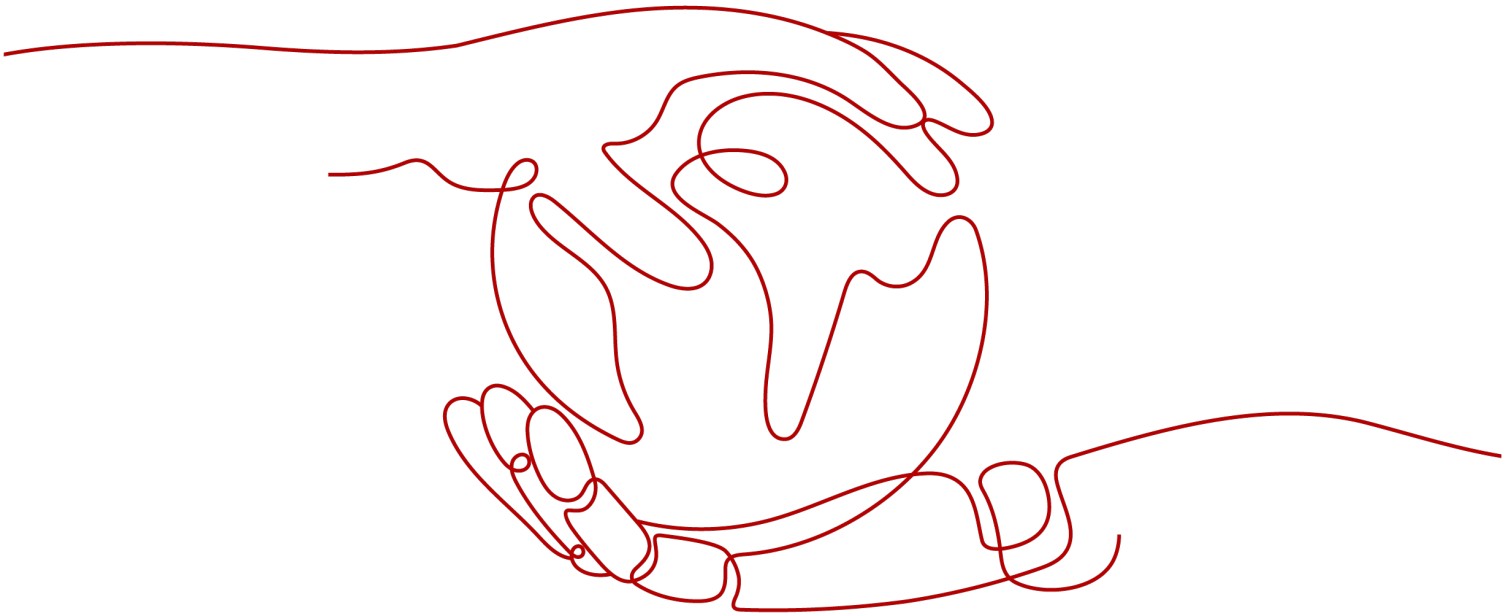
Cloud Container Engine

SDK Reference

Issue 01

Date 2024-10-29



HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of 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.

Contents

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

1 Using client-go to Access CCI

This section describes how to use cci-iam-authenticator (the CCI authentication tool) and Kubernetes client-go to call the APIs.

Installing cci-iam-authenticator

Download, install, and set cci-iam-authenticator by referring to **Using Native kubectl (Recommended)**.

Installing Kubernetes client-go

For details, see **Installing client-go**.

NOTE

- The Kubernetes version corresponding to CCI open APIs is 1.19. According to **Compatibility: client-go <-> Kubernetes clusters**, the recommended SDK version is k8s.io/client-go@kubernetes-1.19.0.
- The clusters used in CCI are shared clusters. All namespaces for the clusters and resources in these namespaces cannot be watched. Only resources in the specified namespace can be watched.

Using the Go SDK

NOTE

The examples have been tested for the following versions:

1. k8s.io/client-go@kubernetes-1.15.0
2. k8s.io/client-go@kubernetes-1.16.0
3. k8s.io/client-go@kubernetes-1.17.0
4. k8s.io/client-go@kubernetes-1.18.0
5. k8s.io/client-go@kubernetes-1.19.0
6. k8s.io/client-go@kubernetes-1.20.0

Using the username and password for authentication:

```
import (
    "fmt"

    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/tools/clientcmd"
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    "k8s.io/client-go/tools/clientcmd/api"
)

const (
    apiVersion = "client.authentication.k8s.io/v1beta1"

        // CCI endpoint: https://developer.huaweicloud.com/intl/en-us/endpoint
    cciEndpoint = "<For example, https://cci.cn-north-4.myhuaweicloud.com indicates CN North-Beijing4.>"

        // IAM endpoint: https://developer.huaweicloud.com/intl/en-us/endpoint
    iamEndpoint = "<For example, https://iam.cn-north-4.myhuaweicloud.com for CN North-Beijing4>"

        // Regions and endpoint: https://developer.huaweicloud.com/intl/en-us/endpoint
        projectName = "<Example: cn-north-4 indicates CN North-Beijing4.>"
)

// userName, domainName, and password indicate the username, account name, and password used for authentication. Keep the username, account name, and password secure.
var userName, domainName, password string

// NewClient indicates creating a Clientset using the username/password.
func NewClient() (*kubernetes.Clientset, error) {
    config, err := clientcmd.BuildConfigFromFlags(cciEndpoint, "")
    if err != nil {
        return nil, err
    }
    var optionArgs []string
    optionArgs = append(optionArgs, fmt.Sprintf("--iam-endpoint=%s", iamEndpoint))
    optionArgs = append(optionArgs, fmt.Sprintf("--project-name=%s", projectName))
    optionArgs = append(optionArgs, fmt.Sprintf("--token-only=false"))
    optionArgs = append(optionArgs, fmt.Sprintf("--domain-name=%s", domainName))
    optionArgs = append(optionArgs, fmt.Sprintf("--user-name=%s", userName))
    optionArgs = append(optionArgs, fmt.Sprintf("--password=%s", password))

    config.ExecProvider = &api.ExecConfig{
        Command:    "cci-iam-authenticator",
        APIVersion: apiVersion,
        Args:       append([]string{"token"}, optionArgs...),
        Env:        make([]api.ExecEnvVar, 0),
    }
    return kubernetes.NewForConfig(config)
}
```

Using an AK/SK for authentication:

```
import (
    "fmt"
    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/tools/clientcmd"
    "k8s.io/client-go/tools/clientcmd/api"
)

const (
    apiVersion = "client.authentication.k8s.io/v1beta1"

        // CCI endpoint: https://developer.huaweicloud.com/intl/en-us/endpoint
    cciEndpoint = "<For example, https://cci.cn-north-4.myhuaweicloud.com indicates CN North-Beijing4.>"

        // IAM endpoint: https://developer.huaweicloud.com/intl/en-us/endpoint
    iamEndpoint = "<For example, https://iam.cn-north-4.myhuaweicloud.com for CN North-Beijing4>"

        // Regions and endpoint: https://developer.huaweicloud.com/intl/en-us/endpoint
    projectName = "<Example: cn-north-4 indicates CN North-Beijing4.>"
)

// Obtaining an AK/SK. Refer to https://support.huaweicloud.com/intl/en-us/devg-cci/cci_kubectl_01.html#cci_kubectl_01__section17023744719.
// ak and sk indicate the user's AK and SK used for authentication. The AK and SK need to be transferred by the user. Exercise caution when using them.
var ak, sk string
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
// NewClient indicates creating a Clientset through AK/SK authentication.
func NewClient() (*kubernetes.Clientset, error) {
    config, err := clientcmd.BuildConfigFromFlags(cciEndpoint, "")
    if err != nil {
        return nil, err
    }
    var optionArgs []string
    optionArgs = append(optionArgs, fmt.Sprintf("--iam-endpoint=%s", iamEndpoint))
    optionArgs = append(optionArgs, fmt.Sprintf("--project-name=%s", projectName))
    optionArgs = append(optionArgs, fmt.Sprintf("--token-only=false"))
    optionArgs = append(optionArgs, fmt.Sprintf("--ak=%s", ak))
    optionArgs = append(optionArgs, fmt.Sprintf("--sk=%s", sk))
    config.ExecProvider = &api.ExecConfig{
        Command:    "cci-iam-authenticator",
        APIVersion: apiVersion,
        Args:       append([]string{"token"}, optionArgs...),
        Env:        make([]api.ExecEnvVar, 0),
    }
    return kubernetes.NewForConfig(config)
}
```

The following example shows how to generate the kubeconfig file for authentication configuration. For details, see the **generate-kubeconfig** subcommand in **cci-iam-authenticator Usage Reference**.

```
import (
    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/tools/clientcmd"
)

// NewClient indicates creating a Clientset using the kubeconfig file.
// The kubeconfig file must contain authentication information. For details, see "cci-iam-authenticator
Usage Reference" at https://support.huaweicloud.com/intl/en-us/devg-cci/cci_kubectl_03.html.
func NewClient() (*kubernetes.Clientset, error) {
    config, err := clientcmd.BuildConfigFromFlags("", "/path/to/kubeconfig")
    if err != nil {
        return nil, err
    }
    return kubernetes.NewForConfig(config)
}
```

FAQ

Question: In the preceding examples, is the request result code 401 returned?

Answer: Generally, if the password or AK/SK is correctly configured, a mechanism provided by client-go for periodically calling cci-iam-authenticator to update the token can ensure that the token does not expire (the validity period of the token is 24 hours) and avoid returning code 401. For details, see **https://github.com/kubernetes/client-go/blob/master/plugin/pkg/client/auth/exec/exec.go**.

However, if the account permissions are changed, the token may also become invalid and the code 401 may be returned.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

2 Using client-go to Access the CRD Resource Network of CCI

Initializing a Project

Create a project named examples.com/cci-examples.

NOTE

The project depends on k8s.io/client-go and k8s.io/code-generator of the following versions (for reference only):

- k8s.io/client-go@kubernetes-1.15.0, k8s.io/code-generator@kubernetes-1.15.0
- k8s.io/client-go@kubernetes-1.16.0, k8s.io/code-generator@kubernetes-1.16.0
- k8s.io/client-go@kubernetes-1.17.0, k8s.io/code-generator@kubernetes-1.17.0
- k8s.io/client-go@kubernetes-1.18.0, k8s.io/code-generator@kubernetes-1.18.0
- k8s.io/client-go@kubernetes-1.19.0, k8s.io/code-generator@kubernetes-1.19.0
- k8s.io/client-go@kubernetes-1.20.0, k8s.io/code-generator@kubernetes-1.20.0

```
mkdir -p examples.com/cci-examples
cd examples.com/cci-examples/
go mod init examples.com/cci-examples
go get k8s.io/client-go@kubernetes-1.15.0
go get k8s.io/code-generator@kubernetes-1.15.0
```

Defining the CRD Resource Network

Create the **pkg/apis/networking.cci.io/v1beta1** folder, where **networking.cci.io** indicates the group of the CRD resource and **v1beta1** indicates the version of the CRD resource.

```
mkdir -p pkg/apis/networking.cci.io/v1beta1
```

Create the following files in the new folder:

doc.go

```
// +k8s:deepcopy-gen=package
// +groupName=networking.cci.io
// +groupGoName=NetworkingCCI
package v1beta1
```

types.go

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
package v1beta1

import (
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)

// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

// NetworkList is a list of network resource in container.
type NetworkList struct {
    metav1.TypeMeta `json:",inline"`
    // Standard list metadata.
    // More info: https://git.k8s.io/community/contributors/devel/api-conventions.md#metadata
    // +optional
    metav1.ListMeta `json:"metadata,omitempty" protobuf:"bytes,1,opt,name=metadata"`
    Items           []Network `json:"items" protobuf:"bytes,2,rep,name=items"`
}

// +genclient
// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object

// Network is a network resource in container.
type Network struct {
    metav1.TypeMeta `json:",inline"`
    // +optional
    metav1.ObjectMeta `json:"metadata,omitempty" protobuf:"bytes,1,opt,name=metadata"`
    // Spec defines the attributes on a network
    // +optional
    Spec NetworkSpec `json:"spec,omitempty" protobuf:"bytes,2,opt,name=spec"`
    // Status describes the network status
    // +optional
    Status NetworkStatus `json:"status,omitempty" protobuf:"bytes,3,opt,name=status"`
}

// NetworkSpec describes the attributes on a network resource.
type NetworkSpec struct {
    // network type
    NetworkType string `json:"networkType,omitempty" protobuf:"bytes,5,opt,name=networkType"`
    // ID of the VPC to attach
    AttachedVPC string `json:"attachedVPC,omitempty" protobuf:"bytes,4,opt,name=attachedVPC"`
    // network ID
    NetworkID string `json:"networkID,omitempty" protobuf:"bytes,7,opt,name=networkID"`
    // Subnet ID
    SubnetID string `json:"subnetID,omitempty" protobuf:"bytes,8,opt,name=subnetID"`
    // available zone
    AvailableZone string `json:"availableZone,omitempty" protobuf:"bytes,9,opt,name=availableZone"`
    // The CIDR of the network
    CIDR string `json:"cidr,omitempty" protobuf:"bytes,3,opt,name=cidr"`
}

// NetworkStatus describes the status of a network
type NetworkStatus struct {
    // State describes the network state
    // +optional
    State string `json:"state" protobuf:"bytes,1,opt,name=state"`
    // Message describes why network is in current state
    // +optional
    Message string `json:"message,omitempty" protobuf:"bytes,2,opt,name=message"`
}

const (
    // NetworkInitializing means the network is initializing
    NetworkInitializing = "Initializing"
    // NetworkPending means the network is processing
    NetworkPending = "Pending"
    // NetworkActive means the network is available
    NetworkActive = "Active"
    // NetworkFailed means the network is not available
    NetworkFailed = "Failed"
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    // NetworkTerminating means the network is undergoing graceful termination
    NetworkTerminating = "Terminating"
)
```

register.go

```
package v1beta1

import (
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
    "k8s.io/apimachinery/pkg/runtime"
    "k8s.io/apimachinery/pkg/runtime/schema"
)

// GroupName is the group name use in this package
const GroupName = "networking.cci.io"

// SchemeGroupVersion is group version used to register these objects
var SchemeGroupVersion = schema.GroupVersion{Group: GroupName, Version: "v1beta1"}

// Resource takes an unqualified resource and returns a Group qualified GroupResource
func Resource(resource string) schema.GroupResource {
    return SchemeGroupVersion.WithResource(resource).GroupResource()
}

var (
    // localSchemeBuilder and AddToScheme will stay in k8s.io/kubernetes.
    SchemeBuilder      = runtime.NewSchemeBuilder(addKnownTypes)
    localSchemeBuilder = &SchemeBuilder
    AddToScheme        = localSchemeBuilder.AddToScheme
)

// Adds the list of known types to the given scheme.
func addKnownTypes(scheme *runtime.Scheme) error {
    scheme.AddKnownTypes(SchemeGroupVersion,
        &Network{},
        &NetworkList{},
    )
    // Add the watch version that applies
    metav1.AddToGroupVersion(scheme, SchemeGroupVersion)
    return nil
}
```

Creating a Build Script

Create the **hack** folder for storing the build script and dependency files.

```
mkdir hack
```

Create the following files in the **hack** folder:

tools.go, which is created to depend on code-generator.

```
// +build tools

/*
Copyright 2019 The Kubernetes Authors.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
*/

// This package imports things required by build scripts, to force `go mod` to see them as dependencies
package tools

import _ "k8s.io/code-generator"
```

update-codegen.sh

```
#!/usr/bin/env bash

# Copyright 2017 The Kubernetes Authors.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

set -o errexit
set -o nounset
set -o pipefail

SCRIPT_ROOT=$(dirname "${BASH_SOURCE[0]}")/..
CODEGEN_PKG=${CODEGEN_PKG:-$(cd "${SCRIPT_ROOT}"; ls -d -1 ./vendor/k8s.io/code-generator 2>/dev/
null || echo ../code-generator)}

# generate the code with:
# --output-base    because this script should also be able to run inside the vendor dir of
#                  k8s.io/kubernetes. The output-base is needed for the generators to output into the vendor dir
#                  instead of the $GOPATH directly. For normal projects this can be dropped.
# generators deepcopy,client,informer,lister
export CLIENTSET_PKG_NAME=networking.cci.io
export CLIENTSET_NAME_VERSIONED=v1beta1
"${CODEGEN_PKG}"/generate-groups.sh "deepcopy,client" \
  examples.com/cci-examples/pkg/client examples.com/cci-examples/pkg/apis \
  networking.cci.io:v1beta1 \
  --output-base "$(dirname "${BASH_SOURCE[0]}")/../../.." \
  --go-header-file "${SCRIPT_ROOT}"/hack/boilerplate.go.txt

# To use your own boilerplate text append:
#   --go-header-file "${SCRIPT_ROOT}"/hack/custom-boilerplate.go.txt
```

Header of the file built by **boilerplate.go.txt**. You can customize it.

```
/*
Copyright The Kubernetes Authors.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/
```

All preparations are complete. The file structure of the directory is as follows:

```
├── go.mod
├── go.sum
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
├── hack
│   ├── boilerplate.go.txt
│   ├── tools.go
│   └── update-codegen.sh
└── pkg
    └── apis
        └── networking.cci.io
            └── v1beta1
                ├── doc.go
                ├── register.go
                └── types.go

5 directories, 8 files
```

Executing the Name Generation Code

Run the following commands in the Linux environment:

```
# Generate the vendor folder.
go mod vendor
# Run the build script.
chmod 755 hack/update-codegen.sh
# hack/update-codegen.sh will execute vendor/k8s.io/code-generator/generate-groups.sh.
chmod 755 vendor/k8s.io/code-generator/generate-groups.sh
./hack/update-codegen.sh
```

After the commands are executed successfully, code is generated. The directory structure is as follows:

```
├── go.mod
├── go.sum
├── hack
│   ├── boilerplate.go.txt
│   ├── tools.go
│   └── update-codegen.sh
└── pkg
    ├── apis
    │   └── networking.cci.io
    │       └── v1beta1
    │           ├── doc.go
    │           ├── register.go
    │           ├── types.go
    │           └── zz_generated.deepcopy.go
    └── client
        └── networking.cci.io
            └── v1beta1
                ├── clientset.go
                ├── doc.go
                ├── fake
                │   ├── clientset_generated.go
                │   ├── doc.go
                │   └── register.go
                ├── scheme
                │   ├── doc.go
                │   └── register.go
                └── typed
                    └── networking.cci.io
                        └── v1beta1
                            ├── doc.go
                            ├── fake
                            │   ├── doc.go
                            │   ├── fake_network.go
                            │   └── fake_networking.cci.io_client.go
                            ├── generated_expansion.go
                            ├── network.go
                            └── networking.cci.io_client.go
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

Example Code: Creating a Network

NOTE

The example has been tested for the following versions:

k8s.io/client-go@kubernetes-1.15.0

k8s.io/code-generator@kubernetes-1.15.0

```
import (
    "time"

    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
    "k8s.io/apimachinery/pkg/util/wait"
    "k8s.io/client-go/tools/clientcmd"

    "examples.com/cci-examples/pkg/apis/networking.cci.io/v1beta1"
    clientset "examples.com/cci-examples/pkg/client/networking.cci.io/v1beta1"
)

const (
    name      = "test-k8s-client-namespace-cn-north-1-default-network"
    namespace = "test-k8s-client-namespace"
)

// CreateNetwork indicates creating a network and waiting until its status changes to Active.
// "Namespace and Network": https://support.huaweicloud.com/intl/en-us/devg-cci/cci_05_0023.html.
// API Reference: https://support.huaweicloud.com/intl/en-us/api-cci/
createNetworkingCciIoV1beta1NamespacedNetwork.html
func CreateNetwork() (*v1beta1.Network, error) {
    config, _ := clientcmd.BuildConfigFromFlags("", "{path to kubeconfig}")
    cs, err := clientset.NewForConfig(config)
    if err != nil {
        return nil, err
    }

    projectId := "<Account ID, which can be obtained from My Credentials>"
    domainId := "<Project ID, which can be obtained from My Credentials>"
    securityGroupID := "<Security group ID, which can be obtained from the Security Groups page of the
Network Console>"
    availableZone := "<AZ name, for example, cn-north-1a, cn-north-4a, or cn-east-3a>"
    vpcID := "VPC ID, which can be obtained from the VPC console"
    cidr := "<Subnet CIDR block, for example, 192.168.128.0/18>"
    networkID := "<Subnet network ID, which can be obtained from the Subnets page of the VPC console>"
    subnetID := "<Subnet ID, which can be obtained from the Subnets page of the VPC console>"

    network := &v1beta1.Network{
        ObjectMeta: metav1.ObjectMeta{
            Annotations: map[string]string{
                "network.alpha.kubernetes.io/default-security-group": securityGroupID,
                "network.alpha.kubernetes.io/domain-id":              domainId,
                "network.alpha.kubernetes.io/project-id":             projectId,
            },
            Name: name,
        },
        Spec: v1beta1.NetworkSpec{
            AvailableZone: availableZone,
            CIDR:          cidr,
            AttachedVPC:   vpcID,
            NetworkID:     networkID,
            NetworkType:   "underlay_neutron",
            SubnetID:      subnetID,
        },
    }
    network, err = cs.NetworkingCCIV1beta1().Networks(namespace).Create(network)
    if err != nil {
        return nil, err
    }

    // Query the network status and wait until the status changes to Active.
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    err = wait.Poll(time.Second*5, time.Second*30, func() (done bool, err error) {
        network, err = cs.NetworkingCCIV1beta1().Networks(namespace).Get(name, metav1.GetOptions{})
        if err != nil {
            return false, err
        }
        if network.Status.State == v1beta1.NetworkActive {
            return true, nil
        }
        return false, nil
    })
    return network, err
}
```

Reference

Kubernetes Deep Dive: Code Generation for CustomResources

code-generator

sample-controller

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

3 Using the Kubernetes Python SDK to Access CCI

This section describes how to use cci-iam-authenticator (the CCI authentication tool) and kubernetes-client/python to call the APIs.

Installing cci-iam-authenticator

Download, install, and set cci-iam-authenticator by referring to **Using Native kubectl (Recommended)**.

Installing kubernetes-client/python

Download and install kubernetes-client/python by referring to **Installation**.

NOTE

The Kubernetes API version of CCI is Kubernetes 1.19. According to **Compatibility**, the recommended SDK version is client 11.y.z.

Using the Python SDK

Run **generate-kubeconfig** to generate the kubeconfig file. For details, see **cci-iam-authenticator Usage Reference**.

NOTE

- The validity period of a cached token is 24 hours. In the following example, the token is periodically refreshed to prevent token expiration. You can also add the operation of retry upon failure to improve availability.
- Periodically refreshing the token is not applicable after the account permission is changed. For example, if the permissions of a subaccount are changed by the main account, the token obtained before the change becomes invalid and needs to be obtained again.

```
# -*- coding: utf-8 -*-
import logging
import time
import threading

from kubernetes import client, config
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
NAMESPACE = "test-k8s-client-namespace"

logging.basicConfig(
    level=logging.INFO,
    datefmt="%Y-%m-%d %H:%M:%S",
    format="%(asctime)s %(levelname)s %(message)s",
)


def create_namespace():
    flavor = "general-computing"
    pool_size = "10"

    namespace = client.V1Namespace(
        metadata=client.V1ObjectMeta(
            name=NAMESPACE,
            annotations={
                "namespace.kubernetes.io/flavor": flavor,
                "network.cci.io/warm-pool-size": pool_size,
            },
            labels={
                "rbac.authorization.cci.io/enable-k8s-rbac": "false",
            }
        )
    )

    logging.info("start to create namespace %s", NAMESPACE)
    client.CoreV1Api().create_namespace(namespace)
    logging.info("namespace created")


def create_network():
    name = "test-k8s-client-namespace-cn-north-7-default-network"
    project_id = "{project_id}"
    domain_id = "{domain_id}"
    security_group_id = "{security_group_id}"
    available_zone = "{available_zone}"
    vpc_id = "{vpc_id}"
    cidr = "{cidr}"
    network_id = "{network_id}"
    subnet_id = "{subnet_id}"

    body = {
        "apiVersion": "networking.cci.io/v1beta1",
        "kind": "Network",
        "metadata": {
            "annotations": {
                "network.alpha.kubernetes.io/default-security-group": security_group_id,
                "network.alpha.kubernetes.io/domain-id": domain_id,
                "network.alpha.kubernetes.io/project-id": project_id,
            },
            "name": name,
        },
        "spec": {
            "availableZone": available_zone,
            "cidr": cidr,
            "attachedVPC": vpc_id,
            "networkID": network_id,
            "networkType": "underlay_neutron",
            "subnetID": subnet_id,
        }
    }

    api = client.CustomObjectsApi()
    logging.info("start to create network")
    api.create_namespaced_custom_object(
        group="networking.cci.io",
        version="v1beta1",
        namespace=NAMESPACE,
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        plural="networks",
        body=body,
    )
    logging.info("network created")


def create_deployment():
    app = "test-k8s-client-deployment"
    image = "library/nginx:stable-alpine-perl"

    body = client.V1Deployment(
        api_version="apps/v1",
        kind="Deployment",
        metadata=client.V1ObjectMeta(name=app),
        spec=client.V1DeploymentSpec(
            replicas=2,
            selector={"matchLabels": {"app": app}},
            template=client.V1PodTemplateSpec(
                metadata=client.V1ObjectMeta(labels={"app": app}),
                spec=client.V1PodSpec(
                    containers=[
                        client.V1Container(
                            name="container-0",
                            image=image,
                            resources=client.V1ResourceRequirements(
                                requests={"cpu": "500m", "memory": "1024Mi"},
                                limits={"cpu": "500m", "memory": "1024Mi"},
                            ),
                        )
                    ],
                    image_pull_secrets=[
                        client.V1LocalObjectReference(name="imagepull-secret")],
                    priority=0),
            ),
        )
    )
    logging.info("start to create deployment %s/%s", NAMESPACE, app)
    client.AppsV1Api().create_namespaced_deployment(NAMESPACE, body)
    logging.info("deployment created")


def get_deployment():
    app = "test-k8s-client-deployment"
    resp = client.AppsV1Api().read_namespaced_deployment(app, NAMESPACE)
    logging.info("deployment detail: %s", resp)


def delete_deployment():
    app = "test-k8s-client-deployment"
    logging.info("start to delete deployment")
    client.AppsV1Api().delete_namespaced_deployment(app, NAMESPACE)
    logging.info("deployment deleted")


def delete_namespace():
    logging.info("start to delete namespace: %s", NAMESPACE)
    client.CoreV1Api().delete_namespace(NAMESPACE)


def main():
    # Configs can be set in Configuration class directly or using helper
    # utility. If no argument provided, the config will be loaded from
    # default location.
    path = '{path to kubeconfig}'
    config.load_kube_config(path)

    # The validity period of a token is 24 hours. Therefore, set a scheduled task for obtaining a new token
every 12 hours.
    # Note: If the account permissions are changed (for example, the permissions of a subaccount are
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
changed by the main account), the token obtained before the change becomes invalid and needs to be
obtained again.
    # You can also add the operation of retry upon failure to improve availability.
    def _refresh():
        while True:
            time.sleep(12 * 3600)
            try:
                config.load_kube_config(path)
            except Exception as e:
                print("load_kube_config error: %s" % e)
    t = threading.Thread(target=_refresh)
    t.daemon = True
    t.start()

    create_namespace()
    create_network()
    # wait for namespace and network to be active
    logging.info("waiting for namespace and network to be active")
    time.sleep(30)
    create_deployment()
    get_deployment()
    delete_deployment()
    delete_namespace()


if __name__ == '__main__':
    main()
```

FAQ

Question: Is the preceding example applicable to other versions of kubernetes-client/python?

Answer: The preceding example has been tested in python3.7.4 for the following versions:

1. 9.0.1
2. 10.1.0
3. 11.0.0
4. 12.0.1
5. 17.17.0
6. 18.17.0a1
7. 19.15.0

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

4 Using the Kubernetes Java SDK to Access CCI

This section describes how to use cci-iam-authenticator (the CCI authentication tool) and **kubernetes-client/java** to call the APIs.

Installing cci-iam-authenticator

Download, install, and set cci-iam-authenticator by referring to **Using Native kubectl (Recommended)**.

Installing kubernetes-client/java

For details, see **Installation**.

NOTE

The Kubernetes version corresponding to the CCI open APIs is 1.19. According to **Versioning and Compatibility**, the recommended SDK version is 11.0.2 or later.

If you want to use GenericKubernetesClient (see **Code Examples**), the version must be 9.0.0 or later.

Using the Java SDK

NOTE

The examples have been tested for the following versions:

1. 11.0.2

Add the following dependencies to the project's POM file:

```
<dependency>
   <groupId>io.kubernetes</groupId>
   <artifactId>client-java</artifactId>
   <version>11.0.2</version>
</dependency>
```

The following example shows how to use the kubeconfig file to create ApiClient. For details, see the **generate-kubeconfig** subcommand in **cci-iam-authenticator Usage Reference**.

```
public class CommonCases {
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
    // ...

    public static void main(String[] args) throws IOException, ApiException, InterruptedException {

        // file path to your KubeConfig
        String kubeConfigPath = "<path to kubeconfig>";

        // loading the out-of-cluster config, a kubeconfig from file-system
        File file = new File(kubeConfigPath);
        KubeConfig config = KubeConfig.loadKubeConfig(new FileReader(file));
        config.setFile(file);
        ApiClient client = buildClient(config).build();

        // ...

    }

    public static ClientBuilder buildClient(KubeConfig config) throws IOException {
        final ClientBuilder builder = new ClientBuilder();

        String server = config.getServer();
        if (!server.contains("://")) {
            if (server.contains(":443")) {
                server = "https://" + server;
            } else {
                server = "http://" + server;
            }
        }

        builder.setVerifyingSsl(config.verifySSL());

        builder.setBasePath(server);
        builder.setAuthentication(new CCIKubeconfigAuthentication(config));
        return builder;
    }
}
```

CCIKubeconfigAuthentication.java

```
package com.huawei.demos;

import io.kubernetes.client.openapi.ApiClient;
import io.kubernetes.client.util.KubeConfig;
import io.kubernetes.client.util.credentials.Authentication;
import okhttp3.Interceptor;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.time.Instant;


/**
 * Uses a {@link KubeConfig} to configure {@link ApiClient} authentication to the CCI Kubernetes API.
 *
 * <p> Only try to use AccessTokenAuthentication mechanisms, which is enough for CCI.
 */
public class CCIKubeconfigAuthentication implements Authentication, Interceptor {

    private static final Logger log = LoggerFactory.getLogger(CCIKubeconfigAuthentication.class);
    private final KubeConfig config;
    private String token;
    private Instant expiry;

    public CCIKubeconfigAuthentication(final KubeConfig config) throws IOException {
        this.config = config;
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        this.expiry = Instant.MIN;
    }

    private String getToken() {
        // get access token every 600 seconds
        if (Instant.now().isAfter(this.expiry)) {
            log.debug("Token expired, get new one from kubeconfig");
            this.token = config.getAccessToken();
            if (this.token != null) {
                this.expiry = Instant.now().plusSeconds(600);
            }
        }
        return this.token;
    }

    @Override
    public void provide(ApiClient client) {
        OkHttpClient httpClient = client.getHttpClient().newBuilder().addInterceptor(this).build();
        client.setHttpClient(httpClient);
    }

    @Override
    public Response intercept(Interceptor.Chain chain) throws IOException {
        Request request = chain.request();
        Request authRequest;
        authRequest = request.newBuilder().header("Authorization", "Bearer " + getToken()).build();
        return chain.proceed(authRequest);
    }
}
```

Accessing CCI:

```
public class CommonCases {

    // ...

    private static void createNamespace(CoreV1Api api) throws ApiException {
        String enableK8sRbac = "false";
        String flavor = "general-computing";
        String warmPoolSize = "10";

        Map<String, String> labels = new HashMap<>();
        labels.put("rbac.authorization.cci.io/enable-k8s-rbac", enableK8sRbac);
        Map<String, String> annotations = new HashMap<>();
        annotations.put("namespace.kubernetes.io/flavor", flavor);
        annotations.put("network.cci.io/warm-pool-size", warmPoolSize);
        V1Namespace namespace = new V1Namespace()
                .metadata(new V1ObjectMeta().name(NAMESPACE).labels(labels).annotations(annotations));
        LOGGER.info("start to create namespace {}", NAMESPACE);
        api.createNamespace(namespace, null, null, null);
        LOGGER.info("namespace created");
    }

    private static void createNetwork(GenericKubernetesApi<Network, NetworkList> networkApi) throws
ApiException {
        String name = NETWORK;
        String projectID = "<Account ID, which can be obtained from My Credentials>";
        String domainID = "<Project ID, which can be obtained from My Credentials>";
        String securityGroupID = "<Security group ID, which can be obtained from the Security Groups page of
the Network Console>";
        String availableZone = "<AZ name, for example, cn-north-1a, cn-north-4a, or cn-east-3a>";
        String vpcID = "VPC ID, which can be obtained from the VPC console";
        String cidr = "<Subnet CIDR block, for example, 192.168.128.0/18>";
        String networkID = "<Subnet network ID, which can be obtained from the Subnets page of the VPC
console>";
        String subnetID = "<Subnet ID, which can be obtained from the Subnets page of the VPC console>";
        String networkType = "underlay_neutron";

        Map<String, String> annotations = new HashMap<>();
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
        annotations.put("network.alpha.kubernetes.io/default-security-group", securityGroupID);
        annotations.put("network.alpha.kubernetes.io/domain-id", domainID);
        annotations.put("network.alpha.kubernetes.io/project-id", projectID);

        Network network = new Network()
                .metadata(new
V1ObjectMeta().name(name).namespace(NAMESPACE).annotations(annotations))
                .spec(new NetworkSpec()
                        .availableZone(availableZone)
                        .cidr(cidr)
                        .attachedVPC(vpcID)
                        .networkID(networkID)
                        .networkType(networkType)
                        .subnetID(subnetID));

        LOGGER.info("start to create network {}/{}", NAMESPACE, name);
        networkApi.create(network).throwsApiException();
        LOGGER.info("network created");
    }

    private static void waitNamespaceActive(CoreV1Api api) throws ApiException, InterruptedException {
        for (int i = 0; i < 5; i++) {
            V1Namespace ns = api.readNamespace(NAMESPACE, null, null, null);
            if (ns.getStatus() != null && NAMESPACE_ACTIVE.equals(ns.getStatus().getPhase())) {
                return;
            }
            Thread.sleep(WAIT_ACTIVE_MILLIS);
        }
        throw new IllegalStateException("namespace not active");
    }

    private static void waitNetworkActive(GenericKubernetesApi<Network, NetworkList> networkApi) throws
ApiException, InterruptedException {
        for (int i = 0; i < 5; i++) {
            Network network = networkApi.get(NAMESPACE, NETWORK).throwsApiException().getObject();
            if (network.getStatus() != null && NETWORK_ACTIVE.equals(network.getStatus().getState())) {
                return;
            }
            Thread.sleep(WAIT_ACTIVE_MILLIS);
        }
        throw new IllegalStateException("network not active");
    }

    private static void createDeployment(AppsV1Api api) throws ApiException {
        String app = APP;
        String cpu = "500m";
        String memory = "1024Mi";
        String containerName = "container-0";
        String image = "library/nginx:stable-alpine-perl";

        Map<String, Quantity> limits = new HashMap<>();
        limits.put("cpu", Quantity.fromString(cpu));
        limits.put("memory", Quantity.fromString(memory));
        V1Container container = new V1Container()
                .name(containerName)
                .image(image)
                .resources(new V1ResourceRequirements().limits(limits).requests(limits));

        Map<String, String> labels = new HashMap<>();
        labels.put("app", app);
        V1PodTemplateSpec podTemplateSpec = new V1PodTemplateSpec()
                .spec(new V1PodSpec()
                        .priority(0)
                        .imagePullSecrets(Collections.singletonList(new
V1LocalObjectReference().name("imagepull-secret")))
                        .containers(Collections.singletonList(container)))
                .metadata(new V1ObjectMeta().labels(labels));

        V1Deployment deployment = new V1Deployment()
```

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.

```
                .metadata(new V1ObjectMeta().name(app))
                .spec(new V1DeploymentSpec()
                        .replicas(2)
                        .selector(new V1LabelSelector().matchLabels(labels))
                        .template(podTemplateSpec));
        LOGGER.info("start to create deployment {}/{}", NAMESPACE, APP);
        api.createNamespacedDeployment(NAMESPACE, deployment, null, null, null);
        LOGGER.info("deployment created");
    }

    private static void getDeployment(AppsV1Api api) throws ApiException {
        V1Deployment deployment = api.readNamespacedDeployment(APP, NAMESPACE, null, null, null);
        LOGGER.info("deployment metadata: {}", deployment.getMetadata());
    }

    private static void deleteDeployment(AppsV1Api api) throws ApiException {
        LOGGER.info("start to delete deployment");
        api.deleteNamespacedDeployment(APP, NAMESPACE, null, null, null, null, null, null);
        LOGGER.info("deployment deleted");
    }

    private static void deleteNamespace(CoreV1Api api) throws ApiException {
        LOGGER.info("start to delete namespace: {}", NAMESPACE);
        api.deleteNamespace(NAMESPACE, null, null, null, null, null, null);
        LOGGER.info("namespace deleted");
    }
}
```

FAQ

Question: Is the preceding example applicable to other versions of kubernetes-client/java?

Answer: The preceding example is not applicable to SDKs earlier than 9.0.0, because it uses GenericKubernetesClient, which requires version 9.0.0 or later (see **Code Examples**).

You need to slightly adjust the preceding example code to apply it to the SDKs of different versions.

 Copyright © Huawei Cloud Computing Technologies Co., Ltd.