

# Configuring and using Service Broker Cloud Consumption Interface

Service Broker Cloud Consumption Interface

December 2022

vRealize Automation SaaS

VMware vRealize Automation Cloud

You can find the most up-to-date technical documentation on the VMware website at:

<https://docs.vmware.com/>

**VMware, Inc.**  
3401 Hillview Ave.  
Palo Alto, CA 94304  
[www.vmware.com](http://www.vmware.com)

Copyright © 2022 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

# Contents

- 1** Configuring and working with the Service Broker Cloud Consumption Interface 4
- 2** Administrator configuration of the Service Broker Cloud Consumption Interface 5
  - Create an additional cloud account for the Cloud Consumption Interface 10
- 3** Using the Service Broker Cloud Consumption Interface 19
  - Create a Supervisor namespace with the Service Broker Cloud Consumption Interface 20
  - Use Service Broker Cloud Consumption Interface to create a deployable workload 22
    - Working with Virtual Machines in the Cloud Consumption Interface 24
    - Working with the Tanzu Kubernetes Grid service in the Cloud Consumption Interface 24
    - Working with the Volumes Service in Cloud Consumption Interface 25
- 4** Using the Cloud Consumption Interface Command Line Interface 27
  - Cloud Consumption Interface Kubernetes API Reference 27
  - Example of manually enabling Service Broker Cloud Consumption Interface for users 35

# Configuring and working with the Service Broker Cloud Consumption Interface

# 1

The Service Broker Cloud Consumption Interface (CCI) enables DevOps users to work with vSphere with Tanzu Supervisor namespaces and associated services to deliver simple, self-service consumption of Kubernetes and cloud infrastructure resources for VMware Cloud environments.

CCI is a flexible vRealize Automation Cloud-based feature set that enables consumers to rapidly create and consume cloud native and traditional IT infrastructure resources. It aggregates and exposes Supervisor DevOps services from across vSphere infrastructure into a common endpoint.

A vSphere+ cloud administrator can activate the Developer Experience service in vSphere+ to grant users access to the Cloud Consumption Interface (CCI) in Service Broker. Designated users receive access to CCI according to the configuration defined by their administrator. Administrators can also use the CCI command line interface to set up infrastructure, such as regions, and to configure user access.

When activated, CCI enables users to create Supervisor namespaces and to consume DevOps services such as the virtual machine service, Tanzu Kubernetes Grid Service, volume service and network service. CCI uses wizards that guide you through the process of using services to create virtual machines, and other resources. When you work with services, CCI automatically generates Kubernetes YAML that you can use to create IaaS resources using the command line or download and save to a suitable repository.

# Administrator configuration of the Service Broker Cloud Consumption Interface

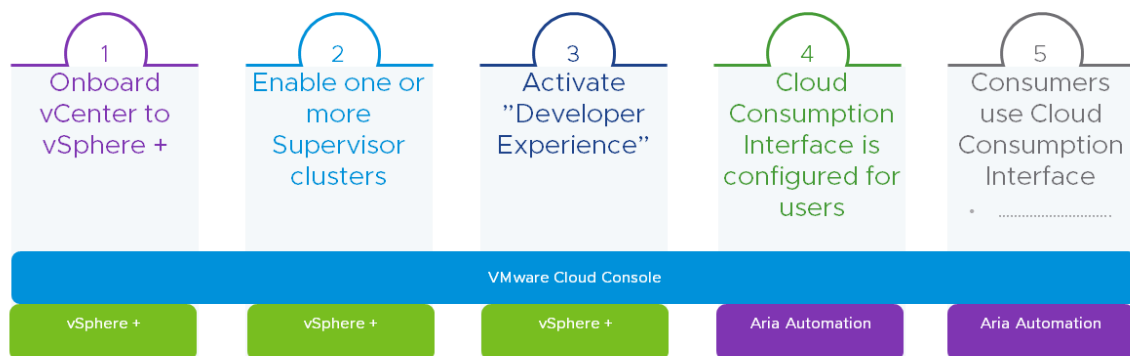
2

A vSphere+ administrator must configure access for users to work with the vRealize Automation Cloud Service Broker Cloud Consumption Interface (CCI). CCI provides user access to Supervisors that enable them to create Supervisor namespaces and any associated IaaS and partner DevOps services.

## Cloud Consumption Interface Enablement Overview

The following diagram outlines the complete high-level workflow required to set up access to CCI for vSphere+ users. The bullets following the diagram provide more details and, where appropriate, links to applicable documentation.

### Enabling Cloud Consumption Interface using Developer Experience



- Step 1 - Onboard vCenter to vSphere+: A vSphere administrator must establish a connection between a vCenter instance that contains clusters to which you want to provide access and vSphere+. See [Connect Your vCenter Server to a vCenter Cloud Gateway](#) for more information.

- Step 2 - Enable one or more Supervisor clusters: A vSphere administrator must configure applicable Supervisor clusters. See [Configuring and Managing a Supervisor Cluster](#) for more information.
- Step 3 - Enable Developer Experience: When an administrator clicks Finish in vSphere+, it initiates a workflow that sets up appropriate projects and related infrastructure for users to work with either traditional or Supervisor clusters or both, depending on the specific configuration. See "Using vRealize Automation Cloud" in Using and Managing vSphere+ for more information.
- Step 4 - Cloud Consumption Interface is configured for users: vRealize Automation Cloud applications are available to users in CSP. This chapter describes the infrastructure that is created and how users can access it.
- Step 5 - Consumers use Cloud Consumption Interface: Users can access designated projects and the associated namespace classes. They can use these namespace classes as templates for new namespaces. CCI provides a customized kubectl for command line control over admin and user tasks. See the following chapters of this document for more information.

## Additional Configuration Notes

Cloud Consumption Interface single sign-on requires customers to use a local Active Directory that has been federated to VMware Cloud and CSP as part of the vSphere+ installation process. Federating the Active Directory domain allows support for maintaining user identity during Supervisor Namespace and IaaS service UI or command line operations, as well as any vSphere+ operations via vSphere+ services.

See [Set up Enterprise Federation with VMware Cloud Services](#) in the vSphere+ *Getting Started with vSphere+* documentation for more information about how vSphere+ uses federation.

Users access CCI services and resources via a dedicated Kubernetes proxy. To maintain user identity as the proxy accesses the vCenter Kubernetes APIs, CCI uses a single sign-on flow similar to that used by vSphere+.

The Service Broker User role includes the necessary privileges to access the Supervisor namespaces as an SSO user. Only users assigned this role can access services within namespaces created on vSphere+ vCenters.

## vRealize Automation Cloud Free Tier configuration for vSphere+ users

The vSphere+ interface enables vSphere+ administrators to activate the vRealize Automation Free Tier feature that enables users to manage vSphere Kubernetes resources using vRealize Automation Cloud functionality, such as the Service Broker with CCI enabled. After a vSphere+ administrative user sets up a connection to a suitable vSphere instance, the interface displays the vSphere Supervisor and traditional clusters available to the user. It also displays a list of users.

To begin configuration, a vSphere administrator clicks the Developer Experience tile under Integrated Services in VMware Cloud Services. The page will display the Supervisor and traditional clusters available on the vSphere instance. Below, the page shows two lists of users and groups under two headings, one for Supervisor clusters and one for traditional clusters. The administrator can select users and groups under each heading to enable access to the clusters for selected users.

When the administrator clicks **Finish**, a configuration workflow is invoked automatically that configures vRealize Automation Cloud with the specified resources and governance constructs for the selected users and groups. This workflow configures all the needed components in vRealize Automation Free Tier for the discovered Supervisors and traditional cluster based on the provided access. The result is a fully functional version of vRealize Automation Cloud as part of free tier.

The following table describes the infrastructure that is created based on cluster selections by an administrator in vSphere+.

vSphere+ Cluster Selection	Infrastructure created in vRealize Automation Free Tier
Traditional Clusters	<ul style="list-style-type: none"> <li>■ Adds a vCenter cloud account.</li> <li>■ Creates a cloud zone for every datacenter that contains traditional clusters to be onboarded. If there are multiple clusters in a datacenter, they are added to the cloud zone for that data center.</li> <li>■ Creates a new project called <i>Default</i>.</li> <li>■ Adds the cloud zones to the project.</li> <li>■ Adds vSphere + VI administrators as Project administrators.</li> <li>■ Adds vCenter/CSP users as Project Users</li> <li>■ Enables multi cloud automation service for customer ORG.</li> <li>■ Assigns vSphere + VI administrators the vRealize Automation Cloud Assembler and Consumption admin role.</li> <li>■ Assign vCenter/CSP users the vRealize Automation Cloud Consumption user role.</li> </ul>
Supervisor Clusters	<ul style="list-style-type: none"> <li>■ Adds a vCenter cloud account. See the vRealize Automation Cloud documentation for information about cloud accounts.</li> <li>■ Creates ccs system project <i>vmware-system-ccs</i> - one per organization.</li> <li>■ Creates a project called "supervisor" - one per organization. <ul style="list-style-type: none"> <li>■ Adds vSphere + VI admin(s) as Project Administrator.</li> <li>■ Adds vCenter/CSP users as Project Users, using project role binding.</li> <li>■ See the vRealize Automation Cloud documentation for more information about projects.</li> </ul> </li> <li>■ Adds default region called <i>onprem</i> - one per organization. A region is a grouping mechanism for Supervisor namespaces.</li> <li>■ Adds a default supervisor namespace class called <i>basic</i> - one per vCenter account.</li> <li>■ Adds region binding config to tie the region <i>onprem</i> / Project <i>supervisor</i> to Supervisor clusters. Supervisor clusters are selected based on the match expression.</li> <li>■ Add Supervisor namespace class binding to tie Supervisor namespace class named <i>basic</i> to the newly created project named <i>supervisor</i>.</li> <li>■ Adds Supervisor namespace class config to specify the Storage policies, Content Libraries and VM Service parameters for the Supervisor namespace. Supervisor clusters are selected based on the match expression.</li> </ul>

When the automatic configuration workflow completes, users will have access to vRealize Automation Cloud components in VMware Cloud Services.



There are, however, some specific limitations on the number of resources that administrators can create. See [Free Tier Limitations](#) for more information about limits.

There are three vRealize Automation Cloud access scenarios for vRealize Automation Free Tier users, depending on the user project membership and whether the project has access to cloud zones, Kubernetes zones or both. These scenarios are outlined below.

- If project users are members of project that is only configured with Kubernetes zones, when they activate Service Broker, they will see the Supervisor Namespaces node for CCI on the left menu for the Consume tab, but they will not see or have access to the Catalog or Deployments nodes on the Service Broker left menu pane.
- If project users are members of a project that is configured with both cloud zones and Kubernetes zones, when they activate Service Broker, they will see the Catalog and Deployments nodes on the left menu, but they will not see or have access to CCI via the Supervisor Namespaces node.
- If project users are members of a project that is configured with both cloud zones and Kubernetes zones, they have access to the Supervisor Namespaces node and to the Catalog and Deployments nodes on the Service Broker left menu.

## The Cloud Consumption Interface kubectl plug-in

CCI provides a plug-in that adds CCI-specific commands to the standard Kubernetes kubectl. Administrators and users may need to use the CCI kubectl to run CLI commands for administrative configuration, maintenance, and troubleshooting. You can download the CCI kubectl executable from the CCI interface in Service Broker. Administrators can use the CCI kubectl to create cloud accounts, create and configure regions, and Supervisor namespace classes. Users also have some more limited access to the CCI kubectl CLI commands.

Before you can use the CCI kubectl, you must obtain an API token. User-level access requires the standard Organization Member and the Service Broker User role. Administrative access requires the Organization Member role and the Service Broker Administrator role.

In order to access Supervisor namespaces as an SSO user, the API token must be assigned the Service Broker User role and its associated vCenter Supervisor Proxy User permission. Do not unselect the permission if the API token will be used to access Supervisor namespaces via the proxy.

- 1 As a user in CSP, navigate to **My Account > API Tokens > Generate a new API Token with Service Broker User Role**
- 2 Select the OpenID check box and click **GENERATE**.
- 3 Copy and save the API Token, as you will need it later.

Use the following command to log in to the CCI kubectl:

```
kubectl ccs login -s (environment) -t (api token)
kubectl config set-context ccs
```

The following information might be helpful to CCI kubectl users:

- Use `ccs [command] --help` to view information about a command.
- You might choose to use the `--skip-set-context` argument. Typically, this argument is used if you're using a TOKEN configured with Service Broker Admin Role, so you don't create Kubeconfig contexts for all Supervisor namespaces if you don't need to.
- The following example shows how you might create a context for a specific supervisor namespace from a specific project:

```
kubectl ccs set-context --project cci-explore --supervisor-namespace elastic-sky
```

## Manual Cloud Consumption Interface configuration for existing vRealize Automation Cloud users

Administrators can also manually configure access to CCI functionality using the Command Line Interface (CLI). To complete this manual configuration, an appropriate administrator runs a series of commands to execute APIs that will set up CCI functionality. See [Example of manually enabling Service Broker Cloud Consumption Interface for users](#) for details about this procedure.

This chapter includes the following topics:

- [Create an additional cloud account for the Cloud Consumption Interface](#)

## Create an additional cloud account for the Cloud Consumption Interface

vSphere+ cloud administrators can create an additional cloud account to support Service Broker Cloud Consumption Interface (CCI) users.

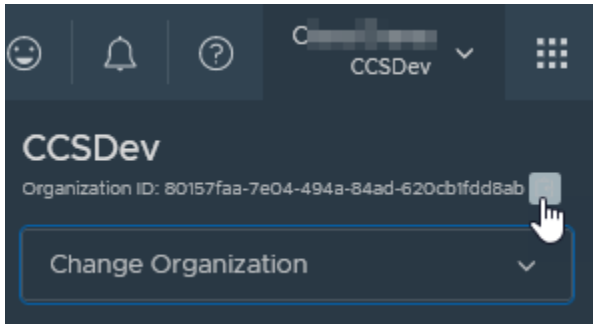
This procedure describes how a vSphere+ cloud administrator uses APIs and other commands to create an additional cloud account for use with CCI and set up federation with the vCenter instance.

As part of the procedure, you use the Cloud Assembly IaaS APIs to get your access token and create a vSphere cloud account. For more information about creating a cloud account with the APIs, see [Add a vSphere Cloud Account](#) in the *vRealize Automation API Programming Guide*.

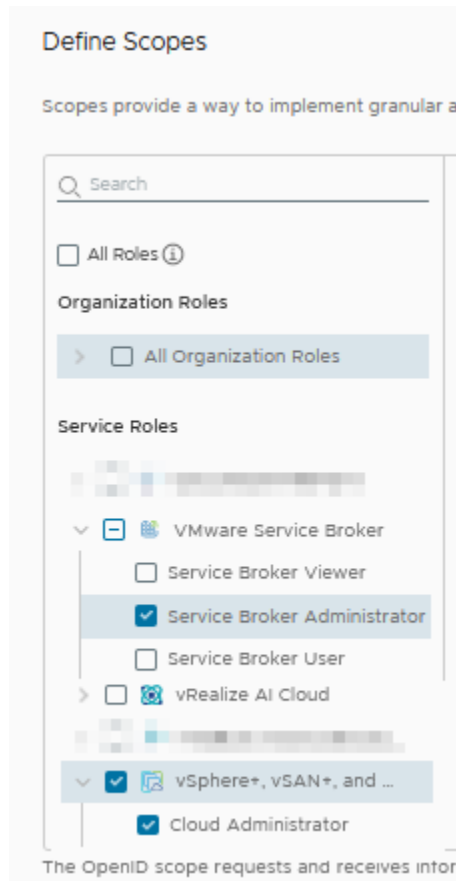
### Prerequisites

- Verify that you have a gateway appliance that is connected to the target organization, and that you have connected the vCenter to the gateway. For more information about gateways in vCenter, see [Connect Your vCenter Server to vCenter Cloud Gateway](#).
- Verify that you are at least an organization member in vRealize Automation Cloud with the Administrator service role for Cloud Assembly or Service Broker.

- Use the vRealize Automation Cloud Console to obtain your organization ID and generate an API token.
  - a Use your **My VMware** credentials to log in to <https://console.cloud.vmware.com/csp/gateway/discovery>.
  - b Click the drop-down arrow by your name.
  - c Under the organization name, click to copy the organization ID. Paste the string in a text file and save the file.



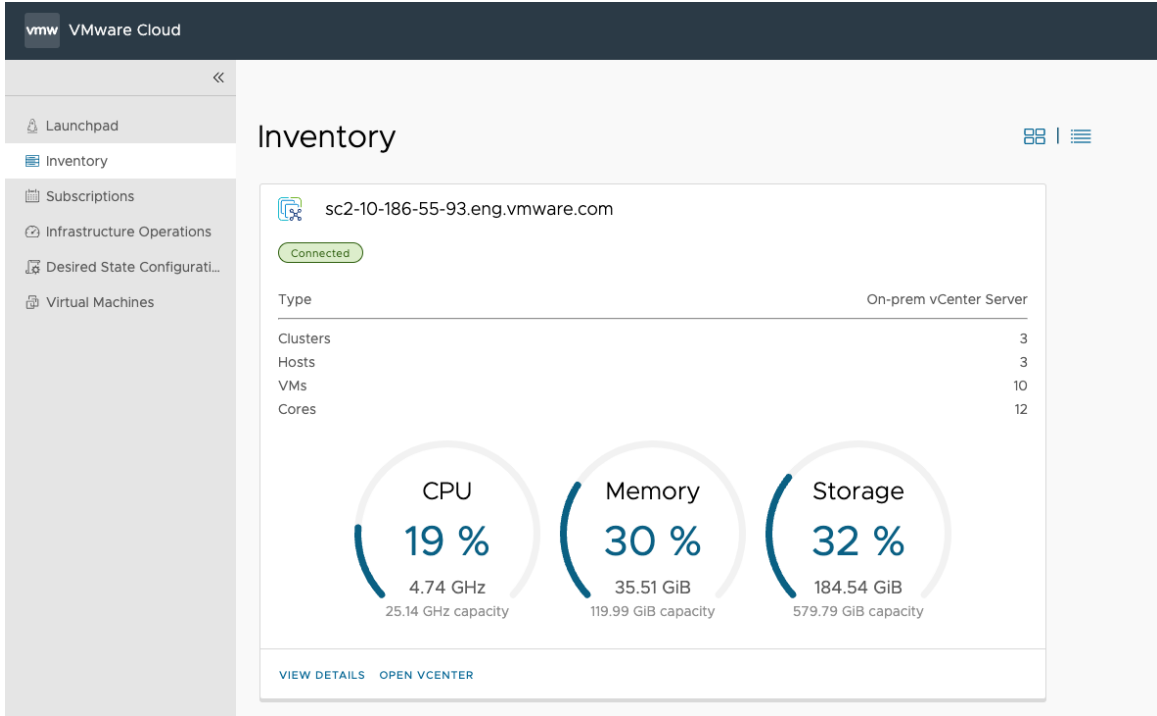
- d Perform the following steps to generate an API token for vSphere+ and for Service Broker service access.
  - 1 Click **My Account**.
  - 2 Click the **API Tokens** tab.
  - 3 Click **Generate Token**.
    - a Enter a Token Name.
    - b Under Define Scopes, select the Service Roles for:
      - **VMware Service Broker > Service Broker Administrator**
      - **vSphere+, vSAN+, and ... > Cloud Administrator**



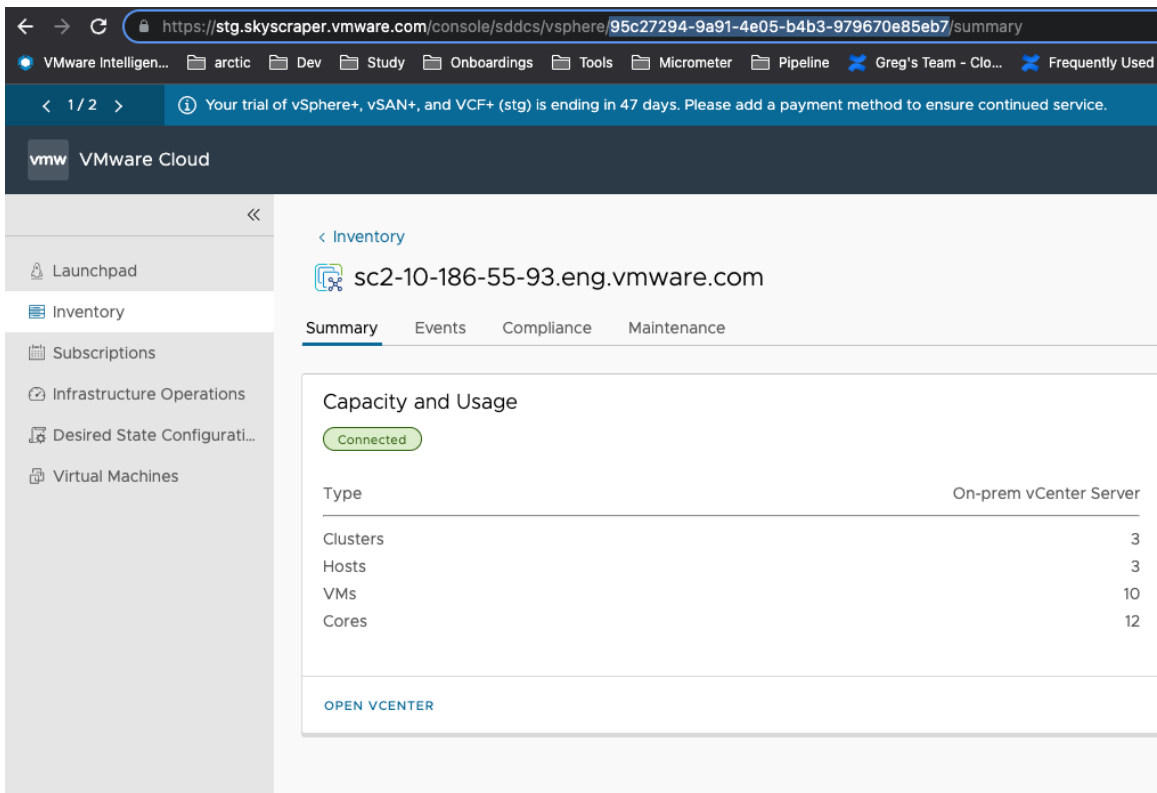
- c Click **Generate**.
- d When the **Token Generated** window displays a token with the name that you specified, click **COPY**. Paste the string in a text file and save the file.

## Procedure

- 1 You get the vCenter ID from the vSphere+ cloud console. Once the gateway and vCenter are connected to the cloud properly, it is visible from the vSphere Inventory.
  - a On the vSphere+ left menu, click **Inventory** and click **View Details** at the bottom left.



- b Get the vCenter ID from the URL.



Assign the vCenter ID variable as in the following example.

```
vcenter_id= '95c27294-9191-4e05-b4b3-979670e85eb7'
```

- 2 Assign a variable for the ID of your organization as in the following example.

```
org_id= '80157faa-7e04-494a-84ad-620cb1fdd8ab'
```

- 3 To to get an access token, use the API token that you generated for vSphere+ and for Service Broker service access.

---

**Note** After 25 minutes of inactivity, the access token times out and you must request it again.

---

- a Assign the API token variable.

```
api_token='<your_API_Token_for_vSphere+_and_Service_Broker>'
```

- b With the API token assigned, get the access token.

```
access_token=`curl -X POST \
  "https://api.mgmt.cloud.vmware.com/iaas/api/login" \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json' \
  -d '{
    "refreshToken": "'$api_token'"
  }' | jq -r .token`
```

- c Verify the access token variable is assigned.

The access token is a long JSON Web Token as in the following example.

```
echo $access_token
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InNpZ25pbmdfMiJ9.eyJzdWIiOiJ2bXdhcmUuY29tOj
...
...
tSQ74_XhszGifZe_gFdxw
```

- 4 Get the gateway ID.

```
curl -X GET \
  'https://vmc.vmware.com/api/vcenter/gatewaymanagement/org/$org_id/vc/$vcenter_id'\
  --header 'Authorization: Bearer $access_token'
```

The response provides the gateway ID as in the following example.

```
{
  "id": "1ed4f476-3022-6303-9a09-0f3a7a620478",
  "org_id": "80157faa-7e04-494a-84ad-620cb1fdd8ab",
  "hostname": "sc2-10-184-81-252.eng.vmware.com",
  "vm_instance_uuid": "4201D361-2B1D-CFCF-5B9E-6D28D675F7A9",
  "vcenter_info": [
    {
      "vcenter_id": "95c27294-9191-4e05-b4b3-979670e85eb7",
      "gateway_id": "1ed4f476-3022-6303-9a09-0f3a7a620478",
      "timestamp": "2022-10-18T23:42:31.047Z"
    }
  ],
}
```

```

    "gateway_version": "8.0.0.10000",
    "status": "healthy",
    "create_time": "2022-10-19T00:45:54.247Z",
    "update_time": "2022-10-19T01:15:00.989Z"
  }

```

## 5 Assign variables as in the following example.

```

gateway_id= '1ed4f476-3022-6303-9a09-0f3a7a620478'
hostname= 'sc2-10-184-81-252.eng.vmware.com'

```

## 6 To list the external region IDs from your vSphere cloud account, start a region enumeration.

```

curl -X POST \
  'https://api.mgmt.cloud.vmware.com/iaas/api/cloud-accounts-vsphere/region-enumeration?
  apiVersion=2021-07-15'
  --header 'Authorization: Bearer $access_token' \
  --header 'Content-Type: application/json' \
  --data-raw '{
    "hostName": "$hostname",
    "dcid": "$gateway_id",
    "environment": "aap",
    "acceptSelfSignedCertificate": true
  }'

```

The response includes a selfLink value that you use to track the progress of the region enumeration, for example 8f119c98-453d-4ace-84cd-88a7ec984e3f.

```

{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "region-enumeration-task",
  "id": "8f119c98-453d-4ace-84cd-88a7ec984e3f",
  "selfLink": "/iaas/api/request-tracker/8f119c98-453d-4ace-84cd-88a7ec984e3f"
}

```

## 7 Track the region enumeration request.

```

curl -X GET \
  'https://api.mgmt.cloud.vmware.com/iaas/api/request-tracker/
  8f119c98-453d-4ace-84cd-88a7ec984e3f?apiVersion=2021-07-15'
  --header 'Authorization: Bearer $access_token' \
  --header 'Content-Type: application/json'

```

When the region enumeration completes, the response shows "status": "FINISHED" and provides a link to the region enumeration in resources.

```

{
  "progress": 100,
  "status": "FINISHED",
  "resources": [
    "/iaas/api/cloud-accounts/region-enumeration/8f119c98-453d-4ace-84cd-88a7ec984e3f"
  ],
}

```

```

    "name": "region-enumeration-task",
    "id": "8f119c98-453d-4ace-84cd-88a7ec984e3f",
    "selfLink": "/iaas/api/request-tracker/8f119c98-453d-4ace-84cd-88a7ec984e3f"
  }

```

- 8 To get the cloud account regions, use the link to the region enumeration from the tracking response.

```

curl -X GET \
  'https://api.mgmt.cloud.vmware.com/iaas/api/cloud-accounts/region-enumeration/8f119c98-453d-4ace-84cd-88a7ec984e3f?apiVersion=2021-07-15' \
  --header 'Authorization: Bearer $access_token' \
  --header 'Content-Type: application/json'

```

The response shows the external regions.

```

{
  "externalRegions": [
    {
      "externalRegionId": "Datacenter:datacenter-53",
      "name": "test-dc"
    },
    {
      "externalRegionId": "Datacenter:datacenter-3",
      "name": "wcp-test-dc"
    }
  ]
}

```

- 9 Create the vSphere cloud account.

- In the `regions` field, specify the external regions from the region enumeration.
- For `createDefaultZones`, depending on whether you want to create default cloud zones for data centers in the endpoint or not, set the value to **true** or **false**. In this example, the value is **false**.

```

curl --location --request POST 'https://api.mgmt.cloud.vmware.com/iaas/api/cloud-accounts-vsphere?apiVersion=2021-07-15' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Bearer $access_token' \
  --data-raw '{
    "hostname": "$hostname",
    "dcid": "$gateway_id",
    "acceptSelfSignedCertificate": true,
    "associatedCloudAccountIds": [],
    "name": "<your_cloud_account_name>",
    "environment": "aap",
    "regions": [
      {
        "externalRegionId": "Datacenter:datacenter-53",
        "name": "test-dc"
      },
      {

```



```

        "externalRegionId": "Datacenter:datacenter-3",
        "name": "wcp-test-dc"
    }
  ],
  "createDefaultZones": false
}'

```

The response includes a `selfLink` value that you use to track the progress of the cloud account creation, for example `2a7ac045-b00b-4b46-94a8-0f053f3bc314`.

```

{
  "progress": 0,
  "status": "INPROGRESS",
  "name": "Cloud account creation/update",
  "id": "2a7ac045-b00b-4b46-94a8-0f053f3bc314", // ID of the tracker
  "selfLink": "/iaas/api/request-tracker/2a7ac045-b00b-4b46-94a8-0f053f3bc314"
}

```

## 10 Track the progress of the cloud account creation.

```

curl -X GET \
  'https://api.mgmt.cloud.vmware.com/iaas/api/request-tracker/2a7ac045-b00b-4b46-94a8-0f053f3bc314?apiVersion=2021-07-15' \
  --header 'Authorization: Bearer $access_token' \
  --header 'Content-Type: application/json'

```

When the cloud account creation completes, the response shows `"status": "FINISHED"` and provides a link to the cloud account in resources.

```

{
  "progress": 100,
  "status": "FINISHED",
  "resources": [
    "/iaas/api/cloud-accounts/8a49db4d-b68a-4366-ac44-8c6a70ec8a49"
  ],
  "name": "Cloud account creation/update",
  "id": "b1c644e8-67b0-4ca4-8cce-bc0d55a882a8",
  "selfLink": "/iaas/api/request-tracker/b1c644e8-67b0-4ca4-8cce-bc0d55a882a8"
}

```

## 11 After full enumeration, the cloud account is ready to use. Use the cloud account ID from the tracking response to get details about the cloud account.

```

curl -X GET \
  'https://api.mgmt.cloud.vmware.com/iaas/api/cloud-accounts-vsphere/8a49db4d-b68a-4366-ac44-8c6a70ec8a49?apiVersion=2021-07-15' \
  --header 'Authorization: Bearer $access_token' \
  --header 'Content-Type: application/json'

```

When the response shows `imageEnumerationTaskState": "FINISHED"` and `"enumerationTaskState": "FINISHED"`, image synchronization and data enumeration are complete.

```
...,
"customProperties": {
  "hostName": "sc2-10-184-81-252.eng.vmware.com",
  "acceptSelfSignedCertificate": "true",
  "lastImageEnumerationTimestampMicro": "1650567942698000",
  "version": "8.0.0",
  "buildNumber": "55223210",
  "lastSuccessfulImageEnumerationTimestampMicro": "1650567942698000",
  "lastSuccessfulEnumerationTimestampMicro": "1650567971775000",
  "imageEnumerationTaskState": "FINISHED",
  "dcId": "1ecc0d7a-2386-68bb-97fa-cb0d690c0196",
  "lastEnumerationTimestampMicro": "1650567971775000",
  "enumerationTaskState": "FINISHED",
  "wcpEnabled": "true",
  "environment": "aap",
  "privateKeyId": "administrator@vsphere.local",
  "vcUuid": "a1f546f2-c234-4ccd-98b2-de896506f884"
},
...
```

## Results

A vRealize Automation Cloud cloud account is created as specified.

# Using the Service Broker Cloud Consumption Interface

## 3

The Service Broker Cloud Consumption Interface (CCI) enables Service Broker DevOps users to work with Supervisor namespaces and associated services to create Kubernetes deployments or IaaS payloads. Supervisor namespaces are vSphere-based Kubernetes entities that enable you to organize resources within clusters. Different users have access to different namespaces and services.

## Getting Started with the Cloud Consumption Interface in Service Broker

The Cloud Consumption Interface uses vRealize Automation Cloud projects and infrastructure and vSphere Kubernetes resources as the foundation on which CCI users can work with namespaces to create deployable services. Services are pluggable UIs that follow SDK guidelines and they are built and tested as separate applications that have been incorporated into CCI. CCI contains wizards that guide you through the process of using services to create virtual machines, and other resources. When working with services, CCI automatically generates YAML code that users can download to use as the basis of deployments or IaaS resources.

To configure virtual machines or other resources for deployment using CCI, users must log in to Service Broker, click the **Consume** tab and select **Supervisor Namespaces**. There will be one or more projects containing namespace classes available to you. Namespace classes, which function as templates that reserve resources for namespaces that users create are defined by administrators. After you select a namespace class, you can create a new namespace. A namespace functions as your personal workspace with a set of resources and services.

---

**Note** Namespaces and other resources created in CCI are unique to that environment. Users should not attempt to manage them in other applications and products such as vRealize Automation Cloud Cloud Assembly and vSphere.

---

Following administrative set up, the CCI Welcome page appears when authorized DevOps users select Supervisor Namespaces in the Service Broker application, and the page lists the projects that are available to those users. Also, it shows the namespace classes that are associated with the selected project. You can use the available namespace classes as templates to create new namespaces. Then, you can use services to create virtual machines or other components as part of their namespaces.

If you dismiss the Welcome page, you can also start working from the CCI Home page. This page lists the namespaces and projects available to you on separate tabs. By default, neither of these pages are displayed for administrative users.

In addition, the tree view on the left shows a list of projects available to you, and you can expand it to view the namespace classes within each project.

This chapter includes the following topics:

- [Create a Supervisor namespace with the Service Broker Cloud Consumption Interface](#)
- [Use Service Broker Cloud Consumption Interface to create a deployable workload](#)

## Create a Supervisor namespace with the Service Broker Cloud Consumption Interface

The Service Broker Cloud Consumption Interface (CCI) enables users to create supervisor namespaces and then add virtual machines and other deployable workloads.

Supervisor namespaces are vSphere-based Kubernetes entities that enable you to organize resources within clusters. Users work with namespaces to create workloads such as virtual machines. Different projects can provide access to different namespace classes and thus different users will have access to different namespaces.

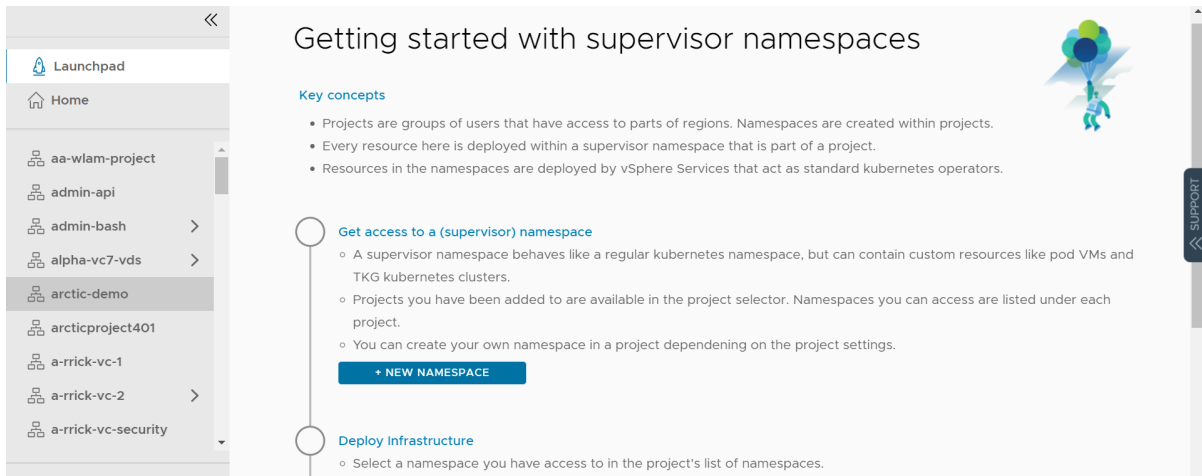
### Prerequisites

- An appropriate administrator must configure CCI using one of two available paths. First, the admin can use vSphere+ to enable CCI using the Developer Experience option for non vRealize Automation Cloud users. The second option is for existing vRealize Automation Cloud customers who can use the supplied APIs to configure CCI.

### Procedure

- 1 Log in to Service Broker, then click the **Consume** tab and select **Supervisor Namespaces**.

The Supervisor Namespaces Consume Launchpad Getting Started page is displayed. This page contains a **New Namespace** button that enables you to begin creating a namespace.

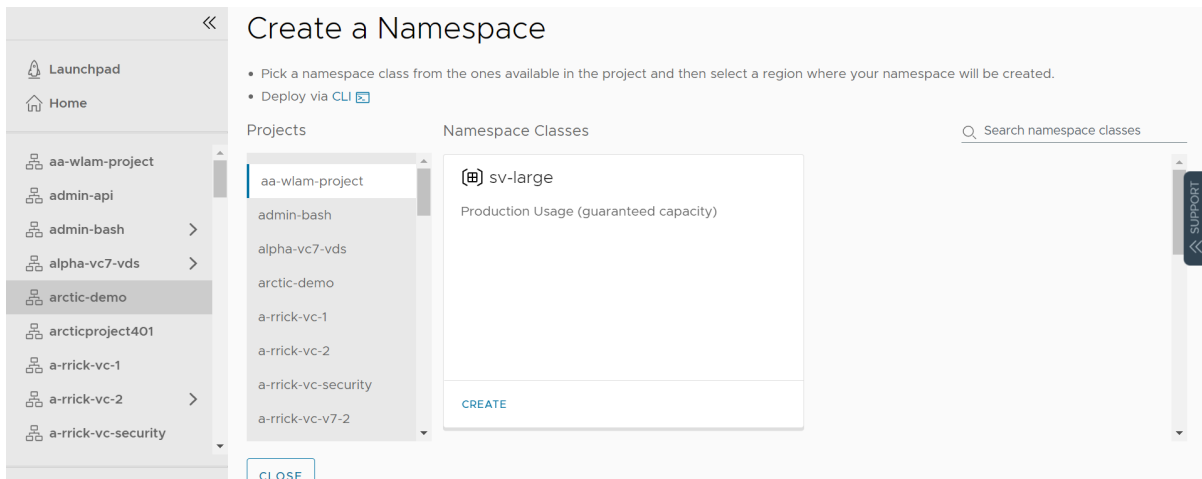


You can use either the CCI Launchpad or Home pages to select a project and a namespace class. Both pages provide access to both components based on user profiles. Each project configured for you provides access to one or more namespace classes that you can use as the basis of namespaces. The Home page lists any existing supervisor namespaces that the user can access.

- 2 Click **New Namespace** on the Launchpad page.

The button is titled **New Supervisor Namespace** on the Home page.

- 3 Select one of the Namespace classes available to you and click **Create** to begin creating a namespace based on that class. Namespace classes function as templates for namespaces.



- 4 Create a namespace. You will need to provide the following information.

- Name and Description - Enter basic identifying information for the project.
- Region- Regions are grouping mechanisms for namespaces created by administrators. By default a region is selected, but you can select other regions that are available within the related project.

New Namespace

×

To be able to request any resources, you will need a namespace. Depending on the selected class, your namespace may have different usage quotas, support HA, and give access to different services.

Project


aa-wlam-project

Namespace class

sv-large

Auto-created

Sharing

 This namespace is accessible to all project users.

Name

Namespace name

May contain lowercase alphanumeric characters and "-"

Description

Region

palo-alto

▼

CANCEL

CREATE

## Results

When you click **Create**, the namespace is created and appears on the list of namespaces for the applicable project. The Cloud Consumption Interface will select the appropriate supervisor within the specified region to create the namespace for the infrastructure resources specified.

The Namespace dialog also contains tabs for Users, Region, and Limits, that show the users and groups associated with a namespace, as well as their role, and the infrastructure limits associated with the namespace. You can't add users, groups or limits, but you can delete them.

## What to do next

After you create a namespace, you can click on it to view the options for creating and working with workloads within that namespace. The namespace includes services that help you to create workloads. For example, one of the offered services is a Virtual Machine service, and you can click on it to view existing virtual machines and initiate a wizard to help you create a new virtual machine. To create a virtual machine within a namespace and select services associated with that namespace.

# Use Service Broker Cloud Consumption Interface to create a deployable workload

The Service Broker Cloud Consumption Interface feature enables users to work with and create vSphere Supervisor namespaces to deploy virtual machines, and other IaaS workloads.

## Prerequisites

- The user must create or select an appropriate Supervisor namespace using CCI.

## Procedure

- 1 Create a namespace as described in [Create a Supervisor namespace with the Service Broker Cloud Consumption Interface](#).

When you click **Create**, the namespace is created and it appears on the list of Namespaces for the applicable project. CCI will select the appropriate supervisor within the specified region to create the namespace for the infrastructure resources specified.

After you create a namespace, you can click on it to view the options for selecting an existing service or creating a new service within the namespace. One of the services will be a Virtual Machine service, and you can click on it to view existing virtual machines and create a new one. The available services are displayed on the Namespace page.

- 2 Click one of the services to select it.

The rest of this procedure assumes that you selected the Virtual Machine service.

- 3 Open the the Virtual Machine page for the namespace page by clicking the Virtual Machine service tile.

- 4 Click **Create VM**.

A New Virtual Machine wizard will start to guide you through the process of creating a new virtual machine.

- 5 Type a name for the virtual machine.

- 6 Select the class that you want to use for the virtual machine. The class selection defines the resources available within the virtual machine.

- 7 Select the image to use with the virtual machine.

- 8 If desired, you can click the **Advanced Configuration** check box to access settings to specify additional resources that are available within the virtual machine.

## Results

As you create a virtual machine, the YAML code that defines the machine is displayed in the YAML section at the right side of the page. You can click the **Download .Zip** button to download a zipped copy of the YAML file generated by the virtual machine wizard.

## What to do next

You can copy and paste the virtual machine YAML code into a command line, incorporate it into a GIT repository as a GitOps style workflow or use it in some other capacity as an IaaS object.

## Working with Virtual Machines in the Cloud Consumption Interface

After you create a namespace, you can use the Cloud Consumption Interface (CCI) to view and create virtual machines that you can deploy on Supervisors or use as an IaaS resource.

To view the virtual machines associated with a namespace, click the namespace on the left tree menu. The Virtual Machine Service page will open listing all virtual machines associated with the namespace. Click the Related objects tab on the Virtual Machine Service page to see all services and other objects associated with virtual machines on the namespace.

Also, you can create a new virtual machine associated with the namespace as described below.

- 1 Open the Virtual Machine page for your namespace by clicking the Virtual Machine service tile.

A New Virtual Machine wizard will start that guides you through the process of creating a new virtual machine.

- 2 Click **Create VM**.
- 3 Type a name for the virtual machine.
- 4 Select the class that you want to use for the virtual machine. The class selection defines the resources available within the virtual machine.
- 5 Select the image to use with the virtual machine.
- 6 If desired, you can click the **Advanced Configuration** check box to access settings to specify additional resources that are available within the virtual machine.

As you create a virtual machine, the YAML code that defines the machine is displayed in the YAML section at the bottom of the page. You can click the **Download.Zip** button to download a zipped copy of the YAML file generated by the virtual machine wizard.

You can copy and paste the virtual machine YAML code into a command line or use it as an IaaS resource. Alternatively, you may choose to incorporate it into a Git repository as a GitOps style workflow.

After you create a virtual machine, it is listed on the Virtual Machines page for the applicable Supervisor namespaces in CCI. You can click on the virtual machine name to open a page that displays Summary information for that machine. The page also contains a **Powered On** toggle that enables you to switch machine from powered on to powered off.

In addition, there is an **Add Volume** button that opens a dialog enabling you to add additional disk volumes in specified sizes to the virtual machine.

## Working with the Tanzu Kubernetes Grid service in the Cloud Consumption Interface

After you create a namespace in the Cloud Consumption Interface (CCI), you can select or create a Tanzu Kubernetes Grid cluster that you can deploy on a Supervisor or use as an IaaS resource.



On a CCI namespace page, you can view and manage existing TKG clusters associated with the selected namespace. To view the TKG cluster resources associated with a particular namespace, click the applicable namespace name on the left tree menu and then click the Tanzu Kubernetes Grid service tile. The Tanzu Kubernetes cluster Service page will open listing all clusters associated with the namespace. Click the Related objects tab on the Tanzu Kubernetes Service page to see all of the services and other objects associated with Tanzu Kubernetes page on the namespace.

Also, you can create a new Tanzu Kubernetes cluster associated with the namespace using the wizard. As you work through the wizard, the pane on the right side of the page displays the YAML code for the cluster you are creating. The following procedure outlines how to use the new Tanzu Kubernetes cluster wizard.

- 1 Open the Tanzu Kubernetes Grid service page for your namespace by clicking the Tanzu Kubernetes Grid service tile.

A New Tanzu Kubernetes cluster wizard will start that guides you through the process of creating a new Tanzu Kubernetes cluster.

- 2 Click **Create**.
- 3 Select the configuration type that you want to use for the Tanzu Kubernetes cluster. The configuration selection defines the resources available within the virtual machine.

You can choose either a default configuration or a custom configuration based on the v1alpha2 API.

- 4 Select the image to use with the virtual machine.
- 5 If desired, you can click the **Advanced Configuration** check box to access settings to specify additional resources that are available within the virtual machine.

As you create a TKG cluster, the YAML code that defines the cluster is displayed in the YAML section at the bottom of the page. You can click the **Download .Zip** button to download a zipped copy of the YAML file generated by the virtual machine wizard.

After you create a TKG cluster, it is listed on the TKG services page for the applicable Supervisor Namespaces in CCI. You can click on the TKG cluster name to open a page that displays Summary information for that instance. You can do several things with the deployed cluster from this page.

- You can edit the control plane for the number of replicas or for the specified virtual machine class.
- You can edit the nodepools for the number of replicas or for the specified virtual machine class.
- You can add Volumes to the cluster.

## Working with the Volumes Service in Cloud Consumption Interface

After you create a namespace, you can create storage volumes in Cloud Consumption Interface, for use in deployments or in IaaS workflows.

- 1 Open the Volume page for your namespace by clicking the Volume service tile.

A New Volume wizard will start that guides you through the process of creating a new virtual machine.

- 2 Click **Create Volume**.
- 3 Type a name for the virtual machine.
- 4 Enter the **Storage Class** and **Capacity** that you want to configure for the volume.
- 5 Select the **Access Modes** for the volume. The options are Read Write Once and Read Write Many. Read Write Many functions only with VSAN enabled with an appropriate file service configuration.
- 6 If desired, you can click the **Advanced Configuration** check box to access settings to specify additional resources that are available within the virtual machine.

As you create a virtual machine, the YAML code that defines the machine is displayed in the YAML section at the bottom of the page. You can click the **Download.Zip** button to download a zipped copy of the YAML file generated by the virtual machine wizard.

You can copy and paste the virtual machine YAML code into a command line or as an IaaS resource. For example, you may choose to incorporate it into a Git repository as a GitOps style workflow.

# Using the Cloud Consumption Interface Command Line Interface

# 4

The Service Broker Cloud Consumption Interface includes a command line interface that enables administrators and , to lesser extent, users to work with CCI functionality without using the provided user interface.

This chapter includes the following topics:

- [Cloud Consumption Interface Kubernetes API Reference](#)
- [Example of manually enabling Service Broker Cloud Consumption Interface for users](#)

## Cloud Consumption Interface Kubernetes API Reference

Administrative and DevOps users can use Cloud Consumption Interface (CCI) API resources that are exposed by the CCI Kubernetes API-server. All CCI resources are implemented using the aggregate api-server and backed by composing existing vRealize Automation APIs.

### Projects and Users

Project

Projects are created by an administrator to group users and set access to infrastructure resources.

Role	Verbs
admin	create get update patch delete list
user	get list

See the createproject example:

```
apiVersion: project.ccs.vmware.com/v1alpha1
kind: Project
metadata:
  name: demo-project
spec:
  description: This is a demo project
  sharedResources: true
```

ProjectRole

After a project is created, project roles are created by the system to reflect available roles (OWNER, EDIT,VIEW). The project roles control access to namespaces by Supervisor Namespaces, and are also set in the vCenter namespace permissions. The permissions associated with the roles are as follows:

- OWNER: This role allows modification and deletion of the namespace.
- EDIT: This role allows modification of the namespace.
- VIEW: This is a read-only role on the namespace.

Role	Verbs
admin	get list
user	get list

```
apiVersion: authorization.ccs.vmware.com/v1alpha1
kind: ProjectRole
metadata:
  name: admin
spec:
  description: project administrator
```

### ProjectRole Binding

Project role binding is created by an administrator to assign a role per user or group. This binding determines which users have permissions to a project and what role they can assume within the project. Supervisor Namespaces validates one resource per user.

Role	Verbs
admin	create get update patch delete list
user	get list

See the create projectrolebinding example:

```
apiVersion: authorization.ccs.vmware.com/v1alpha1
kind: ProjectRoleBinding
metadata:
  # name must match the subject kind and name
  name: ccs:user:vmware.com:hello
  namespace: demo-project
subjects:
  - kind: User # User / Group
    name: hello@vmware.com
roleRef:
  apiGroup: authorization.ccs.vmware.com
  kind: ProjectRole
  name: admin # admin / edit / view
```

## vCenters and Supervisors

This category deals with infrastructure and its functions are for administrators only.

### CloudProxy

Cloud Proxy is created by the system when a cloud proxy registers itself with lemans and vRealize Automation. It is currently not exposed for vSphere+ based projects.

Role	Verbs
admin	get delete list

See the describe cloudproxy example:

```

Name:          2d164fed-bbf3-47cc-8e6b-5226c5277ee4
Namespace:     ccs-config
Labels:        <none>
Annotations:   <none>
API Version:   infrastructure.ccs.vmware.com/v1alpha1
Kind:          CloudProxy
Metadata:
  Creation Timestamp:  2022-10-17T12:06:07Z
  UID:                 492eb469-b6e4-3534-8f48-30bd6f58e904
Spec:
  Nickname:  my-cloud-proxy
  Proxy Id:  2d164fed-bbf3-47cc-8e6b-5226c5277ee4
Status:
  Address:  10.193.5.132
  Conditions:
    Message:      Statuses: cloud-proxy=Active cloudassembly-cmx-agent=RUNNING cloudassembly-
sddc-agent=RUNNING
    Status:       True
    Type:         CloudProxyReady
  Hostname:      xx-rdops-vm08-xxxx-5-111.eng.company.com
  Last Sync Time: 2022-10-24T22:02:00Z
  Phase:        Active
  Events:       <none>

```

### CloudAccount

Cloud accounts are created automatically by vSphere+ or manually by an administrator to register a vCenter and enable datacenters with supervisors.

Role	Verbs
admin	get list

See the describe cloudaccount example:

```

Name:          ccs-ui-volume-service
Namespace:     ccs-config
Labels:        <none>
Annotations:   infrastructure.ccs.vmware.com/data-collection-status: FINISHED
                infrastructure.ccs.vmware.com/last-data-collection-timestamp:
2022-10-24T22:06:08.603Z

```

```

API Version:  infrastructure.ccs.vmware.com/v1alpha1
Kind:          CloudAccount
Metadata:
  Creation Timestamp:  2022-10-17T12:18:28Z
  UID:                2163e7cf-f698-3f1f-afca-f3daa8c730fa
Spec:
  Address:            127.193.29.114
  Cloud Proxy Name:   2d164fed-bbf3-47cc-8e6b-5226c5277ee4
  Username:           admin@vsphere.local
Events:              <none>

```

## Supervisor

A supervisor is created by the system after vCenter data-collection. The administrator can update the Supervisor with capability labels, for placement, and assign to a region.

Role	Verbs
admin	get patch list

See the describe supervisor example:

```

Name:          bugbash-vc:domain-c8
Namespace:     ccs-config
Labels:        environment=bug-bash-9
Annotations:   infrastructure.ccs.vmware.com/cloud-account-id: 33a0b2d0-91c8-4629-
b04a-65448494d54e
API Version:   infrastructure.ccs.vmware.com/v1alpha1
Kind:          Supervisor
Metadata:
  Creation Timestamp:  2022-09-28T04:22:38Z
  UID:                fbd10d08-bc56-4ec2-93f8-693a7a4b2003
Spec:
  Cloud Account Name:  bugbash-vc
  Display Name:        wcp-test-dc-cluster
  External Id:         domain-c8
  Region Names:
    us-demol
Status:
  Power State:  On
Events:        <none>

```

## Topology

### Region

Administrators create regions as a grouping mechanism for one or more supervisors. Regions can be based on geography or infrastructure, etc.; and they can include supervisors from multiple vCenters.

Role	Verbs
admin	create get update patch delete list
user	get list

See the describe region example:

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: Region
metadata:
  name: us-west1
spec:
  description: The us-west1 region
```

## RegionBinding

Region binding is created by an administrator to associate regions with projects. The project is not defined directly, but rather it is specified by a namespace that is associated with the project in Supervisor Namespaces. The resource should not contain any supervisor placement settings.

Role	Verbs
admin	create get update patch delete list
user	get list

See the describe regionbinding example:

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: RegionBinding
metadata:
  name: us-west1
  namespace: demo-project
```

## RegionBindingConfig

Region binding config is defined by an administrator to control the Supervisor placement logic on a per region basis in a project. It allows an administrator to use key:value pairs to further refine the association of specific supervisors to projects. For example, an administrator could use a `key:environment` specification to select a supervisor specified for testing with `value:testing`. The `supervisorSelector` is used to match supervisor labels. The expression operators are limited to support for existing CMX constraints behavior (In/NotIn/Exists/DoesNotExist with one value).

For a project to have access to supervisors, both RegionBinding and RegionBindingConfig must exist.

Role	Verbs
admin	create get update patch delete list

See the create regionbindingconfig example:

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: RegionBindingConfig
metadata:
  name: us-west1
  namespace: demo-project
spec:
  supervisorSelector:
```

```

matchExpressions:
  - key: environment
    operator: In
    values:
      - testing
  - key: storage
    operator: Exists
  - key: storage
    operator: NotIn
    values:
      - encrypted

```

## Supervisor Namespace Classes

### SupervisorNamespaceClass

SupervisorNamespaceClass is created by an administrator to define the schema characteristics of namespace templates with optional parameters. The SupervisorNamespaceClass definitions list the names for namespaces classes and the characteristics, or parameters, for those classes. The resource should not contain any namespace settings.

The default field specifies the parameter value that is used if a user doesn't provide the parameter value on namespace creation. So, "default:false" means the "false" value will be used.

Role	Verbs
admin	create get update patch delete list
user	get list

```

apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: SupervisorNamespaceClass
metadata:
  name: gold
spec:
  description: Gold supervisor namespace class
  parameters:
    - name: encryptedStorage
      type: boolean
      default: false
    - name: dataStoragePolicy
      type: string
      default: standard-storage-policy
      enum:
        - standard-storage-policy
        - tolerant-storage-policy
    - name: maxPods
      type: integer
      minimum: 100
      maximum: 1000
      default: 500
    - name: stringValue
      type: string
      default: 111-222-3333

```



```

minLength: 100
maxLength: 200
pattern: (\\([0-9]{3}\\))?[0-9]{3}-[0-9]{4}

```

### SupervisorNamespaceClassConfig

Whereas SupervisorNamespaceClass defines the schema for namespace classes, SupervisorNamespaceClassConfig defines the implementation specification for them. SupervisorNamespaceClassConfig is defined by an administrator to define namespace class configuration with namespace settings. Parameter values can be used in namespace settings.

- Administrators can use the `supervisorSelector` to match supervisor labels and provide an additional layer of filtering. The expression operators are limited to support existing CMX constraints behavior (In/NotIn/Exists/DoesNotExists with one value).
- The storageClasses will map to automatically created Storage Profiles containing a tag with the policy name (`ccs-storage-policy:wcp-storage-class`)
- For a project to have access to Supervisor Namespace Class, both SupervisorNamespaceClass and SupervisorNamespaceClassConfig must exist.

Role	Verbs
admin	create get update patch delete list

```

apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: SupervisorNamespaceClassConfig
metadata:
  # match the same name of SupervisorNamespaceClass
  name: gold
spec:
  storageClasses:
    - name: wcp-storage-class
      limit: "100"
    - name: ((params.userStorage))
      limit: "100"
  vmClasses:
    - name: big-class-name
    - name: small-class-name
  contentSources:
    - name: staging-content-library-name
      type: Content Library # from VirtualMachineImage spec.imageSourceType
    - name: testing-content-library-name
      type: Content Library
  limits:
    - name: cpu_limit
      limit: "1000"
    - name: cpu_limit_default
      limit: "800"
    - name: pod_count
      limit: "((params.allowPods ? '5' : '0'))"
  # TODO: 'networks' not supported by vRA
  networks:
    - name: network-name

```

```

supervisorSelector:
  matchExpressions:
    # The label key
    - key: capability.ccs/gpu
    # Represents a key's relationship to a set of values.
    # Valid operators are In, NotIn, Exists, DoesNotExist.
    operator: In
    # An array of string values. If the operator is In or NotIn,
    # the values array must be non-empty. If the operator is Exists or DoesNotExist,
    # the values array must be empty.
    values:
      - true

```

## SupervisorNamespaceClassBinding

Administrators define SupervisorNamespaceClassBinding to allow creating a Supervisor Namespace using the Supervisor Namespace Class in a project. The `overrideParameters` are optional and allow forcing a parameter value while ignoring the user provided parameter values during Supervisor Namespace creation.

Role	Verbs
admin	create get update patch delete list
user	get list

```

apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: SupervisorNamespaceClassBinding
metadata:
  name: gold
  namespace: demo-project
spec:
  overrideParameters:
    - name: allowPods
      type: boolean
      const: false
  supervisorNamespaceClassRef:
    apiVersion: infrastructure.ccs.vmware.com/v1alpha1
    kind: SupervisorNamespaceClass
    name: gold

```

## Supervisor Namespaces

### SupervisorNamespace

Unlike most other commands, this command is available to users as well as administrators. SupervisorNamespace allows dev-ops users to create Supervisor namespaces in a specific region while using a Supervisor Namespace Class as a template and to force additional placement constraints.

Role	Verbs
admin	create get delete list
user	create get delete list

See the create supervisornamespace example:

```
apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: SupervisorNamespace
metadata:
  name: demo-ns5
  namespace: demo-project
spec:
  description: Demonstrating supervisor namespace creation
  regionName: us-west2
  className: bronze
```

See the describe supervisornamespace example:

```
Name:          demo-1
Namespace:     sprint-demo-project
Labels:        <none>
Annotations:   infrastructure.ccs.vmware.com/wcp-address: 10.161.81.40
API Version:   infrastructure.ccs.vmware.com/v1alpha1
Kind:          SupervisorNamespace
Metadata:
  Creation Timestamp:  2022-09-13T01:55:57Z
  UID:                 kate-demo-1
Spec:
  Class Name:  demo-class
  Class Parameters:
    Pods:      30
  Description:
  Region Name: us-demo-1
Status:
  Conditions:
    Last Transition Time:  2022-09-13T01:55:58Z
    Status:                True
    Type:                  Ready
  Phase:                  Created
Events:                   <none>
```

## Example of manually enabling Service Broker Cloud Consumption Interface for users

vRealize Automation Cloud cloud administrators may, in some situations, need to work with the command line interface to enable Cloud Consumption Interface (CCI) access and configure governance constructs. The procedure herein shows an example of how this might be accomplished using the Cloud Consumption Interface CLI.

There are several situations in which vRealize Automation Cloud cloud administrators may need to use the following procedure to manually configure the Service Broker Cloud Consumption Interface.

- If the administrators configured user access to CCI using vSphere+, users will have access only to the default projects created by vSphere+. Administrators can use the commands in this procedure to configure access to additional projects and cloud accounts.
- Administrators for existing vRealize Automation Cloud instances who want to use CCI without initiating it through vSphere+

To complete configuration, you must run a series of `kubectl` commands as described in the following procedure. To use these commands you must download and configure the CCI `kubectl` plug-in.

### Prerequisites

- You need a connected vCenter instance containing appropriate namespaces.
- Download the `kubectl-ccs` executable from the Supervisor Namespaces Consume interface. See [Chapter 2 Administrator configuration of the Service Broker Cloud Consumption Interface](#) for more information.

### Procedure

- 1 Create a file for a project and then run the following command to create the project. Administrators use projects to group vRealize Automation Cloud users and set access to infrastructure resources.

```
kubectl create -f project.yaml
```

This command enables an administrator to group users and set access to infrastructure resources. The following is an example `project.yaml` file result.

```
>
apiVersion: project.ccs.vmware.com/v1alpha1
kind: Project
metadata:
  name: <project name>
spec:
  description: <description of project>
  sharedResources: true
```

- 2 Run the following command to create project role binding to assign a user to a project and assign a role to the user. Administrators use project role binding to assign roles to users or groups within a created project.

```
$ kubectl create -f projectrolebinding.yaml
```

This command enables an admin to group users and set access to infrastructure resources. Administrators can add users and groups to projects with the following project roles: admin, edit, view. The following is an example of the contents of the project binding.

```
apiVersion: authorization.ccs.vmware.com/v1alpha1
kind: ProjectRoleBinding
metadata:
  name: ccs:user:vmware.com:<user alias>
  namespace: <project name>
roleRef:
  apiGroup: authorization.ccs.vmware.com
  kind: ProjectRole
  name: admin
subjects:
- kind: User
  name: <username@company.com>
```

### 3 Set up Region Binding.

Run the following command to create region binding to allow project accessing Supervisors in a region. Regions are a grouping mechanism for Supervisor clusters. The resource should not contain any Supervisor placement settings.

```
$ kubectl create -f regionbinding.yaml
```

Example YAML results are shown below.

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: RegionBinding
metadata:
  name: <region name>
  namespace: admin-api
```

### 4 An administrator can also update the Supervisor adding the region and with labels that will be used for namespace placement.

```
$ kubectl -n ccs-config get supervisors
```

The following is an example YAML result.

NAME	AGE
demo-self-service:domain-c50	75d
kate-vcenter-176:domain-c8	33d
priyanka-vcenter:domain-c8	5d18h
anantkumar-vcenter:domain-c8	74d
tjingjing-vcenter:domain-c8	39d
ppuranik-vcenter-2:domain-c8	49d
deva-ldu10:domain-c48	6d22h
rrick-be-2022-03-28:domain-c50	15d

```

jpick-vcenter-186-0:domain-c8    25d
dannyh-vcenter:domain-c57        42d
scale-test-vc-01:domain-c8       56d
parunesh-ccs:domain-c8           27d

```

**5** Update the Supervisor with the region and labels required for Supervisor namespace placement. Refer to the following for usage information:

- Press **Insert** to edit.
- Add both the labels and the regions for which this Supervisor will be a member.
- Use `wq!` to save changes as you would if you were editing a file with Visual Editor (VI) in Linux.
- This process also create Storage Profile for every vCenter storage policy that has a compatible storage in vRealize Automation Cloud.

Run the following command to update the Supervisor adding the region and the labels designated for namespace placement. The labels shown below are examples. Create labels as appropriate for your implementation, but be sure to observe the indentation as shown in the example.

```
$ kubectl -n ccs-config edit supervisor priyanka-vc:domain-c8
```

```

apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: Supervisor
metadata:
  annotations:
    infrastructure.ccs.vmware.com/cloud-account-id: 88abaec7-31c3-43a3-ba29-befce9f559fb
  labels:
    environment: testing
    fipsMode: strict
  name: priyanka-vc:domain-c8
  namespace: ccs-config
  uid: ccd3d154-6404-47b7-8786-bb2d49ad9f5d
spec:
  cloudAccountName: priyanka-vc
  externalId: domain-c8
  externalName: wcp-test-dc-cluster
  regionNames:
    - eu-west1
status:
  powerState: "On"

```

**6** Set up Region Binding.

Run the following command to create region binding to allow projects to access Supervisors in a region. The resource should not contain any Supervisor placement settings.

```
$ kubectl create -f regionbinding.yaml
```

Example YAML results are shown below.

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: RegionBinding
metadata:
  name: <region names>
  namespace: <project name>
```

- 7 Set up Region Binding Config for every region binding that you created. Be sure to observe the following:
  - You can create or edit a region binding config object with additional expressions to match the Supervisor label.
  - When executed, region binding config adds the Supervisor cluster as a managed entity in vRealize Automation Cloud. It then creates the Kubernetes zones with the Supervisor clusters as a provisioning destination for every supervisor in a region and then assigns it to the project.
  - It also configures capability tags on the Supervisor cluster under the provisioning tab within the Kubernetes zone.

Run the following command to create region binding configuration. Create or edit a Supervisor namespace Class Config with additional expressions to match the Supervisor tags.

```
$ kubectl create -f regionbindingconfig.yaml
```

Example YAML results are shown below:

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: RegionBindingConfig
metadata:
  name: eu-west1
  namespace: admin-api
spec:
  supervisorSelector:
    matchExpressions:
      - key: environment
        operator: In
        values:
          - testing
```

- 8 Run the following command to group one or more Supervisors in a region. This command assigns a group to a region. Supervisor resources are visible after vCenter data-collection.

```
$ kubectl create -f region.yaml
```

The following is an example result.

```
region.yaml
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: Region
```

```

metadata:
  name: eu-west1
spec:
  description: The eu-west1 region

```

- 9 Use the following command to create Supervisor Namespace Class to define namespace templates with optional parameters. The resource should not contain any namespace settings.

You can add optional inputs under `parameters` to reflect namespace class parameters provided by users when creating the namespace. All inputs must have default values. The input values can be used to customize the storage profiles, limits and additional constraints.

Types are Integer, String, or Boolean.

```
$ kubectl create -f supervisornamespaceclass.yaml
```

```

$ kubectl create -f supervisornamespaceclass.yaml
supervisornamespaceclass.yaml
apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: SupervisorNamespaceClass
metadata:
  name: bronze
spec:
  description: Bronze supervisor namespace class
  parameters:
    - name: pods
      type: Integer
      default: 40

```

- 10 Run the following command to create Supervisor namespace class configuration.

```
$ kubectl create -f supervisornamespaceclass.yaml
```

The `supervisorSelector` is used to match supervisor labels. The expression operators will be limited to support existing CMX constraints behavior (In/NotIn/Exists/DoesNotExist with one value).

The `storageClasses` will map to the automatically created Storage Profiles containing a tag with the policy name (`ccs-storage-policy:wcp-storage-class`)

For a project to have access to Supervisor Namespace Class, both `SupervisorNamespaceClass` and `SupervisorNamespaceClassConfig` must exist.

Limit names are as follows:

- `config_map_count`
- `cpu_limit`
- `cpu_limit_default`
- `cpu_request_default`



- daemon\_set\_count
- deployment\_count
- job\_count
- memory\_limit
- memory\_limit\_default
- memory\_request\_default
- persistent\_volume\_claim\_count
- pod\_count
- replica\_set\_count
- replication\_controller\_count
- secret\_count
- service\_count
- stateful\_set\_count
- storage\_request\_limit

The only value for contentSource type is currently ContentLibrary.

Example YAML results are shown below:

```
apiVersion: topology.ccs.vmware.com/v1alpha1
kind: SupervisorNamespaceClassConfig
metadata:
  name: bronze
spec:
  storageClasses:
    - name: management-storage-policy-thin
  vmClasses:
    - name: "*"
  contentSources:
    - name: "*"
      type: ContentLibrary
  # Below limits are an EXAMPLE! Setting them may cause unexpected behavior in your
  namespace
  # Either set reasonable limits, or remove the below section to get unlimited resources
  limits:
    - name: cpu_limit
      limit: "1000" # This value is in Mhz
    - name: cpu_limit_default
      limit: "800" # This value is in Mhz
    - name: pod_count
      limit: "((parameters.pods))"
  supervisorSelector:
    matchExpressions:
```

```

- key: environment
  operator: In
  values:
    - testing

```

- 11 Run the following command to create Supervisor namespace class binding. This step associates a Supervisor namespace class with a designated project. The resource should not contain any supervisor placement settings. This command allows creating a Supervisor namespace using the Supervisor Namespace Class in a project. The `overrideParameters` are optional and allow forcing a parameter value while ignoring the user provided parameter values during Supervisor namespace creation.

Use the following Class Binding example and update the inputs to reflect the override parameters for the specified project. Use `inputs const` to prohibit users from changing the class parameter value. Types are Integer, String, or Boolean. Note that the `namespace` parameter should specify the project name.

```
$ kubectl create -f supervisornamespaceclassbinding.yaml
```

Example YAML results are shown below.

```

apiVersion: infrastructure.ccs.vmware.com/v1alpha1
kind: SupervisorNamespaceClassBinding
metadata:
  name: bronze
  namespace: <project name>
spec:
  overrideParameters:
    - name: pods
      type: Integer
      const: 50

```