置上下文条目

示例用法

```
# Set the user field on the gce context entry without touching other valu
es
ac config set-context gce --user=cluster-admin
```

ac config set-credentials

在 kubeconfig 中设置用户条目

示例用法

```
# Set only the "client-key" field on the "cluster-admin"
# entry, without touching other values
ac config set-credentials cluster-admin --client-key=~/.kube/admin.key

# Set basic auth for the "cluster-admin" entry
ac config set-credentials cluster-admin --username=admin --password=uXFGw
eU9l35qcif

# Embed client certificate data in the "cluster-admin" entry
ac config set-credentials cluster-admin --client-certificate=~/.kube/admi
n.crt --embed-certs=true

# Enable the Google Compute Platform auth provider for the "cluster-admi
n" entry
ac config set-credentials cluster-admin --auth-provider=gcp

# Enable the OpenID Connect auth provider for the "cluster-admin" entry w
ith additional arguments
ac config set-credentials cluster-admin --auth-provider=oidc --auth-provi
der-arg=client-id=foo --auth-provider-arg=client-secret=bar

# Remove the "client-secret" config value for the OpenID Connect auth pro
vider for the "cluster-admin" entry
ac config set-credentials cluster-admin --auth-provider=oidc --auth-provi
der-arg=client-secret-

# Enable new exec auth plugin for the "cluster-admin" entry
ac config set-credentials cluster-admin --exec-command=/path/to/the/execu
table --exec-api-version=client.authentication.k8s.io/v1beta1

# Enable new exec auth plugin for the "cluster-admin" entry with interact
ive mode
ac config set-credentials cluster-admin --exec-command=/path/to/the/execu
table --exec-api-version=client.authentication.k8s.io/v1beta1 --exec-inte
ractive-mode=Never

# Define new exec auth plugin arguments for the "cluster-admin" entry
ac config set-credentials cluster-admin --exec-arg=arg1 --exec-arg=arg2

# Create or update exec auth plugin environment variables for the "cluste
r-admin" entry
ac config set-credentials cluster-admin --exec-env=key1=val1 --exec-env=k
ey2=val2
```

```
# Remove exec auth plugin environment variables for the "cluster-admin" e
ntry
ac config set-credentials cluster-admin --exec-env=var-to-remove-
```

ac config sync

将 kubeconfig 与 ACP platform 状态同步

示例用法

```
# Sync current session based on active context
ac config sync

# Sync specific session
ac config sync --session prod

# Sync all sessions
ac config sync --all
```

ac config unset

取消设置 kubeconfig 文件中的单个值

示例用法

```
# Unset the current-context
ac config unset current-context

# Unset namespace in foo context
ac config unset contexts.foo.namespace
```

ac config use-cluster

按集群名称切换到特定的 ACP 集群

示例用法

```
# Switch to existing context for workload-a cluster
ac config use-cluster workload-a

# Create new context for workload-b with namespace
ac config use-cluster workload-b --namespace my-app

# Switch to global cluster
ac config use-cluster global
```

ac config use-context

在 kubeconfig 文件中设置当前上下文

示例用法

```
# Use the context for the minikube cluster
ac config use-context minikube
```

ac config use-session

使用智能上下文选择切换到指定的 ACP 会话

示例用法

```
# Switch to staging session (default global cluster)
ac config use-session staging

# Switch to production session with specific cluster
ac config use-session prod --cluster workload-b

# Switch to session with specific cluster and namespace
ac config use-session staging --cluster workload-a --namespace my-app
```

ac config view

显示合并后的 kubeconfig 设置或指定的 kubeconfig 文件

示例用法

```
# Show merged kubeconfig settings
ac config view

# Show merged kubeconfig settings, raw certificate data, and exposed secr
ets
ac config view --raw

# Get the password for the e2e user
ac config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'
```

ac cp

在容器与本地之间复制文件和目录

示例用法

```
# !!!Important Note!!!
# Requires that the 'tar' binary is present in your container
# image.  If 'tar' is not present, 'ac cp' will fail.
#
# For advanced use cases, such as symlinks, wildcard expansion or
# file mode preservation, consider using 'ac exec'.

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some
-namespace>
tar cf - /tmp/foo | ac exec -i -n <some-namespace> <some-pod> -- tar xf -
-C /tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
ac exec -n <some-namespace> <some-pod> -- tar cf - /tmp/foo | tar xf - -C
/tmp/bar

# Copy /tmp/foo_dir local directory to /tmp/bar_dir in a remote pod in th
e default namespace
ac cp /tmp/foo_dir <some-pod>:/tmp/bar_dir

# Copy /tmp/foo local file to /tmp/bar in a remote pod in a specific cont
ainer
ac cp /tmp/foo <some-pod>:/tmp/bar -c <specific-container>

# Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace <some
-namespace>
ac cp /tmp/foo <some-namespace>/<some-pod>:/tmp/bar

# Copy /tmp/foo from a remote pod to /tmp/bar locally
ac cp <some-namespace>/<some-pod>:/tmp/foo /tmp/bar
```

ac create

从文件或 stdin 创建资源

示例用法

```
# Create a pod using the data in pod.json
ac create -f ./pod.json

# Create a pod based on the JSON passed into stdin
cat pod.json | ac create -f -

# Edit the data in registry.yaml in JSON then create the resource using t
he edited data
ac create -f registry.yaml --edit -o json
```

ac create clusterrole

创建 cluster role

示例用法

```
# Create a cluster role named "pod-reader" that allows user to perform "g
et", "watch" and "list" on pods
ac create clusterrole pod-reader --verb=get,list,watch --resource=pods

# Create a cluster role named "pod-reader" with ResourceName specified
ac create clusterrole pod-reader --verb=get --resource=pods --resource-na
me=readablepod --resource-name=anotherpod

# Create a cluster role named "foo" with API Group specified
ac create clusterrole foo --verb=get,list,watch --resource=rs.apps

# Create a cluster role named "foo" with SubResource specified
ac create clusterrole foo --verb=get,list,watch --resource=pods,pods/stat
us

# Create a cluster role name "foo" with NonResourceURL specified
ac create clusterrole "foo" --verb=get --non-resource-url=/logs/*

# Create a cluster role name "monitoring" with AggregationRule specified
ac create clusterrole monitoring --aggregation-rule="rbac.example.com/agg
regate-to-monitoring=true"
```

ac create clusterrolebinding

为特定的 cluster role 创建 cluster role binding

示例用法

```
# Create a cluster role binding for user1, user2, and group1 using the cl
uster-admin cluster role
ac create clusterrolebinding cluster-admin --clusterrole=cluster-admin --
user=user1 --user=user2 --group=group1
```

ac create configmap

从本地文件、目录或字面值创建 config map

示例用法

```
# Create a new config map named my-config based on folder bar
ac create configmap my-config --from-file=path/to/bar

# Create a new config map named my-config with specified keys instead of
file basenames on disk
ac create configmap my-config --from-file=key1=/path/to/bar/file1.txt --f
rom-file=key2=/path/to/bar/file2.txt

# Create a new config map named my-config with key1=config1 and key2=conf
ig2
ac create configmap my-config --from-literal=key1=config1 --from-literal=
key2=config2

# Create a new config map named my-config from the key=value pairs in the
file
ac create configmap my-config --from-file=path/to/bar

# Create a new config map named my-config from an env file
ac create configmap my-config --from-env-file=path/to/foo.env --from-env-
file=path/to/bar.env
```

ac create cronjob

创建指定名称的 cron job

示例用法

```
# Create a cron job
ac create cronjob my-job --image=busybox --schedule="*/1 * * * *"

# Create a cron job with a command
ac create cronjob my-job --image=busybox --schedule="*/1 * * * *" -- date
```

ac create deployment

创建指定名称的 deployment

示例用法

```
# Create a deployment named my-dep that runs the busybox image
ac create deployment my-dep --image=busybox

# Create a deployment with a command
ac create deployment my-dep --image=busybox -- date

# Create a deployment named my-dep that runs the nginx image with 3 repli
cas
ac create deployment my-dep --image=nginx --replicas=3

# Create a deployment named my-dep that runs the busybox image and expose
port 5701
ac create deployment my-dep --image=busybox --port=5701

# Create a deployment named my-dep that runs multiple containers
ac create deployment my-dep --image=busybox:latest --image=ubuntu:latest
--image=nginx
```

ac create ingress

创建指定名称的 ingress

示例用法

```
# Create a single ingress called 'simple' that directs requests to foo.com/bar to svc
# svc1:8080 with a TLS secret "my-cert"
ac create ingress simple --rule="foo.com/bar=svc1:8080,tls=my-cert"

# Create a catch all ingress of "/path" pointing to service svc:port and Ingress Class as "otheringress"
ac create ingress catch-all --class=otheringress --rule="/path=svc:port"

# Create an ingress with two annotations: ingress.annotation1 and ingress.annotations2
ac create ingress annotated --class=default --rule="foo.com/bar=svc:port" \
--annotation ingress.annotation1=foo \
--annotation ingress.annotation2=bla

# Create an ingress with the same host and multiple paths
ac create ingress multipath --class=default \
--rule="foo.com/=svc:port" \
--rule="foo.com/admin/=svcadmin:portadmin"

# Create an ingress with multiple hosts and the pathType as Prefix
ac create ingress ingress1 --class=default \
--rule="foo.com/path*=svc:8080" \
--rule="bar.com/admin*=svc2:http"

# Create an ingress with TLS enabled using the default ingress certificate and different path types
ac create ingress ingtls --class=default \
--rule="foo.com/=svc:https,tls" \
--rule="foo.com/path/subpath*=othersvc:8080"

# Create an ingress with TLS enabled using a specific secret and pathType as Prefix
ac create ingress ingsecret --class=default \
--rule="foo.com/*=svc:8080,tls=secret1"

# Create an ingress with a default backend
ac create ingress ingdefault --class=default \
--default-backend=defaultsvc:http \
--rule="foo.com/*=svc:8080,tls=secret1"
```

ac create job

创建指定名称的 job

示例用法

```
# Create a job
ac create job my-job --image=busybox

# Create a job with a command
ac create job my-job --image=busybox -- date

# Create a job from a cron job named "a-cronjob"
ac create job test-job --from=cronjob/a-cronjob
```

ac create namespace

创建指定名称的 namespace

示例用法

```
# Create a new namespace named my-namespace
ac create namespace my-namespace
```

ac create poddisruptionbudget

创建指定名称的 pod disruption budget

示例用法

```
# Create a pod disruption budget named my-pdb that will select all pods w
ith the app=rails label
# and require at least one of them being available at any point in time
ac create poddisruptionbudget my-pdb --selector=app=rails --min-available
=1

# Create a pod disruption budget named my-pdb that will select all pods w
ith the app=nginx label
# and require at least half of the pods selected to be available at any p
oint in time
ac create pdb my-pdb --selector=app=nginx --min-available=50%
```

ac create priorityclass

创建指定名称的 priority class

示例用法

```
# Create a priority class named high-priority
ac create priorityclass high-priority --value=1000 --description="high pr
iority"

# Create a priority class named default-priority that is considered as th
e global default priority
ac create priorityclass default-priority --value=1000 --global-default=tr
ue --description="default priority"

# Create a priority class named high-priority that cannot preempt pods wi
th lower priority
ac create priorityclass high-priority --value=1000 --description="high pr
iority" --preemption-policy="Never"
```

ac create quota

创建指定名称的 quota

示例用法

```
# Create a new resource quota named my-quota
ac create quota my-quota --hard=cpu=1,memory=1G,pods=2,services=3,replica
tioncontrollers=2,resourcequotas=1,secrets=5,persistentvolumeclaims=10

# Create a new resource quota named best-effort
ac create quota best-effort --hard=pods=100 --scopes=BestEffort
```

ac create role

创建单条规则的 role

示例用法

```
# Create a role named "pod-reader" that allows user to perform "get", "wa
tch" and "list" on pods
ac create role pod-reader --verb=get --verb=list --verb=watch --resource=
pods

# Create a role named "pod-reader" with ResourceName specified
ac create role pod-reader --verb=get --resource=pods --resource-name=read
ablepod --resource-name=anotherpod

# Create a role named "foo" with API Group specified
ac create role foo --verb=get,list,watch --resource=rs.apps

# Create a role named "foo" with SubResource specified
ac create role foo --verb=get,list,watch --resource=pods,pods/status
```

ac create rolebinding

为特定的 role 或 cluster role 创建 role binding

示例用法

```
# Create a role binding for user1, user2, and group1 using the admin clus
ter role
ac create rolebinding admin --clusterrole=admin --user=user1 --user=user2
--group=group1

# Create a role binding for service account monitoring:sa-dev using the a
dmin role
ac create rolebinding admin-binding --role=admin --serviceaccount=monitor
ing:sa-dev
```

ac create secret

使用指定子命令创建 secret

ac create secret docker-registry

创建用于 Docker registry 的 secret

示例用法

```
# If you do not already have a .dockercfg file, create a dockercfg secret
directly
ac create secret docker-registry my-secret --docker-server=DOCKER_REGISTR
Y_SERVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD
--docker-email=DOCKER_EMAIL

# Create a new secret named my-secret from ~/.docker/config.json
ac create secret docker-registry my-secret --from-file=path/to/.docker/co
nfig.json
```

ac create secret generic

从本地文件、目录或字面值创建 secret

示例用法

```
# Create a new secret named my-secret with keys for each file in folder b
ar
ac create secret generic my-secret --from-file=path/to/bar

# Create a new secret named my-secret with specified keys instead of name
s on disk
ac create secret generic my-secret --from-file=ssh-privatekey=path/to/id_
rsa --from-file=ssh-publickey=path/to/id_rsa.pub

# Create a new secret named my-secret with key1=supersecret and key2=tops
ecret
ac create secret generic my-secret --from-literal=key1=supersecret --from
-literal=key2=topsecret

# Create a new secret named my-secret using a combination of a file and a
literal
ac create secret generic my-secret --from-file=ssh-privatekey=path/to/id_
rsa --from-literal=passphrase=topsecret

# Create a new secret named my-secret from env files
ac create secret generic my-secret --from-env-file=path/to/foo.env --from
-env-file=path/to/bar.env
```

ac create secret tls

创建 TLS secret

示例用法

```
# Create a new TLS secret named tls-secret with the given key pair
ac create secret tls tls-secret --cert=path/to/tls.crt --key=path/to/tls.
key
```

ac create service

使用指定子命令创建 service

ac create service clusterip

创建 ClusterIP 服务

示例用法

```
# Create a new ClusterIP service named my-cs
ac create service clusterip my-cs --tcp=5678:8080

# Create a new ClusterIP service named my-cs (in headless mode)
ac create service clusterip my-cs --clusterip="None"
```

ac create service externalname

创建 ExternalName 服务

示例用法

```
# Create a new ExternalName service named my-ns
ac create service externalname my-ns --external-name bar.com
```

ac create service loadbalancer

创建 LoadBalancer 服务

示例用法

```
# Create a new LoadBalancer service named my-lbs
ac create service loadbalancer my-lbs --tcp=5678:8080
```

ac create service nodeport

创建 NodePort 服务

示例用法

```
# Create a new NodePort service named my-ns
ac create service nodeport my-ns --tcp=5678:8080
```

ac create serviceaccount

创建指定名称的 service account

示例用法

```
# Create a new service account named my-service-account
ac create serviceaccount my-service-account
```

ac create token

请求 service account 令牌

示例用法

```
# Request a token to authenticate to the kube-apiserver as the service ac
count "myapp" in the current namespace
ac create token myapp

# Request a token for a service account in a custom namespace
ac create token myapp --namespace myns

# Request a token with a custom expiration
ac create token myapp --duration 10m

# Request a token with a custom audience
ac create token myapp --audience https://example.com

# Request a token bound to an instance of a Secret object
ac create token myapp --bound-object-kind Secret --bound-object-name myse
cret

# Request a token bound to an instance of a Secret object with a specific
UID
ac create token myapp --bound-object-kind Secret --bound-object-name myse
cret --bound-object-uid 0d4691ed-659b-4935-a832-355f77ee47cc
```

ac delete

根据文件名、stdin、资源和名称，或根据资源和标签选择器删除资源

示例用法

```
# Delete a pod using the type and name specified in pod.json
ac delete -f ./pod.json

# Delete resources from a directory containing kustomization.yaml - e.g.
dir/kustomization.yaml
ac delete -k dir

# Delete resources from all files that end with '.json'
ac delete -f '*.json'

# Delete a pod based on the type and name in the JSON passed into stdin
cat pod.json | ac delete -f -

# Delete pods and services with same names "baz" and "foo"
ac delete pod,service baz foo

# Delete pods and services with label name=myLabel
ac delete pods,services -l name=myLabel

# Delete a pod with minimal delay
ac delete pod foo --now

# Force delete a pod on a dead node
ac delete pod foo --force

# Delete all pods
ac delete pods --all

# Delete all pods only if the user confirms the deletion
ac delete pods --all --interactive
```

ac describe

显示特定资源或资源组的详细信息

示例用法

```
# Describe a node
ac describe nodes kubernetes-node-emt8.c.myproject.internal

# Describe a pod
ac describe pods/nginx

# Describe a pod identified by type and name in "pod.json"
ac describe -f pod.json

# Describe all pods
ac describe pods

# Describe pods by label name=myLabel
ac describe pods -l name=myLabel

# Describe all pods managed by the 'frontend' replication controller
# (rc-created pods get the name of the rc as a prefix in the pod name)
ac describe pods frontend
```

ac diff

将当前运行版本与将要应用的版本进行差异比较

示例用法

```
# Diff resources included in pod.json
ac diff -f pod.json

# Diff file read from stdin
cat service.yaml | ac diff -f -
```

ac edit

编辑服务器上的资源

示例用法

```
# Edit the service named 'registry'
ac edit svc/registry

# Use an alternative editor
KUBE_EDITOR="nano" ac edit svc/registry

# Edit the job 'myjob' in JSON using the v1 API format
ac edit job.v1.batch/myjob -o json

# Edit the deployment 'mydeployment' in YAML and save the modified config
in its annotation
ac edit deployment/mydeployment -o yaml --save-config

# Edit the 'status' subresource for the 'mydeployment' deployment
ac edit deployment mydeployment --subresource='status'
```

ac events

列出事件

示例用法

```
# List recent events in the default namespace
ac events

# List recent events in all namespaces
ac events --all-namespaces

# List recent events for the specified pod, then wait for more events and
list them as they arrive
ac events --for pod/web-pod-13je7 --watch

# List recent events in YAML format
ac events -oyaml

# List recent only events of type 'Warning' or 'Normal'
ac events --types=Warning,Normal
```

ac exec

在容器中执行命令

示例用法

```
# Get output from running the 'date' command from pod mypod, using the fi
rst container by default
ac exec mypod -- date

# Get output from running the 'date' command in ruby-container from pod m
ypod
ac exec mypod -c ruby-container -- date

# Switch to raw terminal mode; sends stdin to 'bash' in ruby-container fr
om pod mypod
# and sends stdout/stderr from 'bash' back to the client
ac exec mypod -c ruby-container -i -t -- bash -il

# List contents of /usr from the first container of pod mypod and sort by
modification time
# If the command you want to execute in the pod has any flags in common
(e.g. -i),
# you must use two dashes (--) to separate your command's flags/arguments
# Also note, do not surround your command and its flags/arguments with qu
otes
# unless that is how you would execute it normally (i.e., do ls -t /usr,
not "ls -t /usr")
ac exec mypod -i -t -- ls -t /usr

# Get output from running 'date' command from the first pod of the deploy
ment mydeployment, using the first container by default
ac exec deploy/mydeployment -- date

# Get output from running 'date' command from the first pod of the servic
e myservice, using the first container by default
ac exec svc/myservice -- date
```

ac explain

获取资源文档

示例用法

```
# Get the documentation of the resource and its fields
ac explain pods

# Get all the fields in the resource
ac explain pods --recursive

# Get the explanation for deployment in supported api versions
ac explain deployments --api-version=apps/v1

# Get the documentation of a specific field of a resource
ac explain pods.spec.containers

# Get the documentation of resources in different format
ac explain deployment --output=plaintext-openapiv2
```

ac expose

将 replication controller、service、deployment 或 pod 暴露为新的 Kubernetes service

示例用法

```
# Create a service for a replicated nginx, which serves on port 80 and co
nnects to the containers on port 8000
ac expose rc nginx --port=80 --target-port=8000

# Create a service for a replication controller identified by type and na
me specified in "nginx-controller.yaml", which serves on port 80 and conn
ects to the containers on port 8000
ac expose -f nginx-controller.yaml --port=80 --target-port=8000

# Create a service for a pod valid-pod, which serves on port 444 with the
name "frontend"
ac expose pod valid-pod --port=444 --name=frontend

# Create a second service based on the above service, exposing the contai
ner port 8443 as port 443 with the name "nginx-https"
ac expose service nginx --port=443 --target-port=8443 --name=nginx-https

# Create a service for a replicated streaming application on port 4100 ba
lancing UDP traffic and named 'video-stream'.
ac expose rc streamer --port=4100 --protocol=UDP --name=video-stream

# Create a service for a replicated nginx using replica set, which serves
on port 80 and connects to the containers on port 8000
ac expose rs nginx --port=80 --target-port=8000

# Create a service for an nginx deployment, which serves on port 80 and c
onnects to the containers on port 8000
ac expose deployment nginx --port=80 --target-port=8000
```

ac get

显示一个或多个资源

示例用法

```
# List all pods in ps output format
ac get pods

# List all pods in ps output format with more information (such as node n
ame)
ac get pods -o wide

# List a single replication controller with specified NAME in ps output f
ormat
ac get replicationcontroller web

# List deployments in JSON output format, in the "v1" version of the "app
s" API group
ac get deployments.v1.apps -o json

# List a single pod in JSON output format
ac get -o json pod web-pod-13je7

# List a pod identified by type and name specified in "pod.yaml" in JSON
output format
ac get -f pod.yaml -o json

# List resources from a directory with kustomization.yaml - e.g. dir/kust
omization.yaml
ac get -k dir/

# Return only the phase value of the specified pod
ac get -o template pod/web-pod-13je7 --template={{.status.phase}}

# List resource information in custom columns
ac get pod test-pod -o custom-columns=CONTAINER:.spec.containers[0].name,
IMAGE:.spec.containers[0].image

# List all replication controllers and services together in ps output for
mat
ac get rc,services

# List one or more resources by their type and names
ac get rc/web service/frontend pods/web-pod-13je7

# List the 'status' subresource for a single pod
ac get pod web-pod-13je7 --subresource status
```

```
# List all deployments in namespace 'backend'
ac get deployments.apps --namespace backend

# List all pods existing in all namespaces
ac get pods --all-namespaces
```

ac kustomize

从目录或 URL 构建 kustomization 目标

示例用法

```
# Build the current working directory
ac kustomize

# Build some shared configuration directory
ac kustomize /home/config/production

# Build from github
ac kustomize https://github.com/kubernetes-sigs/kustomize.git/examples/he
lloWorld?ref=v1.0.6
```

ac label

更新资源上的标签

示例用法

```
# Update pod 'foo' with the label 'unhealthy' and the value 'true'
ac label pods foo unhealthy=true

# Update pod 'foo' with the label 'status' and the value 'unhealthy', ove
rwriting any existing value
ac label --overwrite pods foo status=unhealthy

# Update all pods in the namespace
ac label pods --all status=unhealthy

# Update a pod identified by the type and name in "pod.json"
ac label -f pod.json status=unhealthy

# Update pod 'foo' only if the resource is unchanged from version 1
ac label pods foo status=unhealthy --resource-version=1

# Update pod 'foo' by removing a label named 'bar' if it exists
# Does not require the --overwrite flag
ac label pods foo bar-
```

ac login

登录到 ACP platform

示例用法

```
# Interactive login (prompts for missing parameters)
ac login https://example.com --name prod

# Login with all parameters via flags
ac login https://example.com --name prod --username=myuser --password=myp
assword

# Login using environment variables (automation-friendly)
AC_LOGIN_PLATFORM_URL=https://example.com AC_LOGIN_SESSION=prod \
AC_LOGIN_USERNAME=myuser AC_LOGIN_PASSWORD=mypassword ac login

# Login with specific identity provider
ac login https://example.com --name prod --idp ldap-test

# Login and set specific cluster and namespace
ac login https://example.com --name prod --cluster=my-cluster --namespace
=my-namespace

# Login with custom kubeconfig file
ac login https://example.com --name prod --kubeconfig=/path/to/kubeconfig

# Login with workload cluster LDAP identity provider
ac login https://192.168.1.2:11780 --idp ldap-test --workload --auth-type
ldap --username 'xx' --password 'xx'

# Login with workload cluster OIDC identity provider
ac login https://192.168.1.2:11780 --idp oidc --workload --auth-type oidc
```

ac logout

结束与 ACP platform 的当前会话

示例用法

```
# Log out from current ACP platform session
ac logout

# Log out from specific session
ac logout --session prod

# Log out from all sessions
ac logout --all
```

ac logs

打印 pod 中容器的日志

示例用法

```
# Return snapshot logs from pod nginx with only one container
ac logs nginx

# Return snapshot logs from pod nginx, prefixing each line with the sourc
e pod and container name
ac logs nginx --prefix

# Return snapshot logs from pod nginx, limiting output to 500 bytes
ac logs nginx --limit-bytes=500

# Return snapshot logs from pod nginx, waiting up to 20 seconds for it to
start running.
ac logs nginx --pod-running-timeout=20s

# Return snapshot logs from pod nginx with multi containers
ac logs nginx --all-containers=true

# Return snapshot logs from all pods in the deployment nginx
ac logs deployment/nginx --all-pods=true

# Return snapshot logs from all containers in pods defined by label app=n
ginx
ac logs -l app=nginx --all-containers=true

# Return snapshot logs from all pods defined by label app=nginx, limiting
concurrent log requests to 10 pods
ac logs -l app=nginx --max-log-requests=10

# Return snapshot of previous terminated ruby container logs from pod web
-1
ac logs -p -c ruby web-1

# Begin streaming the logs from pod nginx, continuing even if errors occu
r
ac logs nginx -f --ignore-errors=true

# Begin streaming the logs of the ruby container in pod web-1
ac logs -f -c ruby web-1

# Begin streaming the logs from all containers in pods defined by label a
pp=nginx
ac logs -f -l app=nginx --all-containers=true
```

```
# Display only the most recent 20 lines of output in pod nginx
ac logs --tail=20 nginx

# Show all logs from pod nginx written in the last hour
ac logs --since=1h nginx

# Show all logs with timestamps from pod nginx starting from August 30, 2024, at 06:00:00 UTC
ac logs nginx --since-time=2024-08-30T06:00:00Z --timestamps=true

# Show logs from a kubelet with an expired serving certificate
ac logs --insecure-skip-tls-verify-backend nginx

# Return snapshot logs from first container of a job named hello
ac logs job/hello

# Return snapshot logs from container nginx-1 of a deployment named nginx
ac logs deployment/nginx -c nginx-1
```

ac namespace

显示或切换当前 namespace 上下文

示例用法

```
# Show current namespace and context information
ac namespace

# Switch to a different namespace
ac namespace my-namespace

# Switch to default namespace
ac namespace default
```

ac patch

更新资源字段

示例用法

```
# Partially update a node using a strategic merge patch, specifying the p
atch as JSON
ac patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

# Partially update a node using a strategic merge patch, specifying the p
atch as YAML
ac patch node k8s-node-1 -p $'spec:\n unschedulable: true'

# Partially update a node identified by the type and name specified in "n
ode.json" using strategic merge patch
ac patch -f node.json -p '{"spec":{"unschedulable":true}}'

# Update a container's image; spec.containers[*].name is required because
it's a merge key
ac patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-ser
ve-hostname","image":"new image"}]}}'

# Update a container's image using a JSON patch with positional arrays
ac patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spe
c/containers/0/image", "value":"new image"}]'

# Update a deployment's replicas through the 'scale' subresource using a
merge patch
ac patch deployment nginx-deployment --subresource='scale' --type='merge'
-p '{"spec":{"replicas":2}}'
```

ac plugin

提供与插件交互的实用工具

示例用法

```
# List all available plugins
ac plugin list

# List only binary names of available plugins without paths
ac plugin list --name-only
```

ac plugin list

列出用户 PATH 上所有可见的插件可执行文件

示例用法


```
# List all available plugins
ac plugin list

# List only binary names of available plugins without paths
ac plugin list --name-only
```


ac port-forward

将一个或多个本地端口转发到 pod

示例用法

```
# Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in the pod
ac port-forward pod/mypod 5000 6000

# Listen on ports 5000 and 6000 locally, forwarding data to/from ports 5000 and 6000 in a pod selected by the deployment
ac port-forward deployment/mydeployment 5000 6000

# Listen on port 8443 locally, forwarding to the targetPort of the service's port named "https" in a pod selected by the service
ac port-forward service/myservice 8443:https

# Listen on port 8888 locally, forwarding to 5000 in the pod
ac port-forward pod/mypod 8888:5000

# Listen on port 8888 on all addresses, forwarding to 5000 in the pod
ac port-forward --address 0.0.0.0 pod/mypod 8888:5000

# Listen on port 8888 on localhost and selected IP, forwarding to 5000 in the pod
ac port-forward --address localhost,10.19.21.23 pod/mypod 8888:5000

# Listen on a random port locally, forwarding to 5000 in the pod
ac port-forward pod/mypod :5000
```

ac process

将模板处理为资源列表

示例用法

```
# Convert the template.json file into a resource list and pass to create
ac process -f template.json | ac apply -f -

# Process a file locally instead of contacting the server
ac process -f template.json -o yaml

# Process template while passing a user-defined label
ac process -f template.json -l name=mytemplate

# Convert a stored template into a resource list
ac process foo

# Convert a stored template into a resource list by setting/overriding pa
rameter values
ac process foo -p PARM1=VALUE1 -p PARM2=VALUE2

# Convert a template stored in different namespace into a resource list
ac process cpaas-system//foo

# Convert template.json into a resource list
cat template.json | ac process -f -
```

ac proxy

运行到 Kubernetes API server 的代理

示例用法

```
# To proxy all of the Kubernetes API and nothing else
ac proxy --api-prefix=/

# To proxy only part of the Kubernetes API and also some static files
# You can get pods info with 'curl localhost:8001/api/v1/pods'
ac proxy --www=/my/files --www-prefix=/static/ --api-prefix=/api/

# To proxy the entire Kubernetes API at a different root
# You can get pods info with 'curl localhost:8001/custom/api/v1/pods'
ac proxy --api-prefix=/custom/

# Run a proxy to the Kubernetes API server on port 8011, serving static c
ontent from ./local/www/
ac proxy --port=8011 --www=./local/www/

# Run a proxy to the Kubernetes API server on an arbitrary local port
# The chosen port for the server will be output to stdout
ac proxy --port=0

# Run a proxy to the Kubernetes API server, changing the API prefix to k8
s-api
# This makes e.g. the pods API available at localhost:8001/k8s-api/v1/pod
s/
ac proxy --api-prefix=/k8s-api
```

ac replace

通过文件名或 stdin 替换资源

示例用法

```
# Replace a pod using the data in pod.json
ac replace -f ./pod.json

# Replace a pod based on the JSON passed into stdin
cat pod.json | ac replace -f -

# Update a single-container pod's image version (tag) to v4
ac get pod mypod -o yaml | sed 's/\(image: myimage\):.*$/\1:v4/' | ac rep
lace -f -

# Force replace, delete and then re-create the resource
ac replace --force -f ./pod.json
```

ac rollout

管理资源的 rollout

示例用法

```
# Rollback to the previous deployment
ac rollout undo deployment/abc

# Check the rollout status of a daemonset
ac rollout status daemonset/foo

# Restart a deployment
ac rollout restart deployment/abc

# Restart deployments with the 'app=nginx' label
ac rollout restart deployment --selector=app=nginx
```

ac rollout history

查看 rollout 历史

示例用法

```
# View the rollout history of a deployment
ac rollout history deployment/abc

# View the details of daemonset revision 3
ac rollout history daemonset/abc --revision=3
```

ac rollout pause

将提供的资源标记为已暂停

示例用法

```
# Mark the nginx deployment as paused
# Any current state of the deployment will continue its function; new upd
ates
# to the deployment will not have an effect as long as the deployment is
paused
ac rollout pause deployment/nginx
```

ac rollout restart

重启资源

示例用法

```
# Restart all deployments in the test-namespace namespace
ac rollout restart deployment -n test-namespace

# Restart a deployment
ac rollout restart deployment/nginx

# Restart a daemon set
ac rollout restart daemonset/abc

# Restart deployments with the app=nginx label
ac rollout restart deployment --selector=app=nginx
```

ac rollout resume

恢复已暂停的资源

示例用法

```
# Resume an already paused deployment
ac rollout resume deployment/nginx
```

ac rollout status

显示 rollout 状态

示例用法

```
# Watch the rollout status of a deployment
ac rollout status deployment/nginx
```

ac rollout undo

撤销之前的 rollout

示例用法

```
# Roll back to the previous deployment
ac rollout undo deployment/abc

# Roll back to daemonset revision 3
ac rollout undo daemonset/abc --to-revision=3

# Roll back to the previous deployment with dry-run
ac rollout undo --dry-run=server deployment/abc
```

ac run

在集群上运行指定的镜像

示例用法

```
# Start a nginx pod
ac run nginx --image=nginx

# Start a hazelcast pod and let the container expose port 5701
ac run hazelcast --image=hazelcast/hazelcast --port=5701

# Start a hazelcast pod and set environment variables "DNS_DOMAIN=cluste
r" and "POD_NAMESPACE=default" in the container
ac run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cluster" -
-env="POD_NAMESPACE=default"

# Start a hazelcast pod and set labels "app=hazelcast" and "env=prod" in
the container
ac run hazelcast --image=hazelcast/hazelcast --labels="app=hazelcast,env=
prod"

# Dry run; print the corresponding API objects without creating them
ac run nginx --image=nginx --dry-run=client

# Start a nginx pod, but overload the spec with a partial set of values p
arsed from JSON
ac run nginx --image=nginx --overrides='{ "apiVersion": "v1", "spec": {
... } }'

# Start a busybox pod and keep it in the foreground, don't restart it if
it exits
ac run -i -t busybox --image=busybox --restart=Never

# Start the nginx pod using the default command, but use custom arguments
(arg1 .. argN) for that command
ac run nginx --image=nginx -- <arg1> <arg2> ... <argN>

# Start the nginx pod using a different command and custom arguments
ac run nginx --image=nginx --command -- <cmd> <arg1> ... <argN>
```

ac scale

为 deployment、replica set 或 replication controller 设置新的大小

示例用法

```
# Scale a replica set named 'foo' to 3
ac scale --replicas=3 rs/foo

# Scale a resource identified by type and name specified in "foo.yaml" to
3
ac scale --replicas=3 -f foo.yaml

# If the deployment named mysql's current size is 2, scale mysql to 3
ac scale --current-replicas=2 --replicas=3 deployment/mysql

# Scale multiple replication controllers
ac scale --replicas=5 rc/example1 rc/example2 rc/example3

# Scale stateful set named 'web' to 3
ac scale --replicas=3 statefulset/web
```

ac set

在对象上设置特定功能

ac set env

更新 pod 模板上的环境变量

示例用法

```
# Update deployment 'registry' with a new environment variable
ac set env deployment/registry STORAGE_DIR=/local

# List the environment variables defined on a deployments 'sample-build'
ac set env deployment/sample-build --list

# List the environment variables defined on all pods
ac set env pods --all --list

# Output modified deployment in YAML, and does not alter the object on th
e server
ac set env deployment/sample-build STORAGE_DIR=/data -o yaml

# Update all containers in all replication controllers in the project to
have ENV=prod
ac set env rc --all ENV=prod

# Import environment from a secret
ac set env --from=secret/mysecret deployment/myapp

# Import environment from a config map with a prefix
ac set env --from=configmap/myconfigmap --prefix=MYSQL_ deployment/myapp

# Import specific keys from a config map
ac set env --keys=my-example-key --from=configmap/myconfigmap deployment/
myapp

# Remove the environment variable ENV from container 'c1' in all deployme
nt configs
ac set env deployments --all --containers="c1" ENV-

# Remove the environment variable ENV from a deployment definition on dis
k and
# update the deployment config on the server
ac set env -f deploy.json ENV-

# Set some of the local shell environment into a deployment config on the
server
env | grep RAILS_ | ac set env -e - deployment/registry
```

ac set image

更新 pod 模板的镜像

示例用法

```
# Set a deployment's nginx container image to 'nginx:1.9.1', and its busy box container image to 'busybox'
ac set image deployment/nginx busybox=busybox nginx=nginx:1.9.1

# Update all deployments' and rc's nginx container's image to 'nginx:1.9.1'
ac set image deployments,rc nginx=nginx:1.9.1 --all

# Update image of all containers of daemonset abc to 'nginx:1.9.1'
ac set image daemonset abc *=nginx:1.9.1

# Print result (in yaml format) of updating nginx container image from local file, without hitting the server
ac set image -f path/to/file.yaml nginx=nginx:1.9.1 --local -o yaml
```

ac set resources

更新带有 pod 模板的对象上的资源请求/限制

示例用法

```
# Set a deployments nginx container cpu limits to "200m" and memory to "5
12Mi"
ac set resources deployment nginx -c=nginx --limits=cpu=200m,memory=512Mi

# Set the resource request and limits for all containers in nginx
ac set resources deployment nginx --limits=cpu=200m,memory=512Mi --reques
ts=cpu=100m,memory=256Mi

# Remove the resource requests for resources on containers in nginx
ac set resources deployment nginx --limits=cpu=0,memory=0 --requests=cpu=
0,memory=0

# Print the result (in yaml format) of updating nginx container limits fr
om a local, without hitting the server
ac set resources -f path/to/file.yaml --limits=cpu=200m,memory=512Mi --lo
cal -o yaml
```

ac set selector

设置资源上的选择器

示例用法

```
# Set the labels and selector before creating a deployment/service pair
ac create service clusterip my-svc --clusterip="None" -o yaml --dry-run=c
lient | ac set selector --local -f - 'environment=qa' -o yaml | ac create
-f -
ac create deployment my-dep -o yaml --dry-run=client | ac label --local -
f - environment=qa -o yaml | ac create -f -
```

ac set serviceaccount

更新资源的 service account

示例用法

```
# Set deployment nginx-deployment's service account to serviceaccount1
ac set serviceaccount deployment nginx-deployment serviceaccount1

# Print the result (in YAML format) of updated nginx deployment with the
service account from local file, without hitting the API server
ac set sa -f nginx-deployment.yaml serviceaccount1 --local --dry-run=clie
nt -o yaml
```

ac set subject

更新 role binding 或 cluster role binding 中的用户、组或 service account

示例用法

```
# Update a cluster role binding for serviceaccount1
ac set subject clusterrolebinding admin --serviceaccount=namespace:servic
eaccount1

# Update a role binding for user1, user2, and group1
ac set subject rolebinding admin --user=user1 --user=user2 --group=group1

# Print the result (in YAML format) of updating rolebinding subjects from
a local, without hitting the server
ac create rolebinding admin --role=admin --user=admin -o yaml --dry-run=c
lient | ac set subject --local -f - --user=foo -o yaml
```

ac version

打印客户端和服务器版本信息

示例用法

```
# Print client and server version information
ac version

# Print client version only
ac version --client

# Print version in JSON format
ac version -o json
```

ac wait

实验性：等待一个或多个资源上的特定条件

示例用法

```
# Wait for the pod "busybox1" to contain the status condition of type "Ready"
ac wait --for=condition=Ready pod/busybox1

# The default value of status condition is true; you can wait for other targets after an equal delimiter (compared after Unicode simple case folding, which is a more general form of case-insensitivity)
ac wait --for=condition=Ready=false pod/busybox1

# Wait for the pod "busybox1" to contain the status phase to be "Running"
ac wait --for=jsonpath='{.status.phase}'=Running pod/busybox1

# Wait for pod "busybox1" to be Ready
ac wait --for='jsonpath={.status.conditions[?(@.type=="Ready")].status}=True' pod/busybox1

# Wait for the service "loadbalancer" to have ingress
ac wait --for=jsonpath='{.status.loadBalancer.ingress}' service/loadbalancer

# Wait for the secret "busybox1" to be created, with a timeout of 30s
ac create secret generic busybox1
ac wait --for=create secret/busybox1 --timeout=30s

# Wait for the pod "busybox1" to be deleted, with a timeout of 60s, after having issued the "delete" command
ac delete pod/busybox1
ac wait --for=delete pod/busybox1 --timeout=60s
```

ac whoami

显示当前会话的信息

示例用法

```
# Display the currently authenticated user
ac whoami

# Print the current user context name
ac whoami -c

# Print the current server's REST API URL
ac whoami --show-server

# Print the token the current session is using
ac whoami -t
# Caution: the token printed by ac whoami -t is sensitive; do not expose
it in logs, shell history, screenshots, or shared output.
```

AC CLI 管理员命令参考

本参考提供了 AC CLI 管理员命令的描述和示例命令。使用这些命令需要具备 cluster-admin 或等效权限。

有关开发人员命令，请参见 AC CLI 开发人员命令参考。

运行 `ac adm -h` 列出所有管理员命令，或运行 `ac <command> --help` 获取特定命令的更多详细信息。

目录

ac adm
  示例用法
ac adm certificate
ac adm certificate approve
  示例用法
ac adm certificate deny
  示例用法
ac adm cordon
  示例用法
ac adm drain
  示例用法
ac adm new-project
  示例用法
ac adm new-project-namespace

示例用法

ac adm policy

示例用法

ac adm policy add-cluster-role-to-user

示例用法

ac adm policy add-namespace-role-to-user

示例用法

ac adm policy add-project-role-to-user

示例用法

ac adm policy add-role-to-user

示例用法

ac adm release

示例用法

ac adm release import-manifest

示例用法

ac adm taint

示例用法

ac adm uncordon

示例用法

ac adm upgrade

示例用法

ac adm upgrade status

示例用法

---

ac adm

ACP 集群管理的管理工具

示例用法

```
# 为维护排空节点
ac adm drain NODE_NAME

# 标记节点为不可调度
ac adm cordon NODE_NAME

# new-project 在集群中创建项目
ac adm new-project PROJECT_NAME --cluster CLUSTER_NAME

# 标记节点为可调度
ac adm uncordon NODE_NAME
```

ac adm certificate

修改证书资源

ac adm certificate approve

批准证书签名请求

示例用法

```
# 批准 CSR 'csr-sqgzp'
ac adm certificate approve csr-sqgzp
```

ac adm certificate deny

拒绝证书签名请求

示例用法

```
# 拒绝 CSR 'csr-sqgzp'
ac adm certificate deny csr-sqgzp
```

ac adm cordon

标记节点为不可调度

示例用法

```
# 标记节点 "foo" 为不可调度
ac adm cordon foo
```

ac adm drain

排空节点以准备维护

示例用法

```
# 排空节点 "foo", 即使其上有未被 replication controller、replica set、job、dae
mon set 或 stateful set 管理的 pods
ac adm drain foo --force

# 同上, 但如果存在未被上述控制器管理的 pods 则中止, 且使用 15 分钟的宽限期
ac adm drain foo --grace-period=900
```

ac adm new-project

创建新项目

示例用法

```
# 在指定集群中创建项目
ac adm new-project my-project --cluster cluster1

# 在多个集群中创建项目
ac adm new-project my-project --cluster cluster1,cluster2
```

ac adm new-project-namespace

在项目中创建新命名空间

示例用法

```
# 在指定集群的项目中创建命名空间
ac adm new-project-namespace  my-namespace --project my-project --cluster
cluster1
```

ac adm policy

管理项目或命名空间的 RBAC 策略

示例用法

```
# 将用户分配为项目中的管理员角色
ac adm policy add-project-role-to-user project-admin-system alice --proje
ct my-project

# 将用户分配为项目中集群命名空间的命名空间角色
ac adm policy add-namespace-role-to-user namespace-developer-system alice
--namespace my-namespace --project my-project --cluster business-1

# 给用户 alice 添加 kubernetes 集群角色 view
ac adm policy add-cluster-role-to-user view alice

# 给用户 alice 添加 kubernetes 角色 view
ac adm policy add-role-to-user view alice -n my-namespace
```

ac adm policy add-cluster-role-to-user

为当前上下文集群中的用户分配 kubernetes 集群角色

示例用法

```
# 给用户 alice 添加 kubernetes 集群角色 view
ac adm policy add-cluster-role-to-user view alice
```

ac adm policy add-namespace-role-to-user

为项目中特定集群命名空间中的用户分配平台角色

示例用法

```
# 在项目 my-project 中为用户 alice 分配 namespace-developer-system 角色
ac adm policy add-namespace-role-to-user namespace-developer-system alice
--namespace my-namespace --project my-project --cluster business-1
```

ac adm policy add-project-role-to-user

为项目中的用户分配平台角色

示例用法

```
# 在项目 my-project 中为用户 alice 分配 project-admin-system 角色
ac adm policy add-project-role-to-user project-admin-system alice --proje
ct my-project
```

ac adm policy add-role-to-user

为当前上下文集群中的用户分配 kubernetes 角色

示例用法

```
# 给用户 alice 添加 kubernetes 角色 view
ac adm policy add-role-to-user view alice -n my-namespace
```

ac adm release

管理发布元数据及相关管理工作流

示例用法

```
# 导入发布版本的 ProductManifest
ac adm release import-manifest --version 4.20.0
```

ac adm release import-manifest

将发布元数据导入为 ProductManifest

示例用法

```
# 导入版本 4.20.0 的发布元数据
ac adm release import-manifest --version 4.20.0

# 导入元数据并等待 ProductManifest 变为 Ready 状态
ac adm release import-manifest --version 4.20.0 --wait

# 覆盖等待超时时间
ac adm release import-manifest --version 4.20.0 --wait --timeout=10m
```

ac adm taint

更新一个或多个节点上的污点

示例用法

```
# 为节点 'foo' 添加一个键为 'dedicated'，值为 'special-user'，效果为 'NoSchedul
e' 的污点
# 如果该键和效果的污点已存在，则替换其值
ac adm taint nodes foo dedicated=special-user:NoSchedule

# 从节点 'foo' 移除键为 'dedicated'，效果为 'NoSchedule' 的污点（如果存在）
ac adm taint nodes foo dedicated:NoSchedule-

# 从节点 'foo' 移除所有键为 'dedicated' 的污点
ac adm taint nodes foo dedicated-

# 在标签为 myLabel=X 的节点上添加键为 'dedicated'，效果为 'PreferNoSchedule' 的
污点
ac adm taint node -l myLabel=X  dedicated=foo:PreferNoSchedule

# 为节点 'foo' 添加一个键为 'bar'，无值的污点，效果为 NoSchedule
ac adm taint nodes foo bar:NoSchedule
```

ac adm uncordon

标记节点为可调度

示例用法

```
# 标记节点 "foo" 为可调度
ac adm uncordon foo
```

ac adm upgrade

查看或请求集群升级

示例用法

```
# 查看更新状态和可用的集群更新
ac adm upgrade

# 查看特定集群的摘要
ac adm upgrade --cluster=workload-a

# 升级到最新版本
ac adm upgrade --to-latest

# 从可用更新中升级到指定版本
ac adm upgrade --cluster=workload-a --to=4.15.0

# 允许升级到可用更新之外的明确版本
ac adm upgrade --to=4.15.0 --allow-explicit-upgrade
```

ac adm upgrade status

查看目标集群升级的预检和阶段详细信息

示例用法

```
# 查看默认集群的 Cluster Version Operator 状态
ac adm upgrade status

# 查看特定集群的状态
ac adm upgrade status --cluster=workload-a

# 查看控制器报告的完整详细信息
ac adm upgrade status --verbose
```

升级集群

AC CLI 提供管理员命令，用于准备升级元数据、查看集群升级状态，以及请求集群升级。

当你需要执行以下操作时，请使用这些命令：

- 为目标版本创建一个 `ProductManifest`
- 检查当前集群版本和可用更新
- 请求升级到最新版本
- 请求升级到特定版本
- 查看预检结果和升级执行进度

目录

开始之前

前提条件

创建 ProductManifest

查看升级状态和可用更新

升级到最新版本

升级到特定版本

查看详细升级状态

预检结果

升级阶段

常见信号和故障排查

示例工作流

开始之前

在 ACP 中，`availableUpdates` 不是一个需要你手动维护的静态列表。升级控制器必须先看到目标版本的 `ProductManifest`，才能为集群发布可用的升级目标。

如果你没有先创建 `ProductManifest`，常见症状包括：

- `ac adm upgrade` 不显示任何 `availableUpdates`
- `ac adm upgrade --to-latest` 立即失败
- 你只能使用 `--allow-explicit-upgrade` 手动请求某个版本

建议的流程如下：

1. 确定你要发布或升级到哪个版本。
2. 为该版本创建一个 `ProductManifest`。
3. 等待升级元数据被处理。
4. 运行 `ac adm upgrade`，确认 `availableUpdates` 已填充。
5. 请求升级。

前提条件

在运行与升级相关的命令之前，请确保：

- 你已登录到 ACP 平台。
- 你拥有集群管理员或等效权限。
- 你知道目标集群名称。如果省略，`ac adm upgrade` 默认使用 `global`。
- 目标环境已经安装了 `ProductManifest` CRD 和升级控制器。

你可以先确认当前上下文：

```
ac config current-context
```

当前上下文仍然决定 AC 在命令中使用哪些凭据和 endpoint。

- 对于 `ac adm upgrade` 和 `ac adm upgrade status`，目标集群仍通过 `--cluster` 显式选择，默认值为 `global`。
- 对于 `ac adm release import-manifest`，AC 会先检查当前上下文的 REST URL。如果它指向 ACP workload 路径，AC 会自动将请求重写为对应的 global 路径。如果当前上下文不是 ACP workload/global URL，AC 会按原样使用当前上下文，并且不需要 kubeconfig session extension。

如有需要，先切换到另一个 ACP session：

```
# Switch to another ACP session
ac config use-session production
```

创建 ProductManifest

使用以下命令为目标版本创建一个 `ProductManifest`：

```
ac adm release import-manifest --version 4.20.0
```

此命令会创建控制器所需的最小升级元数据对象：

- `metadata.name` 使用带前缀 `v` 的版本名，例如 `v4.20.0`
- `spec.version` 使用你传入的版本，例如 `4.20.0`

如果你希望命令等待对象变为 Ready，请添加 `--wait`：

```
ac adm release import-manifest --version 4.20.0 --wait
```

你也可以覆盖等待超时时间：

```
ac adm release import-manifest --version 4.20.0 --wait --timeout=10m
```

命令行为：

- 必须提供 `--version`。
- 如果 `ProductManifest` 不存在，AC 会创建它。
- 如果 `ProductManifest` 已存在且版本相同，命令会成功，但不会修改它。
- 如果 `ProductManifest` 已存在但版本不同，命令会失败，不会覆盖现有对象。
- 默认情况下，命令不会等待 Ready。只有显式传入 `--wait` 时才会等待。

查看升级状态和可用更新

在创建 `ProductManifest` 之后，使用以下命令查看默认 `global` 集群的升级摘要：

```
ac adm upgrade
```

此命令通常会显示：

- 当前集群版本
- 期望版本（如果已经请求了升级）
- 当前可用更新列表
- 整体升级条件

当你想快速回答以下问题时，可以使用它：

- 集群当前运行的是什么版本？
- 是否已经请求了目标版本？
- 现在是否有新的升级目标可用？
- 集群当前是 `Ready`、`Reconciling` 还是 `Degraded`？

如需查询特定集群，请添加 `--cluster`：

```
ac adm upgrade --cluster=workload-a
```

如果你刚创建了 `ProductManifest`，但仍然看不到 `availableUpdates`，控制器可能还在处理元数据。稍等片刻后再次运行 `ac adm upgrade`。

升级到最新版本

当 `availableUpdates` 存在时，使用以下命令请求升级到最新版本：

```
ac adm upgrade --to-latest
```

AC 会从 `availableUpdates` 中选择最高版本并提交升级请求。

`--to-latest` 是一个布尔标志，默认值为 `false`，这意味着：

- 如果你不指定它，AC 的行为等同于使用了 `--to-latest=false`。
- 如果你单独指定 `--to-latest`，AC 会将其视为 `true`。
- 你也可以显式写成 `--to-latest=true` 或 `--to-latest=false`。

如果不存在任何 `availableUpdates`，命令会失败，不会提交新的目标版本。

请求升级后，再次查看摘要：

```
ac adm upgrade
```

升级到特定版本

如果你想升级到特定版本，请运行：

```
ac adm upgrade --to=<version>
```

示例：

```
ac adm upgrade --to=4.20.0
```

典型场景包括：

- 你不想使用最新的可用版本

- 你正在遵循发布流程中批准的发布目标
- 你想重试一个已知目标版本

默认情况下，请求的版本必须已经出现在 `availableUpdates` 中。否则，命令会失败。

如果你确实需要请求一个未列在 `availableUpdates` 中的版本，请添加：

```
ac adm upgrade --to=<version> --allow-explicit-upgrade
```

`--allow-explicit-upgrade` 的默认值为 `false`：

- 如果你不指定它，AC 的行为等同于使用了 `--allow-explicit-upgrade=false`。
- 如果你单独指定 `--allow-explicit-upgrade`，AC 会将其视为 `true`。
- 你也可以显式写成 `--allow-explicit-upgrade=true` 或 `--allow-explicit-upgrade=false`。

示例：

```
ac adm upgrade --to=4.20.0 --allow-explicit-upgrade=true
```

此标志只会更改客户端侧校验。升级控制器及其预检检查仍会决定所请求的目标是否可以继续执行。

查看详细升级状态

如果你需要更深入的诊断，请运行：

```
ac adm upgrade status
```

如需查看特定集群：

```
ac adm upgrade status --cluster=workload-a
```

与 `ac adm upgrade` 相比，此命令会展开显示：

- 当前版本、期望版本和整体条件的摘要
- 当前升级目标的预检结果
- 升级阶段和操作进度

预检结果

预检用于描述升级是否可以进入执行阶段。每项检查通常包含：

- 检查名称
- 重入策略
- 状态
- 原因
- 消息

状态含义如下：

- `Passed`：检查已通过。
- `Retry`：检查暂时无法给出最终结果。请稍后再查看。
- `Failed`：存在阻塞条件，必须先处理。

如果当前还没有预检数据，请将其视为“暂无结果”，而不是“所有检查都已通过”。

如果 `AdminAckRequired` 为 `Failed`，则当前升级目标适用管理员确认门控。请查看该门控，完成所需的手动检查，写入确认，然后再次运行 `ac adm upgrade status --cluster=<cluster>`，确认检查通过。

升级阶段

当升级进入执行阶段后，状态输出会显示阶段和操作进度。

阶段输出可能包含：

- 阶段名称
- 优先级
- 阶段状态

操作输出可能包含：

- 操作名称
- 动作
- 当前版本
- 目标版本
- 阶段状态

阶段状态含义如下：

- `Pending`：该阶段尚未开始。
- `Running`：该阶段正在进行中。
- `Finished`：该阶段已完成。

如果当前还没有阶段数据，升级很可能尚未进入执行阶段。

常见信号和故障排查

阅读升级状态时，请参考以下指导：

- `Ready` 通常表示集群已达到期望状态。
- `Reconciling` 通常表示集群仍在应用当前升级请求。
- `Degraded` 通常表示升级被阻塞或发生了错误。

当 `ac adm upgrade` 没有显示任何 `availableUpdates` 时，请先检查以下内容：

1. 是否已为目标版本创建 `ProductManifest`？
2. 如果你使用了 `--wait`，`ProductManifest` 是否已达到 `Ready=True`？
3. 控制器是否有足够时间处理新的元数据？

当 `ac adm upgrade` 显示了目标版本，但升级没有推进时：

1. 运行 `ac adm upgrade status`。
2. 检查是否有任何预检项处于 `Retry` 或 `Failed`。

3. 查看升级阶段是否已经开始。

当 `ac adm upgrade status --cluster=<cluster>` 显示 `AdminAckRequired` 处于 `Failed` 状态时：

1. 从 global 集群检查发行版提供的门控：

```
kubectl -n cpaas-system get configmap admin-gates -o yaml
```

2. 如果目标发行版是 ACP 4.4 或更高版本，且会将集群升级到 Kubernetes 1.35 或更高版本，并且 `admin-gates` 报告了一个节点就绪确认键，请在目标集群的每个生产节点上完成 Kubernetes 1.35 或更高版本节点就绪检查。此要求引入的第一个门控是 `ack-4.4-kubernetes-1.35-kernel-update`；请将其视为示例，因为更晚的目标发行版可能会提供不同的键。
3. 完成门控要求后，从 `admin-gates` 复制适用的键，并将其写入 global 侧的 `admin-acks` ConfigMap：

```
ACK_KEY='<key-from-admin-gates>'
kubectl -n cpaas-system patch configmap admin-acks --type merge \
  -p "{\"data\":{\"${ACK_KEY}\":\"true\"}}"
```

4. 确认预检检查通过：

```
ac adm upgrade status --cluster=<cluster>
```

当升级已经在进行中时：

1. 运行 `ac adm upgrade status`。
2. 查看当前阶段和操作状态。
3. 将当前版本与目标版本进行比较。

示例工作流

```
# 1. Create a ProductManifest for the target version
ac adm release import-manifest --version 4.20.0 --wait

# 2. Review the upgrade summary for the default global cluster
ac adm upgrade

# 3. Review the summary for a specific cluster
ac adm upgrade --cluster=workload-a

# 4. Request an upgrade to the latest available version
ac adm upgrade --to-latest

# 5. Review detailed status
ac adm upgrade status --cluster=workload-a

# 6. Request an upgrade to a specific version
ac adm upgrade --cluster=workload-a --to=4.20.0

# 7. Request a version outside availableUpdates when you explicitly inten
d to do so
ac adm upgrade --cluster=workload-a --to=4.20.0 --allow-explicit-upgrade
```

violet CLI

该平台提供 `violet` 命令行工具，用于从 customer portal 下载软件包并将软件包上传到 ACP。

文档中介绍的下载、清单和上传工作流请使用 **Violet v4.2.13** 或更高版本。

有关详细信息，请参阅 Download Packages 和 Upload Packages。

Web 终端

Alauda 容器平台 **Web** 终端插件让您可以无需额外配置，即可直接通过 Web 界面访问和使用 `kubectl`。

> **Note**
>
> 因为 Alauda Container Platform Web Terminal 的发版周期与灵雀云容器平台不同，所以 Alauda Container Platform Web Terminal 的文档现在作为独立的文档站点托管在 Alauda Container Platform Web Terminal↗。