# Cloud Container Instance (CCI 2.0)

# Developer Guide

**Issue** 01

**Date** 2025-09-29

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

### Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

### Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

## Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
Qianzhong Avenue
Gui'an New District
Gui Zhou 550029
People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

# Contents

**1 Using ccictl ..... 1**
1.1 Constraints ..... 1
1.2 ccictl Configuration Guide ..... 2
1.3 Command Line Tool (ccictl) ..... 6
1.3.1 ccictl Overview ..... 6
1.3.2 ccictl ..... 16
1.3.3 ccictl create ..... 17
1.3.3.1 ccictl create namespace ..... 18
1.3.3.2 ccictl create secret docker-registry ..... 19
1.3.3.3 ccictl create secret generic ..... 21
1.3.3.4 ccictl create secret tls ..... 22
1.3.3.5 ccictl create configmap ..... 23
1.3.3.6 ccictl create service loadbalancer ..... 24
1.3.3.7 ccictl create service externalname ..... 25
1.3.3.8 ccictl create deployment ..... 26
1.3.4 ccictl set ..... 28
1.3.4.1 ccictl set env ..... 28
1.3.4.2 ccictl set image ..... 30
1.3.4.3 ccictl set resources ..... 32
1.3.4.4 ccictl set selector ..... 33
1.3.5 ccictl explain ..... 34
1.3.6 ccictl get ..... 35
1.3.7 ccictl edit ..... 38
1.3.8 ccictl delete ..... 40
1.3.9 ccictl rollout ..... 42
1.3.9.1 ccictl rollout history ..... 43
1.3.9.2 ccictl rollout restart ..... 44
1.3.9.3 ccictl rollout status ..... 45
1.3.9.4 ccictl rollout undo ..... 46
1.3.10 ccictl describe ..... 47
1.3.11 ccictl logs ..... 48
1.3.12 ccictl exec ..... 50
1.3.13 ccictl apply ..... 51

  

1.3.13.1 ccictl apply edit-last-applied....53
1.3.13.2 ccictl apply set-last-applied....54
1.3.13.3 ccictl apply view-last-applied....55
1.3.14 ccictl api-resources....56
1.3.15 ccictl api-versions....57
1.3.16 ccictl config....57
1.3.16.1 ccictl config current-context....58
1.3.16.2 ccictl config get-contexts....58
1.3.16.3 ccictl config set-context....59
1.3.16.4 ccictl config delete-context....60
1.3.16.5 ccictl config rename-context....60
1.3.16.6 ccictl config use-context....60
1.3.16.7 ccictl config get-clusters....61
1.3.16.8 ccictl config set-cluster....61
1.3.16.9 ccictl config delete-cluster....62
1.3.16.10 ccictl config get-users....63
1.3.16.11 ccictl config delete-user....63
1.3.16.12 ccictl config set-credentials....63
1.3.16.13 ccictl config view....65
1.3.17 ccictl version....66
**2 Using Terraform to Manage CCI Resources....67**
2.1 Terraform Configuration Guide....67
2.2 CCI Resources That Can Be Managed by Terraform....69

  

# 1 Using ccictl

## 1.1 Constraints

- CCI resources cannot be operated using ccictl in IAM 5.0 (Landing Zone).
- Currently, ccictl supports only the following commands and resources:

  Supported commands: create, set, explain, get, edit, delete, rollout, describe, logs, exec, apply, api-resources, api-versions, config, version, and options

  The supported resources and commands are as follows:

| Command | Namespace | Network | Deployment | Pod | Service | ConfigMap | Secret | HPA | PVC | PV |
|---|---|---|---|---|---|---|---|---|---|---|
| create | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| set | × | × | √ | √ | √ | × | × | × | × | × |
| explain | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| get | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| edit | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| delete | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| rollout | × | × | √ | × | × | × | × | × | × | × |
| describe | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| logs | × | × | × | √ | × | × | × | × | × | × |
| exec | × | × | × | √ | × | × | × | × | × | × |
| apply | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |

  

- If **--namespace** is not specified, ccictl will use the **default** namespace by default. If you use CCI for the first time, the **default** namespace will not be created automatically.

# 1.2 ccictl Configuration Guide

CCI allows you to use ccictl to create resources such as workloads.

## Downloading ccictl

Download ccictl of the required version from **Table 1-1**.

**Table 1-1** ccictl download addresses

| OS | Download Addresses |
| --- | --- |
| Linux AMD 64-bit | **ccictl_linux-amd64**<br>**ccictl_linux-amd64_sha256** |
| MAC AMD 64-bit | **ccictl_mac-amd64**<br>**ccictl_mac-amd64_sha256** |
| MAC Arm 64-bit | **ccictl_mac-arm64**<br>**ccictl_mac-arm64_sha256** |

## Installing and Configuring ccictl

A Linux OS is used as an example in the following operations.

**Step 1** Grant the execute permission on ccictl downloaded in **Downloading ccictl** and save it to the PATH directory.

```
#chmod +x ./ccictl
#mv ./ccictl $PATH
```

In this command, *$PATH* indicates the PATH directory (for example, **/usr/local/bin**). Replace it with the actual value.

You can run the following command to view the ccictl version:

```
#ccictl version
#Client Version: version.Info{GitVersion:"v0.0.2",
GitCommit:"996a2efcd852473dde345ae2931833342cab1d96", BuildDate:"2025-03-24T00:32:05Z",
GoVersion:"go1.19.6", Compiler:"gc", Platform:"linux/amd64"}
```

**Step 2** Configure IAM authentication information and persistently store it to the local host.

1. Specify a CCI cluster.
   ```
   ccictl config set-cluster <$context-cluster-name> --server=https://<$cci-endpoint>
   ```
   - **$context-cluster-name**: cluster name. You can use any custom name.
   - **$cci-endpoint**: CCI endpoint. Endpoints vary depending on regions. For details, see **Obtaining CCI Endpoints**.

   Example:

  

```
#ccictl config set-cluster cci-cluster --server=https://cci.cn-north-4.myhuaweicloud.com
#Cluster "cci-cluster" set.
```

2. Set the user credential used by ccictl.

```
ccictl config set-credentials <$user> --auth-provider=iam --auth-provider-arg=iam-endpoint=https://<$iam-endpoint> --auth-provider-arg=cache="true" --auth-provider-arg=project-name=<$project-name> --auth-provider-arg=user-name=<$user-name> --auth-provider-arg=domain-name=<$domain-name> --auth-provider-arg=password=<$password>
```

Or

```
ccictl config set-credentials <$user> --auth-provider=iam --auth-provider-arg=iam-endpoint=https://<$iam-endpoint> --auth-provider-arg=cache="true" --auth-provider-arg=project-name=<$project-name> --auth-provider-arg=ak=<$ak> --auth-provider-arg=sk=<$sk>
```

**Table 1-2** Username and password

| Command Flag | Description |
| --- | --- |
| domain-name | Tenant name, which is the account name. |
| user-name | IAM username. |
| password | Password of the account or IAM user. |

**Table 1-3** AK/SK

| Command Flag | Description |
| --- | --- |
| ak | For details about how to obtain the AK and SK, see **Obtaining an AK/SK**. AK is the access key in the file, and SK is the secret key in the file. |
| sk | |

**Table 1-4** Common settings

| Command Flag | Description |
| --- | --- |
| context-user-name | Username, which is the configured context username and can be specified. |
| iam-endpoint | IAM endpoint. This parameter is mandatory. For details, see **Regions and Endpoints**. |
| project-name | Project name. For details, see **Obtaining a Project Name on the Console**. |

  

| Command Flag | Description |
| --- | --- |
| cache | Whether to cache the IAM token to the local host to improve the access performance. The default value is **true**.<br>**CAUTION**<br>In an insecure environment, you are advised to disable this option. |

Example:

```
#ccictl config set-credentials cci-user --auth-provider=iam --auth-provider-arg=iam-endpoint=https://{iam-endpoint} --auth-provider-arg=cache="true" --auth-provider-arg=project-name={region} --auth-provider-arg=ak=ak****** --auth-provider-arg=sk=sk******
#User "cci-user" set.
```

3. Set and use the context of ccictl.

```
ccictl config set-context <$context-name> --cluster=<$context-cluster-name> --user=<$context-user-name>
```

- **$context-name**: context name. You can use any custom name.
- **$context-cluster-name**: cluster name configured in the preceding steps
- **$context-user-name**: username configured in the preceding steps

Example:

```
#ccictl config set-context cci-context1 --cluster=cci-cluster --user=cci-user
#Context "cci-context1" created.
```

Context of using ccictl

```
ccictl config use-context  <$context-name>
```

Example:

```
#ccictl config use-context cci-context1
#Switched to context "cci-context1".
```

**Step 3** After the configuration is complete, you can run ccictl commands to perform operations on CCI resources.

For example, run the **ccictl get namespace** command to view namespace resources.

```
#ccictl get namespace
#No resources found.
```

**----End**

## Configuring ccictl in an Insecure Environment

**Step 1** **Install and configure ccictl**.

**Step 2** Edit the config file and delete sensitive information.

In a Linux OS, the config file is stored in **$HOME/.kubev2/config** by default.

  

**Table 1-5** Sensitive information

| Command Flag | Environment Value | Description |
| --- | --- | --- |
| --domain-name | DOMAIN_NAME | Tenant name |
| --user-name | USER_NAME | Sub-user name |
| --password | PASSWORD | User password |
| --ak | ACCESS_KEY_ID | Access key ID |
| --sk | SECRET_ACCESS_KEY | Secret access key |
| --cache | CREDENTIAL_CACHE | Whether to cache tokens |

**Step 3** Configure the environment variables corresponding to the deleted parameters. For example, configure AK, SK, and whether to cache tokens.

```
#export ACCESS_KEY_ID=xxxxxxx
#export SECRET_ACCESS_KEY=xxxxxxx
#export CREDENTIAL_CACHE=false
#ccictl get ns
```

After the preceding commands are executed, information similar to the following is displayed:

```
#No resources found.
```

**----End**

## Obtaining an AK/SK

AK: access key ID. It is a unique ID associated with an SK. AK is used together with SK to sign requests.

SK: secret access key. It is used together with an AK to sign requests. They can identify request senders and prevent requests from being modified.

**Step 1** Log in to the management console.

**Step 2** Hover over the username and choose **My Credentials** from the drop-down list.

**Step 3** Choose **Access Keys** from the navigation pane.

**Step 4** Click **Create Access Key**, and enter the verification code.

**Step 5** Click **OK** to generate an access key and download it.

NOTE

Keep the AK/SK file confidential to prevent information leakage.

**----End**

## Obtaining CCI Endpoints

An endpoint is the **request address** for calling an API. Endpoints vary with services and regions. An endpoint can be obtained from **Regions and Endpoints**.

Select an endpoint based on your service requirements.

  

**Table 1-6** Regions and endpoints

| Region Name | Region | Endpoint |
| --- | --- | --- |
| TR-Istanbul | tr-west-1 | cci.tr-west-1.myhuaweicloud.com |
| AF-Johannesburg | af-south-1 | cci.af-south-1.myhuaweicloud.com |
| AP-Singapore | ap-southeast-3 | cci.ap-southeast-3.myhuaweicloud.com |
| ME-Riyadh | me-east-1 | cci.me-east-1.myhuaweicloud.com |
| LA-Mexico City2 | la-north-2 | cci.la-north-2.myhuaweicloud.com |
| LA-Sao Paulo1 | sa-brazil-1 | cci.sa-brazil-1.myhuaweicloud.com |
| LA-Santiago | la-south-2 | cci.la-south-2.myhuaweicloud.com |

### Obtaining a Project Name on the Console

To obtain a project name on the console, take the following steps:

1. Log in to the management console.
2. Hover the pointer over the username in the upper right corner and choose **My Credentials** from the drop-down list.

   On the **API Credentials** page, view projects in the project list.

# 1.3 Command Line Tool (ccictl)

## 1.3.1 ccictl Overview

### Introduction to ccictl

ccictl is a CLI tool similar to kubectl. It uses APIs to communicate with CCI 2.0. It can manage various tasks of CCI 2.0 clusters.

ccictl looks for a configuration file named **config** in the **$HOME/.kubev2** directory. You can create other config files as needed by setting the **CLICONFIG** environment variable or the **--cliconfig** parameter.

This topic describes how to use ccictl for declarative application management in CCI 2.0. It also covers some other ccictl functions.

  

**CAUTION**

The group and version of all resources provided by CCI 2.0 are cci and v2, the same for the resources created, queried, and deleted using ccictl. If the resource files where the group and version are used as input parameters and **group/version** is set to **v1** or **apps/v1**, the commands fail to be executed.

## Command Category

Most ccictl commands can be classified into the types in the following table.

**Table 1-7** ccictl command category

| Type | Purpose | Description |
| --- | --- | --- |
| Declarative resource management | Deployment and O&M (such as GitOps) | The commands are used to manage workloads in CCI 2.0. |
| Command-based resource management | Debugging | Command-line parameters and flags are used to manage workloads in CCI 2.0. |
| Workload status printing | Debugging | The commands are used to print information about workloads. |
| Interaction with containers | Debugging | The commands are used for execution, mounting, replication, and logging. |

## Declarative Application Management

A preferred way to manage resources is to use a declarative file named **resource** together with the **ccictl apply** command. This command reads the local (or remote) file structure and modifies the cluster status to reflect the declared intent.

**NOTE**

**Apply**

Apply is preferred for managing resources in CCI 2.0.

## Workload Status Printing

ccictl provides the following commands for debugging:

- Print container logs.
- Print events.
- Execute to a container.

  

## Syntax

Run the ccictl command on the terminal using the following syntax:

```
ccictl [command] [TYPE] [NAME] [flags]
```

**command**, **TYPE**, **NAME**, and **flags** are described as follows:

- **command**: specifies the operation to be performed on one or more resources, for example, create, get, describe, and delete.
- **TYPE**: specifies the resource type. The resource type is case insensitive. You can use the singular or plural form, or an abbreviation. For example, the output of the following commands is the same:

```
ccictl get pod pod1
ccictl get pods pod1
ccictl get po pod1
```

- **NAME**: specifies the name of a resource. The name is case sensitive. If the name is omitted, the details of all resources are displayed, for example, **ccictl get pods**.

  When performing operations on multiple resources, you can specify each resource by type and name, or use one or more files.
- To specify resources by type and name:
- To group all resources of the same type: TYPE1 name1 name2 name<#>

  Example: ccictl get pod example-pod1 example-pod2
- To specify multiple resource types: TYPE1/name1 TYPE1/name2 TYPE2/name3 TYPE<#>/name<#>

  Example: ccictl get pod/example-pod1 deployment/example-deploy1
- Use one or more files to specify resources: -f file1 -f file2 -f file<#&gt
- You are advised to use YAML instead of JSON because YAML is more user-friendly, especially for configuration files.

  Example: ccictl get -f ./pod.yaml
- **flags**: specifies optional parameters. For example, you can use the **-s** or **--server** parameter to specify the address and port of the API server.

**CAUTION**

Parameters specified from the command line overwrite the default values and any corresponding environment variables.

If you need help, run the **ccictl help** command in the terminal.

## Operations

The following table contains a short description and common syntax of all ccictl operations:

  

**Table 1-8** Descriptions and syntax

| Operation | Syntax | Description |
| --- | --- | --- |
| api-resources | ccictl api-resources [flags] | Lists available API resources. |
| api-versions | ccictl api-versions [flags] | Lists available API versions. |
| apply | ccictl apply -f FILENAME [flags] | Applies configuration changes to a resource from a file or stdin. |
| config | ccictl config SUBCOMMAND [flags] | Modifies the **config** file. For details, see the subcommand descriptions. |
| create | ccictl create -f FILENAME [flags] | Creates one or more resources from a file or stdin. |
| delete | ccictl delete (-f FILENAME \| TYPE [NAME \| /NAME \| -l label \| --all]) [flags] | Deletes a resource based on a file or stdin, or by specifying a label selector, name, resource selector, or resource itself. |
| describe | ccictl describe (-f FILENAME \| TYPE [NAME_PREFIX \| /NAME \| -l label]) [flags] | Displays the status of one or more resources. |
| edit | ccictl edit (-f FILENAME \| TYPE NAME \| TYPE/ NAME) [flags] | Uses the default editor to edit and update the definitions of one or more resources on the server. |
| exec | ccictl exec POD [-c CONTAINER] [-i] [-t] [flags] [-- COMMAND [args...]] | Runs commands in the containers of a pod. |
| explain | ccictl explain TYPE [--recursive=false] [flags] | Obtains documents of multiple resources, such as pods, Deployments, and Services. |

  

| Operation | Syntax | Description |
| --- | --- | --- |
| get | cctl get (-f FILENAME \| TYPE [NAME \| /NAME \| -l label]) [--watch] [--sort-by=FIELD] [[-o \| --output]=OUTPUT_FORMAT] [flags] | Lists one or more resources. |
| logs | cctl logs POD [-c CONTAINER] [--follow] [flags] | Prints logs of containers in a pod. |
| rollout | cctl rollout SUBCOMMAND [options] | Manages resource rollouts. A valid resource type is Deployment. |
| set | cctl set SUBCOMMAND [options] | Configures application resources. |
| version | cctl version [flags] | Displays the version of cctl on the client. |

## Resource Types

**Table 1-9** Resource types

| Resource Name | Abbreviation | API Version | By Namespace | Resource Type |
| --- | --- | --- | --- | --- |
| configmaps | cm | cci/v2 | true | ConfigMap |
| deployments | deploy | cci/v2 | true | Deployment |
| horizontalpodautoscalers | hpa | cci/v2 | true | HorizontalPodAutoscaler |
| namespaces | ns | cci/v2 | false | Namespace |
| persistentvolumeclaims | PVC | cci/v2 | true | PersistentVolumeClaim |
| persistentvolumes | PV | cci/v2 | false | PersistentVolume |
| pods | po | cci/v2 | true | Pod |
| replicasets | rs | cci/v2 | true | ReplicaSet |
| secrets | - | cci/v2 | true | Secret |
| services | svc | cci/v2 | true | Service |
| StorageClass | - | cci/v2 | false | StorageClass |

  

| Resource Name | Abbreviation | API Version | By Namespace | Resource Type |
| --- | --- | --- | --- | --- |
| networks | - | yangtse/v2 | true | Network |

**CAUTION**

**Resource Types** lists only resource types supported by CCI 2.0. Not all resource objects in the table support all GET, POST, PUT, DELETE, PATCH APIs. For example, ReplicaSets supports only namespace-level query APIs. The APIs at different sites are also different.

If you encounter any dispute about APIs or resource types, contact the CCI service contact. CCI reserves the right to interpret the information.

## Output Options

### Output format

The default output format of all ccictl commands is human-readable plain text. To output detailed information in a specific format in the terminal, you can add the **-o** or **--output** parameter to the ccictl command.

### Syntax

```
ccictl [command] [TYPE] [NAME] -o <output_format>
```

The following are example outputs.

**Table 1-10**

| Output Format | Description |
| --- | --- |
| -o custom-columns=<spec> | A list of comma-separated **custom columns** |
| -o custom-columns-file=<filename> | A list of **custom columns** based on the template in the <filename> file. |
| -o json | API objects in JSON format. |
| -o jsonpath=<template> | Fields defined by the **JSONPath** expression. |
| -o jsonpath-file=<filename> | Fields defined by the **JSONPath** expression in the <filename> file. |
| -o name | Resource name |
| -o wide | The output is in plain text format and contains all additional information. |
| -o yaml | API objects in YAML format |

  

**Example**

In this example, the command outputs the details of a single pod as an object in YAML format.

```
ccictl get pod web-pod-13je7 -o yaml
```

## Custom Columns

To define custom columns and output only the required details to a list, you can use the **custom-columns** option. You can define an inline (**-o custom-columns=<spec>**) or use a template file (**-o custom-columns-file=<filename>**).

**Example**

Inline:

```
ccictl get pods <pod-name> -o custom-columns=NAME:.metadata.name,RSRC:.metadata.resourceVersion
```

Template file:

```
ccictl get pods <pod-name> -o custom-columns-file=template.txt
```

The **template.txt** file contains the following information:

```
NAME         RSRC
metadata.name metadata.resourceVersion
```

The output of either of the two commands is similar to the following:

```
NAME         RSRC
example-name   610995
```

## Sorting List Objects

To sort objects and output them in the terminal, you can add the **--sort-by** parameter to the supported ccictl command. This parameter allows you to sort objects by specifying any numeric or string field. To specify a field, use the **JSONPath** expression.

**Syntax**

```
ccictl [command] [TYPE] [NAME] --sort-by=<jsonpath_exp>
```

**Example**

To print a list of pods sorted by name, run the following command:

```
ccictl get pods --sort-by=.metadata.name
```

## Common Operation Examples

Use the following examples to help you get familiar with common ccictl operations.

**ccictl apply -** Apply or update resources based on a file or stdin.

```
# Use the definition in example-service.yaml to create a Service.
ccictl apply -f example-service.yaml

# Use the definition in example-deployment.yaml to create a Deployment.
ccictl apply -f example-deployment.yaml
```

  

```
# Use any .yaml, .yml, or .json file in <directory> to create an object.
ccictl apply -f <directory>
```

**ccictl get -** List one or more resources.

```
# List all pods in plain text format.
ccictl get pods

# List all pods in plain text format, including additional information.
ccictl get pods -o wide

# List all pods and Services in plain text format.
ccictl get po,services

# List all pods in the Pending state.
ccictl get pods --field-selector=status.phase=Pending
```

**ccictl describe -** Display the status of one or more resources, including uninitialized resources.

```
# Display the details about the Service named <service-name>.
ccictl describe svc <service-name>

# Display the details about the pod named <pod-name>.
ccictl describe pods/<pod-name>

# Display the details about all pods for the workload named <deploy-name>.
ccictl describe pods <deploy-name>

# Describe all pods.
ccictl describe pods
```

NOTE

The **ccictl get** command is usually used to retrieve one or more resources of the same resource class. You can specify the output format using the **-o** or **--output** parameter. You can also specify the **-w** or **--watch** parameter to start monitoring updates for a specific object. The **ccictl describe** command focuses more on describing many relevant aspects of a specified resource. It can invoke multiple API calls to the API server to build views for users. For example, the **ccictl describe pod** command retrieves information about the pod and the events generated for the pod.

**ccictl delete -** Delete a resource based on a file or stdin, or by specifying a label selector, name, resource selector, or resource itself.

```
# Use the type and name specified in the pod.yaml file to delete a pod.
ccictl delete -f pod.yaml

# Delete all pods and Services with the <label-key>=<label-value> label.
ccictl delete pods,services -l <label-key>=<label-value>

# Delete all pods, including uninitialized pods.
ccictl delete pods --all
```

**ccictl exec -** Run commands in the containers of a pod.

```
# Obtain the output by running 'date' from pod <pod-name>. By default, the output comes from the first container.
ccictl exec <pod-name> -- date

# Obtain the output by running 'date' from container <container-name> in pod <pod-name>.
ccictl exec <pod-name> -c <container-name> -- date

# Obtain an interactive TTY and run /bin/bash in pod <pod-name>. By default, the output comes from the first container.
ccictl exec -ti <pod-name> -- /bin/bash
```

**ccictl logs -** Print logs of containers in a pod.

  

```
# Return the log snapshot of pod <pod-name>.
ccictl logs <pod-name>

# Start to transfer logs in streaming mode from pod <pod-name>. This is similar to the tail -f Linux command.
ccictl logs -f <pod-name>
```

## JSONPath Support

ccictl supports the JSONPath template as the output format.

A JSONPath template consists of a JSONPath expression enclosed by braces ({}). ccictl uses the JSONPath expression to filter specific fields in a JSON object and format the output. In addition to the original JSONPath template syntax, the following functions and syntax are also valid:

- Use double quotation marks to enclose the text in the JSONPath expression.
- Use the range and end operators to iterate the list.
- Back up a list using a negative index. A negative index does not "surround" the list and is valid as long as the \( ( - index + listLength ) \ge 0 \) is valid.

NOTE

- The $ operator is optional because the expression always starts from the root object by default.
- The result object is output as its String() function.

### Functions in JSONPath

Given JSON input:

```
{
  "kind": "List",
  "items":[
    {
      "kind":"None",
      "metadata":{
        "name":"127.0.0.1",
        "labels":{
          "kubernetes.io/hostname":"127.0.0.1"
        }
      },
      "status":{
        "capacity":{"cpu":"4"},
        "addresses":[{"type": "LegacyHostIP", "address":"127.0.0.1"}]
      }
    },
    {
      "kind":"None",
      "metadata":{"name":"127.0.0.2"},
      "status":{
        "capacity":{"cpu":"8"},
        "addresses":[
          {"type": "LegacyHostIP", "address":"127.0.0.2"},
          {"type": "another", "address":"127.0.0.3"}
        ]
      }
    }
  ],
  "users":[
    {
      "name": "myself",
      "user": {}
    },
    {
```

```
    "name": "e2e",
    "user": {"username": "admin", "password": "secret"}
  }
 ]
}
```

**Table 1-11** Function description

| Function | Description | Example | Result |
| --- | --- | --- | --- |
| text | Text only | kind is {.kind} | kind is List |
| @ | Current object | {@} | Same as the input |
| . or [] | Operator | {.kind}, {['kind']}, or {['name \.type']} | List |
| .. | Recursive descent | {..name} | 127.0.0.1 127.0.0.2 myself e2e |
| * | Wildcard mask. All objects are obtained. | {.items[*].metadata.name} | [127.0.0.1 127.0.0.2] |
| [start:end:step] | Indicator operator | {.users[0].name} | myself |
| [,] | Union operator | {.items[*]['metadata.name', 'status.capacity']} | 127.0.0.1 127.0.0.2 map[cpu:4] map[cpu:8] |
| ?() | Filter | {.users[?(@.name=="e2e")].user.password} | Secret |
| range, end | Iteration list | {range .items[*]}[{.metadata.name}, {.status.capacity}] {end} | [127.0.0.1, map[cpu:4]] [127.0.0.2, map[cpu:8]] |
| '' | Reference explanation execution string | {range .items[*]}{.metadata.name}{'\t'}{end} | 127.0.0.1 127.0.0.2 |
| \ | Escape terminator | {.items[0].metadata.labels.kubernetes\.io/hostname} | 127.0.0.1 |

#### Using JSONPath Expressions Through ccictl

The following is an example of using ccictl and JSONPath expressions:

```
ccictl get pods -o json
ccictl get pods -o=jsonpath='{@}'
ccictl get pods -o=jsonpath='{.items[0]}'
ccictl get pods -o=jsonpath='{.items[0].metadata.name}'
ccictl get pods -o=jsonpath="{.items[*]['metadata.name', 'status.capacity']}"
```

  

```
ccictl get pods -o=jsonpath='{range .items[*]}{.metadata.name}{"\t"}{.status.startTime}{"\n"}{end}'
ccictl get pods -o=jsonpath='{.items[0].metadata.labels.kubernetes\.io/hostname}'
```

Regular expression in JSONPath

JSONPath regular expressions are not supported. To use regular expressions for matching, you can use tools such as jq.

```
# The JSONPath output of ccictl does not support regular expressions.
# The following command does not take effect.
ccictl get pods -o jsonpath='{.items[?(@.metadata.name=~/^test$/)].metadata.name}'

# The following command can obtain the required result.
ccictl get pods -o json | jq -r '.items[] | select(.metadata.name | test("test-")).metadata.name'
```

# 1.3.2 ccictl

## Scenario

ccictl is used to control the CCI 2.0 cluster manager.

```
ccictl [flag]
```

## Options

```
--cache-dir string     Default: "$HOME/.kubev2/cache"
```

Default cache directory

```
--cliconfig string
```

Path of the **cliconfig** file to be used by the CLI request.

```
--cluster string
```

Name of the cluster to be used in **cliconfig**

```
--context string
```

Name of the **cliconfig** context to be used

```
--insecure-skip-tls-verify
```

If the value is **true**, the validity of the server certificate is not checked.

```
-n, --namespace string
```

If present, it is the namespace scope of the CLI request.

```
--password string
```

Password used for basic authentication to the API server

```
--request-timeout string     Default: "0"
```

The length of time to wait before giving up on a single server request. Non-zero values should contain a corresponding time unit (for example, 1s, 2 min, 3h). The value **0** means that the requests will not time out.

```
-s, --server string
```

Address and port of the API server

```
--token string
```

Bearer token for authentication to the API server

  

```
--user string
```

Name of the cliconfig user to use

```
--username string
```

Username used for basic authentication to the API server

```
-v, --v int
```

Log level of the command. A larger value indicates more log details.

# 1.3.3 ccictl create

## Scenario

Create a resource based on a file or stdin.

Both JSON and YAML are supported.

```
ccictl create -f FILENAME
```

## Examples

```
# Use the data in pod.json to create a pod.
ccictl create -f ./pod.json

# Create a pod based on stdin in the JSON format.
cat pod.json | ccictl create -f -

# Edit the data in registry.yaml in JSON format and use the edited data to create resources.
ccictl create -f registry.yaml --edit -o json
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
--edit
```

Edit API resources before creating a resource.

```
-f, --filename strings
```

File name, directory, or file URL used to create a resource

```
-h, --help
```

Help information for create

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--raw string
```

Original URI used to send a POST request to the server. Use the transfer mode specified in the **cliconfig** file.

```
-R, --recursive
```

  

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

```
--validate string[="strict"]     Default: "strict"
```

The value must be one of the following: **"strict"** (or **"true"**), **"warn"**, and **"ignore"** (or **"false"**). **"true"** or **"strict"** will use the pattern definition to validate the input. If the input is invalid, the request fails.

**"false"** or **"ignore"** will not perform any schema definition checks, but will silently delete all unknown or duplicate fields.

```
--windows-line-endings
```

This option is related only when **--edit** is set to **true**. The default value is the native line end format of your platform.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.1 ccictl create namespace

## Scenario

Create a namespace with a specified name.

```
ccictl create namespace NAME [options]
```

## Examples

```
# Create a namespace named my-namespace.
ccictl create namespace my-namespace
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-h, --help
```

  

Help information for create namespace

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run **ccictl apply** on the object later.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.2 ccictl create secret docker-registry

## Scenario

Create a secret for use with Docker registries

Dockercfg secrets are used to authenticate against Docker registries.

When using the Docker command line to push images, you can authenticate to a given registry by running:

```
docker login DOCKER_REGISTRY_SERVER --username=DOCKER_USER --password=DOCKER_PASSWORD --email=DOCKER_EMAIL
```

This command produces a **~/.dockercfg** file that is used by subsequent **docker push** and **docker pull** commands to authenticate to the registry. The email address is optional.

When creating applications, you may have a Docker registry that requires authentication. In order for the nodes to pull images on your behalf, they must have the credentials. You can provide this information by creating a dockercfg secret and attaching it to your service account.

```
ccictl create secret docker-registry NAME --docker-username=user --docker-password=password --docker-email=email [--docker-server=string] [--from-file=[key=]source]
```

## Examples

```
# If the .dockercfg file does not exist, you can directly create a dockercfg secret.
ccictl create secret docker-registry my-secret --docker-server=DOCKER_REGISTRY_SERVER --docker-username=DOCKER_USER --docker-password=DOCKER_PASSWORD --docker-email=DOCKER_EMAIL

# Create a secret named my-secret based on ~/.docker/config.json.
ccictl create secret docker-registry my-secret --from-file=path/to/.docker/config.json
```

  

## Options

--allow-missing-template-keys Default: true

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

--append-hash

Append a hash of the secret to its name.

--docker-email string

Email for Docker registry

--docker-password string

Password for Docker registry authentication

--docker-server string Default: "https://index.docker.io/v1/"

Address of the server where the Docker registry is located

--docker-username string

Username for Docker registry authentication

--from-file strings

Key files can be specified using their file path, in which case a default name **.dockerconfigjson** will be given to them, or optionally with a name and file path, in which case the given name will be used. Specifying a directory will iterate each named file in the directory that is a valid secret key. For this command, the key should always be .dockerconfigjson.

-h, --help

Help information for create docker-registry

-o, --output string

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

--save-config

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

--template string

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

  

### 1.3.3.3 ccictl create secret generic

## Scenario

Create a secret based on a file, directory, or specified literal value.

A secret can contain one or more key-value pairs.

When creating a secret based on a file, the key will default to the basename of the file, and the value will default to the file content. If the basename is an invalid key or you want to choose your own, you may specify an alternate key.

When creating a secret based on a directory, each file whose basename is a valid key in the directory will be packaged into the secret. Any directory entries (such as subdirectories, symlinks, devices, pipes) except regular files are ignored.

```
ccictl create secret generic NAME [--type=string] [--from-file=[key=]source] [--from-literal=key1=value1]
```

## Examples

```
# Create a secret named my-secret with keys as each file in the bar folder.
ccictl create secret generic my-secret --from-file=path/to/bar

# Create a secret named my-secret with specified keys instead of the names on the disk.
ccictl create secret generic my-secret --from-file=ssh-privatekey=path/to/id_rsa --from-file=ssh-
publickey=path/to/id_rsa.pub

# Create a secret named my-secret with key1=supersecret and key2=topsecret.
ccictl create secret generic my-secret --from-literal=key1=supersecret --from-literal=key2=topsecret

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

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

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
--append-hash
```

Append a hash of the secret to its name.

```
--from-env-file strings
```

Specify the file path to read lines of key=val pairs to create a secret.

```
--from-file strings
```

Key files can be specified using their file path, in which case a default name will be given to them, or optionally with a name and file path, in which case the given name will be used. Specifying a directory will iterate each named file in the directory that is a valid secret key.

```
--from-literal strings
```

Specify a key and literal value (for example, mykey=somevalue) to insert in the secret.

  

-h, --help

Help information for create secret generic

-o, --output string

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

--save-config

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

--template string

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

--type string

Type of the secret to be created

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.4 ccictl create secret tls

## Scenario

Create a TLS secret using a given public/private key pair.

The public/private key pair must exist beforehand. The public key certificate must be in .PEM format and match the given private key.

```
ccictl create secret tls NAME --cert=path/to/cert/file --key=path/to/key/file
```

## Examples

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

## Options

--allow-missing-template-keys Default: true

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

--append-hash

Append a hash of the secret to its name.

--cert string

Path of the PEM-encoded public key certificate

-h, --help

  

Help information for create secret tls

```
--key string
```

Path of the private key associated with the given certificate

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.5 ccictl create configmap

## Scenario

Create a ConfigMap based on a file, directory, or specified literal value.

A ConfigMap can contain one or more key-value pairs.

When creating a ConfigMap based on a file, the key will default to the basename of the file, and the value will default to the file content. If the basename is an invalid key, you may specify an alternate key.

When creating a ConfigMap based on a directory, each file whose basename is a valid key in the directory will be packaged into the ConfigMap. Any directory entries (such as subdirectories, symlinks, devices, pipes) except regular files are ignored.

```
ccictl create configmap NAME [--from-file=[key=]source] [--from-literal=key1=value1]
```

## Examples

```
# Create a ConfigMap named my-config based on the bar folder.
ccictl create configmap my-config --from-file=path/to/bar

# Create a ConfigMap named my-config with specified keys instead of the names on the disk.
ccictl create configmap my-config --from-file=key1=/path/to/bar/file1.txt --from-file=key2=/path/to/bar/file2.txt

# Create a ConfigMap named my-config with key1=config1 and key2=config2.
ccictl create configmap my-config --from-literal=key1=config1 --from-literal=key2=config2

# Create a ConfigMap named my-config from the key-value pair in the file.
ccictl create configmap my-config --from-file=path/to/bar
```

  

```
# Create a ConfigMap named my-config from env files.
ccictl create configmap my-config --from-env-file=path/to/foo.env --from-env-file=path/to/bar.env
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
--append-hash
```

Append a hash of the ConfigMap to its name.

```
--from-env-file strings
```

Specify the file path to read lines of key=val pairs to create a ConfigMap.

```
--from-file strings
```

The key file can be specified using its file path. In this case, the base name of the file is used as the key of the ConfigMap. In addition, the key file can be specified using the key and file path. In this case, the specified key is used. Specifying a directory will traverse all named files in this directory (with valid ConfigMap keys as their base names).

```
--from-literal strings
```

Specify a key and literal value (for example, mykey=somevalue) to insert in the ConfigMap.

```
-h, --help
```

Help information for create configmap

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.6 ccictl create service loadbalancer

## Scenario

Create a LoadBalancer Service with a specified name.

  

```
ccictl create service loadbalancer NAME [--tcp=port:targetPort] [--elb-id:targetElbID]
```

## Examples

```
# Create a LoadBalancer Service named my-lbs.
ccictl create service loadbalancer my-lbs --tcp=5678:8080 --elb-id=xxx
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-h, --help
```

Help information for create service loadbalancer

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

```
--tcp strings
```

The port pair can be specified as *"<port>:<target port>"*.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

```
--elb-id string
```

Elastic load balancer ID

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.7 ccictl create service externalname

## Scenario

Create an ExternalName Service with a specified name.

An ExternalName Service references to an external DNS address instead of only pods. This allows application authors to reference services that exist off the platform, on other clusters, or locally.

```
ccictl create service externalname NAME --external-name external.name
```

  

## Examples

```
# Create an ExternalName Service named my-ns.
ccictl create service externalname my-ns --external-name bar.com
```

## Options

--allow-missing-template-keys Default: true

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

--external-name string

Service name

-h, --help

Help information for create service externalname

-o, --output string

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

--save-config

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

--tcp strings

The port pair can be specified as *"<port>:<target port>"*.

--template string

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.3.8 ccictl create deployment

## Scenario

Create a Deployment with a specified name.

```
ccictl create deployment NAME --image=image -- [COMMAND] [args...]
```

## Examples

```
# Create a Deployment named my-dep to run the busybox image. The pod has 1 vCPU and 2-GiB memory.
ccictl create deployment my-dep --image=busybox --pod-size-specs=1.0_2.00

# Create a Deployment with commands and set the pod CPU and memory to 2 vCPUs and 4 GiB.
ccictl create deployment my-dep --image=busybox --pod-size-specs=2.0_4.00 -- date
```

  

```
# Create a Deployment named my-dep that runs the nginx image and has three replicas. The pod has 1 vCPU and 2-GiB memory.
ccictl create deployment my-dep --image=nginx --pod-size-specs=1.0_2.00 --replicas=3

# Create a Deployment named my-dep that runs the busybox image and exposes port 5701. The pod has 1 vCPU and 2-GiB memory.
ccictl create deployment my-dep --image=busybox --pod-size-specs=1.0_2.00 --port=5701

# Create a Deployment named my-dep. The pod has multiple containers and 1 vCPU and 2-GiB memory.
ccictl create deployment my-dep --image=busybox:latest --pod-size-specs=1.0_2.00 --image=ubuntu:latest --image=nginx
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-h, --help
```

Help information for create deployment

```
--image strings
```

Image names. A deployment can have multiple images set for a multi-container pod.

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--port int32     Default: -1
```

Container port exposed by the Deployment.

```
-r, --replicas int32     Default: 1
```

Number of replicas to be created. The default value is **1**.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run the **ccictl apply** command on the object.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

```
--pod-size-specs
```

Pod specifications

**The following ccictl options can also be used in subcommands:**

**Parent command options**

  

# 1.3.4 ccictl set

## Scenario

Configure application resources.

These commands help you make changes to existing application resources.

```
ccictl set SUBCOMMAND
```

## Options

```
-h, --help
```

Help information for the set operations

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.4.1 ccictl set env

## Scenario

Update environment variables in a pod template.

List environment variable definitions in one or more pods, pod templates. Add, update, or remove container environment variable definitions in one or more pod templates (within replication controllers or Deployment configurations). View or modify the environment variable definitions on all containers in the specified pods or pod templates, or just those that match a wildcard.

If **--env -** is passed, environment variables can be read from stdin using the standard env syntax.

Possible resources include (case insensitive):

```
pods (po) and Deployments (deploy)
```

```
ccictl set env RESOURCE/NAME KEY_1=VAL_1 ... KEY_N=VAL_N
```

## Examples

```
# Update the registry Deployment with a new environment variable.
ccictl set env deployment/registry STORAGE_DIR=/local

# List the environment variables defined on the sample-build Deployment.
ccictl set env deployment/sample-build --list

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

# Output a modified Deployment in YAML without altering the object on the server.
ccictl set env deployment/sample-build STORAGE_DIR=/data -o yaml

# Update all containers in all replication controllers in the project to have ENV=prod.
ccictl set env deploy --all ENV=prod

# Import the environment from a secret.
ccictl set env --from=secret/mysecret deployment/myapp
```

  

```
# Import the environment from a ConfigMap with a prefix.
ccictl set env --from=configmap/myconfigmap --prefix=MYSQL_ deployment/myapp

# Import a specific key from a ConfigMap.
ccictl set env --keys=my-example-key --from=configmap/myconfigmap deployment/myapp

# Remove the environment variable ENV from the c1 container in all Deployment configurations.
ccictl set env deployments --all --containers="c1" ENV-

# Remove the environment variable ENV from a Deployment definition on the disk and update the Deployment configuration on the server.
ccictl set env -f deploy.json ENV-

# Set some local Shell environment variables into the Deployment configuration on the server.
env | grep RAILS_ | ccictl set env -e - deployment/registry
```

## Options

```
--all
```

If the value is **true**, all resources in the namespace of the specified resource type are selected.

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-c, --containers string     Default: "*"
```

Names of the containers to be changed in the selected pod template. Wildcard characters can be used.

```
-e, --env strings
```

Specify a key-value pair for an environment variable to set into each container.

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resources whose environments are to be updated.

```
--from string
```

Name of a resource from which to inject environment variables

```
-h, --help
```

Help information for set env

```
--keys strings
```

List of comma-separated keys to be imported from the specified resource

```
--list
```

If the value is **true**, the environment and all changes are displayed in the standard format. This flag is removed when the **ccictl view env** command is used.

```
--local
```

If the value is **true**, **set env** will not communicate with the API server and will run locally.

```
-o, --output string
```

  

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--overwrite    Default: true
```

If the value is **true**, the environment can be overwritten. Otherwise, the update to overwrite the existing environment is rejected.

```
--prefix string
```

Prefix to be added to the variable names

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--resolve
```

If the value is **true**, the secret or ConfigMap reference is displayed when the variable is listed.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.4.2 ccictl set image

## Scenario

Update existing container images of resources.

Possible resources include (case insensitive):

```
pods (po) and Deployments (deploy)
```

The command syntax is as follows:

```
ccictl set image (-f FILENAME | TYPE NAME) CONTAINER_NAME_1=CONTAINER_IMAGE_1 ... CONTAINER_NAME_N=CONTAINER_IMAGE_N
```

## Examples

```
# Set a Deployment's nginx container image to 'nginx:1.9.1' and the busybox container image to 'busybox'.
ccictl set image deployment/nginx busybox=busybox nginx=nginx:1.9.1

# Update all Deployments' nginx container image to 'nginx:1.9.1'.
ccictl set image deployments nginx=nginx:1.9.1 --all

# Update all container images of the abc Deployment to "nginx:1.9.1".
```

  

```
ccictl set image deploy abc *=nginx:1.9.1

# Print result (in YAML format) of updating the nginx container image from the local file. No request is sent to the server.
ccictl set image -f path/to/file.yaml nginx=nginx:1.9.1 --local -o yaml
```

## Options

```
--all
```

If the value is **true**, all resources in the namespace of the specified resource type are selected.

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resources whose environments are to be updated.

```
-h, --help
```

Help information for set image

```
--local
```

If the value is **true**, **set image** will not communicate with the API server and will run locally.

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

  

### 1.3.4.3 ccictl set resources

## Scenario

Specify compute resource requirements (vCPU and memory) for any resource that defines a pod template. If a pod is successfully scheduled, the resource requests are guaranteed, but it may burst up to the specified limits.

For each type of compute resource, if only the limit is specified and the request is omitted, the request will default to the limit.

Possible resources include (case insensitive): Run **ccictl api-resources** to view a complete list of supported resources.

```
ccictl set resources (-f FILENAME | TYPE NAME)  ([--limits=LIMITS & --requests=REQUESTS]
```

## Examples

```
# Set the container vCPU limit of the nginx Deployment to "200m" and memory limit to "512Mi".
ccictl set resources deployment nginx -c=nginx --limits=cpu=200m,memory=512Mi

# Set the resource requests and limits for all containers in the nginx Deployment.
ccictl set resources deployment nginx --limits=cpu=200m,memory=512Mi --
requests=cpu=100m,memory=256Mi

# Remove the requests for resources on containers in the nginx Deployment.
ccictl set resources deployment nginx --limits=cpu=0,memory=0 --requests=cpu=0,memory=0

# Print the result (in YAML format) of updating the container restrictions based on the local list. No request is sent to the server.
ccictl set resources -f path/to/file.yaml --limits=cpu=200m,memory=512Mi --local -o yaml
```

## Options

```
--all
```

If the value is **true**, all resources in the namespace of the specified resource type are selected.

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-c, --containers string     Default: "*"
```

Names of the containers to be changed in the selected pod template. Wildcard characters can be used.

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resources whose environments are to be updated.

```
-h, --help
```

Help information for set resources

```
--limits string
```

Resource limits of a specified container, for example, 'cpu=100m,memory=256Mi'. Note that server side components may assign requests depending on the server configuration, such as limit ranges.

  

```
--local
```

If the value is **true**, **set image** will not communicate with the API server and will run locally.

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--requests string
```

Resource requests of a specified container, for example, 'cpu=100m,memory=256Mi'. Note that server side components may assign requests depending on the server configuration, such as limit ranges.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following cctl options can also be used in subcommands:**

**Parent command options**

## 1.3.4.4 cctl set selector

## Scenario

Set the selector on a resource. A selector must begin with a letter or number, and may contain letters, numbers, hyphens, dots, and underscores, up to 63 characters. If **--resource-version** is specified, the updates will use this resource version. Otherwise, the existing resource version will be used.

```
cctl set selector (-f FILENAME | TYPE NAME) EXPRESSIONS [--resource-version=version]
```

## Constraints

- Currently, selectors can only be set on Service objects.
- The new selector will overwrite the old selector if the resource had one prior to the invocation of **set selector**.

## Examples

```
# Set the selector after a Service is created.
cctl create service loadbalancer my-svc --tcp=8088:8088 --elb-id="xxx" -o yaml -nt1 | cctl set selector --
```

  

```
local -f - 'environ
ment=qa' -oyaml
```

## Options

```
--all
```

If the value is **true**, all resources in the namespace of the specified resource type are selected.

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-f, --filename strings
```

File names to distinguish different resources

```
-h, --help
```

Help information for set selector

```
--local
```

If the value is **true**, **set selector** will not communicate with the API server and will run locally.

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--resource-version string
```

If the value is not empty, the selector update succeeds only when the given value is the current resource version of the object. This option is valid only when a single resource is specified.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following cctl options can also be used in subcommands:**

**Parent command options**

# 1.3.5 cctl explain

## Scenario

Describe the fields and structures of various resources.

  

This command describes the fields associated with each supported API resource. These fields are identified by a simple JSONPath identifier:

```
<type>.<field name>[.<field name>]
```

The information about each field is retrieved from the server in OpenAPI format.

Use **ccictl api-resources** to obtain a complete list of supported resources.

```
ccictl explain TYPE [--recursive=FALSE|TRUE] [--api-version=api-version-group] [-o|--output=plaintext|plaintext-openapiv2]
```

## Examples

```
# Obtain the document of the resource and its fields.
ccictl explain pods

# Obtain all fields in the resource.
ccictl explain pods --recursive

# Obtain the description of Deployment in the supported API version.
ccictl explain deployments --api-version=apps/v1

# Obtain the documentation of a specific field of a resource.
ccictl explain pods.spec.containers
```

## Options

```
--api-version string
```

API version (group/version) of the resource

```
-h, --help
```

Help information for explain

```
--recursive
```

If the value is **true**, the names of all fields are printed recursively. Otherwise, the available fields and their descriptions are printed.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.6 ccictl get

## Scenario

Display one or more resources.

Print a table that contains the most important information related to a specified resource. You can use a label selector and the --selector flag to filter the list. If the requested resource type is namespaced, you will only see the result in the current namespace unless you pass the **--all-namespaces** parameter.

By specifying the output as 'template' and providing a Go template as the value of the **--template** flag, you can filter the attributes of the fetched resources.

Use **ccictl api-resources** to obtain a complete list of supported resources.

  

```
ccictl get [(-o|--output=)json|yaml|name|go-template|go-template-file|template|templatefile|jsonpath|
jsonpath-as-json|jsonpath-file|custom-columns|custom-columns-file|wide] (TYPE[.VERSION][.GROUP]
[NAME | -l label] | TYPE[.VERSION][.GROUP]/NAME ...) [flags]
```

## Examples

```
# List all pods in ps output format.
ccictl get pods

# List all pods in ps output format and provide more information (such as pod IP addresses).
ccictl get pods -o wide

# List a single pod with a specified name in ps output format.
ccictl get po web

# List Deployments in the v2 version of the cci API group in JSON format.
ccictl get deployments.v2.cci -o json

# List a single pod in JSON format.
ccictl get -o json pod web-pod-13je7

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

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

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

# List all Deployments and Services in ps output format.
ccictl get deploy,services

# List one or more resources by type and name.
ccictl get deploy/web service/frontend pods/web-pod-13je7

# List all Deployments in the backend namespace.
ccictl get deployments --namespace backend

# List all pods in all namespaces.
ccictl get pods --all-namespaces
```

## Options

```
-A, --all-namespaces
```

If this flag exists, the requested object is listed across all namespaces. Even if a namespace is specified using **--namespace**, the namespace in the current context is ignored.

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
--chunk-size int     Default: 500
```

Return large lists in chunks rather than all at once. The value **0** indicates that the function is disabled.

```
--field-selector string
```

Selector used for filtering (field query). The value can be **=**, **==**, or **!=**, for example, --field-selector key1=value1,key2=value2. The server only supports a limited number of field queries per type.

  

-f, --filename strings

List of file names, directories, or file URLs, which are used to identify the resource to be obtained from the server.

-h, --help

Help information for get

--ignore-not-found

If the requested object does not exist, the command returns the exit code 0.

-L, --label-columns strings

Accept a comma-separated list of labels that will be used as different columns in the printed table. The names are case sensitive. You can also use multiple flag options, for example, -L label1 -L label2...

--no-headers

When the default or custom column output format is used, do not print the headers (which are printed by default).

-o, --output string

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, **jsonpath-file**, **custom-columns**, **custom-columns-file**, and **wide**. For details, refer to **Custom Columns**, Golang template (http://golang.org/pkg/text/template/#pkg-overview), and **JSONPath Support** in the JSONPath template.

--output-watch-events

When **--watch** or **--watch-only** is used, the watched event object is output. Existing objects are output as initial ADDED events.

--raw string

Raw URI to request from the server. Use the transfer mode specified in the **cliconfig** file.

-R, --recursive

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

-l, --selector string

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

--server-print Default: true

If the value is **true**, the server returns the appropriate table output. Extended APIs and CRDs are supported.

--show-kind

If this flag is present, the resource types of the requested objects are listed.

--show-labels

During printing, all labels are displayed in the last column (which is hidden by default).

  

```
--sort-by string
```

If the value is not empty, this column is used to sort the list type. The field specification is expressed as a JSONPath expression (for example, {.metadata.name}). The field in the API resource specified by the JSONPath expression must be an integer or a string.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

```
-w, --watch
```

After listing or obtaining the requested objects, watch their changes.

```
--watch-only
```

Watch for the changes to the requested objects, without listing or getting the objects first.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.7 ccictl edit

## Scenario

Use the default editor to edit resources.

- The **edit** command allows you to directly edit any API resource that can be retrieved using the command-line tool. It opens the editor defined by the *EDITOR* environment variable, or rolls back to using the "vi" of Linux or the "notepad" of Windows. When attempting to open the editor, it first attempts to use the Shell defined in the *SHELL* environment variable. If no Shell is defined, the default Shell is used. In Linux, the default Shell is /bin/bash. In Windows, the default Shell is cmd.
- You can edit multiple objects, but only one change is applied at a time. This command accepts the file names and command-line arguments, although the files you point to must be previously saved versions of the resources.
- Editing is done with the API version used to fetch the resource. To edit using a specific API version, fully qualify the resource, version, and group.
- The default format is YAML. To edit in JSON, specify **-o json**.
- The **--windows-line-endings** flag can be used to force Windows line endings. Otherwise, the default value for the OS is used.
- If an error occurs during the update, a temporary file containing the changes that are not applied will be created on the disk. The most common error when updating resources is that another editor has changed the resource on the server. When this happens, you must update the resource to the newer version, or update the temporarily saved copy to include the latest resource version.

```
ccictl edit (RESOURCE/NAME | -f FILENAME)
```

  

## Examples

```
# Edit the Service named registry.
cctl edit svc/registry

# Use an alternative editor.
EDITOR="nano" cctl edit svc/registry

# Edit the mydeploy Deployment in JSON using the CCI/v2 API format.
cctl edit deployment.v2.cci/mydeploy -o json

# Edit the mydeployment Deployment in YAML and save the modified configuration in its annotations.
cctl edit deployment/mydeployment -o yaml --save-config
```

## Options

```
-f, --filename strings
```

List of file names, directories, or file URLs used to edit resources.

```
-h, --help
```

Help information for edit

```
-o, --output string
```

Output format. The value can be **json** or **yaml**.

```
--output-patch
```

If the resource is edited, a patch is output.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--save-config
```

If the value is **true**, the configuration of the object is saved in its annotation. Otherwise, the annotation remains unchanged. This flag is useful when you want to run **cctl apply** on the object later.

```
--validate string[="strict"]     Default: "strict"
```

The value must be one of the following: **"strict"** (or **"true"**), **"warn"**, and **"ignore"** (or **"false"**). **"true"** or **"strict"** will use the pattern definition to validate the input. If the input is invalid, the request fails. **"false"** or **"ignore"** will not perform any schema definition checks, but will silently delete all unknown or duplicate fields.

```
--windows-line-endings
```

The default value is the native line end format of your platform.

**The following cctl options can also be used in subcommands:**

**Parent command options**

  

# 1.3.8 ccictl delete

## Scenario

Delete resources based on the file name, stdin, resource, and name, or based on the resource and label selector.

Both JSON and YAML are supported. Only one type of argument can be specified: file name, resource and name, or resource label selector.

Some resources (such as pods) support graceful deletion. These resources define the default duration (grace period) before a forcible termination, but you can override the value using the **--grace-period** flag or pass **--now** to set the grace period to 1. Deletion may not be confirmed immediately because these resources usually represent entities in the cluster. If the API server is not accessible, the termination time may be much longer than the grace period. To force the deletion of a resource, you must specify the **--force** flag. Note: Only some resources support graceful deletion. If graceful deletion is not supported, the **--grace-period** flag will be ignored.

Important: If you forcibly delete a pod, the system does not wait for the pod process to be terminated. As a result, the pod process may continue to run until the node detects the deletion request and completes graceful deletion. You can forcibly delete a pod only when you are sure that the pod has been terminated or your application can tolerate multiple copies of the same pod running at the same time.

Note that the delete command does not check the resource version. If someone submits an update to a resource right when you submit a deletion request, their update will be lost along with the rest of the resource.

```
ccictl delete ([-f FILENAME] | TYPE [(NAME | -l label | --all)])
```

## Examples

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

# Delete resources based on the content in the directory.
ccictl delete -f dir/

# Delete the resources in all files that end with .json.
ccictl delete -f '*.json'

# Delete a pod based on the type and name transferred to stdin in JSON.
cat pod.json | ccictl delete -f -

# Delete the pods and Services named baz and foo.
ccictl delete pod,service baz foo

# Delete the pod and Service with the name=myLabel label.
ccictl delete pods,services -l name=myLabel

# Delete a pod with the minimum delay.
ccictl delete pod foo --now

# Forcibly delete a pod.
ccictl delete pod foo --force

# Delete all pods.
ccictl delete pods --all
```

  

## Options

--all

Delete all resources in the namespace of a specified resource type.

-A, --all-namespaces

If this option is present, all objects requested across namespaces are listed. Even if a namespace is specified using **--namespace**, the namespace in the current context is ignored.

--cascade string[="background"]  Default: "background"

Select the cascading policy for deleting dependent objects (for example, pods created by Deployment). The value must be **background**, **orphan**, or **foreground**, and the default value is **background**.

--field-selector string

Selector used for filtering (field query). The value can be **=**, **==**, or **!=**, for example, --field-selector key1=value1,key2=value2. The server only supports a limited number of field queries per type.

-f, --filename strings

Names of the files that contain the resources to be deleted

--force

If the value is **true**, the resource is immediately removed from the API, and the graceful deletion process is bypassed. Note that deleting some resources immediately may cause inconsistency or data loss, and you need to confirm the operation.

--grace-period int  Default: -1

Period of time given to the resource to terminate gracefully, in seconds. If the value is a negative number, it is ignored. If the value is **1**, the resource is immediately terminated. This parameter can only be set to **0** when **--force** is set to **true** (forcible deletion).

-h, --help

Help information for delete

--ignore-not-found

"resource not found" is considered as a successful deletion. When the **--all** parameter is specified, the default value is **true**.

--now

If the value is **true**, the resource will be marked for immediate shutdown (equivalent to **--grace-period=1**).

-o, --output string

Output mode. Use **-o name** to obtain a shorter output (resource/name).

--raw string

Raw URI to request from the server. Use the transfer mode specified in the **cliconfig** file.

-R, --recursive

  

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--timeout duration
```

The length of time to wait before the deletion is canceled. The value **0** indicates that the timeout duration is determined based on the object size.

```
--wait     Default: true
```

If the value is **true**, the resource is returned after it disappears. This waits for the finalizer.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.9 ccictl rollout

## Scenario

Manage the rollout of one or more resources.

Valid resource types include:

Deployments

```
ccictl rollout SUBCOMMAND
```

## Examples

```
# Roll back the Deployment to the previous version.
ccictl rollout undo deployment/abc

# Check the status of a Deployment.
ccictl rollout status deploy/foo

# Restart a Deployment.
ccictl rollout restart deployment/abc

# Restart the Deployment with the app=nginx label.
ccictl rollout restart deployment --selector=app=nginx
```

## Options

```
-h, --help
```

Help information for rollout

**The following ccictl options can also be used in subcommands:**

**Parent command options**

  

### 1.3.9.1 ccictl rollout history

## Scenario

View the previous revisions and configurations that have been brought online.

```
ccictl rollout history (TYPE NAME | TYPE/NAME) [flags]
```

## Examples

```
# View the rollout history of a Deployment.
ccictl rollout history deployment/abc

# View the details about revision 3 of the Deployment.
ccictl rollout history deployment/abc --revision=3
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resource to be obtained from the server.

```
-h, --help
```

Help information for rollout history

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--revision int
```

View details, including the pod template of the specified revision.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

  

## 1.3.9.2 ccictl rollout restart

### Scenario

Restart resources.

The resources will go online again.

```
ccictl rollout restart RESOURCE
```

### Examples

```
# Restart all Deployments in the test-namespace namespace.
ccictl rollout restart deployment -n test-namespace

# Restart a Deployment.
ccictl rollout restart deployment/nginx

# Restart the Deployment with the app=nginx label.
ccictl rollout restart deployment --selector=app=nginx
```

### Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resource to be obtained from the server.

```
-h, --help
```

Help information for rollout restart

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

  

**Parent command options**

## 1.3.9.3 ccictl rollout status

### Scenario

Show the status of the rollout.

By default, **rollout status** will watch the latest rollout status until it is complete. If you do not want to wait for rollout to complete, you can use set **--watch** to **false**. Note that if a new rollout starts in-between, **rollout status** continues watching the latest revision. If you want to watch a specific revision and stop it when it is overwritten by another revision, use **--revision=N**, where *N* is the revision you want to watch.

```
ccictl rollout status (TYPE NAME | TYPE/NAME) [flags]
```

### Examples

```
# Watch the rollout status of a Deployment.
ccictl rollout status deployment/nginx
```

### Options

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resource to be obtained from the server.

```
-h, --help
```

Help information for rollout status

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--revision int
```

Pin to a specific revision for showing its status. The default value is **0** (latest revision version).

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--timeout duration
```

The length of time to wait before watch is ended. The value **0** indicates that watch never ends. Any other values should contain a corresponding time unit (for example, 1s, 2 min, or 3h).

```
-w, --watch     Default: true
```

Watch the status of the rollout until it is done.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

  

## 1.3.9.4 ccictl rollout undo

### Scenario

Roll back to a previously released version.

```
ccictl rollout undo (TYPE NAME | TYPE/NAME) [flags]
```

### Examples

```
# Roll back the Deployment to the previous status.
ccictl rollout undo deployment/abc

# Roll back to Deployment revision 3.
ccictl rollout undo deploy/abc --to-revision=3
```

### Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
-f, --filename strings
```

List of file names, directories, or file URLs, which are used to identify the resource to be obtained from the server.

```
-h, --help
```

Help information for rollout undo

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

```
--to-revision int
```

Revision version to be rolled back to. The default value is **0** (latest revision version).

**The following ccictl options can also be used in subcommands:**

  

**Parent command options**

# 1.3.10 ccictl describe

## Scenario

Show details of a specific resource or group of resources.

Print a detailed description of the selected resources, including related resources such as events or controllers. You may select a single object by name, all objects of that type, provide a name prefix, or label selector. The following is an example:

```
ccictl describe TYPE NAME_PREFIX
```

This command first checks for an exact match with **TYPE** and **NAME_PREFIX**. If no such resource exists, it outputs the details of all resources whose names are prefixed with **NAME_PREFIX**.

Use **ccictl api-resources** to obtain a complete list of supported resources.

```
ccictl describe (-f FILENAME | TYPE [NAME_PREFIX | -l label] | TYPE/NAME)
```

## Examples

```
# Describe a node.
ccictl describe deploy example-deploy

# Describe a pod.
ccictl describe pods/nginx

# Describe the pods identified by type and name in pod.json.
ccictl describe -f pod.json

# Describe all pods.
ccictl describe pods

# Describe the pod with the name=myLabel label.
ccictl describe pods -l name=myLabel

# Describe all pods for the frontend Deployment.
ccictl describe pods frontend
```

## Options

```
-A, --all-namespaces
```

If this flag exists, the requested object is listed across all namespaces. Even if a namespace is specified using **--namespace**, the namespace in the current context is ignored.

```
--chunk-size int     Default: 500
```

Return large lists in chunks rather than all at once. The value **0** indicates that the function is disabled.

```
-f, --filename strings
```

List of file names, directories, or file URLs that contain the resources to be described.

```
-h, --help
```

Help information for describe

  

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.11 ccictl logs

## Scenario

Print the logs of a container in a pod or a specified resource. If a pod has only one container, the container name is optional.

```
ccictl logs [-f] [-p] (POD | TYPE/NAME) [-c CONTAINER]
```

## Examples

```
# Return snapshot logs of the nginx pod that contains only one container.
ccictl logs nginx

# Return snapshot logs from the nginx pod, with the source pod and container name as the prefix of each line.
ccictl logs nginx --prefix

# Return snapshot logs from the nginx pod. The output is limited to 500 bytes.
ccictl logs nginx --limit-bytes=500

# Return snapshot logs from the nginx pod and wait for the pod to start for a maximum of 20 seconds.
ccictl logs nginx --pod-running-timeout=20s

# Return snapshot logs of the nginx pod that contains multiple containers.
ccictl logs nginx --all-containers=true

# Return the snapshot logs of all containers in the pods with the app=nginx label.
ccictl logs -l app=nginx --all-containers=true

# Return snapshot logs of containers in the pods with the app=nginx label. Concurrent log requests can be sent to a maximum of 10 pods.
ccictl logs -l app=nginx --max-log-requests=10

# Return the logs of the ruby container that was terminated in the web-1 pod.
ccictl logs -p -c ruby web-1

# Start streaming logs from the Nginx pod, even if an error occurs.
ccictl logs nginx -f --ignore-errors=true

# Start streaming logs of the ruby container in the web-1 pod.
ccictl logs -f -c ruby web-1

# Start streaming logs of all containers in the pods with the app=nginx label.
ccictl logs -f -l app=nginx --all-containers=true

# Display only the last 20 lines of the nginx pod output.
ccictl logs --tail=20 nginx

# Display all logs written by the nginx pod in the last hour.
```

  

```
ccictl logs --since=1h nginx

# Display all logs that contain timestamps in the nginx pod from 06:00:00 UTC on August 30, 2024.
ccictl logs nginx --since-time=2024-08-30T06:00:00Z --timestamps=true

# Display the logs of the nginx pod and skip the kubelet certificate verification.
ccictl logs --insecure-skip-tls-verify-backend nginx

# Return snapshot logs of the nginx-1 container of the nginx Deployment.
ccictl logs deployment/nginx -c nginx-1
```

## Options

```
--all-containers
```

Obtain logs of all containers in a pod.

```
-c, --container string
```

Print logs of a container in a pod.

```
-f, --follow
```

Specify whether the logs should be streamed.

```
-h, --help
```

Help information for logs

```
--ignore-errors
```

If you are watching or following pod logs, any non-fatal errors are allowed.

```
--insecure-skip-tls-verify-backend
```

Skip kubelet identity authentication for the log request source. Theoretically, attackers may provide invalid log content. If the certificate provided by kubelet has expired, you may need to use this option.

```
--limit-bytes int
```

Maximum number of bytes of logs to return. No limit is specified by default.

```
--max-log-requests int     Default: 5
```

Specify the maximum number of concurrent logs that must be followed when a selector is used. The default value is **5**.

```
--pod-running-timeout duration     Default: 20s
```

The length of time to wait until at least one pod is running (the value must be greater than 0, for example, 5s, 2 min, or 3h).

```
--prefix
```

Add the log source (pod name and container name) prefix to the beginning of each log line.

```
-p, --previous
```

If the value is **true**, print the logs of the previous instance of the container in the pod (if it exists).

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

  

```
--since duration
```

Only logs that are newer than a relative duration (like 5s, 2 min, or 3h) are returned. By default, all logs are returned. Only one of **since-time** and **since** can be used.

```
--since-time string
```

Return logs generated after a specific date (RFC3339). By default, all logs are returned. Only one of **since-time** and **since** can be used.

```
--tail int     Default: -1
```

Number of lines in the latest log file to be displayed. If no selector is specified, the default value is **-1**, indicating that all log lines are displayed. If a selector is provided, the default value is **10**.

```
--timestamps
```

Each line of the log output contains a timestamp.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.12 ccictl exec

## Scenario

Execute commands in a container.

```
ccictl exec (POD | TYPE/NAME) [-c CONTAINER] [flags] -- COMMAND [args...]
```

## Examples

```
# Run the date command in the mypod pod to obtain the output. By default, the command is executed in the first container.
ccictl exec mypod -- date

# Run the date command in the ruby-container container of the mypod pod and obtain the output.
ccictl exec mypod -c ruby-container -- date

# Switch to the raw terminal mode, send stdin from the mypod pod to 'bash' in the ruby-container container, and send stdout/stderr from 'bash' back to the client.
ccictl exec mypod -c ruby-container -i -t -- bash -il

# List the content of /usr in the first container of the mypod pod and sort the content by modification time.
# If the command to be executed in the pod has any flag that overlaps with ccictl (for example, -i), use two hyphens (--) to separate the command's flags/arguments.
# Also note that you should not enclose your command and its flags/arguments in quotes, unless this is the way how you would execute it normally (for example, execute ls -t /usr, not "ls -t /usr").
ccictl exec mypod -i -t -- ls -t /usr
```

## Options

```
-c, --container string
```

Container name. If the name is omitted, the **kubectl.kubernetes.io/default-container** annotation is used to select the container to be mounted. Otherwise, the first container in the pod is selected.

```
-f, --filename strings
```

Files used to execute the command into resources.

  

```
-h, --help
```

Help information for exec

```
--pod-running-timeout duration     Default: 1m0s
```

The length of time to wait until at least one pod is running (the value must be greater than 0, for example 5s, 2m, or 3h).

```
-q, --quiet
```

Print only the output of remote sessions.

```
-i, --stdin
```

Pass stdin to the container.

```
-t, --tty
```

stdin is a TTY.

**The following cctl options can also be used in subcommands:**

**Parent command options**

# 1.3.13 cctl apply

## Scenario

Apply a configuration to a resource by file name or stdin. The resource name must be specified. If the resource does not exist, the resource will be created. To use the **apply** command, you should always use **apply** or **create --save-config** when initially creating the resource.

Both JSON and YAML are supported.

```
cctl apply -f FILENAME
```

## Examples

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

# Apply resources from the directory.
cctl apply -f dir/

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

# Apply the configurations in all files that end with .json.
cctl apply -f '*.json'
```

## Options

```
--all
```

Select all resources in the namespace of the specified resource types.

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

  

```
--cascade string[="background"]    Default: "background"
```

The value must be **background**, **orphan**, or **foreground**. This is to select the cascading policy for deleting dependent objects (for example, pods created by Deployment). The default value is **background**.

```
-f, --filename strings
```

Files that contain the configurations to be applied.

```
--force
```

If the value is **true**, the resource is immediately removed from the API, and the graceful deletion process is bypassed. Note that deleting some resources immediately may cause inconsistency or data loss, and you need to confirm the operation.

```
--grace-period int    Default: -1
```

Period of time given to the resource to terminate gracefully, in seconds. If the value is a negative number, it is ignored. If the value is **1**, the resource is immediately terminated. This parameter can only be set to **0** when **--force** is set to **true** (forcible deletion).

```
-h, --help
```

Help information for apply

```
--openapi-patch    Default: true
```

If the value to **true**, OpenAPI is used to calculate the diff when OpenAPI exists, and resources can be found in the OpenAPI specifications. Otherwise, the built-in type is used.

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--overwrite    Default: true
```

The value in the modified configuration is used to automatically resolve the conflict between the modified configuration and the real-time configuration.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

```
--timeout duration
```

  

The length of time to wait before the deletion is canceled. The value **0** indicates that the timeout duration is determined based on the object size.

```
--validate string[="strict"]     Default: "strict"
```

The value must be one of the following: **"strict"** (or **"true"**), **"warn"**, and **"ignore"** (or **"false"**). **"true"** or **"strict"** will use the pattern definition to validate the input. If the input is invalid, the request fails. **"false"** or **"ignore"** will not perform any schema definition checks, but will silently delete all unknown or duplicate fields.

```
--wait
```

If the value is **true**, the resource is returned after it disappears. This waits for the finalizer.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.13.1 ccictl apply edit-last-applied

## Scenario

Use the default editor to edit the latest **last-applied-configuration** annotation of the resource.

- The **edit-last-applied** command allows you to directly edit any API resource that can be retrieved using the command-line tool. It opens the editor defined by the *EDITOR* environment variable, or "vi" on Linux or "notepad" on Windows by default. You can edit multiple objects, but the changes you make can only be applied one at a time. This command accepts file names and command line arguments, but the file you point to must be a previously saved version of the resource.
- The default format is YAML. To edit in JSON, specify **-o json**.
- The **--windows-line-endings** flag can be used to force the use of Windows-style line endings. Otherwise, the default settings of the OS will be used.
- If an error occurs during the update, a temporary file containing the changes that are not applied is created on the disk. The most common error during an update is that another editor is changing the resource on the server. When this occurs, you will have to apply your changes to the newer version of the resource, or update your temporary saved copy to include the latest resource version.

```
ccictl apply edit-last-applied (RESOURCE/NAME | -f FILENAME)
```

## Examples

```
# Edit the last-applied-configuration annotation by type or name in the YAML file.
ccictl apply edit-last-applied deployment/nginx

# Edit the last-applied-configuration annotation in JSON.
ccictl apply edit-last-applied -f deploy.yaml -o json
```

## Options

```
-f, --filename strings
```

  

List of file names, directories, or file URLs used to edit resources.

```
-h, --help
```

Help information for edit-last-applied

```
-o, --output string
```

Output format. The value can be **json** or **yaml**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
--validate string[="strict"]     Default: "strict"
```

The value must be one of the following: **"strict"** (or **"true"**), **"warn"**, and **"ignore"** (or **"false"**). **"true"** or **"strict"** will use the pattern definition to validate the input. If the input is invalid, the request fails. **"false"** or **"ignore"** will not perform any schema definition checks, but will silently delete all unknown or duplicate fields.

```
--windows-line-endings
```

The default value is the native line ending format of your platform.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.13.2 ccictl apply set-last-applied

## Scenario

Set the **last-applied-configuration** annotation to match the content of a file. This updates the **last-applied-configuration** annotation, as if you had run **ccictl apply -f <file>**, without updating any other part of the object.

```
ccictl apply set-last-applied -f FILENAME
```

## Examples

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

# Execute the set-last-applied operation against each configuration file in a directory.
ccictl apply set-last-applied -f path/

# Set the last-applied-configuration annotation for a resource to match the content of a file. If no such annotation exists, an annotation will be created.
ccictl apply set-last-applied -f deploy.yaml --create-annotation=true
```

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
--create-annotation
```

  

If the current object does not have the **last-applied-configuration** annotation, the annotation will be created.

```
-f, --filename strings
```

List of file names, directories, or URLs that contain the **last-applied-configuration** annotation.

```
-h, --help
```

Help information for set-last-applied

```
-o, --output string
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following cctl options can also be used in subcommands:**

**Parent command options**

## 1.3.13.3 cctl apply view-last-applied

## Scenario

View the latest **last-applied-configuration** annotations by type, name or file.

By default, the output is printed to stdout in YAML format. You can use the **-o** option to change the output format.

```
cctl apply view-last-applied (TYPE [NAME | -l label] | TYPE/NAME | -f FILENAME)
```

## Examples

```
# View the last-applied-configuration annotation by name or type or in YAML.
cctl apply view-last-applied deployment/nginx

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

## Options

```
--all
```

Select all resources in the namespace of the specified resource types.

```
-f, --filename strings
```

List of file names, directories, or URLs that contain the **last-applied-configuration** annotation.

```
-h, --help
```

Help information for view-last-applied

```
-o, --output string     Default: "yaml"
```

  

Output format. The value must be **yaml** or **json**.

```
-R, --recursive
```

Process the directory used in **-f** or **--filename** recursively. This option is useful when you want to manage related manifests organized within the same directory.

```
-l, --selector string
```

Selector used for filtering (label query). The value can be **=**, **==**, or **!=**, for example, -l key1=value1,key2=value2. Matched objects must meet all specified label constraints.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.14 ccictl api-resources

## Scenario

Print the supported API resources on the server.

```
ccictl api-resources [flags]
```

## Examples

```
# Print the supported API resources on the server.
ccictl api-resources

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

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

# Print the supported namespaced resources.
ccictl api-resources --namespaced=true

# Print the supported non-namespaced resources.
ccictl api-resources --namespaced=false

Print the supported API resources with a specific API group.
ccictl api-resources --api-group=cci
```

## Options

```
-api-group string
```

Only the resources in a specified API group are printed.

```
--cached
```

If available, the cached resource list will be used.

```
-h, --help
```

Help information for api-resources

```
--namespaced     Default: true
```

If the value is **false**, non-namespaced resources will be returned. Otherwise, namespaced resources are returned by default.

  

```
--no-headers
```

When the default or custom column output format is used, do not print the headers (which are printed by default).

```
-o, --output string
```

Output format. The value can be **wide** or **name**.

```
--sort-by string
```

If the value is not empty, the resource list is sorted using the specified field, which can be **"name"** or **"kind"**.

```
--verbs strings
```

Filter the resources that support the specified verb.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.15 ccictl api-versions

## Scenario

Print the API versions supported by the server in the format of *group/version*.

```
ccictl api-versions
```

## Examples

```
# Print the supported API versions.
ccictl api-versions
```

## Options

```
-h, --help
```

Help information for api-versions

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.16 ccictl config

## Scenario

Use subcommands such as **ccictl config set current-context my-context** to modify the **cliconfig** file.

The loading order follows the following rules:

- If the **--cliconfig** flag is set, only that file is loaded. This flag can be set only once, and no merging occurs.
- If the *$CLICONFIG* environment variable is set, it is used as a list of paths (normal path delimiting rules of the system). These paths are merged. When a value is modified, the value is also modified in the file that defines the

  

corresponding content. When a value is created, the value is also created in the first file that exists. If no file exists in the chain, the last file in the list is created.

Otherwise, **${HOME}/.kubev2/config** is used, and no merging occurs.

```
ccictl config SUBCOMMAND
```

## Options

```
-h, --help
```

Help information for config

```
--cliconfig string
```

Use a specific **cliconfig** file.

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.1 ccictl config current-context

## Scenario

Display the current context.

```
ccictl config current-context [flags]
```

## Examples

```
# Display the current context.
ccictl config current-context
```

## Options

```
-h, --help
```

Help information for current-context

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.2 ccictl config get-contexts

## Scenario

Display one or more contexts in the **cliconfig** file.

```
ccictl config get-contexts [(-o|--output=)name)]
```

## Examples

```
# List all contexts in the cliconfig file.
ccictl config get-contexts

# Describe a context in the cliconfig file.
ccictl config get-contexts my-context
```

  

## Options

-h, --help

Help information for get-contexts

--no-headers

When the default or custom column output format is used, do not print the headers (which are printed by default).

-o, --output string

Output format. The value can be **name**.

**The following cctl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.3 cctl config set-context

## Scenario

Set a context entry in **cliconfig**.

Specifying a name that already exists will merge new fields on top of existing values for those fields.

```
cctl config set-context [NAME | --current] [--cluster=cluster_nickname] [--user=user_nickname] [--namespace=namespace]
```

## Examples

```
# Set the user field in the gce context entry without affecting other values.
cctl config set-context gce --user=cluster-admin
```

## Options

--cluster string

Cluster for the context entry in **cliconfig**.

--current

Modify the current context.

-h, --help

Help information for set-context

--namespace string

Namespace for the context entry in **cliconfig**.

--user string

User for the context entry in **cliconfig**.

**The following cctl options can also be used in subcommands:**

**Parent command options**

  

## 1.3.16.4 ccictl config delete-context

### Scenario

Delete a specified context from **cliconfig**.

```
ccictl config delete-context NAME
```

### Examples

```
# Delete the context for the example-context cluster.
ccictl config delete-context example-context
```

### Options

```
-h, --help
```

Help information for delete-context

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.5 ccictl config rename-context

### Scenario

Rename the context in the **cliconfig** file.

- **CONTEXT_NAME** indicates the name of the context to be changed.
- **NEW_NAME** indicates the new name to be set.
- Note: If the context to be renamed is the current context, this field will be updated.

```
ccictl config rename-context CONTEXT_NAME NEW_NAME
```

### Examples

```
# Rename the context (from old-name to new-name) in the cliconfig file.
ccictl config rename-context old-name new-name
```

### Options

```
-h, --help
```

Help information for rename-context

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.6 ccictl config use-context

### Scenario

Set the current context in the **cliconfig** file.

  

```
ccictl config use-context CONTEXT_NAME
```

## Examples

```
# Use the context named exampleCtx1.
kubectl config use-context exampleCtx1
```

## Options

```
-h, --help
```

Help information for use-context

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.7 ccictl config get-clusters

## Scenario

Display the clusters defined in **cliconfig**.

```
ccictl config get-clusters [flags]
```

## Examples

```
# List clusters that ccictl knows about.
ccictl config get-clusters
```

## Options

```
-h, --help
```

Help information for get-clusters

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.8 ccictl config set-cluster

## Scenario

Set a cluster entry in **cliconfig**.

Specifying a name that already exists will merge new fields on top of existing values for those fields.

```
ccictl config set-cluster NAME [--server=server] [--certificate-authority=path/to/certificate/authority] [--insecure-skip-tls-verify=true] [--tls-server-name=example.com]
```

## Examples

```
# Set only the server field of the e2e cluster entry, without affecting other values.
ccictl config set-cluster e2e --server=https://1.2.3.4

# Add the certificate data to the e2e cluster entry.
ccictl config set-cluster e2e --embed-certs --certificate-authority=~/.kubev2/e2e/ca.crt
```

  

```
# Disable the certificate check in the e2e cluster entry.
ccictl config set-cluster e2e --insecure-skip-tls-verify=true

# Set the custom TLS server name for verifying the e2e cluster entry.
ccictl config set-cluster e2e --tls-server-name=my-cluster-name

# Set the proxy URL of the e2e cluster entry.
ccictl config set-cluster e2e --proxy-url=https://1.2.3.4
```

## Options

```
--certificate-authority string
```

Path to the CA file for the cluster entry in **cliconfig**

```
--embed-certs tristate[=true]
```

Embed the certificate of the cluster entry in **cliconfig**.

```
-h, --help
```

Help information for set-cluster

```
--insecure-skip-tls-verify tristate[=true]
```

Set the **insecure-skip-tls-verify** field in the cluster entry in **cliconfig**.

```
--proxy-url string
```

Proxy URL of the cluster entry in **cliconfig**

```
--server string
```

The **server** field of the cluster entry in **cliconfig**

```
--tls-server-name string
```

The **tls-server-name** field of the cluster entry in **cliconfig**

**The following ccictl options can also be used in subcommands:**

**Parent command options**

# 1.3.16.9 ccictl config delete-cluster

## Scenario

Delete a specified cluster from **cliconfig**.

```
ccictl config delete-cluster NAME
```

## Examples

```
# Delete the example-cluster cluster.
ccictl config delete-cluster example-cluster
```

## Options

```
-h, --help
```

Help information for delete-cluster

**The following ccictl options can also be used in subcommands:**

  

**Parent command options**

## 1.3.16.10 ccictl config get-users

### Scenario

Display the users defined in **cliconfig**.

```
ccictl config get-users [flags]
```

### Examples

```
# List users ccictl that knows about.
ccictl config get-users
```

### Options

```
-h, --help
```

Help information for get-users

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.11 ccictl config delete-user

### Scenario

Delete a specified user from **cliconfig**.

```
ccictl config delete-user NAME
```

### Examples

```
# Delete user user1.
ccictl config delete-user user1
```

### Options

```
-h, --help
```

Help information for delete-user

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.12 ccictl config set-credentials

### Scenario

Set a user entry in **cliconfig**.

- Specifying a name that already exists will merge new fields on top of existing values for those fields.

  Client certificate flag: --client-certificate=certfile --client-key=keyfile

  

Bearer token flag: --token=bearer_token

Basic authentication flag: --username=basic_user --password=basic_password

- The bearer token and basic authentication are mutually exclusive (cannot be used at the same time).

```
ccictl config set-credentials NAME [--client-certificate=path/to/certfile] [--client-key=path/to/keyfile] [--token=bearer_token] [--username=basic_user] [--password=basic_password] [--auth-provider=provider_name] [--auth-provider-arg=key=value] [--exec-command=exec_command] [--exec-api-version=exec_api_version] [--exec-arg=arg] [--exec-env=key=value]
```

## Examples

```
# Set only the client-key field in the cluster-admin entry.
ccictl config set-credentials cluster-admin --client-key=~/.kubev2/admin.key

# Set basic identity authentication for the cluster-admin entry.
ccictl config set-credentials cluster-admin --username=admin --password=uXFGweU9l35qcif

# Embed the client certificate data in the cluster-admin entry.
ccictl config set-credentials cluster-admin --client-certificate=~/.kubev2/admin.crt --embed-certs=true

# Enable the IAM identity authentication provider for the cluster-admin entry.
ccictl config set-credentials cluster-admin --auth-provider=iam --auth-provider-arg=iam-endpoint=example.com

# Delete the iam-endpoint configuration value of the IAM identity provider for the cluster-admin entry.
ccictl config set-credentials cluster-admin --auth-provider=iam --auth-provider-arg=iam-endpoint-

# Enable the new exec authentication plug-in for the cluster-admin entry.
ccictl config set-credentials cluster-admin --exec-command=/path/to/the/executable --exec-api-version=client.authentication.k8s.io/v1beta1

# Define the new exec authentication plug-in for the cluster-admin entry.
ccictl config set-credentials cluster-admin --exec-arg=arg1 --exec-arg=arg2

# Create or update the exec authentication plug-in arguments for the cluster-admin entry.
ccictl config set-credentials cluster-admin --exec-env=key1=val1 --exec-env=key2=val2

# Delete the exec authentication plug-in arguments for the cluster-admin entry.
ccictl config set-credentials cluster-admin --exec-env=var-to-remove-
```

## Options

```
--auth-provider string
```

Authentication provider for user entries in **cliconfig**

```
--auth-provider-arg strings
```

Identity authentication provider arguments, in the format of key=value

```
--client-certificate string
```

Path to the client certificate file for the user entry in **cliconfig**

```
--client-key string
```

Path to the client key file for the user entry in **cliconfig**

```
--embed-certs tristate[=true]
```

Embed the client certificate/key for the user entry in **cliconfig**.

```
--exec-api-version string
```

API version of the exec credential plug-in for the user entry in **cliconfig**

```
--exec-arg strings
```

  

New arguments for the exec credential plug-in command for the user entry in **cliconfig**

```
--exec-command string
```

Command for the exec credential plug-in command for the user entry in **cliconfig**

```
--exec-env strings
```

Environment variables for the exec credential plug-in, in the key=value format

```
-h, --help
```

Help information for set-credentials

```
--password string
```

Password for the user entry in **cliconfig**

```
--token string
```

Token for the user entry in **cliconfig**

```
--username string
```

Username for the user entry in **cliconfig**

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.16.13 ccictl config view

## Scenario

Display merged cliconfig settings or a specified cliconfig file.

You can use **--output jsonpath={...}** to extract specific values using the JSONPath expression.

```
ccictl config view [flags]
```

## Examples

```
# Display merged cliconfig settings.
ccictl config view

# Display merged cliconfig settings, raw certificate data, and exposed secrets.
ccictl config view --raw

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

## Options

```
--allow-missing-template-keys     Default: true
```

If the value is **true**, the error in the template is ignored when a field or mapping key is missing in the template. This option applies only to the Golang and JSONPath output formats.

```
--flatten
```

  

Flatten the generated cliconfig file into a self-contained output (useful for creating portable cliconfig files).

```
-h, --help
```

Help information for view

```
--merge tristate[=true]     Default: true
```

Merge the full hierarchy of cliconfig files.

```
--minify
```

Remove all information not used by the current context from the output.

```
-o, --output string     Default: "yaml"
```

Output format. The value options include **json**, **yaml**, **name**, **go-template**, **go-template-file**, **template**, **templatefile**, **jsonpath**, **jsonpath-as-json**, and **jsonpath-file**.

```
--raw string
```

Display raw byte data and sensitive data.

```
--template string
```

Template character string or template file path used when **-o** is set to **go-template** or **go-template-file**. The Golang template format is [http://golang.org/pkg/text/template/#pkg-overview].

**The following ccictl options can also be used in subcommands:**

**Parent command options**

## 1.3.17 ccictl version

### Scenario

Print the client and server version information for the current context.

```
ccictl version [flags]
```

### Examples

```
# Print the client and server versions for the current context.
ccictl version
```

### Options

```
-h, --help
```

Help information about version

```
-o, --output string
```

The value can be **'yaml'** or **'json'**.

**The following ccictl option can also be used in subcommands:**

**Parent command options**

  

# 2 Using Terraform to Manage CCI Resources

## 2.1 Terraform Configuration Guide

### What Is Terraform?

**Terraform** is an infrastructure as code tool that you can use to build, change, and version CCI resources safely and efficiently. For more information, see **Terraform**.

### Installing Terraform

Terraform is distributed as a single binary. Download a Terraform package and decompress it to a directory specified by the **PATH** environment variable of the OS.

**CAUTION**

Terraform supports multiple OSs, such as Linux, macOS, and Windows. You can log in to the **Terraform official website**, download the installation package of the corresponding OS, and add the executable Terraform file to the environment variable of the corresponding OS.

The following uses Linux as an example to describe how to install Terraform.

1. Go to **Terraform official website** and download a Terraform package that matches your OS.
2. Run the following command to decompress the package, grant the execute permission on Terraform, and save the package to the directory specified by **PATH**: *$PATH* indicates the specified directory (for example, **/usr/local/bin**). Replace it with the actual path.

   ```
   unzip terraform_1.xx.x_
   chmod +x ./terraform
   mv ./terraform $PATH
   ```

3. Run the following command in the command-line interface (CLI) to check whether the path is correctly configured:

  

```
terraform -version
```

If the following information is displayed, the configuration is correct and the Terraform can run properly.


## Authentication

Before using Terraform to manage CCI resources, you need to obtain the access key (AK) and secret key (SK) and configure them on Terraform for authentication.

You can configure Terraform using static credentials or environment variables.

Static credentials are simple to use. However, they require AKs and SKs to be stored in configuration files in plaintext, which risks secret leakage. It is recommended that you provide credentials as environment variables.

- Static credentials

```
provider "huaweicloud" {
  region     = "cn-north-4"
  access_key = "my-access-key"
  seccret_key = "my-secret-key"
}
```

**region**: region where the resources are to be created and managed. You can **query Huawei Cloud regions**.

**access_key**: access secret ID (AK). For details, see **Access Keys**.

**secret_key**: secret access key (SK). For details, see **Access Keys**.

- Environment variables

Configure the region, AK, and SK as environment variables.

```
export HW_REGION_NAME="cn-north-4"
export HW_ACCESS_KEY="my-access-key"
export HW_SECRET_KEY="my-secret-key"
```

**HW_REGION_NAME**: region where the resources are to be created and managed. You can **query Huawei Cloud regions**.

**HW_ACCESS_KEY**: access secret ID (AK). For details, see **Access Keys**.

**HW_SECRET_KEY**: secret access key (SK). For details, see **Access Keys**.

## Initializing the Working Directory

1. Create a working directory.

```
mkdir test
```

2. Create the **versions.tf** file in the working directory and specify Huawei Cloud as the registry source and the version of Huawei Cloud Provider. The file content is as follows:

```
terraform {
  required_providers {
    huaweicloud = {
      source = "huaweicloud/huaweicloud"
      version = ">=1.xx.xx" # Version of the provider to be loaded. You can use `=` to specify the version or use `>=` to specify the minimum provider version. The latest version is preferentially loaded.
    }
  }
}
```

  

> **NOTE**
>
> For details about how to query the version, see **Huawei Cloud Provider**.

3. Create the **main.tf** file and configure the Huawei Cloud provider. The file content is as follows:

```
# Configure the HuaweiCloud Provider
provider "huaweicloud" {
  region      = "cn-north-4"
  access_key  = "my-access-key"
  secret_key  = "my-secret-key"
}
```

This is an example configuration (including the AK/SK for authentication) of Huawei Cloud provider. For details on how to configure these parameters, see **Authentication**. If you provide credentials using environment variables, omit this part.

4. Run the following command to perform initialization:

```
terraform init
```

If the following command output is displayed, Huawei Cloud Provider has been downloaded and installed when you run this command for the first time:

```
[root@ecs-33d5-4f10 test]# terraform init
Initializing the backend...
Initializing provider plugins...
- Finding huaweicloud/huaweicloud versions matching ">= 1.20.0"...
- Installing huaweicloud/huaweicloud v1.76.5...
- Installed huaweicloud/huaweicloud v1.76.5 (self-signed, key ID 4FFE1736199213B8)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://developer.hashicorp.com/terraform/cli/plugins/signing
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

If you need to modify the configuration in the **versions.tf** or **main.tf** file, run the following command to upgrade the file:

```
terraform init -upgrade
```

# 2.2 CCI Resources That Can Be Managed by Terraform

**Table 2-1** CCI resources that can be managed by Terraform

| | create | update | get |
|---|---|---|---|
| Namespace | √ | × | √ |
| Network | √ | √ | √ |
| Deployment | √ | √ | √ |
| Pod | √ | √ | √ |
| Service | √ | √ | √ |

  

| | create | update | get |
|---|---|---|---|
| ConfigMap | √ | √ | √ |
| Secret | √ | √ | √ |
| HPA | √ | √ | √ |
| PVC | √ | √ | √ |
| PV | √ | √ | √ |
| PoolBinding | √ | × | × |
| StorageClass | × | × | √ |

NOTE

- For more information about Terraform, see **terraform-provider-huaweicloud**.
- For more information about Terraform syntax, see **Huawei Cloud Terraform Documentation**.

  