

# ACP CLI (ac)

---

[Начало работы с ACP CLI](#)[Настройка ACP CLI](#)[Используй](#)[Расширение ACP CLI с помощью плагинов](#)[AC CLI Dev](#)[AC CLI Справочник команд администратор](#)

# Начало работы с ACP CLI

## Содержание

### О ACP CLI

#### Установка

Установка из бинарного файла

Установка ACP CLI на Linux

Установка ACP CLI на macOS

Установка ACP CLI на Windows

#### Первые шаги

Вход в ACP Platform

Интерактивный вход

Вход с параметрами

Вход с использованием переменных окружения

#### Быстрое управление конфигурацией

Просмотр текущего статуса

Переключение кластеров

Переключение пространств имён

Основные операции с ресурсами

Управление несколькими средами

#### Ваше первое приложение

Создание простого pod-a

Просмотр статуса приложения

Очистка

#### Получение помощи

Встроенная система помощи

Общая помощь

Помощь по конкретной команде

Документация по ресурсам

Выход из системы

---

## О ACP CLI

С помощью ACP CLI (ac) вы можете управлять платформами и кластерами ACP из терминала. ACP CLI предоставляет опыт, похожий на kubectl, оптимизированный для централизованной, прокси-ориентированной мультикластерной архитектуры ACP.

ACP CLI идеально подходит в следующих случаях:

- Работа с платформами ACP и несколькими кластерами через единый интерфейс
- Прямая работа с исходным кодом проектов, скриптование операций платформы ACP и автоматизация рабочих процессов
- Управление проектами при ограниченных ресурсах пропускной способности и отсутствии доступа к веб-консоли
- Управление приложениями в разных средах ACP (production, staging, development)

## Установка

### Установка из бинарного файла

Вы можете установить ACP CLI (ac), загрузив бинарный файл для вашей операционной системы.

Выполните следующие шаги, чтобы скачать нужный пакет:

1. Откройте в браузере [страницу загрузки Alauda Cloud](#) ↗.
  2. Выберите **CLI Tools**, чтобы перейти на страницу загрузки CLI.
-

3. Найдите раздел **ACP CLI (ac)**.
4. Скачайте бинарный файл, соответствующий вашей операционной системе и архитектуре процессора (например, `ac-linux-amd64`).

## Установка ACP CLI на Linux

1. Выполните шаги загрузки выше, чтобы получить бинарный файл для Linux (например, `ac-linux-amd64` или `ac-linux-arm64`).
2. Сделайте бинарный файл исполняемым:

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

Замените `ac-linux-amd64` на имя скачанного файла.

3. Переместите бинарный файл в каталог из PATH и переименуйте его в `ac`:

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

При необходимости измените имя файла, если скачали другую версию.

4. Проверьте установку:

```
ac version
```

## Установка ACP CLI на macOS

1. Выполните шаги загрузки выше, чтобы получить бинарный файл для macOS (например, `ac-darwin-amd64` или `ac-darwin-arm64`).
2. Сделайте бинарный файл исполняемым:

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

Замените `ac-darwin-amd64` на имя скачанного файла.

3. Переместите бинарный файл в каталог из PATH и переименуйте его в `ac`:

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

При необходимости измените имя файла, если скачали другую версию.

4. Проверьте установку:

```
ac version
```

## Установка ACP CLI на Windows

1. Выполните шаги загрузки выше, чтобы получить бинарный файл для Windows (например, `ac-windows-amd64.exe`).
2. Переместите бинарный файл `ac-windows-amd64.exe` в каталог из PATH и при желании переименуйте в `ac.exe`. Можно оставить исходное имя, главное — чтобы каталог файла был в PATH.
3. Проверьте установку:

```
ac version
```

## Первые шаги

### Вход в ACP Platform

Команда `ac login` — это ваша точка входа для подключения к платформам ACP. Она выполняет аутентификацию и автоматически настраивает доступ ко всем доступным кластерам.

### Интерактивный вход

Для простейшего варианта запустите `ac login` без параметров и следуйте интерактивным подсказкам:

```
$ 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
```

## Вход с параметрами

Вы также можете передать параметры напрямую:

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

## Вход с использованием переменных окружения

Для автоматизации и скриптинга используйте переменные окружения:

```
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
```

## Быстрое управление конфигурацией

После входа ACP CLI предоставляет удобные команды для повседневных операций:

## Просмотр текущего статуса

Используйте `ac namespace`, чтобы увидеть текущий рабочий контекст:

```
$ 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/
```

## Переключение кластеров

Переключайтесь между кластерами в рамках текущей сессии:

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

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

## Переключение пространств имён

Измените активное пространство имён:

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

## Основные операции с ресурсами

Используйте стандартные команды `kubectl` для управления ресурсами:

```
# Список pod-ов в текущем пространстве имён
$ ac get pods

# Описание конкретного pod-а
$ ac describe pod my-pod

# Получить сервисы во всех пространствах имён
$ ac get services --all-namespaces

# Применить конфигурационный файл
$ ac apply -f deployment.yaml
```

## Управление несколькими средами

Для пользователей, работающих с несколькими платформами ACP:

Список всех настроенных сессий:

```
$ 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
```

Переключение между платформами:

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

## Ваше первое приложение

Создадим и посмотрим простое приложение, чтобы убедиться, что всё работает:

### Создание простого pod-а

## 1. Создайте базовую конфигурацию pod-a:

```
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. Примените конфигурацию:

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

## Просмотр статуса приложения

### 1. Список pod-ов для просмотра приложения:

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

### 2. Подробная информация о pod-e:

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

### 3. Просмотр логов pod-a:

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

## Очистка

Удалите тестовый pod после завершения:

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

## Получение помощи

### Встроенная система помощи

ACP CLI предоставляет подробную помощь на разных уровнях:

#### Общая помощь

Обзор всех доступных команд:

```
ac help
```

#### Помощь по конкретной команде

Подробная помощь по любой команде:

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

#### Документация по ресурсам

Информация о Kubernetes-ресурсах:

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

# Выход из системы

Когда вы закончите работу или нужно сменить учётные данные, используйте команду выхода:

```
$ 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
```

Команда logout:

- Удаляет токены аутентификации из локальной конфигурации
- Очищает все записи о кластерах и контекстах для сессии
- Отзывает используемые токены в ACP
- Обеспечивает отсутствие «зависших» конфигураций

# Настройка ACP CLI

---

## Содержание

### Автодополнение в оболочке

Включение автодополнения для Bash

Требования

Процедура

Включение автодополнения для Zsh

Требования

Процедура

Доступ к kibesconfig с помощью ACP CLI

Требования

Процедура

Обработка конфигурации для нескольких кластеров

Вопросы безопасности

---

## Автодополнение в оболочке

Вы можете включить автодополнение по нажатию клавиши Tab для оболочек Bash или Zsh.

### Включение автодополнения для Bash

---

После установки ACP CLI (ac) вы можете включить автодополнение, чтобы автоматически дополнять команды ac или предлагать варианты при нажатии Tab. Следующая процедура включает автодополнение для оболочки Bash.

## Требования

- У вас должен быть установлен ACP CLI (ac).
- Должен быть установлен пакет bash-completion.

## Процедура

1. Сохраните код автодополнения Bash в файл:

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

2. Скопируйте файл в каталог `/etc/bash_completion.d/`:

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

Также вы можете сохранить файл в локальный каталог и подключать его из вашего файла `.bashrc`.

Автодополнение будет включено при открытии нового терминала.

## Включение автодополнения для Zsh

После установки ACP CLI (ac) вы можете включить автодополнение, чтобы автоматически дополнять команды ac или предлагать варианты при нажатии Tab. Следующая процедура включает автодополнение для оболочки Zsh.

## Требования

У вас должен быть установлен ACP CLI (ac).

## Процедура

Чтобы добавить автодополнение для `ac` в ваш файл `.zshrc`, выполните следующую команду:

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

Автодополнение будет включено при открытии нового терминала.

## Доступ к kubernetes с помощью ACP CLI

Вы можете использовать ACP CLI (`ac`) для входа в вашу платформу ACP и получения файла `kubernetes` для доступа к кластерам из командной строки. В отличие от традиционного экспорта `kubernetes` для одного кластера, команда `ac login` создает комплексную конфигурацию для нескольких кластеров через обнаружение платформы.

### Требования

У вас должен быть доступ к конечной точке платформы ACP и действительные учетные данные для аутентификации.

### Процедура

1. Войдите в вашу платформу ACP, выполнив следующую команду:

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

- `<platform-url>`: базовый URL платформы ACP (например, <https://acp.prod.example.com>)

- `<session-name>` : удобочитаемое имя для этого подключения к платформе (например, "prod", "staging")

2. Процесс входа автоматически:

- Аутентифицируется на платформе ACP
- Обнаруживает все доступные кластеры на платформе
- Создает записи `kubeconfig` для всех кластеров с метаданными, специфичными для ACP
- Устанавливает контекст по умолчанию, указывающий на глобальный кластер

3. Чтобы экспортировать конфигурацию в отдельный файл, выполните:

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

4. Установите переменную окружения `KUBECONFIG`, указывающую на экспортированный файл:

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

5. Используйте `ac` для взаимодействия с вашими кластерами ACP:

```
$ ac get nodes
```

## Обработка конфигурации для нескольких кластеров

Процесс входа ACP CLI создает комплексную структуру `kubeconfig`, которая включает:

- **Несколько записей кластеров:** по одной для каждого доступного кластера на платформе
- **Метаданные сессии:** URL платформы, имя сессии и описания кластеров, сохраненные в расширенных полях
- **Единая аутентификация:** одна учетная запись пользователя, работающая для всех кластеров платформы

- **Интеллектуальное именование:** уникальные имена без конфликтов в формате

```
acp:<session>:<cluster>
```

## Вопросы безопасности

**Важно:** экспортированный файл `kubecfg` содержит токены аутентификации, которые предоставляют доступ к вашим кластерам платформы ACP.

- Храните файл в надежном месте с соответствующими правами доступа
- Никогда не коммитьте файлы `kubecfg` в системы контроля версий
- Учитывайте срок действия токенов и требования к их обновлению
- Используйте разные имена сессий для разных окружений (`prod`, `staging`, `dev`) для четкого разделения

Если вы планируете повторно использовать экспортированный файл `kubecfg` между сессиями или машинами, убедитесь, что он хранится безопасно и регулярно синхронизируется с помощью `ac config sync` для поддержания актуального списка кластеров.

# Использование команд `ас` и `kubectl`

Командный интерфейс Kubernetes (CLI) `kubectl` можно использовать для выполнения команд в кластере Kubernetes. Поскольку АСР является платформой, совместимой с Kubernetes, вы можете использовать поддерживаемые бинарные файлы `kubectl`, поставляемые с АСР CLI, либо получить расширенный функционал, используя бинарный файл `ас`.

## Содержание

### [Бинарный файл `ас`](#)

- Интеграция с платформой АСР

- Интеллектуальная маршрутизация ресурсов

  - Пример маршрутизации ресурсов

- Дополнительные команды

- Бинарный файл `kubectl`

## Бинарный файл `ас`

Бинарный файл `ас` предлагает те же возможности, что и `kubectl`, но дополнительно нативно поддерживает дополнительные функции платформы АСР, включая:

### Интеграция с платформой АСР

ACP CLI обеспечивает встроенную поддержку централизованной, прокси-ориентированной мультикластерной архитектуры ACP:

- **Аутентификация на платформе** — встроенная команда `login` для безопасной аутентификации на платформах ACP
- **Управление сессиями** — мультиплатформенное управление сессиями с помощью команд `ac login`, `ac config use-session` и `ac logout`
- **Расширенная конфигурация** — дополнительные команды, такие как `ac config use-cluster`, упрощающие работу с мультикластерными средами ACP

## Интеллектуальная маршрутизация ресурсов

ACP CLI автоматически направляет ресурсы уровня платформы, такие как `User` и `Project`, в глобальный кластер, поскольку эти ресурсы существуют только на уровне платформы. Это позволяет обращаться к ним из любого контекста кластера без необходимости ручного переключения. Все остальные ресурсы работают в соответствии с текущим контекстом кластера.

### Пример маршрутизации ресурсов

```
# Текущий контекст указывает на кластер нагрузки
$ ac config current-context
prod/workload-a

# Пользователь запрашивает глобальный ресурс — ACP CLI автоматически направляет запрос в глобальный кластер
$ ac get projects
(i) Note: Targeting global cluster for this command only, as 'projects' is a global resource.
NAME          STATUS  AGE
project-a     Active  32d
project-b     Active  18d

# Пользователь запрашивает ресурс нагрузки — работает с текущим кластером
$ ac get pods
NAME                                READY  STATUS  RESTARTS  AGE
my-app-7d4f8c9b6-xyz123            1/1    Running  0          2h
```

## Дополнительные команды

ACP CLI включает дополнительные команды, упрощающие рабочие процессы на платформе ACP:

- `ac login` — аутентификация на платформах ACP и настройка доступа к мультикластеру
- `ac logout` — завершение сессий платформы и очистка конфигурации
- `ac config get-sessions` — список всех настроенных сессий платформ ACP
- `ac config use-session <session_name>` — переключение между платформами ACP
- `ac config use-cluster <cluster_name>` — переключение кластеров в рамках текущей сессии
- `ac namespace` — расширенное управление namespace с отображением контекста платформы
- `ac config sync` — синхронизация конфигурации с состоянием платформы

## Бинарный файл kubectl

Бинарный файл kubectl предоставляется для поддержки существующих рабочих процессов и скриптов для новых пользователей ACP CLI, пришедших из стандартной среды Kubernetes, или для тех, кто предпочитает использовать CLI kubectl.

Существующие пользователи kubectl могут продолжать использовать этот бинарный файл для взаимодействия с примитивами Kubernetes без изменений в платформе ACP.

Для получения дополнительной информации о kubectl смотрите [документацию kubectl](#).

# Управление профилями CLI

Файл конфигурации CLI позволяет настраивать различные профили или контексты для использования с ACP CLI (ac). Контекст состоит из данных аутентификации пользователя и информации о сервере платформы ACP, связанной с псевдонимом.

## Содержание

### Удобное управление конфигурацией

Управление платформой и сессиями

`ac login` — аутентификация и настройка доступа к платформам ACP

`ac logout` — завершение сессий платформы и очистка конфигурации

`ac config get-sessions` — список всех настроенных сессий платформ ACP

`ac config use-session <session_name>` — переключение между платформами ACP

Ежедневные операции

`ac config use-cluster <cluster_name>` — переключение кластеров в текущей сессии

`ac namespace` — просмотр текущего статуса и переключение пространств имён

`ac config sync` — синхронизация конфигурации платформы

Структура конфигурации ACP CLI

Расширенная структура kubernetes ACP CLI

Структура и организация метаданных

Идентификация на основе метаданных

Конвенции именования

Ручная настройка профилей CLI

Стандартные команды конфигурации

Примеры ручных операций

## Удобное управление конфигурацией

ACP CLI предоставляет расширенные команды, которые значительно упрощают управление конфигурацией по сравнению с традиционной манипуляцией `kubecfg`. Эти команды разработаны для бесшовной работы в мультикластерной среде ACP.

## Управление платформой и сессиями

`ac login` — аутентификация и настройка доступа к платформам ACP

Команда `ac login` служит основным входом для установления соединений с платформами ACP. Она аутентифицирует пользователей и автоматически настраивает все необходимые записи `kubecfg`.

```
# Интерактивный вход на платформу ACP
ac login https://prod.acp.com --name prod

# Вход с указанием конкретного кластера и пространства имён
ac login https://prod.acp.com --name prod --cluster workload-a --namespace my-app

# Вход с использованием переменных окружения (для автоматизации)
AC_LOGIN_PLATFORM_URL=https://prod.acp.com AC_LOGIN_SESSION=prod AC_LOGIN_USERNAME=user AC_LOGIN_PASSWORD=secret ac login
```

Процесс входа:

1. Аутентификация на платформе ACP
2. Обнаружение всех доступных кластеров в платформе
3. Создание записей кластеров и пользователей в вашем `kubecfg`
4. Создание и активация контекста:

- Если указан `--cluster`: создаётся контекст для этого конкретного кластера
- Если указан `--namespace`: пространство имён устанавливается в контексте
- Если кластер не указан: по умолчанию используется глобальный кластер
- Имя контекста формируется по шаблону: `<session_name>/<cluster_name>`

## `ac logout` — завершение сессий платформы и очистка конфигурации

```
# Выйти из текущей сессии платформы
ac logout

# Выйти из конкретной сессии
ac logout --session prod
```

Команда `logout` удаляет все записи конфигурации, связанные с сессией, включая кластеры, пользователей и контексты.

## `ac config get-sessions` — список всех настроенных сессий платформ АСР

```
ac config get-sessions
```

Пример вывода:

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		

Эта команда отображает:

- **CURRENT**: указывает, принадлежит ли текущий контекст этой сессии (отмечено `*`)

- **SESSION:** имя сессии (задаётся пользователем при входе)
- **PLATFORM:** базовый URL платформы
- **USER:** имя аутентифицированного пользователя для сессии
- **CLUSTERS:** количество доступных кластеров в сессии

`ac config use-session <session_name>` — переключение между платформами АСР

```
# Переключиться на платформу staging (по умолчанию глобальный кластер)
ac config use-session staging

# Переключиться на конкретный кластер в сессии
ac config use-session prod --cluster workload-a

# Переключение с указанием пространства имён
ac config use-session staging --cluster workload-b --namespace my-app
```

Эта команда интеллектуально выбирает или создаёт соответствующие контексты на основе вашей сессии и требований к кластеру.

## Ежедневные операции

`ac config use-cluster <cluster_name>` — переключение кластеров в текущей сессии

```
# Переключиться на кластер workload в текущей сессии
ac config use-cluster workload-a

# Создать новый контекст с указанием пространства имён
ac config use-cluster workload-b --namespace my-app
```

Команда находит или создаёт контексты для указанного кластера в рамках текущей сессии платформы.

## `ac namespace` — просмотр текущего статуса и переключение пространств имён

Показать текущий статус:

```
ac namespace
```

Пример вывода:

Вы находитесь в пространстве имён "my-app-dev".

```
Контекст:   prod/workload-a
Кластер:   acp:prod:workload-a
Сервер:     https://acp.prod.example.com/kubernetes/workload-a/
Платформа: https://acp.prod.example.com/
Сессия:    prod
```

Переключить пространство имён:

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

## `ac config sync` — синхронизация конфигурации платформы

```
# Синхронизировать текущую сессию платформы
ac config sync

# Синхронизировать конкретную сессию
ac config sync --session prod

# Синхронизировать все сессии
ac config sync --all
```

Команда `sync` обновляет вашу конфигурацию с последней информацией с платформ АСР, добавляя новые кластеры и обновляя учётные данные по мере необходимости.

# Структура конфигурации ACP CLI

ACP CLI хранит всю информацию конфигурации в стандартном файле

`~/.kube/config`, обеспечивая полную совместимость с `kubectl` и другими инструментами Kubernetes, при этом добавляя специфичные для ACP расширения.

## Расширенная структура `kubeconfig` ACP CLI

ACP CLI расширяет стандартный формат `kubeconfig` метаданными ACP для улучшенной интеграции с платформой:



```
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 behavior, do not edit this item.

```

## Структура и организация метаданных

ACP CLI использует расширения метаданных для организации и идентификации записей конфигурации:

### Идентификация на основе метаданных

- **Идентификация платформы:** используется `platformUrl` для определения родительской платформы
- **Ассоциация сессии:** используется `sessionName` для группировки связанных кластеров, пользователей и контекстов
- **Обнаружение глобального кластера:** используется поле `isGlobal` для определения управляющих кластеров
- **Расположение учётных данных пользователя:** совпадение `sessionName` и `platformUrl` в расширениях пользователя

### Конвенции именования

ACP CLI использует единообразные правила именования при создании новых записей:

- **Записи кластеров:** `acp:<session_name>:<cluster_name>` (например, `acp:prod:global`)
- **Записи пользователей:** `acp:<session_name>:user` (например, `acp:prod:user`)
- **Записи контекстов:** `<session_name>/<cluster_name>` (например, `prod/global`)

## NOTE

Префикс `аср:` гарантирует, что записи, управляемые ACP CLI, не конфликтуют с существующими записями `kubecfg`. Пользователи могут вручную переименовывать эти записи — ACP CLI использует метаданные для идентификации, а не имена.

## Ручная настройка профилей CLI

Для продвинутых пользователей, которым требуется точный контроль над конфигурацией, ACP CLI поддерживает все стандартные команды `kubectl config` для ручного управления `kubecfg`.

## TIP

Большинству пользователей рекомендуется использовать удобные команды, описанные выше.

Команды ручной настройки полезны для продвинутых сценариев:

- **Пользовательское именование контекстов** — создание контекстов, не следующих конвенциям ACP CLI
- **Среды без ACP** — управление традиционными контекстами `kubectl` наряду с сессиями ACP
- **Сложные мультиконтекстные сценарии** — продвинутые рабочие процессы, требующие точного контроля контекстов
- **Отладка проблем с конфигурацией** — диагностика или исправление проблем конфигурации

## Стандартные команды конфигурации

ACP CLI обеспечивает полную совместимость с подкомандами `kubectl config`:

Подкоманда	Использование
<code>set-cluster</code>	Устанавливает запись кластера в файле конфигурации CLI
<code>set-context</code>	Устанавливает запись контекста в файле конфигурации CLI
<code>use-context</code>	Устанавливает текущий контекст по указанному имени
<code>set</code>	Устанавливает отдельное значение в файле конфигурации CLI
<code>unset</code>	Снимает отдельные значения в файле конфигурации CLI
<code>view</code>	Отображает объединённую конфигурацию CLI, используемую в данный момент

## Примеры ручных операций

Создать пользовательский контекст:

```
# Создать контекст с пользовательским именем
ac config set-context my-custom-context --cluster=acp:prod:workload-a --namespace=my-app

# Переключиться на пользовательский контекст
ac config use-context my-custom-context
```

Просмотреть текущую конфигурацию:

```
# Показать объединённую конфигурацию
ac config view

# Показать конфигурацию из конкретного файла
ac config view --config=/path/to/config
```

Обновить пространство имён контекста:

```
# Установить пространство имён для текущего контекста
ac config set-context `ac config current-context` --namespace=my-namespace
```

## Правила загрузки и слияния

Вы можете следовать этим правилам при выполнении операций CLI для порядка загрузки и слияния конфигурации CLI:

- Файлы конфигурации CLI загружаются с вашей рабочей станции, используя следующую иерархию и правила слияния:
  - Если задан параметр `--config`, загружается только этот файл. Флаг устанавливается один раз, слияние не происходит.
  - Если установлена переменная окружения `$KUBECONFIG`, используется она. Переменная может содержать список путей, которые сливаются вместе. При изменении значения оно изменяется в файле, где определён соответствующий раздел. При создании значения оно создаётся в первом существующем файле. Если ни один файл из списка не существует, создаётся последний файл в списке.
  - В противном случае используется файл `~/.kube/config` без слияния.
- Контекст для использования определяется по первому совпадению в следующем порядке:
  - Значение параметра `--context`.
  - Значение `current-context` из файла конфигурации CLI.
  - На этом этапе допускается пустое значение.
- Пользователь и кластер для использования определяются следующим образом. На этом этапе контекст может быть или не быть; они строятся по первому совпадению в следующем порядке, который выполняется отдельно для пользователя и для кластера:
  - Значение параметра `--user` для имени пользователя и `--cluster` для имени кластера.
  - Если присутствует параметр `--context`, используется значение из контекста.

- На этом этапе допускается пустое значение.
- Определяется фактическая информация о кластере. На этом этапе информация о кластере может быть или отсутствовать. Каждое поле информации о кластере строится по первому совпадению в следующем порядке:
  - Значения любых из следующих параметров командной строки: `--server`, `--api-version`, `--certificate-authority`, `--insecure-skip-tls-verify`
  - Если информация о кластере и значение атрибута присутствуют, используется оно.
  - Если адрес сервера отсутствует, возникает ошибка.
- Определяется фактическая информация о пользователе. Пользователи строятся по тем же правилам, что и кластеры, за исключением того, что у пользователя может быть только один метод аутентификации; конфликтующие методы вызывают ошибку операции. Параметры командной строки имеют приоритет над значениями из файла конфигурации. Допустимые параметры командной строки:
  - `--auth-path`
  - `--client-certificate`
  - `--client-key`
  - `--token`
- Для любой отсутствующей информации используются значения по умолчанию, а также запрашивается дополнительная информация.

# Расширение ACP CLI с помощью плагинов

Вы можете писать и устанавливать плагины для расширения стандартных команд `ac`, что позволит выполнять новые и более сложные задачи с помощью ACP CLI и интеграции с платформой ACP.

## Содержание

### [Написание плагинов для CLI](#)

Создание простого плагина

Требования к разработке плагинов

Дополнительные ресурсы

Установка и использование плагинов CLI

Предварительные требования

Процедура установки

## Написание плагинов для CLI

Вы можете написать плагин для ACP CLI (`ac`) на любом языке программирования или скрипте, который позволяет создавать команды командной строки. Обратите внимание, что плагин не может перезаписывать существующую команду `ac`.

### Создание простого плагина

В этой процедуре создаётся простой Bash-плагин, который выводит сообщение в терминал при выполнении команды `ac foo`.

## Процедура

1. Создайте файл с именем `ac-foo`. При выборе имени файла плагина учитывайте следующее:

- Файл должен начинаться с `ac-` или `kubectl-`, чтобы распознаваться как плагин
- Имя файла определяет команду, которая вызывает плагин. Например, плагин с именем файла `ac-foo-bar` вызывается командой `ac foo bar`
- Можно использовать подчёркивания, если хотите, чтобы команда содержала дефисы. Например, плагин с именем файла `ac-foo_bar` вызывается командой `ac foo-bar`

2. Добавьте в файл следующий код:

```
#!/bin/bash

# Обработка необязательных аргументов
if [[ "$1" == "version" ]]; then
    echo "1.0.0"
    exit 0
fi

# Обработка необязательных аргументов
if [[ "$1" == "config" ]]; then
    echo $KUBECONFIG
    exit 0
fi

echo "Я плагин с именем ac-foo"
```

После установки этого плагина для ACP CLI его можно вызвать командой `ac foo`.

## Требования к разработке плагинов

- **Язык программирования:** используйте любой язык программирования или скрипт, поддерживающий интерфейс командной строки

- **Правила именования:** файлы плагинов должны соответствовать шаблону `ac-<plugin-name>` или `kubectl-<plugin-name>`
- **Исполняемый файл:** файлы плагинов должны иметь права на выполнение
- **Перезапись команд:** плагины не могут перезаписывать существующие команды ACP CLI
- **Обработка аргументов:** плагины должны корректно обрабатывать стандартные аргументы и флаги командной строки

## Дополнительные ресурсы

- Ознакомьтесь с руководствами по разработке плагинов `kubectl` для изучения шаблонов реализации и лучших практик
- Используйте утилиты CLI runtime для разработки плагинов на Go
- Учитывайте интеграцию с платформой ACP при проектировании плагинов, взаимодействующих с ресурсами кластера

## Установка и использование плагинов CLI

После написания пользовательского плагина для ACP CLI его необходимо установить перед использованием.

### Предварительные требования

- Установлен ACP CLI (`ac`)
- Имеется файл плагина CLI, начинающийся с `ac-` или `kubectl-`

### Процедура установки

1. При необходимости сделайте файл плагина исполняемым:

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

2. Поместите файл в любую директорию из вашего PATH, например, `/usr/local/bin/`:

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

3. Выполните команду `ac plugin list`, чтобы убедиться, что плагин отображается в списке:

```
ac plugin list
```

### Пример вывода

```
The following compatible plugins are available:
```

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

Если ваш плагин не отображается, проверьте, что файл начинается с `ac-` или `kubectl-`, имеет права на выполнение и находится в PATH.

4. Вызовите новую команду или опцию, добавленную плагином.

Например, если вы создали и установили плагин `ac-ns`, вы можете использовать команду для просмотра текущего namespace:

```
ac ns
```

Обратите внимание, что команда для вызова плагина зависит от имени файла плагина. Например, плагин с именем файла `ac-foo-bar` вызывается командой `ac foo bar`.

# AC CLI Developer Command Reference

Этот справочник содержит описания и примеры команд разработчика AC CLI. Для команд администратора смотрите справочник команд администратора AC CLI.

Выполните `ac help`, чтобы вывести список всех команд, или `ac <command> --help`, чтобы получить дополнительные сведения по конкретной команде.

## Содержание

### [ac annotate](#)

Пример использования

### [ac api-resources](#)

Пример использования

### [ac api-versions](#)

Пример использования

### [ac apply](#)

Пример использования

### [ac apply edit-last-applied](#)

Пример использования

### [ac apply set-last-applied](#)

Пример использования

### [ac apply view-last-applied](#)

Пример использования

### [ac attach](#)

Пример использования

### [ac auth](#)

ac auth can-i

Пример использования

ac auth reconcile

Пример использования

ac auth whoami

Пример использования

ac autoscale

Пример использования

ac cluster-info

Пример использования

ac cluster-info dump

Пример использования

ac completion

Пример использования

ac config

ac config current-context

Пример использования

ac config delete-cluster

Пример использования

ac config delete-context

Пример использования

ac config delete-user

Пример использования

ac config get-clusters

Пример использования

ac config get-contexts

Пример использования

ac config get-sessions

Пример использования

ac config get-users

Пример использования

ac config rename-context

Пример использования

```
ac config set
```

Пример использования

```
ac config set-cluster
```

Пример использования

```
ac config set-context
```

Пример использования

```
ac config set-credentials
```

Пример использования

```
ac config sync
```

Пример использования

```
ac config unset
```

Пример использования

```
ac config use-cluster
```

Пример использования

```
ac config use-context
```

Пример использования

```
ac config use-session
```

Пример использования

```
ac config view
```

Пример использования

```
ac cp
```

Пример использования

```
ac create
```

Пример использования

```
ac create clusterrole
```

Пример использования

```
ac create clusterrolebinding
```

Пример использования

```
ac create configmap
```

Пример использования

```
ac create cronjob
```

Пример использования

```
ac create deployment
```

Пример использования

```
ac create ingress
```

Пример использования

```
ac create job
```

Пример использования

```
ac create namespace
```

Пример использования

```
ac create poddisruptionbudget
```

Пример использования

```
ac create priorityclass
```

Пример использования

```
ac create quota
```

Пример использования

```
ac create role
```

Пример использования

```
ac create rolebinding
```

Пример использования

```
ac create secret
```

```
ac create secret docker-registry
```

Пример использования

```
ac create secret generic
```

Пример использования

```
ac create secret tls
```

Пример использования

```
ac create service
```

```
ac create service clusterip
```

Пример использования

```
ac create service externalname
```

Пример использования

```
ac create service loadbalancer
```

Пример использования

```
ac create service nodeport
```

Пример использования

```
ac create serviceaccount
```

Пример использования

```
ac create token
```

Пример использования

```
ac delete
```

Пример использования

```
ac describe
```

Пример использования

```
ac diff
```

Пример использования

```
ac edit
```

Пример использования

```
ac events
```

Пример использования

```
ac exec
```

Пример использования

```
ac explain
```

Пример использования

```
ac expose
```

Пример использования

```
ac get
```

Пример использования

```
ac kustomize
```

Пример использования

```
ac label
```

Пример использования

```
ac login
```

Пример использования

```
ac logout
```

Пример использования

```
ac logs
```

Пример использования

```
ac namespace
```

Пример использования

```
ac patch
```

Пример использования

```
ac plugin
```

Пример использования

```
ac plugin list
```

Пример использования

```
ac port-forward
```

Пример использования

```
ac process
```

Пример использования

```
ac proxy
```

Пример использования

```
ac replace
```

Пример использования

```
ac rollout
```

Пример использования

```
ac rollout history
```

Пример использования

```
ac rollout pause
```

Пример использования

```
ac rollout restart
```

Пример использования

```
ac rollout resume
```

Пример использования

```
ac rollout status
```

Пример использования

```
ac rollout undo
```

Пример использования

```
ac run
```

Пример использования

```
ac scale
```

Пример использования

```
ac set
```

```
ac set env
```

Пример использования

```
ac set image
```

Пример использования

```
ac set resources
```

Пример использования

```
ac set selector
```

Пример использования

```
ac set serviceaccount
```

Пример использования

```
ac set subject
```

Пример использования

```
ac version
```

Пример использования

```
ac wait
```

Пример использования

---

## ac annotate

Обновить аннотации ресурса

### Пример использования

---

```
# Обновить pod 'foo' с аннотацией 'description' и значением 'my frontend'
# Если одна и та же аннотация указана несколько раз, будет применено только последнее значение
ac annotate pods foo description='my frontend'

# Обновить pod, указанный по типу и имени в "pod.json"
ac annotate -f pod.json description='my frontend'

# Обновить pod 'foo' с аннотацией 'description' и значением 'my frontend running nginx', перезаписывая существующее значение
ac annotate --overwrite pods foo description='my frontend running nginx'

# Обновить все pods в namespace
ac annotate pods --all description='my frontend running nginx'

# Обновить pod 'foo' только если ресурс не изменился с версии 1
ac annotate pods foo description='my frontend running nginx' --resource-version=1

# Удалить аннотацию с именем 'description' у pod 'foo', если она существует
# Флаг --overwrite не требуется
ac annotate pods foo description-
```

## ac api-resources

Вывести поддерживаемые API ресурсы на сервере

## Пример использования

```
# Вывести поддерживаемые API ресурсы
ac api-resources

# Вывести поддерживаемые API ресурсы с дополнительной информацией
ac api-resources -o wide

# Вывести поддерживаемые API ресурсы, отсортированные по столбцу
ac api-resources --sort-by=name

# Вывести поддерживаемые ресурсы с namespace
ac api-resources --namespaced=true

# Вывести поддерживаемые ресурсы без namespace
ac api-resources --namespaced=false

# Вывести поддерживаемые API ресурсы с определённой APIGroup
ac api-resources --api-group=rbac.authorization.k8s.io
```

## ac api-versions

Вывести поддерживаемые версии API на сервере в формате "group/version"

### Пример использования

```
# Вывести поддерживаемые версии API
ac api-versions
```

## ac apply

Применить конфигурацию к ресурсу по имени файла или stdin

### Пример использования

```
# Применить конфигурацию из pod.json к pod
ac apply -f ./pod.json

# Применить ресурсы из директории с kustomization.yaml, например dir/kust
omization.yaml
ac apply -k dir/

# Применить JSON, переданный через stdin, к pod
cat pod.json | ac apply -f -

# Применить конфигурацию из всех файлов, оканчивающихся на '.json'
ac apply -f '*.json'

# Примечание: --prune пока в Alpha
# Применить конфигурацию из manifest.yaml, соответствующую метке app=ngin
x, и удалить все остальные ресурсы с этой меткой, не входящие в файл
ac apply --prune -f manifest.yaml -l app=nginx

# Применить конфигурацию из manifest.yaml и удалить все остальные config
maps, не входящие в файл
ac apply --prune -f manifest.yaml --all --prune-allowlist=core/v1/ConfigM
ap
```

## ac apply edit-last-applied

Редактировать последние аннотации last-applied-configuration ресурса/объекта

### Пример использования

```
# Редактировать last-applied-configuration по типу/имени в YAML
ac apply edit-last-applied deployment/nginx

# Редактировать last-applied-configuration по файлу в JSON
ac apply edit-last-applied -f deploy.yaml -o json
```

## ac apply set-last-applied

Установить аннотацию `last-applied-configuration` на живом объекте в соответствии с содержимым файла

## Пример использования

```
# Установить last-applied-configuration ресурса в соответствии с содержимым файла
ac apply set-last-applied -f deploy.yaml

# Выполнить set-last-applied для каждого файла конфигурации в директории
ac apply set-last-applied -f path/

# Установить last-applied-configuration ресурса в соответствии с содержимым файла; создать аннотацию, если её ещё нет
ac apply set-last-applied -f deploy.yaml --create-annotation=true
```

## ac apply view-last-applied

Просмотреть последние аннотации `last-applied-configuration` ресурса/объекта

## Пример использования

```
# Просмотреть last-applied-configuration по типу/имени в YAML
ac apply view-last-applied deployment/nginx

# Просмотреть last-applied-configuration по файлу в JSON
ac apply view-last-applied -f deploy.yaml -o json
```

## ac attach

Подключиться к запущенному контейнеру

## Пример использования

```
# Получить вывод из запущенного pod mypod; использовать аннотацию 'ac.kubernetes.io/default-container'  
# для выбора контейнера, к которому подключаться, или будет выбран первый контейнер в pod  
ac attach mypod  
  
# Получить вывод из контейнера ruby-container в pod mypod  
ac attach mypod -c ruby-container  
  
# Переключиться в режим raw terminal; отправляет stdin в 'bash' в ruby-container из pod mypod  
# и отправляет stdout/stderr от 'bash' обратно клиенту  
ac attach mypod -c ruby-container -i -t  
  
# Получить вывод из первого pod replica set с именем nginx  
ac attach rs/nginx
```

## ac auth

Проверить авторизацию

## ac auth can-i

Проверить, разрешено ли действие

## Пример использования

```
# Проверить, могу ли я создавать pods в любом namespace
ac auth can-i create pods --all-namespaces

# Проверить, могу ли я просматривать deployments в текущем namespace
ac auth can-i list deployments.apps

# Проверить, может ли сервисный аккаунт "foo" из namespace "dev" просматривать pods в namespace "prod"
# Для использования глобальной опции "--as" необходимо разрешение на имперсонацию
ac auth can-i list pods --as=system:serviceaccount:dev:foo -n prod

# Проверить, могу ли я делать всё в текущем namespace ("*" означает все)
ac auth can-i '*' '*'

# Проверить, могу ли я получить job с именем "bar" в namespace "foo"
ac auth can-i list jobs.batch/bar -n foo

# Проверить, могу ли я читать логи pod
ac auth can-i get pods --subresource=log

# Проверить, могу ли я получить доступ к URL /logs/
ac auth can-i get /logs/

# Проверить, могу ли я одобрять certificates.k8s.io
ac auth can-i approve certificates.k8s.io

# Вывести все разрешённые действия в namespace "foo"
ac auth can-i --list --namespace=foo
```

## ac auth reconcile

Согласовать правила для объектов RBAC role, role binding, cluster role и cluster role binding

## Пример использования

```
# Согласовать RBAC ресурсы из файла  
ac auth reconcile -f my-rbac-rules.yaml
```

## ac auth whoami

Экспериментально: проверить атрибуты субъекта

### Пример использования

```
# Получить атрибуты субъекта  
ac auth whoami  
  
# Получить атрибуты субъекта в формате JSON  
ac auth whoami -o json
```

## ac autoscale

Автоматически масштабировать deployment, replica set, stateful set или replication controller

### Пример использования

```
# Автоматически масштабировать deployment "foo" с количеством pod от 2 до  
10, без указания целевого использования CPU, будет применена политика авт  
оскейлинга по умолчанию  
ac autoscale deployment foo --min=2 --max=10  
  
# Автоматически масштабировать replication controller "foo" с количеством  
pod от 1 до 5, целевое использование CPU 80%  
ac autoscale rc foo --max=5 --cpu-percent=80
```

## ac cluster-info

Показать информацию о кластере

### Пример использования

```
# Вывести адрес управляющей плоскости и сервисов кластера  
ac cluster-info
```

## ac cluster-info dump

Снять дамп релевантной информации для отладки и диагностики

### Пример использования

```
# Снять дамп текущего состояния кластера в stdout  
ac cluster-info dump  
  
# Снять дамп текущего состояния кластера в /path/to/cluster-state  
ac cluster-info dump --output-directory=/path/to/cluster-state  
  
# Снять дамп всех namespace в stdout  
ac cluster-info dump --all-namespaces  
  
# Снять дамп набора namespace в /path/to/cluster-state  
ac cluster-info dump --namespaces default,kube-system --output-directory  
=/path/to/cluster-state
```

## ac completion

Вывести код автодополнения для указанного shell (bash, zsh, fish или powershell)

### Пример использования



```
# Установка автодополнения bash на macOS через homebrew
## Если используется Bash 3.2, включённый в macOS
brew install bash-completion
## или, если используется Bash 4.1+
brew install bash-completion@2
## Если ac установлен через homebrew, автодополнение начнёт работать сразу
## Если установлен другим способом, возможно, потребуется добавить автодополнение в каталог completions
ac completion bash > $(brew --prefix)/etc/bash_completion.d/ac
```

```
# Установка автодополнения bash на Linux
## Если bash-completion не установлен, установите пакет 'bash-completion' через менеджер пакетов дистрибутива.
## Загрузить код автодополнения ac для bash в текущую сессию shell
source <(ac completion bash)
## Записать код автодополнения bash в файл и подключить его из .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
```

```
# Загрузить код автодополнения ac для zsh[1] в текущую сессию shell
source <(ac completion zsh)
# Установить автодополнение ac для zsh[1] для автозагрузки при старте
ac completion zsh > "${fpath[1]}/_ac"
```

```
# Загрузить код автодополнения ac для fish[2] в текущую сессию shell
ac completion fish | source
# Чтобы загружать автодополнения в каждой сессии, выполните один раз:
ac completion fish > ~/.config/fish/completions/ac.fish
```

```
# Загрузить код автодополнения ac для powershell в текущую сессию shell
ac completion powershell | Out-String | Invoke-Expression
# Установить автодополнение ac для powershell для запуска при старте
## Сохранить код автодополнения в скрипт и выполнить его в профиле
ac completion powershell > $HOME\.kube\completion.ps1
Add-Content $PROFILE "$HOME\.kube\completion.ps1"
```

```
## Выполнить код автодополнения в профиле
Add-Content $PROFILE "if (Get-Command ac -ErrorAction SilentlyContinue) {
ac completion powershell | Out-String | Invoke-Expression
}"
## Добавить код автодополнения напрямую в скрипт $PROFILE
ac completion powershell >> $PROFILE
```

## ac config

Изменять kubernetes файлы

## ac config current-context

Показать текущий контекст

## Пример использования

```
# Показать текущий контекст
ac config current-context
```

## ac config delete-cluster

Удалить указанный кластер из kubernetes

## Пример использования

```
# Удалить кластер minikube
ac config delete-cluster minikube
```

## ac config delete-context

Удалить указанный контекст из kubernetes

## Пример использования

```
# Удалить контекст для кластера minikube  
ac config delete-context minikube
```

## ac config delete-user

Удалить указанного пользователя из kubeconfig

## Пример использования

```
# Удалить пользователя minikube  
ac config delete-user minikube
```

## ac config get-clusters

Показать кластеры, определённые в kubeconfig

## Пример использования

```
# Вывести список кластеров, известных ac  
ac config get-clusters
```

## ac config get-contexts

Описать один или несколько контекстов

## Пример использования

```
# Вывести все контексты в kubeconfig
```

```
ac config get-contexts
```

```
# Описать один контекст в kubeconfig
```

```
ac config get-contexts my-context
```

## ac config get-sessions

Вывести все настроенные сессии АСР платформы

### Пример использования

```
# Вывести все настроенные сессии АСР
```

```
ac config get-sessions
```

## ac config get-users

Показать пользователей, определённых в kubeconfig

### Пример использования

```
# Вывести список пользователей, известных ac
```

```
ac config get-users
```

## ac config rename-context

Переименовать контекст в kubeconfig

### Пример использования

```
# Переименовать контекст 'old-name' в 'new-name' в kubeconfig  
ac config rename-context old-name new-name
```

## ac config set

Установить отдельное значение в kubeconfig файле

### Пример использования

```
# Установить поле server для кластера my-cluster в https://1.2.3.4  
ac config set clusters.my-cluster.server https://1.2.3.4  
  
# Установить поле certificate-authority-data для кластера my-cluster  
ac config set clusters.my-cluster.certificate-authority-data $(echo "cert  
_data_here" | base64 -i -)  
  
# Установить поле cluster в контексте my-context в my-cluster  
ac config set contexts.my-context.cluster my-cluster  
  
# Установить поле client-key-data для пользователя cluster-admin с опцией  
--set-raw-bytes  
ac config set users.cluster-admin.client-key-data cert_data_here --set-ra  
w-bytes=true
```

## ac config set-cluster

Установить запись кластера в kubeconfig

### Пример использования

```
# Установить только поле server для записи кластера e2e без изменения других значений
ac config set-cluster e2e --server=https://1.2.3.4

# Встроить данные сертификата для записи кластера e2e
ac config set-cluster e2e --embed-certs --certificate-authority=~/.kube/e2e/kubernetes.ca.crt

# Отключить проверку сертификата для записи кластера e2e
ac config set-cluster e2e --insecure-skip-tls-verify=true

# Установить пользовательское имя TLS сервера для проверки в записи кластера e2e
ac config set-cluster e2e --tls-server-name=my-cluster-name

# Установить URL прокси для записи кластера e2e
ac config set-cluster e2e --proxy-url=https://1.2.3.4
```

## ac config set-context

Установить запись контекста в kubeconfig

### Пример использования

```
# Установить поле user для записи контекста gce без изменения других значений
ac config set-context gce --user=cluster-admin
```

## ac config set-credentials

Установить запись пользователя в kubeconfig

### Пример использования



```
# Установить только поле "client-key" для "cluster-admin"
# без изменения других значений
ac config set-credentials cluster-admin --client-key=~/.kube/admin.key

# Установить базовую аутентификацию для "cluster-admin"
ac config set-credentials cluster-admin --username=admin --password=uXFGw
eU9l35qcif

# Встроить данные клиентского сертификата в запись "cluster-admin"
ac config set-credentials cluster-admin --client-certificate=~/.kube/admi
n.crt --embed-certs=true

# Включить провайдера аутентификации Google Compute Platform для "cluster
-admin"
ac config set-credentials cluster-admin --auth-provider=gcp

# Включить провайдера аутентификации OpenID Connect для "cluster-admin" с
дополнительными аргументами
ac config set-credentials cluster-admin --auth-provider=oidc --auth-provi
der-arg=client-id=foo --auth-provider-arg=client-secret=bar

# Удалить конфигурационное значение "client-secret" для провайдера OpenID
Connect для "cluster-admin"
ac config set-credentials cluster-admin --auth-provider=oidc --auth-provi
der-arg=client-secret-

# Включить новый exec auth плагин для "cluster-admin"
ac config set-credentials cluster-admin --exec-command=/path/to/the/execu
table --exec-api-version=client.authentication.k8s.io/v1beta1

# Включить новый exec auth плагин для "cluster-admin" с интерактивным реж
имом
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

# Определить новые аргументы exec auth плагина для "cluster-admin"
ac config set-credentials cluster-admin --exec-arg=arg1 --exec-arg=arg2

# Создать или обновить переменные окружения exec auth плагина для "cluste
r-admin"
ac config set-credentials cluster-admin --exec-env=key1=val1 --exec-env=k
ey2=val2
```

```
# Удалить переменные окружения exec auth плагина для "cluster-admin"  
ac config set-credentials cluster-admin --exec-env=var-to-remove-
```

## ac config sync

Синхронизировать kubernetes с состоянием ACP платформы

### Пример использования

```
# Синхронизировать текущую сессию на основе активного контекста  
ac config sync  
  
# Синхронизировать конкретную сессию  
ac config sync --session prod  
  
# Синхронизировать все сессии  
ac config sync --all
```

## ac config unset

Сбросить отдельное значение в kubernetes файле

### Пример использования

```
# Сбросить current-context  
ac config unset current-context  
  
# Сбросить namespace в контексте foo  
ac config unset contexts.foo.namespace
```

## ac config use-cluster

Переключиться на конкретный ACP кластер по имени кластера

## Пример использования

```
# Переключиться на существующий контекст для кластера workload-a
ac config use-cluster workload-a

# Создать новый контекст для workload-b с namespace
ac config use-cluster workload-b --namespace my-app

# Переключиться на глобальный кластер
ac config use-cluster global
```

## ac config use-context

Установить current-context в kubeconfig файле

## Пример использования

```
# Использовать контекст для кластера minikube
ac config use-context minikube
```

## ac config use-session

Переключиться на указанную сессию ACP с интеллектуальным выбором контекста

## Пример использования

```
# Переключиться на сессию staging (по умолчанию глобальный кластер)
ac config use-session staging

# Переключиться на сессию production с конкретным кластером
ac config use-session prod --cluster workload-b

# Переключиться на сессию с конкретным кластером и namespace
ac config use-session staging --cluster workload-a --namespace my-app
```

## ac config view

Показать объединённые настройки kubernetes или указанный kubernetes файл

### Пример использования

```
# Показать объединённые настройки kubernetes
ac config view

# Показать объединённые настройки kubernetes, сырые данные сертификатов и раскрытые секреты
ac config view --raw

# Получить пароль для пользователя e2e
ac config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'
```

## ac cp

Копировать файлы и директории в контейнеры и из них

### Пример использования

```
# !!!Важное замечание!!!
# Требуется, чтобы в вашем контейнере был установлен бинарник 'tar'.
# Если 'tar' отсутствует, 'ас ср' завершится с ошибкой.
#
# Для продвинутых случаев, таких как символические ссылки, расширение шаблонов или сохранение прав доступа, рассмотрите использование 'ас ехес'.

# Копировать локальный файл /tmp/foo в /tmp/bar в удалённом pod в namespace <some-namespace>
tar cf - /tmp/foo | ас ехес -i -n <some-namespace> <some-pod> -- tar xf - -C /tmp/bar

# Копировать /tmp/foo из удалённого pod в локальный /tmp/bar
ас ехес -n <some-namespace> <some-pod> -- tar cf - /tmp/foo | tar xf - -C /tmp/bar

# Копировать локальную директорию /tmp/foo_dir в /tmp/bar_dir в удалённом pod в default namespace
ас ср /tmp/foo_dir <some-pod>:/tmp/bar_dir

# Копировать локальный файл /tmp/foo в /tmp/bar в удалённом pod в конкретном контейнере
ас ср /tmp/foo <some-pod>:/tmp/bar -c <specific-container>

# Копировать локальный файл /tmp/foo в /tmp/bar в удалённом pod в namespace <some-namespace>
ас ср /tmp/foo <some-namespace>/<some-pod>:/tmp/bar

# Копировать /tmp/foo из удалённого pod в локальный /tmp/bar
ас ср <some-namespace>/<some-pod>:/tmp/foo /tmp/bar
```

## ас create

Создать ресурс из файла или stdin

## Пример использования

```
# Создать pod из данных в pod.json
```

```
ac create -f ./pod.json
```

```
# Создать pod из JSON, переданного через stdin
```

```
cat pod.json | ac create -f -
```

```
# Отредактировать данные в registry.yaml в JSON, затем создать ресурс с и  
использованием отредактированных данных
```

```
ac create -f registry.yaml --edit -o json
```

## ac create clusterrole

Создать cluster role

### Пример использования

```
# Создать cluster role с именем "pod-reader", разрешающую пользователю вы  
полнять "get", "watch" и "list" на pods
```

```
ac create clusterrole pod-reader --verb=get,list,watch --resource=pods
```

```
# Создать cluster role с именем "pod-reader" с указанным ResourceName
```

```
ac create clusterrole pod-reader --verb=get --resource=pods --resource-na  
me=readablepod --resource-name=anotherpod
```

```
# Создать cluster role с именем "foo" с указанной API Group
```

```
ac create clusterrole foo --verb=get,list,watch --resource=rs.apps
```

```
# Создать cluster role с именем "foo" с указанным SubResource
```

```
ac create clusterrole foo --verb=get,list,watch --resource=pods,pods/stat  
us
```

```
# Создать cluster role с именем "foo" с указанным NonResourceURL
```

```
ac create clusterrole "foo" --verb=get --non-resource-url=/logs/*
```

```
# Создать cluster role с именем "monitoring" с указанным AggregationRule
```

```
ac create clusterrole monitoring --aggregation-rule="rbac.example.com/agg  
regate-to-monitoring=true"
```

# ac create clusterrolebinding

Создать cluster role binding для конкретной cluster role

## Пример использования

```
# Создать cluster role binding для user1, user2 и group1 с использованием cluster-admin cluster role
ac create clusterrolebinding cluster-admin --clusterrole=cluster-admin --user=user1 --user=user2 --group=group1
```

# ac create configmap

Создать config map из локального файла, директории или литерального значения

## Пример использования

```
# Создать новый config map с именем my-config на основе папки bar
ac create configmap my-config --from-file=path/to/bar

# Создать новый config map с именем my-config с указанными ключами вместо имён файлов на диске
ac create configmap my-config --from-file=key1=/path/to/bar/file1.txt --from-file=key2=/path/to/bar/file2.txt

# Создать новый config map с именем my-config с key1=config1 и key2=config2
ac create configmap my-config --from-literal=key1=config1 --from-literal=key2=config2

# Создать новый config map с именем my-config из ключ=значение в файле
ac create configmap my-config --from-file=path/to/bar

# Создать новый config map с именем my-config из env файла
ac create configmap my-config --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

# ac create cronjob

Создать cron job с указанным именем

## Пример использования

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

# Создать cron job с командой
ac create cronjob my-job --image=busybox --schedule="*/1 * * * *" -- date
```

# ac create deployment

Создать deployment с указанным именем

## Пример использования

```
# Создать deployment с именем my-dep, запускающий образ busybox
ac create deployment my-dep --image=busybox

# Создать deployment с командой
ac create deployment my-dep --image=busybox -- date

# Создать deployment с именем my-dep, запускающий образ nginx с 3 реплика
ми
ac create deployment my-dep --image=nginx --replicas=3

# Создать deployment с именем my-dep, запускающий образ busybox и открыва
ющий порт 5701
ac create deployment my-dep --image=busybox --port=5701

# Создать deployment с именем my-dep, запускающий несколько контейнеров
ac create deployment my-dep --image=busybox:latest --image=ubuntu:latest
--image=nginx
```

# ac create ingress

Создать ingress с указанным именем

## Пример использования

```
# Создать одиночный ingress с именем 'simple', который направляет запросы
foo.com/bar к svc
# svc1:8080 с TLS секретом "my-cert"
ac create ingress simple --rule="foo.com/bar=svc1:8080,tls=my-cert"

# Создать catch all ingress для "/path", направляющий на сервис svc:port
и класс Ingress "otheringress"
ac create ingress catch-all --class=otheringress --rule="/path=svc:port"

# Создать ingress с двумя аннотациями: ingress.annotation1 и ingress.anno
tation2
ac create ingress annotated --class=default --rule="foo.com/bar=svc:port" \
\
--annotation ingress.annotation1=foo \
--annotation ingress.annotation2=bla

# Создать ingress с одним хостом и несколькими путями
ac create ingress multipath --class=default \
--rule="foo.com/=svc:port" \
--rule="foo.com/admin/=svcadmin:portadmin"

# Создать ingress с несколькими хостами и pathType как Prefix
ac create ingress ingress1 --class=default \
--rule="foo.com/path*=svc:8080" \
--rule="bar.com/admin*=svc2:http"

# Создать ingress с включённым TLS, используя сертификат по умолчанию и р
азные типы путей
ac create ingress ingtls --class=default \
--rule="foo.com/=svc:https,tls" \
--rule="foo.com/path/subpath*=othersvc:8080"

# Создать ingress с включённым TLS, используя конкретный секрет и pathTyp
e как Prefix
ac create ingress ingsecret --class=default \
--rule="foo.com/=svc:8080,tls=secret1"

# Создать ingress с backend по умолчанию
ac create ingress ingdefault --class=default \
--default-backend=defaultsvc:http \
--rule="foo.com/=svc:8080,tls=secret1"
```

## ac create job

Создать job с указанным именем

### Пример использования

```
# Создать job
ac create job my-job --image=busybox

# Создать job с командой
ac create job my-job --image=busybox -- date

# Создать job из cron job с именем "a-cronjob"
ac create job test-job --from=cronjob/a-cronjob
```

## ac create namespace

Создать namespace с указанным именем

### Пример использования

```
# Создать новый namespace с именем my-namespace
ac create namespace my-namespace
```

## ac create poddisruptionbudget

Создать pod disruption budget с указанным именем

### Пример использования

```
# Создать pod disruption budget с именем my-pdb, который выбирает все pods с меткой app=rails
# и требует, чтобы хотя бы один из них был доступен в любой момент времени
ac create poddisruptionbudget my-pdb --selector=app=rails --min-available=1

# Создать pod disruption budget с именем my-pdb, который выбирает все pods с меткой app=nginx
# и требует, чтобы было доступно не менее половины выбранных pods в любой момент времени
ac create pdb my-pdb --selector=app=nginx --min-available=50%
```

## ac create priorityclass

Создать priority class с указанным именем

### Пример использования

```
# Создать priority class с именем high-priority
ac create priorityclass high-priority --value=1000 --description="high priority"

# Создать priority class с именем default-priority, считающийся глобальным приоритетом по умолчанию
ac create priorityclass default-priority --value=1000 --global-default=true --description="default priority"

# Создать priority class с именем high-priority, который не может вытеснить pods с более низким приоритетом
ac create priorityclass high-priority --value=1000 --description="high priority" --preemption-policy="Never"
```

## ac create quota

Создать quota с указанным именем

## Пример использования

```
# Создать новый resource quota с именем 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

# Создать новый resource quota с именем best-effort
ac create quota best-effort --hard=pods=100 --scopes=BestEffort
```

## ac create role

Создать роль с одним правилом

## Пример использования

```
# Создать роль с именем "pod-reader", разрешающую выполнять "get", "watc
h" и "list" на pods
ac create role pod-reader --verb=get --verb=list --verb=watch --resource=
pods

# Создать роль с именем "pod-reader" с указанным ResourceName
ac create role pod-reader --verb=get --resource=pods --resource-name=read
ablepod --resource-name=anotherpod

# Создать роль с именем "foo" с указанной API Group
ac create role foo --verb=get,list,watch --resource=rs.apps

# Создать роль с именем "foo" с указанным SubResource
ac create role foo --verb=get,list,watch --resource=pods,pods/status
```

## ac create rolebinding

Создать role binding для конкретной роли или cluster role

## Пример использования

```
# Создать role binding для user1, user2 и group1 с использованием cluster
role admin
ac create rolebinding admin --clusterrole=admin --user=user1 --user=user2
--group=group1

# Создать role binding для service account monitoring:sa-dev с использо
ванием роли admin
ac create rolebinding admin-binding --role=admin --serviceaccount=monitor
ing:sa-dev
```

## ac create secret

Создать secret с использованием указанной подкоманды

## ac create secret docker-registry

Создать secret для использования с Docker registry

## Пример использования

```
# Если у вас ещё нет файла .dockercfg, создайте dockercfg secret напрямую
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

# Создать новый secret с именем my-secret из ~/.docker/config.json
ac create secret docker-registry my-secret --from-file=path/to/.docker/co
nfig.json
```

## ac create secret generic

Создать secret из локального файла, директории или литерального значения

## Пример использования

```
# Создать новый secret с именем my-secret с ключами для каждого файла в папке bar
ac create secret generic my-secret --from-file=path/to/bar

# Создать новый secret с именем my-secret с указанными ключами вместо имен на диске
ac create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-file=ssh-publickey=path/to/id_rsa.pub

# Создать новый secret с именем my-secret с key1=supersecret и key2=topsecret
ac create secret generic my-secret --from-literal=key1=supersecret --from-literal=key2=topsecret

# Создать новый secret с именем my-secret, используя комбинацию файла и литерала
ac create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-literal=passphrase=topsecret

# Создать новый secret с именем my-secret из env файлов
ac create secret generic my-secret --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

## ac create secret tls

Создать TLS secret

## Пример использования

```
# Создать новый TLS secret с именем tls-secret с указанной парой ключей
ac create secret tls tls-secret --cert=path/to/tls.crt --key=path/to/tls.key
```

## ac create service

Создать сервис с использованием указанной подкоманды

## ac create service clusterip

Создать сервис типа ClusterIP

### Пример использования

```
# Создать новый ClusterIP сервис с именем my-cs
ac create service clusterip my-cs --tcp=5678:8080

# Создать новый ClusterIP сервис с именем my-cs (в режиме headless)
ac create service clusterip my-cs --clusterip="None"
```

## ac create service externalname

Создать сервис типа ExternalName

### Пример использования

```
# Создать новый ExternalName сервис с именем my-ns
ac create service externalname my-ns --external-name bar.com
```

## ac create service loadbalancer

Создать сервис типа LoadBalancer

### Пример использования

```
# Создать новый LoadBalancer сервис с именем my-lbs  
ac create service loadbalancer my-lbs --tcp=5678:8080
```

## ac create service nodeport

Создать сервис типа NodePort

### Пример использования

```
# Создать новый NodePort сервис с именем my-ns  
ac create service nodeport my-ns --tcp=5678:8080
```

## ac create serviceaccount

Создать service account с указанным именем

### Пример использования

```
# Создать новый service account с именем my-service-account  
ac create serviceaccount my-service-account
```

## ac create token

Запросить токен service account

### Пример использования

```
# Запросить токен для аутентификации в kube-apiserver как service account
"myapp" в текущем namespace
ac create token myapp

# Запросить токен для service account в пользовательском namespace
ac create token myapp --namespace myns

# Запросить токен с пользовательским временем жизни
ac create token myapp --duration 10m

# Запросить токен с пользовательской аудиторией
ac create token myapp --audience https://example.com

# Запросить токен, связанный с экземпляром объекта Secret
ac create token myapp --bound-object-kind Secret --bound-object-name myse
cret

# Запросить токен, связанный с экземпляром объекта Secret с конкретным UI
D
ac create token myapp --bound-object-kind Secret --bound-object-name myse
cret --bound-object-uid 0d4691ed-659b-4935-a832-355f77ee47cc
```

## ac delete

Удалить ресурсы по именам файлов, stdin, ресурсам и именам или по ресурсам и селектору меток

## Пример использования

```
# Удалить pod по типу и имени, указанным в pod.json
ac delete -f ./pod.json

# Удалить ресурсы из директории с kustomization.yaml, например dir/kustomization.yaml
ac delete -k dir

# Удалить ресурсы из всех файлов, оканчивающихся на '.json'
ac delete -f '*.json'

# Удалить pod по типу и имени из JSON, переданного через stdin
cat pod.json | ac delete -f -

# Удалить pods и services с именами "baz" и "foo"
ac delete pod,service baz foo

# Удалить pods и services с меткой name=myLabel
ac delete pods,services -l name=myLabel

# Удалить pod с минимальной задержкой
ac delete pod foo --now

# Принудительно удалить pod на мёртвом узле
ac delete pod foo --force

# Удалить все pods
ac delete pods --all

# Удалить все pods только после подтверждения пользователя
ac delete pods --all --interactive
```

## ac describe

Показать детали конкретного ресурса или группы ресурсов

## Пример использования

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

# Описать pod
ac describe pods/nginx

# Описать pod, указанный по типу и имени в "pod.json"
ac describe -f pod.json

# Описать все pods
ac describe pods

# Описать pods с меткой name=myLabel
ac describe pods -l name=myLabel

# Описать все pods, управляемые replication controller 'frontend'
# (pods, созданные rc, получают имя rc в качестве префикса)
ac describe pods frontend
```

## ac diff

Показать различия между живой версией и версией, которая будет применена

## Пример использования

```
# Показать различия для ресурсов из pod.json
ac diff -f pod.json

# Показать различия для файла, прочитанного из stdin
cat service.yaml | ac diff -f -
```

## ac edit

Редактировать ресурс на сервере

## Пример использования

```
# Редактировать сервис с именем 'registry'
ac edit svc/registry

# Использовать альтернативный редактор
KUBE_EDITOR="nano" ac edit svc/registry

# Редактировать job 'myjob' в JSON с использованием API версии v1
ac edit job.v1.batch/myjob -o json

# Редактировать deployment 'mydeployment' в YAML и сохранить изменённую к
онфигурацию в аннотациях
ac edit deployment/mydeployment -o yaml --save-config

# Редактировать подресурс 'status' для deployment 'mydeployment'
ac edit deployment mydeployment --subresource='status'
```

## ac events

Список событий

## Пример использования

```
# Вывести последние события в default namespace
```

```
ac events
```

```
# Вывести последние события во всех namespace
```

```
ac events --all-namespaces
```

```
# Вывести последние события для указанного pod, затем ждать и выводить новые события по мере их появления
```

```
ac events --for pod/web-pod-13je7 --watch
```

```
# Вывести последние события в формате YAML
```

```
ac events -oyaml
```

```
# Вывести только события типов 'Warning' или 'Normal'
```

```
ac events --types=Warning,Normal
```

## ac exec

Выполнить команду в контейнере

## Пример использования

# Получить вывод команды 'date' из pod mypod, по умолчанию из первого контейнера

```
ac exec mypod -- date
```

# Получить вывод команды 'date' из контейнера ruby-container в pod mypod

```
ac exec mypod -c ruby-container -- date
```

# Переключиться в raw terminal режим; отправляет stdin в 'bash' в ruby-container из pod mypod

# и отправляет stdout/stderr от 'bash' обратно клиенту

```
ac exec mypod -c ruby-container -i -t -- bash -il
```

# Вывести содержимое /usr из первого контейнера pod mypod, отсортированное по времени изменения

# Если команда содержит флаги, совпадающие с флагами ac exec (например, -i),

# используйте двойной дефис (--) для разделения флагов команды и ac exec

# Не заключайте команду и её аргументы в кавычки, если обычно не делаете этого

```
ac exec mypod -i -t -- ls -t /usr
```

# Получить вывод команды 'date' из первого pod deployment mydeployment, по умолчанию из первого контейнера

```
ac exec deploy/mydeployment -- date
```

# Получить вывод команды 'date' из первого pod сервиса myservice, по умолчанию из первого контейнера

```
ac exec svc/myservice -- date
```

## ac explain

Получить документацию по ресурсу

## Пример использования

# Получить документацию по ресурсу и его полям

```
ac explain pods
```

# Получить все поля ресурса

```
ac explain pods --recursive
```

# Получить объяснение для deployment в поддерживаемых версиях api

```
ac explain deployments --api-version=apps/v1
```

# Получить документацию по конкретному полю ресурса

```
ac explain pods.spec.containers
```

# Получить документацию по ресурсам в другом формате

```
ac explain deployment --output=plaintext-openapi2
```

## ac expose

Взять replication controller, service, deployment или pod и открыть его как новый Kubernetes сервис

## Пример использования

```
# Создать сервис для реплицированного nginx, который слушает порт 80 и подключается к контейнерам на порту 8000
ac expose rc nginx --port=80 --target-port=8000

# Создать сервис для replication controller, указанный по типу и имени в "nginx-controller.yaml", который слушает порт 80 и подключается к контейнерам на порту 8000
ac expose -f nginx-controller.yaml --port=80 --target-port=8000

# Создать сервис для pod valid-pod, который слушает порт 444 с именем "frontend"
ac expose pod valid-pod --port=444 --name=frontend

# Создать второй сервис на основе вышеуказанного, открывая порт контейнера 8443 как порт 443 с именем "nginx-https"
ac expose service nginx --port=443 --target-port=8443 --name=nginx-https

# Создать сервис для реплицированного streaming приложения на порту 4100, балансирующего UDP трафик, с именем 'video-stream'.
ac expose rc streamer --port=4100 --protocol=UDP --name=video-stream

# Создать сервис для replica set nginx, который слушает порт 80 и подключается к контейнерам на порту 8000
ac expose rs nginx --port=80 --target-port=8000

# Создать сервис для deployment nginx, который слушает порт 80 и подключается к контейнерам на порту 8000
ac expose deployment nginx --port=80 --target-port=8000
```

## ac get

Показать один или несколько ресурсов

## Пример использования



# Вывести все pods в формате ps

```
ac get pods
```

# Вывести все pods в формате ps с дополнительной информацией (например, имя node)

```
ac get pods -o wide
```

# Вывести один replication controller с указанным NAME в формате ps

```
ac get replicationcontroller web
```

# Вывести deployments в формате JSON, в версии "v1" API группы "apps"

```
ac get deployments.v1.apps -o json
```

# Вывести один pod в формате JSON

```
ac get -o json pod web-pod-13je7
```

# Вывести pod, указанный по типу и имени в "pod.yaml" в формате JSON

```
ac get -f pod.yaml -o json
```

# Вывести ресурсы из директории с kustomization.yaml, например dir/kustomization.yaml

```
ac get -k dir/
```

# Вывести только значение поля phase указанного pod

```
ac get -o template pod/web-pod-13je7 --template={{.status.phase}}
```

# Вывести информацию о ресурсе в пользовательских столбцах

```
ac get pod test-pod -o custom-columns=CONTAINER:.spec.containers[0].name,IMAGE:.spec.containers[0].image
```

# Вывести все replication controllers и services вместе в формате ps

```
ac get rc,services
```

# Вывести один или несколько ресурсов по типу и имени

```
ac get rc/web service/frontend pods/web-pod-13je7
```

# Вывести подресурс 'status' для одного pod

```
ac get pod web-pod-13je7 --subresource status
```

# Вывести все deployments в namespace 'backend'

```
ac get deployments.apps --namespace backend
```

# Вывести все pods во всех namespace

```
ac get pods --all-namespaces
```

## ac kustomize

Построить kustomization цель из директории или URL

### Пример использования

```
# Построить текущую рабочую директорию
ac kustomize

# Построить из общей директории конфигурации
ac kustomize /home/config/production

# Построить из github
ac kustomize https://github.com/kubernetes-sigs/kustomize.git/examples/helloWorld?ref=v1.0.6
```

## ac label

Обновить метки ресурса

### Пример использования

```
# Обновить pod 'foo' с меткой 'unhealthy' и значением 'true'
ac label pods foo unhealthy=true

# Обновить pod 'foo' с меткой 'status' и значением 'unhealthy', перезаписывая существующее значение
ac label --overwrite pods foo status=unhealthy

# Обновить все pods в namespace
ac label pods --all status=unhealthy

# Обновить pod, указанный по типу и имени в "pod.json"
ac label -f pod.json status=unhealthy

# Обновить pod 'foo' только если ресурс не изменился с версии 1
ac label pods foo status=unhealthy --resource-version=1

# Удалить метку с именем 'bar' у pod 'foo', если она существует
# Флаг --overwrite не требуется
ac label pods foo bar-
```

## ac login

Войти в ACP платформу

## Пример использования

```
# Интерактивный вход (запрашивает отсутствующие параметры)
ac login https://example.com --name prod

# Вход со всеми параметрами через флаги
ac login https://example.com --name prod --username=myuser --password=mypassword

# Вход с использованием переменных окружения (для автоматизации)
AC_LOGIN_PLATFORM_URL=https://example.com AC_LOGIN_SESSION=prod \
AC_LOGIN_USERNAME=myuser AC_LOGIN_PASSWORD=mypassword ac login

# Вход с указанием конкретного провайдера идентификации
ac login https://example.com --name prod --idp ldap-test

# Вход с указанием конкретного кластера и namespace
ac login https://example.com --name prod --cluster=my-cluster --namespace=my-namespace

# Вход с использованием кастомного kubeconfig файла
ac login https://example.com --name prod --kubeconfig=/path/to/kubeconfig
```

## ac logout

Завершить текущую сессию с АСР платформой

### Пример использования

```
# Выйти из текущей сессии АСР платформы
ac logout

# Выйти из конкретной сессии
ac logout --session prod

# Выйти из всех сессий
ac logout --all
```

# ac logs

Вывести логи контейнера в pod

## Пример использования



```
# Получить снимок логов из pod nginx с одним контейнером
ac logs nginx

# Получить снимок логов из pod nginx, добавляя префикс с именем pod и кон
тейнера к каждой строке
ac logs nginx --prefix

# Получить снимок логов из pod nginx, ограничив вывод 500 байтами
ac logs nginx --limit-bytes=500

# Получить снимок логов из pod nginx, ожидая до 20 секунд запуска pod
ac logs nginx --pod-running-timeout=20s

# Получить снимок логов из pod nginx с несколькими контейнерами
ac logs nginx --all-containers=true

# Получить снимок логов из всех pods в deployment nginx
ac logs deployment/nginx --all-pods=true

# Получить снимок логов из всех контейнеров в pods с меткой app=nginx
ac logs -l app=nginx --all-containers=true

# Получить снимок логов из всех pods с меткой app=nginx, ограничив одновр
еменные запросы логов до 10 pods
ac logs -l app=nginx --max-log-requests=10

# Получить снимок предыдущих логов завершённого ruby контейнера из pod we
b-1
ac logs -p -c ruby web-1

# Начать потоковую передачу логов из pod nginx, продолжая даже при ошибка
х
ac logs nginx -f --ignore-errors=true

# Начать потоковую передачу логов ruby контейнера из pod web-1
ac logs -f -c ruby web-1

# Начать потоковую передачу логов из всех контейнеров в pods с меткой app
=nginx
ac logs -f -l app=nginx --all-containers=true

# Показать только последние 20 строк вывода из pod nginx
ac logs --tail=20 nginx
```

```
# Показать все логи из pod nginx за последний час
ac logs --since=1h nginx

# Показать все логи с отметками времени из pod nginx начиная с 30 августа
2024, 06:00:00 UTC
ac logs nginx --since-time=2024-08-30T06:00:00Z --timestamps=true

# Показать логи kubelet с просроченным сертификатом
ac logs --insecure-skip-tls-verify-backend nginx

# Получить снимок логов из первого контейнера job с именем hello
ac logs job/hello

# Получить снимок логов из контейнера nginx-1 deployment с именем nginx
ac logs deployment/nginx -c nginx-1
```

## ac namespace

Показать или переключить текущий namespace контекст

### Пример использования

```
# Показать текущий namespace и информацию о контексте
ac namespace

# Переключиться на другой namespace
ac namespace my-namespace

# Переключиться на default namespace
ac namespace default
```

## ac patch

Обновить поля ресурса

### Пример использования

```
# Частично обновить node с помощью стратегического merge patch, указав па
тч в JSON
ac patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

# Частично обновить node с помощью стратегического merge patch, указав па
тч в YAML
ac patch node k8s-node-1 -p '$spec:\n unschedulable: true'

# Частично обновить node, указанный по типу и имени в "node.json" с помощ
ью стратегического merge patch
ac patch -f node.json -p '{"spec":{"unschedulable":true}}'

# Обновить образ контейнера; spec.containers[*].name обязателен, так как
это ключ слияния
ac patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-ser
ve-hostname","image":"new image"}]}}'

# Обновить образ контейнера с помощью JSON patch с позиционными массивами
ac patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spe
c/containers/0/image", "value":"new image"}]'

# Обновить количество реплик deployment через подресурс 'scale' с помощью
merge patch
ac patch deployment nginx-deployment --subresource='scale' --type='merge'
-p '{"spec":{"replicas":2}}'
```

## ac plugin

Утилиты для взаимодействия с плагинами

### Пример использования

```
# Вывести список всех доступных плагинов
ac plugin list

# Вывести только имена бинарников доступных плагинов без путей
ac plugin list --name-only
```

## ac plugin list

Вывести все видимые исполняемые плагины в PATH пользователя

### Пример использования

```
# Вывести список всех доступных плагинов
ac plugin list

# Вывести только имена бинарников доступных плагинов без путей
ac plugin list --name-only
```

## ac port-forward

Переадресовать один или несколько локальных портов на pod

### Пример использования

```
# Прослушивать локально порты 5000 и 6000, переадресуя данные на порты 5000 и 6000 в pod
ac port-forward pod/mypod 5000 6000

# Прослушивать локально порты 5000 и 6000, переадресуя данные на порты 5000 и 6000 в pod, выбранном deployment
ac port-forward deployment/mydeployment 5000 6000

# Прослушивать локально порт 8443, переадресуя на targetPort сервиса с именем порта "https" в pod, выбранном сервисом
ac port-forward service/myservice 8443:https

# Прослушивать локально порт 8888, переадресуя на 5000 в pod
ac port-forward pod/mypod 8888:5000

# Прослушивать локально порт 8888 на всех адресах, переадресуя на 5000 в pod
ac port-forward --address 0.0.0.0 pod/mypod 8888:5000

# Прослушивать локально порт 8888 на localhost и выбранном IP, переадресуя на 5000 в pod
ac port-forward --address localhost,10.19.21.23 pod/mypod 8888:5000

# Прослушивать локально случайный порт, переадресуя на 5000 в pod
ac port-forward pod/mypod :5000
```

## ac process

Обработать шаблон в список ресурсов

## Пример использования

```
# Преобразовать файл template.json в список ресурсов и передать в create
ac process -f template.json | ac apply -f -

# Обработать файл локально без обращения к серверу
ac process -f template.json -o yaml

# Обработать шаблон с передачей пользовательской метки
ac process -f template.json -l name=mytemplate

# Преобразовать сохранённый шаблон в список ресурсов
ac process foo

# Преобразовать сохранённый шаблон в список ресурсов с установкой/переопределением значений параметров
ac process foo -p PARM1=VALUE1 -p PARM2=VALUE2

# Преобразовать шаблон, сохранённый в другом namespace, в список ресурсов
ac process cpaas-system//foo

# Преобразовать template.json в список ресурсов
cat template.json | ac process -f -
```

## ac proxy

Запустить прокси к Kubernetes API серверу

## Пример использования

```
# Проксировать весь Kubernetes API и ничего больше
ac proxy --api-prefix=/

# Проксировать только часть Kubernetes API и некоторые статические файлы
# Можно получить информацию о pods с помощью 'curl localhost:8001/api/v1/pods'
ac proxy --www=/my/files --www-prefix=/static/ --api-prefix=/api/

# Проксировать весь Kubernetes API с другим корнем
# Можно получить информацию о pods с помощью 'curl localhost:8001/custom/api/v1/pods'
ac proxy --api-prefix=/custom/

# Запустить прокси к Kubernetes API серверу на порту 8011, обслуживая статический контент из ./local/www/
ac proxy --port=8011 --www=./local/www/

# Запустить прокси к Kubernetes API серверу на произвольном локальном порту
# Выбранный порт будет выведен в stdout
ac proxy --port=0

# Запустить прокси к Kubernetes API серверу, изменив префикс API на k8s-api
# Это сделает API pods доступным по адресу localhost:8001/k8s-api/v1/pods/
ac proxy --api-prefix=/k8s-api
```

## ac replace

Заменить ресурс по имени файла или stdin

## Пример использования

```
# Заменить pod, используя данные из pod.json
ac replace -f ./pod.json

# Заменить pod, используя JSON, переданный через stdin
cat pod.json | ac replace -f -

# Обновить версию образа (тег) pod с одним контейнером на v4
ac get pod mypod -o yaml | sed 's/\(image: myimage\):.*$/\1:v4/' | ac replace -f -

# Принудительно заменить ресурс: удалить и затем создать заново
ac replace --force -f ./pod.json
```

## ac rollout

Управлять развертыванием ресурса

### Пример использования

```
# Откат к предыдущему deployment
ac rollout undo deployment/abc

# Проверить статус rollout daemonset
ac rollout status daemonset/foo

# Перезапустить deployment
ac rollout restart deployment/abc

# Перезапустить deployments с меткой 'app=nginx'
ac rollout restart deployment --selector=app=nginx
```

## ac rollout history

Просмотреть историю rollout

## Пример использования

```
# Просмотреть историю rollout deployment
ac rollout history deployment/abc

# Просмотреть детали ревизии 3 daemonset
ac rollout history daemonset/abc --revision=3
```

## ac rollout pause

Пометить указанный ресурс как приостановленный

## Пример использования

```
# Пометить deployment nginx как приостановленный
# Текущее состояние deployment продолжит работать; новые обновления не будут применяться, пока deployment приостановлен
ac rollout pause deployment/nginx
```

## ac rollout restart

Перезапустить ресурс

## Пример использования

```
# Перезапустить все deployments в namespace test-namespace
ac rollout restart deployment -n test-namespace

# Перезапустить deployment
ac rollout restart deployment/nginx

# Перезапустить daemon set
ac rollout restart daemonset/abc

# Перезапустить deployments с меткой app=nginx
ac rollout restart deployment --selector=app=nginx
```

## ac rollout resume

Возобновить приостановленный ресурс

### Пример использования

```
# Возобновить уже приостановленный deployment
ac rollout resume deployment/nginx
```

## ac rollout status

Показать статус rollout

### Пример использования

```
# Следить за статусом rollout deployment
ac rollout status deployment/nginx
```

## ac rollout undo

Отменить предыдущий rollout

## Пример использования

```
# Откат к предыдущему deployment
ac rollout undo deployment/abc

# Откат к ревизии 3 daemonset
ac rollout undo daemonset/abc --to-revision=3

# Откат к предыдущему deployment с dry-run
ac rollout undo --dry-run=server deployment/abc
```

## ac run

Запустить конкретный образ в кластере

## Пример использования

```
# Запустить pod nginx
ac run nginx --image=nginx

# Запустить pod hazelcast и открыть порт 5701 в контейнере
ac run hazelcast --image=hazelcast/hazelcast --port=5701

# Запустить pod hazelcast и установить переменные окружения "DNS_DOMAIN=c
luster" и "POD_NAMESPACE=default" в контейнере
ac run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMAIN=cluster" -
-env="POD_NAMESPACE=default"

# Запустить pod hazelcast и установить метки "app=hazelcast" и "env=prod"
в контейнере
ac run hazelcast --image=hazelcast/hazelcast --labels="app=hazelcast,env=
prod"

# Dry run; вывести соответствующие API объекты без создания
ac run nginx --image=nginx --dry-run=client

# Запустить pod nginx, но перегрузить spec частичным набором значений в ф
ормате JSON
ac run nginx --image=nginx --overrides='{ "apiVersion": "v1", "spec": {
... } }'
```

```
# Запустить pod busybox и держать его на переднем плане, не перезапускать
при выходе
ac run -i -t busybox --image=busybox --restart=Never

# Запустить pod nginx с командой по умолчанию, но с пользовательскими арг
ументами (arg1 .. argN)
ac run nginx --image=nginx -- <arg1> <arg2> ... <argN>

# Запустить pod nginx с другой командой и пользовательскими аргументами
ac run nginx --image=nginx --command -- <cmd> <arg1> ... <argN>
```

## ac scale

Установить новый размер для deployment, replica set или replication controller

## Пример использования

```
# Масштабировать replica set с именем 'foo' до 3
ac scale --replicas=3 rs/foo

# Масштабировать ресурс, указанный по типу и имени в "foo.yaml" до 3
ac scale --replicas=3 -f foo.yaml

# Если текущий размер deployment mysql равен 2, масштабировать mysql до 3
ac scale --current-replicas=2 --replicas=3 deployment/mysql

# Масштабировать несколько replication controllers
ac scale --replicas=5 rc/example1 rc/example2 rc/example3

# Масштабировать stateful set с именем 'web' до 3
ac scale --replicas=3 statefulset/web
```

## ac set

Установить конкретные параметры на объектах

## ac set env

Обновить переменные окружения в шаблоне pod

## Пример использования

```
# Обновить deployment 'registry' с новой переменной окружения
ac set env deployment/registry STORAGE_DIR=/local

# Вывести список переменных окружения, определённых в deployment 'sample-
build'
ac set env deployment/sample-build --list

# Вывести список переменных окружения, определённых во всех pods
ac set env pods --all --list

# Вывести изменённый deployment в формате YAML, не изменяя объект на серв
ере
ac set env deployment/sample-build STORAGE_DIR=/data -o yaml

# Обновить все контейнеры во всех replication controllers в проекте, уста
новив ENV=prod
ac set env rc --all ENV=prod

# Импортировать переменные окружения из secret
ac set env --from=secret/mysecret deployment/myapp

# Импортировать переменные окружения из config map с префиксом
ac set env --from=configmap/myconfigmap --prefix=MYSQL_ deployment/myapp

# Импортировать конкретные ключи из config map
ac set env --keys=my-example-key --from=configmap/myconfigmap deployment/
myapp

# Удалить переменную окружения ENV из контейнера 'c1' во всех deployment
configs
ac set env deployments --all --containers="c1" ENV-

# Удалить переменную окружения ENV из deployment на диске и обновить depl
oyment на сервере
ac set env -f deploy.json ENV-

# Установить часть локальных переменных окружения shell в deployment на с
ервере
env | grep RAILS_ | ac set env -e - deployment/registry
```

## ac set image

Обновить образ в шаблоне pod

### Пример использования

```
# Установить образ контейнера nginx в deployment на 'nginx:1.9.1', а образ
# контейнера busybox на 'busybox'
ac set image deployment/nginx busybox=busybox nginx=nginx:1.9.1

# Обновить образ контейнера nginx во всех deployments и rc на 'nginx:1.9.
# 1'
ac set image deployments,rc nginx=nginx:1.9.1 --all

# Обновить образ всех контейнеров daemonset abc на 'nginx:1.9.1'
ac set image daemonset abc *=nginx:1.9.1

# Вывести результат (в формате yaml) обновления образа контейнера nginx и
# з локального файла без обращения к серверу
ac set image -f path/to/file.yaml nginx=nginx:1.9.1 --local -o yaml
```

## ac set resources

Обновить запросы/лимиты ресурсов на объектах с шаблонами pod

### Пример использования

```

# Установить лимиты cpu до "200m" и памяти до "512Mi" для контейнера nginx
x в deployment
ac set resources deployment nginx -c=nginx --limits=cpu=200m,memory=512Mi

# Установить запросы и лимиты ресурсов для всех контейнеров в nginx
ac set resources deployment nginx --limits=cpu=200m,memory=512Mi --requests=cpu=100m,memory=256Mi

# Удалить запросы ресурсов для контейнеров в nginx
ac set resources deployment nginx --limits=cpu=0,memory=0 --requests=cpu=0,memory=0

# Вывести результат (в формате yaml) обновления лимитов контейнера nginx
из локального файла без обращения к серверу
ac set resources -f path/to/file.yaml --limits=cpu=200m,memory=512Mi --local -o yaml

```

## ac set selector

Установить селектор на ресурсе

### Пример использования

```

# Установить метки и селектор перед созданием пары deployment/service
ac create service clusterip my-svc --clusterip="None" -o yaml --dry-run=client | 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

Обновить service account ресурса

## Пример использования

```
# Установить service account deployment nginx-deployment в serviceaccount
1
ac set serviceaccount deployment nginx-deployment serviceaccount1

# Вывести результат (в формате YAML) обновления nginx deployment с servic
e account из локального файла без обращения к API серверу
ac set sa -f nginx-deployment.yaml serviceaccount1 --local --dry-run=clie
nt -o yaml
```

## ac set subject

Обновить пользователя, группу или service account в role binding или cluster role binding

## Пример использования

```
# Обновить cluster role binding для serviceaccount1
ac set subject clusterrolebinding admin --serviceaccount=namespace:servic
eaccount1

# Обновить role binding для user1, user2 и group1
ac set subject rolebinding admin --user=user1 --user=user2 --group=group1

# Вывести результат (в формате YAML) обновления subjects rolebinding из л
окального файла без обращения к серверу
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

Вывести информацию о версиях клиента и сервера

## Пример использования

```
# Вывести информацию о версиях клиента и сервера
```

```
ac version
```

```
# Вывести только версию клиента
```

```
ac version --client
```

```
# Вывести версию в формате JSON
```

```
ac version -o json
```

## ac wait

Экспериментально: ждать определённого условия на одном или нескольких ресурсах

## Пример использования

```
# Ждать, пока pod "busybox1" не будет содержать статус condition типа "Ready"
```

```
ac wait --for=condition=Ready pod/busybox1
```

```
# Значение статуса condition по умолчанию true; можно ждать другие значения после знака равенства (сравнение с учётом Unicode case folding)
```

```
ac wait --for=condition=Ready=false pod/busybox1
```

```
# Ждать, пока pod "busybox1" не будет содержать статус фазы "Running"
```

```
ac wait --for=jsonpath='{.status.phase}'=Running pod/busybox1
```

```
# Ждать, пока pod "busybox1" не будет Ready
```

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

```
# Ждать, пока сервис "loadbalancer" не получит ingress
```

```
ac wait --for=jsonpath='{.status.loadBalancer.ingress}' service/loadbalancer
```

```
# Ждать создания секрета "busybox1" с таймаутом 30 секунд
```

```
ac create secret generic busybox1
```

```
ac wait --for=create secret/busybox1 --timeout=30s
```

```
# Ждать удаления pod "busybox1" с таймаутом 60 секунд после выполнения команды удаления
```

```
ac delete pod/busybox1
```

```
ac wait --for=delete pod/busybox1 --timeout=60s
```

# AC CLI Справочник команд администратора

Этот справочник содержит описания и примеры команд администратора AC CLI. Для использования этих команд необходимо иметь права `cluster-admin` или эквивалентные.

Для команд разработчика смотрите справочник команд разработчика AC CLI.

Выполните `ac adm -h`, чтобы вывести список всех команд администратора, или `ac <command> --help` для получения дополнительной информации по конкретной команде.

## Содержание

### ac adm

Пример использования

`ac adm certificate`

`ac adm certificate approve`

Пример использования

`ac adm certificate deny`

Пример использования

`ac adm cordon`

Пример использования

`ac adm drain`

Пример использования

`ac adm new-project`

Пример использования

`ac adm new-project-namespace`

Пример использования

```
ac adm policy
```

Пример использования

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

Пример использования

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

Пример использования

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

Пример использования

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

Пример использования

```
ac adm taint
```

Пример использования

```
ac adm uncordon
```

Пример использования

---

## ac adm

Административные инструменты ACP для управления кластером

### Пример использования

```
# Подготовить узел к техническому обслуживанию (drain)
ac adm drain NODE_NAME

# Заблокировать узел (отметить как недоступный для планирования)
ac adm cordon NODE_NAME

# new-project для создания проекта с кластером
ac adm new-project PROJECT_NAME --cluster CLUSTER_NAME

# Разблокировать узел (отметить как доступный для планирования)
ac adm uncordon NODE_NAME
```

## ac adm certificate

Изменение ресурсов сертификатов

## ac adm certificate approve

Одобрить запрос на подпись сертификата

### Пример использования

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

## ac adm certificate deny

Отклонить запрос на подпись сертификата

### Пример использования

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

## ac adm cordon

Отметить узел как недоступный для планирования

### Пример использования

```
# Отметить узел "foo" как недоступный для планирования  
ac adm cordon foo
```

## ac adm drain

Освободить узел в подготовке к техническому обслуживанию

### Пример использования

```
# Освободить узел "foo", даже если на нем есть поды, не управляемые repli  
cation controller, replica set, job, daemon set или stateful set  
ac adm drain foo --force  
  
# То же, но прервать, если есть поды, не управляемые replication controll  
er, replica set, job, daemon set или stateful set, и использовать период  
ожидания 15 минут  
ac adm drain foo --grace-period=900
```

## ac adm new-project

Создать новый проект

## Пример использования

```
# Создать проект с указанным кластером
ac adm new-project my-project --cluster cluster1

# Создать проект с несколькими кластерами
ac adm new-project my-project --cluster cluster1,cluster2
```

## ac adm new-project-namespace

Создать новое пространство имён в проекте

## Пример использования

```
# Создать пространство имён в проекте с указанным кластером
ac adm new-project-namespace my-namespace --project my-project --cluster
cluster1
```

## ac adm policy

Управление политиками RBAC для проекта или пространства имён

## Пример использования

```
# Назначить пользователю роль администратора проекта
ac adm policy add-project-role-to-user project-admin-system alice --project my-project

# Назначить пользователю роль в пространстве имён кластера в проекте
ac adm policy add-namespace-role-to-user namespace-developer-system alice --namespace my-namespace --project my-project --cluster business-1

# Добавить роль кластера Kubernetes view пользователю alice
ac adm policy add-cluster-role-to-user view alice

# Добавить роль Kubernetes view пользователю alice
ac adm policy add-role-to-user view alice -n my-namespace
```

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

Назначить роль кластера Kubernetes пользователю в текущем контексте кластера

### Пример использования

```
# Добавить роль кластера Kubernetes view пользователю alice
ac adm policy add-cluster-role-to-user view alice
```

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

Назначить платформенную роль пользователю в определённом пространстве имён кластера в проекте

### Пример использования

```
# Назначить роль namespace-developer-system пользователю alice в проекте my-project
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

Назначить платформенную роль пользователю в проекте

### Пример использования

```
# Назначить роль project-admin-system пользователю alice в проекте my-project
ac adm policy add-project-role-to-user project-admin-system alice --project my-project
```

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

Назначить роль Kubernetes пользователю в текущем контексте кластера

### Пример использования

```
# Добавить роль Kubernetes view пользователю alice
ac adm policy add-role-to-user view alice -n my-namespace
```

## ac adm taint

Обновить taint'ы на одном или нескольких узлах

### Пример использования

```
# Обновить узел 'foo' с taint с ключом 'dedicated', значением 'special-user' и эффектом 'NoSchedule'  
# Если taint с таким ключом и эффектом уже существует, его значение будет заменено  
ac adm taint nodes foo dedicated=special-user:NoSchedule  
  
# Удалить с узла 'foo' taint с ключом 'dedicated' и эффектом 'NoSchedule', если он существует  
ac adm taint nodes foo dedicated:NoSchedule-  
  
# Удалить с узла 'foo' все taint'ы с ключом 'dedicated'  
ac adm taint nodes foo dedicated-  
  
# Добавить taint с ключом 'dedicated' на узлах с меткой myLabel=X  
ac adm taint node -l myLabel=X dedicated=foo:PreferNoSchedule  
  
# Добавить на узел 'foo' taint с ключом 'bar' без значения  
ac adm taint nodes foo bar:NoSchedule
```

## ac adm uncordon

Отметить узел как доступный для планирования

### Пример использования

```
# Отметить узел "foo" как доступный для планирования  
ac adm uncordon foo
```