



IBM Cloud
Container Registry
Product guide

Edition notices

This PDF was created on 2023-07-29 as a supplement to *Container Registry* in the IBM Cloud docs. It might not be a complete set of information or the latest version. For the latest information, see the IBM Cloud documentation at <https://cloud.ibm.com/docs/Registry>.

© IBM Corp. 2023

Getting started with Container Registry

IBM Cloud® Container Registry provides a multi-tenant private image [registry](#) that you can use to store and share your [container images](#) with users in your IBM Cloud account.

The IBM Cloud console includes a brief Quick Start. To find out more about how to use the IBM Cloud console, see [Managing image security with Vulnerability Advisor](#).



Important: Do not put personal information in your container images, namespace names, description fields, or in any image configuration data (for example, image names or image labels).

Before you begin

Install the IBM Cloud CLI so that you can run the IBM Cloud `ibmcloud` commands, see [Getting started with the IBM Cloud CLI](#).

Step 1: Install the Container Registry CLI

1. Install the `container-registry` CLI plug-in by running the following command:

```
$ ibmcloud plugin install container-registry
```

For more information about installing plug-ins, see [Extending IBM Cloud CLI with plug-ins](#).

Set up a namespace

Create a [namespace](#). The namespace is created in the [resource group](#) that you specify so that you can configure access to resources within the namespace at the resource group level. If you don't specify a resource group, and you don't target a resource group, the default resource group is used. Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console.

1. Log in to IBM Cloud.

```
$ ibmcloud login
```



Tip: If you have a federated ID, use `ibmcloud login --sso` to log in. Enter your username and use the provided URL in your CLI output to retrieve your one-time passcode. If you have a federated ID, the login fails without the `--sso` and succeeds with the `--sso` option.



Note: You don't need to log in to Container Registry until you want to push an image, see [Step 5: Push images to your namespace](#).

2. Add a namespace to create your own image [repository](#). Replace `<my_namespace>` with your preferred namespace.



Tip: The namespace must be unique across all IBM Cloud accounts in the same region. Namespaces must have 4 - 30 characters, and contain lowercase letters, numbers, hyphens (-), and underscores (_) only. Namespaces must start and end with a letter or number.

```
$ ibmcloud cr namespace-add <my_namespace>
```

You can put the namespace in a resource group of your choice by using one of the following options.

- Before you create the namespace, run the `ibmcloud target -g <resource_group>` command, where `<resource_group>` is the resource group.
- Specify the resource group by using the `-g` option on the `ibmcloud cr namespace-add` command.



Tip: If you have a problem when you try to create a namespace, see [Why can't I add a namespace?](#) for assistance.

3. To ensure that your namespace is created, run the `ibmcloud cr namespace-list` command.

```
$ ibmcloud cr namespace-list -v
```

Pull images from a registry to your local computer


1. Install Docker or a tool of your choice, such as Podman.

- Install the [Docker Engine CLI](#). For Windows® 8, or OS X Yosemite 10.10.x or earlier, install [Docker Desktop](#) instead. For more information about the version of Docker that is supported by IBM Cloud Container Registry, see [Support for Docker](#).
 - Install [Podman](#).
2. Download (*pull*) the image to your local computer. Replace `<source_image>` with the repository of the image and `<tag>` with the [tag](#) of the image that you want to use, for example, `latest`. For example, depending on the tool that you are using, run one of the following commands.
- If you are using Docker, run the following command.

```
$ docker pull <source_image>:<tag>
```

Example, where `<source_image>` is `hello-world` and `<tag>` is `latest`:

```
$ docker pull hello-world:latest
```

 **Tip:** If you have problem when you try to pull a Docker image, see [Why can't I push or pull a Docker image?](#) for assistance. If you can't pull the most recent image by using the `latest` tag, see [Why can't I pull the newest image by using the latest tag?](#) for assistance.

- If you are using Podman, run the following command.


```
$ podman pull <source_image>:<tag>
```

Example, where `<source_image>` is `hello-world` and `<tag>` is `latest`:

```
$ podman pull hello-world:latest
```

Tag the image

To tag the image, replace `<source_image>` with the repository and `<tag>` with the tag of your local image that you pulled earlier. Replace `<region>` with the name of your [region](#). Replace `<my_namespace>` with the namespace that you created in [Set up a namespace](#). Define the repository and tag of the image that you want to use in your namespace by replacing `<new_image_repo>` and `<new_tag>`. For example, depending on the tool that you are using, run one of the following commands.

 **Tip:** To find the name of your region, run the [ibmcloud cr region](#) command.

- If you are using Docker, run the following command.

```
$ docker tag <source_image>:<tag> <region>.icr.io/<my_namespace>/<new_image_repo>:<new_tag>
```

Example, where `<source_image>` is `hello-world`, `<tag>` is `latest`, `<region>` is `uk`, `<my_namespace>` is `namespace1`, `<new_image_repo>` is `hw_repo`, and `<new_tag>` is `1`:

```
$ docker tag hello-world:latest uk.icr.io/namespace1/hw_repo:1
```

- If you are using Podman, run the following command.

```
$ podman tag <source_image>:<tag> <region>.icr.io/<my_namespace>/<new_image_repo>:<new_tag>
```

Example, where `<source_image>` is `hello-world`, `<tag>` is `latest`, `<region>` is `uk`, `<my_namespace>` is `namespace1`, `<new_image_repo>` is `hw_repo`, and `<new_tag>` is `1`:

```
$ podman tag hello-world:latest uk.icr.io/namespace1/hw_repo:1
```

Push images to your namespace

1. Log in to IBM Cloud Container Registry by using one of the following options.
- To log in by using Docker, run the `ibmcloud cr login` command to log your local Docker daemon in to IBM Cloud Container Registry.

```
$ ibmcloud cr login --client docker
```

- To log in by using Podman, run the `ibmcloud cr login` command to log in to IBM Cloud Container Registry.

```
$ ibmcloud cr login --client podman
```

- To log in by using other clients, see [Accessing your namespaces interactively](#).



Tip: If you have a problem when you try to log in, see [Why can't I log in to Container Registry?](#) for assistance.

2. Upload (*push*) the image to your namespace. Replace `<my_namespace>` with the namespace that you created in [Set up a namespace](#). Replace `<image_repo>` and `<tag>` with the repository and the tag of the image that you chose when you tagged the image. For example, depending on the tool that you are using, run one of the following commands.

- If you are using Docker, run the following command.

```
$ docker push <region>.icr.io/<my_namespace>/<image_repo>:<tag>
```

Example, where `<region>` is `uk`, `<my_namespace>` is `namespace1`, `<image_repo>` is `hw_repo`, and `<tag>` is `1`:

```
$ docker push uk.icr.io/namespace1/hw_repo:1
```



Tip: If you have a problem when you try to push a Docker image, see [Why can't I push or pull a Docker image?](#) for assistance.

- If you are using Podman, run the following command.

```
$ podman push <region>.icr.io/<my_namespace>/<image_repo>:<tag>
```

Example, where `<region>` is `uk`, `<my_namespace>` is `namespace1`, `<image_repo>` is `hw_repo`, and `<tag>` is `1`:

```
$ podman push uk.icr.io/namespace1/hw_repo:1
```

Verify that the image was pushed

Verify that the image was pushed successfully by running the following command.

```
$ ibmcloud cr image-list
```

You set up a namespace in IBM Cloud Container Registry and pushed your first image to your namespace.

Next steps in Container Registry

- [Manage image security with Vulnerability Advisor.](#)
- [Review your service plans.](#)
- [Store and manage more images in your namespace.](#)
- [Define access policies.](#)
- [Set up clusters and worker nodes.](#)

About Container Registry

Use IBM Cloud® Container Registry to store and access private container images in a highly available and scalable architecture.

IBM Cloud Container Registry provides a multi-tenant, highly available, scalable, and encrypted private image *registry* that is hosted and managed by IBM. You can use Container Registry by setting up your own image *namespace* and pushing container images to your namespace.

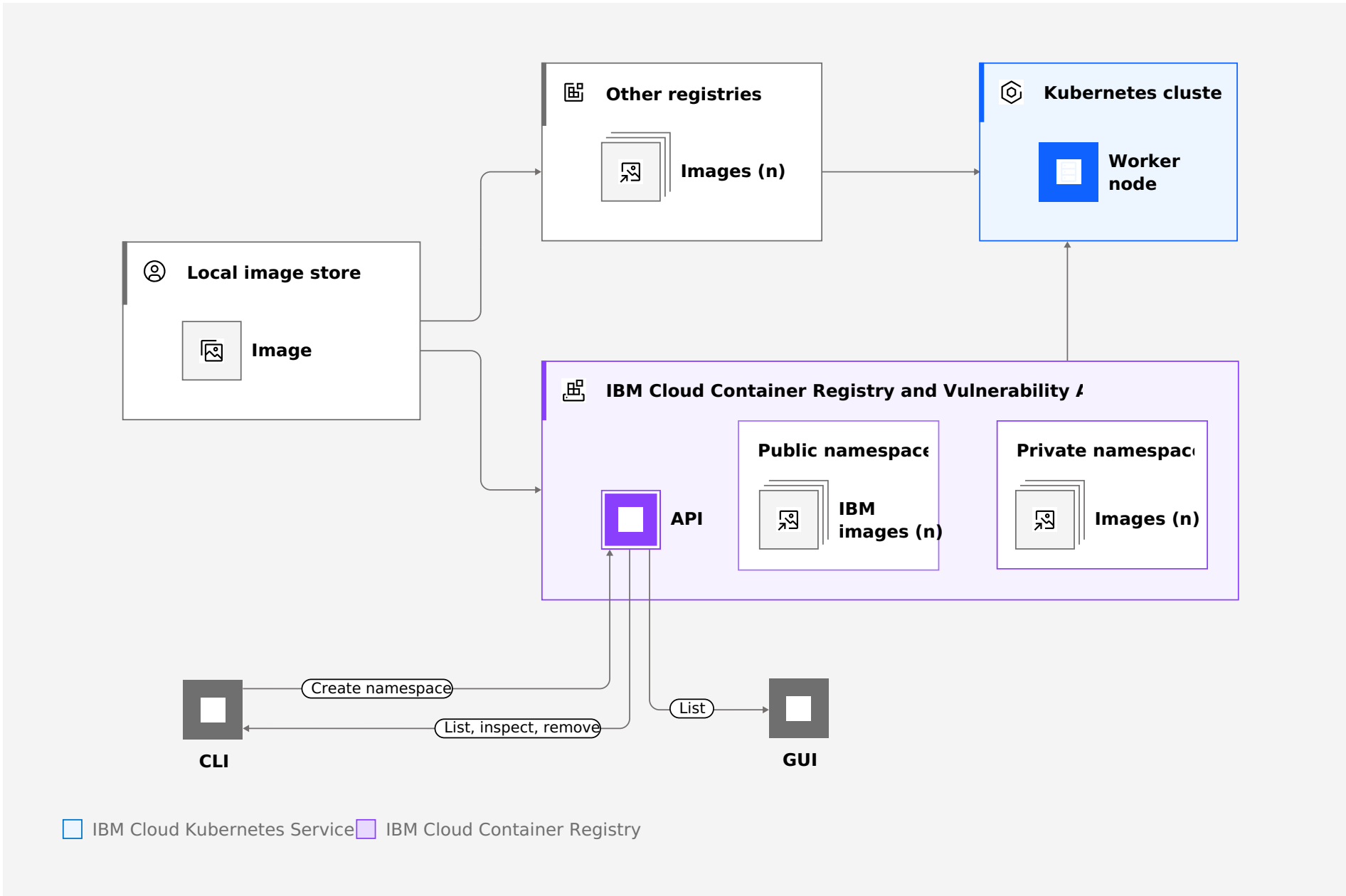


Figure 1. How Container Registry interacts with your images

A Docker image is the basis for every container that you create. An image is created from a *Dockerfile*, which is a file that contains instructions to build the image. A Dockerfile might reference build artifacts in its instructions that are stored separately, such as an app, the configuration of the app, and its dependencies. Images are typically stored in a registry that can either be accessible by the public (public registry) or set up with limited access for a small group of users (private registry). By using Container Registry, only users with access to your IBM Cloud account can access your images.

When you push images to Container Registry, you benefit from the built-in Vulnerability Advisor features that scan for potential security issues and vulnerabilities. Vulnerability Advisor checks for vulnerable packages in specific Docker base images, and known vulnerabilities in app configuration settings. When vulnerabilities are found, information about the vulnerability is provided. You can use this information to resolve security issues so that containers are not deployed from vulnerable images.

Review the following table to find an overview of benefits of using Container Registry.

Benefit	Description
Highly available and scalable private registry.	Set up your own image namespace in a multi-tenant, highly available, scalable, encrypted private registry that is hosted and managed by IBM. Store your private Docker images and share them with users in your IBM Cloud account.
Image security compliance with Vulnerability Advisor.	Benefit from automatic scanning of images in your namespace. Review recommendations that are specific to the operating system to fix potential vulnerabilities and protect your containers from being compromised.
Quota limits for storage and pull traffic.	Benefit from free storage and pull traffic to your private images until you reach your free quota. Set custom quota limits for the amount of storage and pull traffic per month to avoid exceeding your preferred payment level.

Table 1. Container Registry benefits

Service plans

You can choose between the free or standard Container Registry service plans to store your Docker images and make these images available to users in your IBM Cloud account.

The IBM Cloud Container Registry service plan determines the amount of storage and pull traffic that you can use for your private images. The service plan is associated with your IBM Cloud account, and limits for storage and image pull traffic apply to all namespaces that you set up in your account.



Note: Service plans are scoped to the specific registry instance (one of the regional registries or the global registry) that you're currently working with. Plan settings must all be managed separately for your account in each registry instance. For more information, see [Regions](#).

The following table shows available IBM Cloud Container Registry service plans and their characteristics. For more information about how billing works and what happens when you exceed service plan limits, see [Quota limits and billing](#).

Characteristics	Free	Standard
Description.	Try out Container Registry to store and share your Docker images. This plan is the default service plan when you set up your first namespace in Container Registry.	Benefit from unlimited storage and pull traffic usage to manage the Docker images for all namespaces in your IBM Cloud account.
Amount of storage for images.	500 MB	Unlimited
Pull traffic.	5 GB per month	Unlimited
Billing.	If you exceed your storage or pull traffic limits, you cannot push or pull images to and from your namespace. For more information, see Quota limits and billing .	Storage. You are charged by Gigabyte-Months of usage. The first 0.5 GB-Months are free. Then, you are charged as stated in the offering details page, see Container Registry . Pull traffic. You are charged by Gigabyte usage per month. The first 5 GB are free. Then, you are charged as stated in the offering details page, see Container Registry . If you exceed your storage or pull traffic limits, you can't push or pull images to and from your namespace. For more information about storage, pull traffic, and the cost estimator, see Quota limits and billing .

Table 2. Container Registry plans

Quota limits and billing

Find information and examples for how the billing process and quota limits work in Container Registry.

Every image is built from a number of layers that each represent an incremental change from the base image. When you push or pull an image, the amount of storage and pull traffic that is needed for each layer is added to your monthly usage. Identical layers are automatically shared between images in your IBM Cloud account and are reused when you create other images. The storage for each identical layer is charged only once, regardless of how many images in your account reference the layer. Layers that are only referenced by deleted images in the trash are not charged.



Note: From 1 February 2022, both [tagged](#) and [untagged](#) images are charged for.



Note: Quota limits and billing are scoped to the specific registry instance (one of the regional registries or the global registry) that you're currently working with. Quota settings must be managed separately for your account in each registry instance. For more information, see [Regions](#).



Note: Pull traffic across public connections counts toward usage and quota. Pull traffic across private connections doesn't count.

The following example is for pushing images:

You push an image to your namespace that is based on the Ubuntu image. The Ubuntu image contains several layers. Because you do not have these layers in your account yet, the amount of storage that these layers require is added to your monthly usage.

Later, you create a second image that is based on the Ubuntu image. You change the Ubuntu base image, such as by adding more commands or files to your Dockerfile. Each change represents a new image layer. When you push the second image, Container Registry recognizes that all layers of the

base Ubuntu image are already stored in your account. You're not charged for storing these layers a second time, even if you pushed your image to another namespace. Container Registry determines the size of all new layers and adds the amount of storage to your monthly usage.

Billing for storage and pull traffic

Depending on the service plan that you choose, you are charged for the storage and pull traffic that you use per month in each region.

Storage charges

Every IBM Cloud Container Registry service plan comes with a certain amount of storage that you can use to store your Docker images in the namespaces of your IBM Cloud account. If you're on the standard plan, you are charged by GB-Months of usage. The first 0.5 GB-Months are free. If you're on the free plan, you can store your images in Container Registry for free until you reach the quota limits for the free plan. A GB-Month is an average of 1 GB of storage for a month (730 hours).

The following example is for the standard plan:

You use 5 GB for exactly half the month and then you push several images to your namespace and use 10 GB for the rest of the month. Your monthly usage is calculated as shown in the following example:

$$(5 \text{ GB} \times 0.5 \text{ (months)}) + (10 \text{ GB} \times 0.5 \text{ (months)}) = 2.5 + 5 = 7.5 \text{ GB-Months}$$

In the standard plan, the first 0.5 GB-Months are free, so you get charged for 7 GB-Months (7.5 GB-Months - 0.5 GB-Months).

Pull traffic charges

Every IBM Cloud Container Registry service plan includes a certain amount of free pull traffic to your private images that are stored in your namespace. Pull traffic is the bandwidth that you use when you pull a layer of an image from your namespace to your local computer. If you're on the standard plan, you're charged by GB of usage per month. The first 5 GB each month is free. If you're on the free plan, you can pull images from your namespace until you reach the quota limit for the free plan.



Note: Pull traffic across public connections counts toward usage and quota. Pull traffic across private connections doesn't count.

The following example is for the standard plan:

In the month, you pulled images that contain layers with a total size of 14 GB. Your monthly usage is calculated as shown in the following example:

In the standard plan, the first 5 GB per month is free, so you get charged for 9 GB (14 GB - 5 GB).

Quota limits for storage and pull traffic

Depending on the service plan that you choose, you can push and pull images to and from your namespace until you reach your plan-specific or custom quota limits for each region.

Storage quota limits

When you reach or exceed the quota limits for your plan, you can't push any images to the namespaces in your IBM Cloud account until you complete one of the following tasks.

- [Free up space by removing images](#) from your namespaces.
- [Upgrade to the standard plan](#).
- If you set quota limits for storage in your free or standard plan, you can also [increase this quota limit](#) to enable the pushing of new images again.

The following example is for the standard plan:

Your current quota limit for storage is set to 1 GB. All private images that are stored in the namespaces of your IBM Cloud account already use 900 MB of this storage. You have 100 MB storage available until you reach your quota limit. One user wants to push an image with a size of 2 GB on the local computer. Because the quota limit is not yet reached, Container Registry allows the user to push this image.

After the push, Container Registry determines the actual size of the image in your namespace, which can vary from the size on your local computer, and checks whether the limit for storage is reached. In this example, the storage usage increases from 900 MB by 2 GB. With your current quota limit

set to 1 GB, Container Registry prevents you from pushing more images to the namespace.

Pull traffic quota limits

When you reach or exceed the quota limits for your plan, you can't pull any images from the namespaces in your IBM Cloud account until you complete one of the following tasks.

- Wait for the next billing period to start.
- [Upgrade to the standard plan.](#)
- [Increase your quota limits for pull traffic.](#)



Note: Pull traffic across public connections counts toward usage and quota. Pull traffic across private connections doesn't count.

The following example is for the standard plan:

In the month, your quota limit for pull traffic is set to 5 GB. You already pulled images from your namespaces and used 4.5 GB of this pull traffic. You have 0.5 GB pull traffic available until you reach your quota limit. One user wants to pull an image from your namespace with a size of 1 GB. Because the quota limit is not yet reached, Container Registry allows the user to pull this image.

After the image is pulled, Container Registry determines the bandwidth that you used during the pull and checks whether the limit for pull traffic is reached. In this example, the pull traffic usage increases from 4.5 GB to 5.5 GB. With your current quota limit set to 5 GB, Container Registry prevents you from pulling images from your namespace.

Cost of Container Registry

You can see the costs of IBM Cloud Container Registry in the pricing plans section of the offering details page, see [Container Registry](#).

Upgrading your service plan

You can upgrade your service plan to benefit from unlimited storage and pull traffic usage to manage the Docker images for all namespaces in your IBM Cloud account.



Tip: If you want to find out what service plan you have for the registry region that you're targeting, run the `ibmcloud cr plan` command.

To upgrade your service plan, complete the following steps.

1. Log in to IBM Cloud.

```
$ ibmcloud login
```



Tip: If you have a federated ID, use `ibmcloud login --sso` to log in to the IBM Cloud CLI. Enter your username and use the provided URL in your CLI output to retrieve your one-time passcode. If you have a federated ID, the login fails without the `--sso` and succeeds with the `--sso` option.

2. Target the region for which you want to upgrade the plan.

```
$ ibmcloud cr region-set
```

For more information, see [ibmcloud cr region-set](#) and [Regions](#).

3. Upgrade to the standard plan.

```
$ ibmcloud cr plan-upgrade standard
```



Tip: If you have an IBM Cloud lite plan, you must upgrade to an IBM Cloud Pay-as-you-go or Subscription account before you run `ibmcloud cr plan-upgrade`.

For more information, see [ibmcloud cr plan-upgrade](#).

Terms that are used in IBM Cloud Container Registry

Information about the terms that are used in IBM Cloud Container Registry.

To learn more about Docker-specific terms, see [Docker glossary](#).

Container image

A file system and its execution parameters that are used within a container runtime to create a container. The file system consists of a series of layers, which are combined at run time, that are created as the container image is built by successive updates. The container image does not retain state as the container runs.

Container images are stored in a repository that is stored in a namespace.

Digest

Digests are used as immutable references to various objects in the registry such as image manifests, layers, and configuration items.

In the context of the registry, an image digest is an immutable reference to an image that identifies an image by using the `sha256` hash of the [image manifest](#). You can use an image digest to ensure that you always reference the same version of an image. Use the long format of the image digest to work with images, such as pulling, pushing, and deleting images.

To find the image digest, run the `ibmcloud cr image-digests` command. The `ibmcloud cr image-list` command also returns the image digest, but, by default, it is in a truncated format. You can add an option to the `ibmcloud cr image-list` command to return the image digest in the long format.



Tip: When you are using the image digest to identify an image, always use the long format.



Note: In Container Registry, any reference to "digest" means "image digest".

Dockerfile

A Dockerfile is a text file that contains instructions to build a Docker image.

Typically, a container image is built upon a base image that contains a base operating system, such as Ubuntu. You can incrementally change the base image with your Dockerfile instructions to define the environment that the app needs to run. Every change to the base image describes a new layer of the image, and you can make multiple changes in a single Dockerfile line. The instructions in a Dockerfile also might reference build artifacts that are stored separately, such as an app, the configuration of the app, and its dependencies. For more information about Dockerfile, see [Dockerfile reference](#).

Docker V2 container images

A container image that is compliant with the [Docker: Image Manifest V2, schema 2](#) specification.

The media type for Docker Image Manifest V2, schema 2 is `application/vnd.docker.distribution.manifest.v2+json` and the media type for the manifest list is `application/vnd.docker.distribution.manifest.list.v2+json`. A Docker V2 container image is a type of OCI container image. For more information about support for Docker, see [Docker](#).

Domain name

The name of a host system. A domain name consists of a sequence of subnames that are separated by a delimiter character, for example, `www.ibm.com`.

The domain names that Container Registry uses are in the format `us.icr.io`. Earlier domain names that Container Registry used are in the format `registry.ng.bluemix.net`. Both formats of domain name refer to the same registry and content. The Container Registry service responds to earlier and canonical domain names equally. You can push or pull images by using either domain name interchangeably.

The domain name is only significant in the following situations:

- When Kubernetes is selecting a pull-secret, it chooses one that matches the domain name.
- When `ibmcloud cr login` is helping you to log in, it uses domain names in the format `us.icr.io` only.
- When images are signed, the signature includes the domain name that was used at the time of signing.

For more information about the domain names that Container Registry uses, see [Regions](#).

Image manifest

An image manifest is a `.json` document that references the configuration object and image layers that are required to pull and run the image. The `sha256` hash of the image manifest is the `digest`, which is used to identify the image. You can view the image manifest by running the `ibmcloud cr manifest-inspect` command.

OCI container images

A container image that is compliant with the [OCI Image Format](#) specification.

The media type for OCI container images is `application/vnd.oci.image.manifest.v1+json`.

Registry

A public or private container image storage and distribution service.

Storage is provided for *OCI container images* (also known as Docker container images). OCI container images can be accessed or "pulled" by OCI clients that use the appropriate registry domain name. Container images can be accessed by anyone (public images) or access can be limited to a group (private images). Container Registry provides a multi-tenant, highly available, private image registry that is hosted and managed by IBM. You can use the registry by adding a namespace that is private to your account and then push images to your namespace.

Registry namespace

A collection of repositories that store your container images in Container Registry. The registry namespace is associated with your IBM Cloud account. You can have multiple registry namespaces in an account.

A registry namespace is made up of one or more repositories.

When you set up your own namespace in Container Registry, the namespace is appended to the registry URL, `<region>.icr.io/my_namespace`. The namespace must be unique across all IBM Cloud accounts in the same region. Every user in your IBM Cloud account can view and work with images that are stored in your registry namespace.



Note: You can have 100 namespaces in each region.

Namespaces are created in a [resource group](#) that you specify so that you can configure access to resources within the namespace at the resource group level. If you don't specify a resource group, and a resource group isn't targeted, the default resource group is used. If you have an older namespace that is not in a resource group, you can assign it to a resource group and then set permissions for that namespace at the resource group level. For more information about resource groups, see [Assigning existing namespaces to resource groups](#).

Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console.

Repository

A collection of related container images in the container registry that are distinguished by tag or digest only.

Repository is often used interchangeably with container image, but a repository potentially holds multiple tagged variants of a container image.

A repository stores container images, and is itself stored in a registry namespace.

Tag

An identifier that is attached to container images within a repository. Tags can be reassigned or deleted from images.

You can use *tags* to distinguish different versions of the same base image within a repository. When you run a Docker command and do not specify the tag of a repository image, then the image tagged `latest` is used by default.

Untagged image

An image that has no *tag* is an untagged image. Images that are untagged can be referenced by using the digest reference format `<repository>@<digest>` as opposed to the tag reference format `<repository>:<tag>`. Untagged images are typically the result of an image that is pushed with a pre-existing `<repository>:<tag>` combination. In this case, the tag is overwritten and the original image becomes untagged.

You can view untagged images by using the [ibmcloud cr image-digests](#) command, and clean up untagged images by using the [ibmcloud cr image-prune-untagged](#) command.

Regions

The default instance of Container Registry is the global registry. The global registry doesn't include a region in its [domain name](#) (`icr.io`).

Use the global instance of the registry unless you have a specific requirement, for example, data sovereignty, to store your data in a particular region. In which case, you can use Container Registry in [local regions](#).

All registry artifacts are scoped to the specific registry instance (one of the regional registries or the global registry) that you are currently working with. For example, namespaces, images, quota settings, and plan settings must all be managed separately for your account in each registry instance.

Global registry


A global registry is available. The global registry doesn't include a region in its name (`icr.io`). In addition to hosting user namespaces and images, this registry also hosts public images that are provided by IBM.

The global instance of Container Registry is available by using the [domain names](#) that are shown in the following table.

Registry	Domain name	Private domain name	Deprecated domain name
Global	<code>icr.io</code>	<code>private.icr.io</code>	<code>registry.bluemix.net</code>

Table 3. Domain name for the global registry

To learn about connecting to Container Registry by using the private domain names, see [Using private network connections](#).

 **Deprecated:** The existing `bluemix.net` domain names are deprecated, but you can continue to use them for the moment. An end of support date is not available yet.


Targeting the global registry

You can target the global registry by running the `ibmcloud cr region-set` command.

1. To target the global registry, run the following command.

```
$ ibmcloud cr region-set global
```

2. To log your local Docker daemon into the global registry, run the `ibmcloud cr login` command.

 **Tip:** Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).


Local regions

Regional instances of Container Registry are available by using the [domain names](#) that are shown in the following table.

Local registry region	Domain name	Private domain name	Deprecated domain name
ap-north	<code>jp.icr.io</code>	<code>private.jp.icr.io</code>	Not applicable
ap-south	<code>au.icr.io</code>	<code>private.au.icr.io</code>	<code>registry.au-syd.bluemix.net</code>
br-sao	<code>br.icr.io</code>	<code>private.br.icr.io</code>	Not applicable
ca-tor	<code>ca.icr.io</code>	<code>private.ca.icr.io</code>	Not applicable
eu-central	<code>de.icr.io</code>	<code>private.de.icr.io</code>	<code>registry.eu-de.bluemix.net</code>
jp-osa	<code>jp2.icr.io</code>	<code>private.jp2.icr.io</code>	Not applicable
uk-south	<code>uk.icr.io</code>	<code>private.uk.icr.io</code>	<code>registry.eu-gb.bluemix.net</code>
us-south	<code>us.icr.io</code>	<code>private.us.icr.io</code>	<code>registry.ng.bluemix.net</code>

Table 4. Domain names for local regions

To learn about connecting to Container Registry by using the private domain names, see [Using private network connections](#).

 **Deprecated:** The existing `bluemix.net` domain names are deprecated, but you can continue to use them for the moment. An end of support date is not available yet.

Targeting a local region

If you want to use a region other than your local region, you can target the region that you want to access by running the `ibmcloud cr region-set`

command. You can run the command with no options to get a list of available regions, or you can specify the region as an option.

1. To run the command with options, replace `<region>` with the name of the [region](#).

```
$ ibmcloud cr region-set <region>
```

For example, to target the `eu-central` region, run the following command.

```
$ ibmcloud cr region-set eu-central
```

2. To log your local Docker daemon into the registry so that you can push or pull images, run the [ibmcloud cr login](#) command.



Tip: Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).

Supported clients

Support for Docker

IBM Cloud Container Registry supports versions of Docker Engine that Docker supports.

Docker is required only if you want to push or pull images.

Docker V2 schema 2 images are supported. Manifest lists are also supported. For more information, see [Docker: Image Manifest Version 2, schema 2](#).



Note: Docker V2 schema 1 images are discontinued and you can't push them to Container Registry anymore.

Support for other clients

IBM Cloud Container Registry supports supported versions of clients that are compliant with the OCI Distribution spec version 1, or later, such as Buildah, Podman, and Skopeo.

IBM Cloud Container Registry architecture and workload

IBM Cloud® Container Registry is a multi-tenant, highly available, scalable, and encrypted private image [registry](#) that is hosted and managed by IBM.

Both the control plane (management of images and configuration) and data plane (pushing and pulling your images) are multi-tenant. All parts of the service are hosted in an IBM service account, which is not shared with users or other services.

In each regional instance of the [registry](#), the service runs in three physically separate data centers to ensure availability. All data and the configuration for each instance of the registry is retained within the region in which it is hosted. The global instance is also hosted in physically separate data centers. The data centers might not be in the same region as each other. For more information about regions, see [Regions](#).

IBM Cloud Container Registry runs in IBM Cloud Kubernetes Service clusters, and uses IBM Cloud Object Storage to store images. Image data in IBM Cloud Object Storage is encrypted at rest.

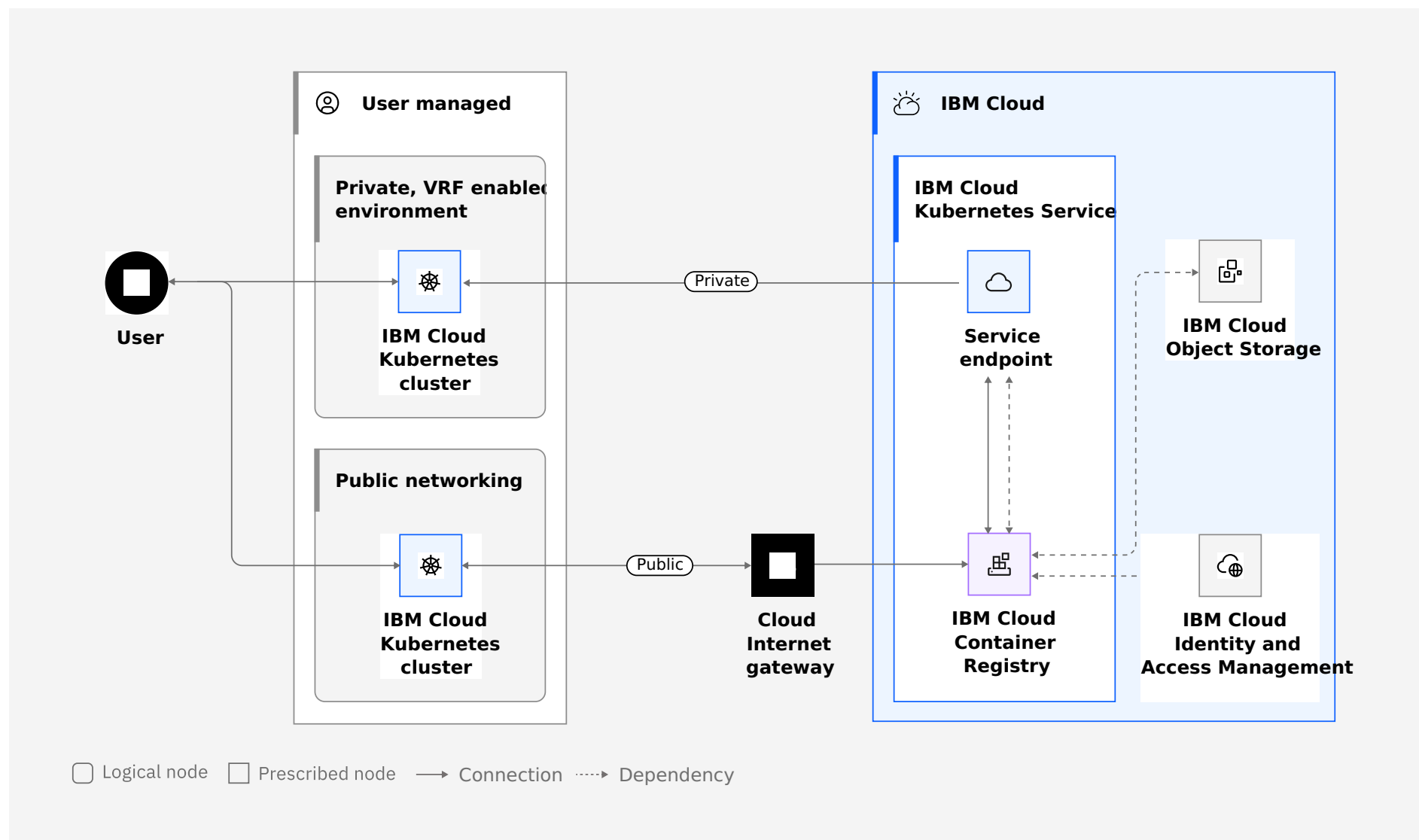


Figure 1. Diagram showing deployment

Segmentation of data

Segmentation of data within IBM Cloud Container Registry is achieved by using private [namespaces](#), which are strictly owned by single accounts.

You can control access to [namespaces](#) within the account by using Cloud Identity and Access Management (IAM) access policies. Storage in IBM Cloud Object Storage is not segmented, but user accounts do not have direct access to the IBM Cloud Object Storage that contains the image data. For more information, see [Managing IAM access for IBM Cloud Container Registry](#).

All traffic to the registry, and from the service to IBM Cloud Container Registry dependencies is encrypted in transit. No additional network level segmentation of traffic is provided. The control plane and data plane are not separated from each other.

Private connections

You can decide whether your data plane interactions use private connections. Additionally, you can choose to prohibit public data plane connections for your account.

The flow of all customer data between IBM Cloud Container Registry and its dependencies uses private network connections. For more information about private connections, see [Securing your connection to IBM Cloud Container Registry](#).

Public IBM images

You can access the images that are provided by IBM by using the IBM Cloud® Container Registry command line.



Note: You can't access the public IBM images by using the IBM Cloud console anymore.

Accessing the public IBM images by using the CLI

You can access the public IBM images by using the command line.

Before you begin, complete the following tasks.

1. Ensure that the IBM Cloud Container Registry CLI is installed, see [Installing the container-registry CLI plug-in](#).
2. Log in to [IBM Cloud](#).

```
$ ibmcloud login
```

To list the public images, complete the following steps.

1. Target the global registry:

```
$ ibmcloud cr region-set global
```

2. List the IBM public images.

```
$ ibmcloud cr images --include-ibm
```

Notifications about Container Registry

Notifications about changes that affect IBM Cloud® Container Registry and Vulnerability Advisor.

Notifications topics

Notifications about changes.

- [Upcoming public networking changes for Container Registry](#).
- [Update Vulnerability Advisor to version 4 by 19 June 2023](#).
- [Changes to Container Registry VPE gateways from 11 November 2022](#).
- [Changes to private IP addresses from 15 December 2022](#).
- [Container Registry CLI stops returning security status results in lists by default from version 1.0.0](#).
- [Container Registry private IP addresses changed on 5 July 2022](#).
- [IAM access policies are required from 5 July 2022](#).
- [Minimum supported Docker version from 1 March 2022](#).

Notifications about Container Registry

Notifications about changes that affect IBM Cloud® Container Registry and Vulnerability Advisor.

Notifications topics

Notifications about changes.

- [Upcoming public networking changes for Container Registry](#).
- [Update Vulnerability Advisor to version 4 by 19 June 2023](#).
- [Changes to Container Registry VPE gateways from 11 November 2022](#).
- [Changes to private IP addresses from 15 December 2022](#).
- [Container Registry CLI stops returning security status results in lists by default from version 1.0.0](#).
- [Container Registry private IP addresses changed on 5 July 2022](#).
- [IAM access policies are required from 5 July 2022](#).
- [Minimum supported Docker version from 1 March 2022](#).

Upcoming public networking changes for Container Registry

Users that use the public network to access IBM Cloud® Container Registry are advised to use a wildcard domain, such as `*.icr.io`, in their firewalls rules. The use of a wildcard domain accommodates future changes that might issue a redirect for certain requests such as an image layer download for traffic optimization.



Note: This advice is a change from existing advice that states that you must allow only the public domain of the Container Registry that you are using.

Update Vulnerability Advisor to version 4 by 19 June 2023

The Vulnerability Advisor component of IBM Cloud® Container Registry is being updated. From 19 June 2023, Vulnerability Advisor version 3 will be replaced as the default by Vulnerability Advisor version 4.

Vulnerability Advisor version 3 is being deprecated as the default on 19 June 2023. From 19 June 2023, the default will be Vulnerability Advisor version 4. If you have version 3 set as the default, you can continue to use version 3 until the end of support date. An end of support date is not available yet.

What you need to know about this change

If you use the IBM Cloud console to access Vulnerability Advisor, no action is required. The IBM Cloud console is automatically updated to Vulnerability Advisor version 4.

If you use the IBM Cloud CLI and you want to use version 4 as the default, you must update the Container Registry CLI plug-in to version 1.0.0, or later, by 19 June 2023. Updating the Container Registry CLI plug-in to version 1.0.0, or later, enables the `ibmcloud cr va` command and the `--va` option on the `ibmcloud cr images` and `ibmcloud cr digests` commands to work with Vulnerability Advisor version 4.

On 19 June 2023, when the default changes to Vulnerability Advisor version 4, the Container Registry CLI automatically starts to use this version unless the `ibmcloud cr va-version-set v3` command was run, in which case Vulnerability Advisor version 3 continues to be used. You can use the `ibmcloud cr va-version` command to determine which Vulnerability Advisor version is being used and the `ibmcloud cr va-version-set v4` command to switch to Vulnerability Advisor version 4. When Vulnerability Advisor version 3 reaches its end of support date, any Container Registry CLI commands that access Vulnerability Advisor version 3 cease to work. An end of support date is not available yet.

If you use the Vulnerability Advisor REST API to access Vulnerability Advisor, you must update your client call from `/va/api/v3` APIs to `/va/api/v4` APIs.

If you use one of the Vulnerability Advisor version 3 SDKs to access Vulnerability Advisor, you must update to the Vulnerability Advisor version 4 SDK.

Any exemptions that you previously defined continue to work. However, the security notice value that comes back in Vulnerability Advisor version 4 might not be the same as for Vulnerability Advisor version 3 because different sources of data are used. Therefore, if the returned value isn't the same as for Vulnerability Advisor version 3, you might have to update any existing exemptions that specify a security notice. Red Hat® security notices are unaffected. Exemptions that are defined by CVE value are also unaffected.

Differences in Vulnerability Advisor version 4 behavior are documented in [About Vulnerability Advisor](#).

What actions you must take by 19 June 2023

You can choose whether to update to use version 4, the default, or to continue to use version 3, which is deprecated.

- If you want to use Vulnerability Advisor version 4 as the default, update the following items as described in [What you need to know about this change](#):
 - The Container Registry CLI plug-in and, if you have explicitly run the `ibmcloud cr va-version-set v3` command previously, run the following command.
- If you want to continue to use Vulnerability Advisor version 3, run the following command:

```
$ ibmcloud cr va-version-set v4
```

- Any code that calls Vulnerability Advisor version 3 either through the API or through the SDK.
- You might have to update any existing exemptions that specify a security notice.

```
$ ibmcloud cr va-version-set v3
```

Changes to Container Registry VPE gateways from 11 November 2022

If you're using an IBM Cloud® Container Registry virtual private endpoint (VPE) gateway that was created before 11 November 2022, you must re-create your VPE gateway before 15 December 2022. If you use Cloud Identity and Access Management restricted IP address lists and Container Registry VPE gateways, you must also update your restricted IP address lists.

What you need to know about this change

On 11 November 2022, virtual private endpoints (VPEs) for IBM Cloud Container Registry are being updated and the existing VPE version is being deprecated on 15 December 2022. If you use Container Registry VPE gateways, you must create new VPE gateways and remove your VPE gateways that were created before 11 November 2022 at the earliest opportunity so that you pick up these changes. VPE gateways that were created before 11 November 2022 are deprecated and will not work after 15 December 2022.

If you create a new Container Registry VPE gateway after 11 November 2022 and also use Cloud Identity and Access Management (IAM) [restricted IP address lists](#), you must ensure that your restricted IP address list contains the Cloud Service Endpoint (CSE) source IP addresses of the VPCs in which your Container Registry VPE gateways exist. This requirement is related to a previous change to how Container Registry works over the private network that the new VPE version also uses, see [Container Registry private IP addresses changed on 5 July 2022](#).

How you benefit from this change

Container Registry VPEs are being updated so that they can access all Container Registry service regions by using a single VPE gateway. Previously, a Container Registry VPE gateway provided access to Container Registry only in the region in which the VPC and VPE gateway existed. If a user wanted to access another Container Registry region, they had to use the private registry domain names (for example, `private.us.icr.io`) and required extra VPC configuration for these domains.

With the new VPE version, VPC users can privately access all Container Registry regions by using a single Container Registry VPE gateway that uses a public domain name (for example, `us.icr.io`) regardless of the region in which the VPE gateway exists and without having to provide extra configuration.

Additionally, the new version of the VPE is in line with previous changes to private networking within Container Registry where the real source IP addresses of requests to the Container Registry are now maintained.

Previously, when connections came in over private networks, including through VPE gateways, the source IP addresses that you saw in IBM Cloud Activity Tracker and that were configured for IAM restricted IP address lists, were documented Container Registry IP addresses, see [Permit worker nodes to communicate with IBM Cloud Container Registry](#).

When you connect to Container Registry now, the real IP address of your VPC [Cloud service endpoint source addresses](#) is maintained in a request, which means that IAM restricted IP lists can be configured to specifically allow requests from your VPC, which improves security.

Understanding if you are impacted by this change

You are affected by this change if you are using a Container Registry VPE gateway that was created before 11 November 2022. VPE gateways created before this date can be identified by differences in their target CRN. You can find the target CRN of a VPE gateway by viewing the VPE gateway details in the IBM Cloud console (GUI) or CLI, see [Viewing details of an endpoint gateway](#).

VPE gateways that were created before 11 November 2022 are in the format:

```
crn:v1:bluemix:public:container-registry:<cloud-region>::endpoint:vpe.<cloud-region>.container-registry.cloud.ibm.com
```

VPE gateways that are created after 11 November 2022 are in the format:

```
crn:v1:bluemix:public:container-registry:<cloud-region>::endpoint:<registry-region-domain>
```

For a list of updated Container Registry VPE CRNs, see [Setting up a VPE for IBM Cloud Container Registry](#).

If you are using a Container Registry VPE gateway that was created before 11 November 2022 and also maintain IAM restricted IP address lists, you must change your restricted IP address list to contain your VPC Cloud Service Endpoint source addresses.

What actions you must take

- If you use a Container Registry VPE gateway that was created before 11 November 2022, you must re-create your Container Registry VPE gateway before 15 December 2022. If you don't re-create your VPE gateway, the VPE gateway will not connect to Container Registry after that date.

To replace the VPE gateway, complete the following steps:

1. Create a VPE gateway for Container Registry in the required VPC, see [Creating an endpoint gateway](#). If you're using the CLI, you might want to refer to the Container Registry VPE CRNs. For more information, see [Setting up a VPE for IBM Cloud Container Registry](#).
 2. Remove the VPE gateway that was created before 11 November 2022, see [Deleting an endpoint gateway](#).
- If you use a Container Registry VPE gateway that was created before 11 November 2022 and you use IAM restricted IP address lists, you must re-create your Container Registry VPE gateway and update your IAM restricted IP address lists before 15 December 2022. If you don't re-create your VPE gateway and update your IP address lists, the VPE gateway will not connect to Container Registry after that date.

To replace the VPE gateway and update your IAM restricted IP address lists, complete the following steps:

1. Update your IAM restricted IP address lists to access the [Cloud service endpoint source addresses](#) of your VPC. You can find the CSEs by viewing the details of your VPC in either the IBM Cloud console or CLI.
2. Create a VPE gateway for Container Registry in the required VPC, see [Creating an endpoint gateway](#). If you're using the CLI, you might want to

refer to the Container Registry VPE CRNs. For more information, see [Setting up a VPE for IBM Cloud Container Registry](#).

3. Remove the VPE gateway that was created before 11 November 2022, see [Deleting an endpoint gateway](#).

Changes to private IP addresses from 15 December 2022

The IBM Cloud® Container Registry private IP addresses that were replaced on 5 July 2022 are being decommissioned on 15 December 2022.

What you need to know about this change

IP addresses for accessing Container Registry over the private network changed on 5 July 2022, see [Container Registry private IP addresses changed on 5 July 2022](#). You continue to be able to use these old IP addresses, but they are due to be decommissioned on 15 December 2022. After this date, you will not be able to access Container Registry by using these IP addresses.

IBM Cloud Kubernetes Service cluster workers are configured so that you can access Container Registry over the private network even when the public domain is used. This process caches the DNS result on worker startup and is not refreshed until a cluster worker is reloaded, replaced, or rebooted. If the cluster workers have not been reloaded, replaced, or rebooted since 5 July 2022, it is likely that they still refer to the old Container Registry private IP addresses. If these workers are not reloaded or replaced, they will not function correctly after the old Container Registry IP addresses are decommissioned.

If you have any other custom configuration that uses the old Container Registry private IP addresses and you do not update your configuration to use the new Container Registry IP addresses, see [Permit worker nodes to communicate with IBM Cloud Container Registry](#), you might have a problem with accessing Container Registry over the private network.

What actions you need to take

You must ensure that any cluster workers that have not been reloaded, replaced, or rebooted since 5 July 2022 are reloaded or replaced before 15 December 2022. For more information about how to reload workers for classic clusters, see [ibmcloud ks worker reload](#). For more information about how to replace workers for VPC clusters, see [ibmcloud ks worker replace](#). You can also reboot affected workers but this action is not recommended due to the risks associated with rebooting. For more information about rebooting workers, see [ibmcloud ks worker reboot](#).

If you have any other custom configuration that uses the old Container Registry private IP addresses, such as for firewall rules or Calico policies, ensure that you update your configuration to use the new Container Registry IP addresses and remove the old IP addresses. Both the old and new private IP addresses are documented in the IBM Cloud Kubernetes Service documentation, see [Permit worker nodes to communicate with IBM Cloud Container Registry](#).

The IBM Cloud Kubernetes Service [Calico policy samples](#) will be updated to remove the old IP addresses on 15 December 2022. If you use these samples, ensure that you pull in the most recent samples after this date.

Container Registry CLI stops returning security status results in lists by default from version 1.0.0

From IBM Cloud® Container Registry CLI plug-in version 1.0.0, when you use the `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands to list images, they return Vulnerability Advisor security status results only if you use the `--va` option.

The `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands currently return the security status of each image by default. Some users store many images in the registry, and returning security status for every image when listing takes more processing time, which can lead to a poor experience and a smaller limit on the maximum number of images that can be listed.

To improve the user experience, a new major version, version 1.0.0, of the IBM Cloud Container Registry CLI plug-in changes the default behavior for the `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands so that they return Vulnerability Advisor security status results only if you use the `--va` option.

Action required now

If you want to continue to receive security status with your lists, prepare to upgrade by adding the new `--va` option to your commands.

If you require security status with your lists, add the `--restrict` option when you're using the `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands to receive just the information that you require.

If you are an API user and require security status with your lists, ensure that you are restricting the repositories that you are requesting to ensure that you receive list results quickly and reliably.

Container Registry private IP addresses changed on 5 July 2022

By 23 June 2022, if you connect to IBM Cloud® Container Registry over the private network and you use Cloud Identity and Access Management (IAM) restricted IP address lists, you must change your IAM restricted IP list. This change also affects you if you have allowlists or a firewall rule.



Note: On 23 June 2022, only the `br-sao` and `ca-tor` regions changed. The remaining regions changed on 5 July 2022. This change was originally due to take place on 23 May 2022 but was delayed.

What you need to know about this change

From 5 July 2022 (23 June 2022 for the `br-sao` and `ca-tor` regions), when connections are made to IBM Cloud Container Registry, the real source IP of the request is used. Previously, when connections came in over private networks, the source IP addresses that you saw in IBM Cloud Activity Tracker and that were configured for [IAM restricted IP address lists](#) were documented Container Registry [IP addresses](#).

This change improves the security of IBM Cloud Container Registry. With this change, you can configure real, account specific, private client IP addresses in IAM restricted IP lists, instead of the documented list of shared IP addresses. You must now allow private subnet and IP addresses of your own hosts (for example, worker nodes in a classic IBM Cloud Kubernetes Service cluster or the egress IP of a VPC network).

Also, as part of this change, the IBM Cloud Container Registry service private IP addressees changed, which might require updates to your firewall configuration.

If you use [Calico](#), the samples are updated to take account of the change.



Important: You must not remove the IBM Cloud Container Registry private IP addresses from your IAM restricted IP list until an announcement advises you to do so.



Important: You must take the [appropriate actions](#) before this change happens on 23 June 2022, otherwise your requests to Container Registry might fail to be authorized.

How you benefit from this change

This change increases security for any IBM Cloud Container Registry users that use private connections and IAM restricted IP address lists. After the change, you must configure the restricted IP address list to allow the private subnet and IP addresses of your own host. This change means that you can ensure Container Registry OAuth requests originate only from hosts that you own.

If you use IBM Cloud Activity Tracker, you can see the true source IP address in any audit logs, where previously you saw a Container Registry owned IP that was private.

Understanding if you are impacted by this change

You are impacted if you are accessing Container Registry over the private network.

You are accessing Container Registry over the private network if one of the following statements is true:

- You're using one of the `private.*` domains, for example, `private.us.icr.io`.
- You're using an IBM Cloud Kubernetes Service cluster in a [configuration](#) that automatically talks to the registry over a private connection.
- You're accessing Container Registry through a virtual private cloud (VPC) virtual private endpoint gateway (VPE gateway).
- You're using the Container Registry private IP addresses for configuring network access, for example, in firewalls or Access Control Lists (ACL).

If any of the previous statements is true when this change takes effect, the IP addresses in the IBM Cloud Activity Tracker logs change, but you don't need to do anything unless you are also using IAM IP address access restrictions.

What actions you need to take



Important: You must take the appropriate actions before 23 June 2022. If you don't make the appropriate updates, your requests to push and pull from Container Registry might fail.

Depending on which of the following scenarios you fit, take the appropriate action.

You access Container Registry over the private network and maintain a list of restricted IP addresses in IAM.

You must update your IAM restricted IP address list to include any IP addresses or subnets of hosts in your account that make requests to Container Registry. You must keep the current Container Registry private IP addresses in your restricted IP list until an announcement indicates it is safe to remove them. For more information about how to update a restricted IP address list, see [Allowing specific IP addresses](#) in the Cloud Identity and Access Management (IAM) documentation.

Your firewalls are configured with the Container Registry private IP addresses.

You must include the new private IP addresses in your firewall configuration. For more information about connecting to Container Registry over the private network, see [Securing your connection to Container Registry](#).

For more information about the new and current Container Registry private IP addresses, see the following topics:

- For IBM Cloud Kubernetes Service, see [Permit worker nodes to communicate with IBM Cloud Container Registry](#).
- For Red Hat® OpenShift® on IBM Cloud®, see [Permit worker nodes to communicate with IBM Cloud Container Registry](#).

IAM access policies are required from 5 July 2022

From 5 July 2022, to access IBM Cloud® Container Registry you must be using Cloud Identity and Access Management (IAM) access policies.



Important: If you started to use Container Registry before the availability of [IAM API key policies in Container Registry](#) in February 2019, you must ensure that you are using IAM access policies to manage access to the IBM Cloud Container Registry service.

Policy-free authorization is being discontinued in the following Container Registry regions:

- `us-south` (`us.icr.io`)
- `uk-south` (`uk.icr.io`)
- `eu-central` (`de.icr.io`)
- `ap-south` (`au.icr.io`)
- `ap-north` (`jp.icr.io`)

Other regions are unaffected because they already require IAM access policies for all accounts.

What are the changes?

Before 7 June 2019, all account users had full access to the images and settings that are associated with the account. Accounts that use Container Registry for the first time since 7 June 2019 are required to have IAM access policies and other accounts optionally require them. From 5 July 2022, all accounts require IAM access policies.

By default, account owners already have appropriate policies that give them full access to Container Registry. If policy-free authorization is in use, any other account user IDs and service IDs must be granted appropriate policies so that they can continue to access images and settings.

Check whether these changes affect you

To check whether policy-free authorization is in effect, run the `ibmcloud cr iam-policies-status` command. If the CLI reports that `IAM policy enforcement is disabled`, you must [prepare for the changes](#).

The policy status setting is specific to each Container Registry region. Check every region in which you have Container Registry namespaces by running the `ibmcloud cr region-set` command.

Prepare for the changes

If the changes affect you, you must create IAM [access policies](#) that apply to each service ID and user ID that accesses images in your Container Registry namespaces, or accesses Container Registry settings that are associated with your account and region.

1. Identify each service ID and user ID that accesses your Container Registry images and settings. You can use [IBM Cloud Activity Tracker](#) to help find this information.
2. For each access that is identified in the previous step, [create an IAM access policy](#) that allows the correct access. You can also use access groups to apply policies to IDs.
3. (Optional) If you want to upgrade the account to use IAM access policy authorization at a more convenient time, rather than on the date of the change, run the `ibmcloud cr iam-policies-enable` command.



Important: This change cannot be reversed.

This change applies to the currently targeted region only. Remember to check all regions where you have Container Registry namespaces.

What if I did not implement the changes in time?

To recover any access that is lost, follow the steps in [Prepare for the changes](#).

Minimum supported Docker version from 1 March 2022

From 1 March 2022, the minimum version of Docker Engine that is supported by IBM Cloud® Container Registry is v17.07, or later.

Release notes for Container Registry

Learn about the changes to IBM Cloud® Container Registry and Vulnerability Advisor. The changes are grouped by date.

24 July 2023

JSON output option added to several IBM Cloud Container Registry commands

The following IBM Cloud Container Registry commands now have an output option for JSON format:

- `ibmcloud cr exemption-add`
- `ibmcloud cr exemption-list`
- `ibmcloud cr exemption-types`
- `ibmcloud cr image-list`
- `ibmcloud cr namespace-list`
- `ibmcloud cr plan`
- `ibmcloud cr quota`
- `ibmcloud cr retention-policy-list`

For more information about the IBM Cloud Container Registry commands, see [IBM Cloud Container Registry CLI](#)

The `--json` option for IBM Cloud Container Registry commands is deprecated

The `--json` option is replaced with the `--output json` option in the following commands:

- `ibmcloud cr image-digests`
- `ibmcloud cr image-prune-untagged`
- `ibmcloud cr image-retention-run`
- `ibmcloud cr trash-list`

For more information about the IBM Cloud Container Registry commands, see [IBM Cloud Container Registry CLI](#)

19 June 2023

Vulnerability Advisor version 3 is deprecated from 19 June 2023

For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

19 May 2023

Update Vulnerability Advisor to version 4 by 19 June 2023

The Vulnerability Advisor component of IBM Cloud® Container Registry is being updated.

Vulnerability Advisor version 3 is being deprecated as the default on 19 June 2023. From 19 June 2023, the default will be Vulnerability Advisor version 4. If you have version 3 set as the default, you can continue to use version 3 until the end of support date. An end of support date is not available yet.

For more information, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

26 April 2023

Using Portieris to block the deployment of images with issues is deprecated.

The use of Portieris to block the deployment of images with issues that are found by Vulnerability Advisor is deprecated.

11 November 2022

Change to virtual private endpoints

Virtual private endpoints are changing.

On 11 November 2022, virtual private endpoints (VPEs) for IBM Cloud Container Registry are being updated and the existing VPE version is being deprecated on 15 December 2022. If you use Container Registry VPE gateways, you must create new VPE gateways and remove your VPE gateways that were created before 11 November 2022 at the earliest opportunity so that you pick up these changes. VPE gateways that were created before 11 November 2022 are deprecated and will not work after 15 December 2022.

If you create a new Container Registry VPE gateway after 11 November 2022 and also use Cloud Identity and Access Management (IAM) [restricted IP address lists](#), you must ensure that your restricted IP address list contains the Cloud Service Endpoint (CSE) source IP addresses of the VPCs in which your Container Registry VPE gateways exist. This requirement is related to a previous change to how Container Registry works over the private network that the new VPE version also uses, see [Container Registry private IP addresses changed on 5 July 2022](#).

For more information, see [Changes to Container Registry VPE gateways from 11 November 2022](#).

2 November 2022

Changes to private IP addresses from 15 December 2022

The IBM Cloud Container Registry private IP addresses that were replaced on 5 July 2022 are being decommissioned on 15 December 2022.

IP addresses for accessing Container Registry over the private network changed on 5 July 2022, see [Container Registry private IP addresses changed on 5 July 2022](#). You continue to be able to use these old IP addresses, but they are due to be decommissioned on 15 December 2022. After this date, you will not be able to access Container Registry by using these IP addresses.

For more information, see [Changes to private IP addresses from 15 December 2022](#).

15 September 2022

Container Registry plug-in 1.0.0 is available

A new version, version 1.0.0, of the Container Registry CLI plug-in is available. To update the version of your Container Registry CLI plug-in, see [Updating the container-registry CLI plug-in](#).

Version 1.0.0 includes [Vulnerability Advisor 4](#).

In version 1.0.0, the `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands no longer include security status by default. To include security status, you can either add the `--va` option to the command, or use the `ibmcloud cr va` command to query the security status for an individual image.

For more information, see [Container Registry CLI stops returning security status results in lists by default from version 1.0.0](#).

All releases of Container Registry plug-in 0.1 are deprecated

All releases of version 0.1 of the Container Registry CLI plug-in are deprecated. You can continue to use releases of version 0.1, but version 1.0.0 is available for you to use. Version 0.1 will continue to be updated with any required updates until 15 September 2023. To update the version of your CLI plug-in, see [Updating the container-registry CLI plug-in](#).

Vulnerability Advisor 4 is available from Container Registry plug-in 1.0.0

From Container Registry plug-in 1.0.0, you can choose whether to use Vulnerability Advisor version 3 or version 4 to run your commands. Vulnerability Advisor 4 is available from version 1.0.0 of the Container Registry plug-in. Vulnerability Advisor 3 is the default.

If you want to continue to use version 3, you don't need to do anything.

If you want to use version 4 to run the `ibmcloud cr va`, `ibmcloud cr image-list`, or `ibmcloud cr image-digests` commands, see

[Setting the Vulnerability Advisor version.](#)

For more information about Vulnerability Advisor, see [About Vulnerability Advisor](#). For more information about Vulnerability Advisor API 4, see [Vulnerability Advisor 4 for IBM Cloud Container Registry](#).

New commands for setting and checking the Vulnerability Advisor version are available from Container Registry plug-in 1.0.0

From Container Registry plug-in 1.0.0, you can use new commands to check and set Vulnerability Advisor versions.

If you want to continue to use version 3, you don't need to do anything.

If you want to use version 4, you can set the version by running the `ibmcloud cr va-version-set` command.

For more information about setting the version by using the `ibmcloud cr va-version-set` command, see [ibmcloud cr va-version-set](#) and [Setting the Vulnerability Advisor version](#).

To find out which version of Vulnerability Advisor that you're running, see [ibmcloud cr va-version](#).

3 August 2022

The CLI stops returning security status results in lists by default from version 1.0.0

From IBM Cloud Container Registry CLI plug-in version 1.0.0, when you use the `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands to list images, they return Vulnerability Advisor security status results only if you use the `--va` option.

If you want to continue to receive security status with your lists, prepare to upgrade by adding the new `--va` option to your commands.

For more information, see [Container Registry CLI stops returning security status results in lists by default from version 1.0.0](#).

8 July 2022

Context-based restrictions

You can use context-based restrictions to define and enforce access restrictions for IBM Cloud resources based on the network location of access requests.

For more information, see [Context-based restrictions](#).

5 July 2022

Change to Container Registry private IP addresses in all regions

If you are using Cloud Identity and Access Management (IAM) restricted IP address lists and you are connecting to Container Registry over the private network, your lists of allowed IP addresses must now include the private subnet and IP addresses of your own hosts. The IBM Cloud Container Registry private IP addressees also changed. This change also affects you if you have allowlists or a firewall rule.

For more information, see [Container Registry private IP addresses changed on 5 July 2022](#) and [Using IAM IP address access restrictions](#).

All accounts require IAM access policies

To access IBM Cloud Container Registry, you must be using Cloud Identity and Access Management (IAM) access policies. You must ensure that you are using IAM access policies to manage access to the Container Registry service.

Policy-free authorization is discontinued in the following Container Registry regions:

- `us-south` (`us.icr.io`)
- `uk-south` (`uk.icr.io`)
- `eu-central` (`de.icr.io`)
- `ap-south` (`au.icr.io`)
- `ap-north` (`jp.icr.io`)

Other regions are unaffected because they already require IAM access policies for all accounts.

For more information, see [IAM access policies are required from 5 July 2022](#) and [Defining IAM access policies](#).

23 June 2022

Change to Container Registry private IP addresses in the following regions only: **br-sao**, **ca-tor**

If you're using Cloud Identity and Access Management (IAM) restricted IP address lists and you're connecting to Container Registry over the private network in the **br-sao** and **ca-tor** regions, your lists of allowed IP addresses must now include the private subnet and IP addresses of your own hosts. The IBM Cloud Container Registry private IP addressees also changed. This change also affects you if you have allowlists or a firewall rule. Other regions are not affected yet.

For more information, see [Container Registry private IP addresses changed on 5 July 2022](#) and [Using IAM IP address access restrictions](#).

20 April 2022

Container Registry private IP addresses are changing from 23 June 2022

By 23 June 2022, if you're using Cloud Identity and Access Management (IAM) restricted IP address lists and you're connecting to Container Registry over the private network, you must update your lists of allowed IP addresses to include the private subnet and IP addresses of your own hosts. The IBM Cloud Container Registry private IP addressees are also changing, which might require updates to your firewall configuration.

For more information, see [Container Registry private IP addresses changed on 5 July 2022](#).

1 March 2022

Amendment to the minimum supported Docker version for Container Registry

From 1 March 2022, the minimum version of Docker Engine that is supported by Container Registry is v17.07, or later.

9 February 2022

All accounts will require IAM access policies from 5 July 2022

From 5 July 2022, to access IBM Cloud Container Registry, you must be using Cloud Identity and Access Management (IAM) access policies. If you started to use Container Registry before the availability of [IAM API key policies in Container Registry](#) in February 2019, you must now ensure that you're using IAM access policies to manage access to the Container Registry service.

Policy-free authorization will be discontinued in the following Container Registry regions:

- **us-south** (**us.icr.io**)
- **uk-south** (**uk.icr.io**)
- **eu-central** (**de.icr.io**)
- **ap-south** (**au.icr.io**)
- **ap-north** (**jp.icr.io**)

Other regions are unaffected because they already require IAM access policies for all accounts.

For more information, see [IAM access policies are required from 5 July 2022](#) and [Defining IAM access policies](#).

2 February 2022

Replication of exemption policies between IBM regions is discontinued

From 2 February 2022, all regions require separate exemption policy management. Exemption policies are used to exclude any matching vulnerabilities or configuration issues from Vulnerability Advisor reports. You can set an exemption policy by running the `ibmcloud cr exemption-add` command, see [Setting organizational exemption policies](#).

For more information, see [IBM Cloud Container Registry Is Ending Exemption Synchronization Across Regions](#).

1 February 2022

Storage that is used by untagged images is being charged for

From 1 February 2022, IBM Cloud Container Registry is charging for the storage that is used by `untagged` images. To reduce the amount that you're charged, you can [clean up your namespaces by deleting untagged images](#). You can also [free up used storage and change service plans or quota limits to stay within given quota limits](#).

For more information, see [Starting 1 February 2022, IBM Cloud Container Registry will be billing for the storage that is used by untagged images](#).

17 January 2022

View IBM Cloud Activity Tracker events for Red Hat signing

You can view Activity Tracker events for [Red Hat Signing](#) operations.

For more information, see [Auditing events for Container Registry](#).

7 December 2021

Define configuration rules for Container Registry

As a security or compliance focal, you can use the **Govern resources** section of the Security and Compliance Center dashboard to define *configuration rules* for Container Registry.

For more information, see [Managing security and compliance for Container Registry](#).

1 November 2021

Using Notary v1 for signing images is discontinued

The Notary v1 service that supports [Docker Content Trust](#) and `docker trust` commands in Container Registry is discontinued from 1 November 2021.

Container Registry supports the [Red Hat Signing](#) model. For more information, see [Signing images for trusted content by using Red Hat signatures](#).

5 October 2021

Container Registry container builds are discontinued

The `ibmcloud cr build` command is discontinued from 5 October 2021. You can use [Tekton pipelines](#) instead.

For more information, see [IBM Cloud Container Registry is Deprecating Container Builds](#).

2 September 2021

Namespaces that are assigned to a resource group show in the Resource list page

Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console. For more information, see [Planning namespaces](#).

30 August 2021

New region in Brazil

A new region in Sao Paulo, Brazil is available. The new region is **br-sao** and the domain name is **br.icr.io**. For more information, see [Local regions](#).

19 August 2021

Using registry tokens is discontinued

From 19 August 2021, registry tokens are no longer accepted for authentication. For more information, see [IBM Cloud Container Registry Deprecates Registry Tokens for Authentication](#).

9 August 2021

The **ibmcloud cr login** command logs you into the **<region>.icr.io** registry domain only

From version 0.1.541 of the Container Registry CLI, the **ibmcloud cr login** command logs you in to the **<region>.icr.io** registry domain only, where **<region>** is your [region](#). If you want to log in to the deprecated domain, **registry.<region>.bluemix.net**, use **docker login**, see [Authentication](#).

8 July 2021

Using Notary v1 for signing images is deprecated

The Notary v1 service for signing images is deprecated. It is being removed from Container Registry on 31 August 2021.

As an alternative approach, Container Registry supports the [Red Hat Signing](#) model. For more information, see [Signing images for trusted content by using Red Hat signatures](#).

21 June 2021

Global registry

The global registry (**icr.io**) is now available for hosting user images and namespaces. Previously the global registry hosted only public images that are provided by IBM. For more information, see [Global registry](#).

10 May 2021

New region in Canada

A new region in Toronto, Canada is available. The new region is **ca-tor** and the domain name is **ca.icr.io**. For more information, see [Local regions](#).

4 May 2021

The **ibmcloud cr ppa-archive-load** command is discontinued

The **ibmcloud cr ppa-archive-load** command is discontinued. Containerized software is distributed in IBM Cloud Paks, see [IBM Cloud Paks](#). To run IBM Cloud Paks on Red Hat OpenShift on IBM Cloud, see [Adding Cloud Paks](#). For more information about setting up your IBM Cloud Kubernetes Service cluster to pull entitled software, see [Setting up a cluster to pull entitled software](#).

18 February 2021

Increase the performance of the **ibmcloud cr image-list** and **ibmcloud cr image-digests** commands by using the **--no-va** option

If you don't need the security status (Vulnerability Advisor) results as part of the output of the [ibmcloud cr image-list](#) or [ibmcloud cr image-digests](#) commands, you can use the **--no-va** option to increase performance.

15 February 2021

New region in Japan

A new region in Japan is available. The new region is **jp-osa** and the domain name is **jp2.icr.io**. For more information, see [Local regions](#).

19 November 2020

You can use Portieris to enforce container image security

With effect from 19 November 2020, Container Image Security Enforcement is deprecated. To enforce container image security, you can use [Portieris](#). You can use Portieris to control where images are deployed from, enforce Vulnerability Advisor policies, and ensure that [content trust](#) is properly applied to the image.

For more information, see [Enforcing container image security by using Portieris](#).

21 October 2020

Find out about the usage on your account by using platform metrics

You can use the [ibmcloud cr platform-metrics](#) command to enable and disable platform metrics, and to find out whether you have platform metrics set up on your account.

For more information, see [Monitoring metrics for IBM Cloud Container Registry](#).

6 October 2020

Container Registry container builds are deprecated

The `ibmcloud cr build` command, which builds an image in IBM Cloud and pushes it to Container Registry, is deprecated from 6 October 2020. You can use [Tekton pipelines](#) instead.

For more information, see [IBM Cloud Container Registry is Deprecating Container Builds](#).

27 August 2020

Setting exemption policies by digest

When you want to set up an exemption policy, you can set the scope by namespace, repository, digest, or tag. You can set an exemption policy by running the `ibmcloud cr exemption-add` command.

For more information, see [Setting organizational exemption policies](#).

12 August 2020

Using UAA tokens is discontinued

From 12 August 2020, UAA tokens are no longer accepted for authentication. At the moment, registry tokens continue to be accepted, but their use is deprecated.

For more information, see [Announcing End of IBM Cloud Container Registry Support for UAA Tokens](#).

30 July 2020

New access roles are required for Vulnerability Advisor exemption policies

If you want to manage your Vulnerability Advisor exemption policies for security issues, depending on the task that you want to complete, you might have to update your access role.

For more information, see [Access roles for configuring Container Registry](#).

29 July 2020

You can set permissions so that access to resources within a namespace can be configured at the resource group level

Namespaces are created in resource groups so that access to resources within a namespace can be configured at the resource group level. If a namespace isn't already in a resource group, you can assign the namespace to a resource group.

- Namespaces created in version 0.1.485 of the Container Registry CLI or later, or in the IBM Cloud console on or after 29 July 2020, are created in a resource group that you specify so that you can configure access to resources within the namespace at the [resource group](#) level. However, you can still set permissions for the namespace at the account level or in the namespace itself. For more information, see [Set up a namespace](#) and `ibmcloud cr namespace-add`.
- Namespaces created in version 0.1.484 of the Container Registry CLI or earlier, or in the IBM Cloud console before 29 July 2020 aren't assigned to resource groups. You can assign an existing namespace to a resource group so that you can configure access to resources within a namespace at the resource group level. For more information, see [Assigning existing namespaces to resource groups](#) and `ibmcloud cr namespace-assign`.
- You can find out which namespaces are assigned to resource groups and which are unassigned by running the `ibmcloud cr namespace-`

[list](#) command with the `-v` option.

13 July 2020

To work with namespaces, you must have the Manager role at the account level

If you want to add or remove namespaces, you must have the Manager role, see [Access roles for configuring Container Registry](#).

24 June 2020

Restoring all tags for a digest in a repository is now an option

When you want to restore an image, you can restore either by tag or by digest. Restoring by digest restores the digest and all its tags in the repository that aren't already in the live repository. You can restore an image by running the `ibmcloud cr image-restore` command.

For more information, see [Restoring images](#).

18 May 2020

Retaining untagged images is now an option when you clean up your namespaces

Retention policies now include untagged images by default when they calculate what to retain. You can use the `retain-untagged` option for the `ibmcloud cr retention-policy-set` and `ibmcloud cr retention-run` commands to keep all untagged images and delete only tagged images. You can view the value of `retain-untagged` for each retention policy by running the `ibmcloud cr retention-policy-list` command.

For more information, see [Cleaning up your namespaces](#).

30 April 2020

`ibmcloud cr image-prune-untagged` command is available

The `ibmcloud cr image-prune-untagged` command deletes all [untagged](#) images in your Container Registry account.

For more information, see [ibmcloud cr image-prune-untagged](#) and [Clean up your namespaces by deleting untagged images](#).

16 April 2020

You can use private network connections to securely route your data in Container Registry

If you use cloud-based services for production workloads, you can use a secure private connection so that you ensure that you adhere to any compliance regulations. You can use IBM Cloud service endpoints to connect to IBM Cloud services over the private IBM Cloud network.

For more information, see [Securing your connection to Container Registry](#) and [IBM Cloud Container Registry architecture and workload](#).

`ibmcloud cr private-only` command is available

You can use the `ibmcloud cr private-only` command to prevent image pulls or pushes over public network connections for your account.

For more information, see [ibmcloud cr private-only](#).

3 February 2020

Using Container Registry tokens is deprecated

The use of UAA and Container Registry tokens is deprecated. From 12 August 2020, UAA tokens are not accepted for authentication.

For more information, see [Announcing End of IBM Cloud Container Registry Support for UAA Tokens](#).

31 January 2020

`ibmcloud cr image-digests` command is available

The `ibmcloud cr image-digests` command displays all images in your account, including untagged images.

For more information, see [ibmcloud cr image-digests](#).

4 December 2019

Support for Red Hat signatures is available

Container Registry now supports Red Hat signatures.

For more information, see [Signing images for trusted content by using Red Hat® signatures](#).

25 October 2019

IBM Log Analysis platform services logs are available

Container Registry generates platform services logs that are displayed in your logging instances.

For more information, see [IBM Log Analysis platform services logs](#).

14 October 2019

`ibmcloud cr manifest-inspect` command is available

The `ibmcloud cr manifest-inspect` command displays the contents of the manifest for an image.

For more information, see [ibmcloud cr manifest-inspect](#).

23 September 2019

You can create retention policies for your images

Image retention policies retain the specified number of images for each repository within a namespace in Container Registry. All other images in the namespace are deleted.

The following commands are available for you to use to create retention policies so that you can manage your images:

- The [ibmcloud cr retention-policy-set](#) command sets up a policy for retaining images for each repository within a namespace.

- The `ibmcloud cr retention-policy-list` command lists the image retention policies for your account.

For more information, see [Retaining images](#).

You can restore deleted images from the trash

All deleted images are stored in the trash for 30 days.

The following commands are available for you to use to restore images:

- The `ibmcloud cr trash-list` command displays all images in the trash in your IBM Cloud account. Images remain in the trash for 30 days after deletion.
- The `ibmcloud cr image-restore` command restores a deleted image from the trash.

For more information, see [Listing images in the trash](#) and [Restoring images](#).

1 August 2019

`ibmcloud cr retention-run` command is available

The `ibmcloud cr retention-run` command cleans up your namespaces by retaining images for each repository within a namespace in Container Registry by applying specified criteria. All other images in the namespace are deleted.

For more information, see [ibmcloud cr retention-run](#) and [Retaining images](#).

25 July 2019

IBM Cloud Activity Tracker available for Container Registry

Use the IBM Cloud Activity Tracker service to track how users and applications interact with the Container Registry service in IBM Cloud.

For more information, see [Auditing events for Container Registry](#).

1 July 2019

`ibmcloud cr token-add` command is no longer available

The `ibmcloud cr token-add` command is discontinued. You can't add registry tokens in either the CLI or the API anymore.

27 June 2019

Container Scanner is no longer available

The Container Scanner is discontinued. Vulnerability Advisor still scans images that are pushed to Container Registry.

13 June 2019

Remove tags from images

Remove a tag, or tags, from each specified image in Container Registry. To remove a specific tag from an image and leave the underlying image and any other tags in place, use the `ibmcloud cr image-untag` command. If you want to delete the underlying image, and all its tags, use the `ibmcloud cr image-rm` command instead.

For more information, see [Removing tags from images in your private repository](#).

13 May 2019

End of support for Container Scanner

Container Scanner is now deprecated and is no longer usable.

2 April 2019

General Availability of Container Image Security Enforcement

Use Container Image Security Enforcement to verify your container images before you deploy them to your cluster in IBM Cloud Kubernetes Service. You can control where images are deployed from, enforce Vulnerability Advisor policies, and ensure that [content trust](#) is properly applied to the image.

14 March 2019

IBM Cloud Activity Tracker available for Container Registry

Use the IBM Cloud Activity Tracker service to track how users and applications interact with the Container Registry service in IBM Cloud.

For more information, see [Auditing events for Container Registry](#).

25 February 2019

New domain names

Container Registry is adopting new domain names. The new domain names are available in the IBM Cloud console and the CLI. You can use the new `icr.io` domain names now. The existing `registry.bluemix.net` domain names are deprecated, but you can continue to use them until the end of support date. An end of support date is not available yet.

For more information, see [Regions](#) and [Introducing New IBM Cloud Container Registry Domain Names](#).

Signatures apply to the whole image name, which includes the domain name. If you're using content trust, you must add new signatures so that you can use content trust under the new domain name.

For more information about signing images, see [Signing images for trusted content](#).

Adding IAM API key policies to control access to resources

The new cluster image pull secrets for the `icr.io` domains are authorized by using an Cloud Identity and Access Management (IAM) API key. Therefore, if you want more control over access to your Container Registry resources, you can add IAM access policies. For example, you can change the API key policies in the cluster's pull secret so that images are pulled from a certain registry region or namespace only.

For more information, see [Understanding how to authorize your cluster to pull images from a registry](#).

New region in ap-north

A new region is available in `ap-north`. You can use the new region by using the domain name `jp.icr.io`.

For more information, see [Regions](#).

21 February 2019

Automating access to your namespaces

Using tokens to automate pushing and pulling Docker images to and from your namespaces is deprecated. You must now use API keys to automate access to your Container Registry namespaces so that you can push and pull images.

For more information, see [Creating a user API key manually](#).

8 January 2019

End of support for Vulnerability Advisor API version 2

Vulnerability Advisor's API version 2 is deprecated and is no longer usable. Use version 3 of the API, see [Vulnerability Advisor for IBM Cloud Container Registry](#).

4 October 2018

Managing user access

Use IBM Cloud Identity and Access Management (IAM) to control access by users in your account to Container Registry. When IAM access policies are enabled for your account in Container Registry, every user that accesses the service in your account must be assigned an IAM *access policy* with an IAM user role defined. That policy determines the role that the user has within the context of the service, and what actions the user can perform.

For more information, see [Managing IAM access with Cloud Identity and Access Management](#), [Defining IAM access policies](#), and [Granting access to Container Registry resources tutorial](#).

7 August 2018

Exemption policies available in Vulnerability Advisor

If you want to manage the security of an IBM Cloud organization, you can use your policy setting to determine whether an issue is exempt or not. You can use Portieris to ensure that deployment is allowed only from images that contain no security issues after accounting for any issues that are exempted by your policy.

For more information, see [Setting organizational exemption policies](#).

25 July 2018

IBM Cloud Activity Tracker available for Vulnerability Advisor

Use the IBM Cloud Activity Tracker service to track how users and applications interact with the Container Registry service in IBM Cloud.

For more information, see [Auditing events for Container Registry](#).

12 July 2018

Vulnerability Advisor API version 3

Version 3 of the API changes the behavior of the API endpoints that are used to list exemptions. You must check that your use of the API does not rely on the behavior of version 2.

For more information, see [Vulnerability Advisor for IBM Cloud Container Registry](#).

31 May 2018

Use Helm for Passport Advantage images

Import IBM software that is downloaded from [IBM Passport Advantage Online for customers](#) and packaged for use with Helm into your Container Registry namespace.

21 March 2018

Container Scanner

Container Scanner enables Vulnerability Advisor to report any problems that are found in running containers that are not present in the container's base image.

16 March 2018

Container Image Security Enforcement beta

Use Container Image Security Enforcement beta to verify your container images before you deploy them to your cluster in IBM Cloud Kubernetes Service. You can control where images are deployed from, enforce Vulnerability Advisor policies, and ensure that [content trust](#) is properly applied to the image.

20 February 2018

Trusted content

Container Registry provides trusted content technology so that you can sign images to ensure the integrity of images in your registry namespace. By pulling and pushing signed images, you can verify that your images were pushed by the correct party, such as your continuous integration (CI) tools.

For more information, see [Signing images for trusted content](#).

6 November 2017

Global registry

A global registry is available. The global registry domain name doesn't include a region (`icr.io`). Only public images that are provided by IBM are hosted in this registry.

For more information, see [Global registry](#).

29 September 2017

Build Docker images

The `ibmcloud cr build` command is now available for running container builds. You can build a Docker image directly in IBM Cloud or create

your own Docker image on your local computer and upload (push) it to your namespace in Container Registry.

For more information, see [Building Docker images to use them with your namespace](#).

24 August 2017

Graphical user interface released

The IBM Cloud Container Registry graphical user interface is available in the catalog, see [Container Registry](#).

27 June 2017

Introducing IBM Cloud Container Registry

IBM Cloud Container Registry is available as a service in IBM Cloud. [Container Registry](#) provides a multi-tenant private image registry that you can use to store and share your container images with users in your IBM Cloud account.

For more information about how to use Container Registry, see [Getting started with Container Registry](#).

Container Registry and Vulnerability Advisor workflow tutorial

Use this tutorial to find out about the basic functions of both IBM Cloud® Container Registry and Vulnerability Advisor.

These two services are pre-integrated and work together seamlessly in IBM Cloud, and their features provide a robust but straightforward workflow for users of containers. You can use these services to store your container images, ensure the security of your images and Kubernetes clusters, control the images that you can use to deploy to your clusters, and more.

Much of the information that is provided in this tutorial is available in greater detail in the "How To" sections of the documentation. This tutorial combines all those tasks into a workflow that helps you to use IBM Cloud Container Registry and Vulnerability Advisor. To learn more about each task, click the relevant link.

Objectives

The tutorial has the following objectives.

- Understand the core features of IBM Cloud Container Registry and Vulnerability Advisor.
- Use the functions of these services to create a workflow.

Services used

This tutorial uses the following IBM Cloud services:

- [IBM Cloud Container Registry](#)
- [IBM Cloud Kubernetes Service](#)

Before you begin

Before you begin, complete the following tasks:

- [Install Git](#).
- [Install IBM Cloud Developer Tools](#), a script to install `docker`, `kubectl`, `helm`, `ibmcloud` CLI, and required plug-ins by following the instructions in the `README.md` file in the repository.
- [Create a cluster](#).
- Ensure that you have the correct access permissions for adding and removing `namespaces`, see [Access roles for configuring IBM Cloud Container Registry](#).
- Ensure that you are using Vulnerability Advisor version 4.

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

Step 1: From code to a running container

Using [IBM Cloud Container Registry](#) to store your container images is the easiest way to get an application up and running with IBM Cloud Kubernetes Service. The following steps show you how to build a container image, store it in IBM Cloud Container Registry, and create a Kubernetes deployment that uses that image.

Create a namespace

Create a `namespace` to store your container images in IBM Cloud Container Registry. Namespaces are created in a `resource group`. For more information, see [Planning namespaces](#).

1. To log in to IBM Cloud and target the `us-south` region, run the following command.

```
$ ibmcloud login -r us-south [--sso]
```

✓ **Tip:** If you have a federated ID, use `ibmcloud login -r us-south --sso` to log in. Enter your username and use the provided URL in your CLI output to retrieve your one-time passcode. If you have a federated ID, the login fails without the `--sso` and succeeds with the `--sso` option.

2. Set `us-south` as the target region for the IBM Cloud Container Registry commands.

```
$ ibmcloud cr region-set us-south
```

3. Create a namespace by running the following command. Choose a name for your namespace, and replace `<my_namespace>` with that name.

✓ **Tip:** The namespace must be unique across all IBM Cloud accounts in the same region. Namespaces must have 4 - 30 characters and include lowercase letters, numbers, hyphens (-), and underscores (_) only. Namespaces must start and end with a letter or number.

✓ **Tip:** If you want to create the namespace in a specific resource group, see [Set up a namespace](#).

```
$ ibmcloud cr namespace-add <my_namespace>
```

Throughout this tutorial, replace `<my_namespace>` with your chosen namespace.

✓ **Tip:** If you have a problem when you try to add a namespace, see [Why can't I add a namespace?](#) for assistance.

Build and push an image

To [build a container image and push it to IBM Cloud Container Registry](#), you require an application and a *Dockerfile*. To get the application and the Dockerfile, and the other artifacts that you require, clone the [GitHub repository](#) that is associated with this tutorial. For the rest of this tutorial, ensure that you run all commands from the directory of the cloned repository.

1. To build the image, run the following command:

```
$ docker build -t us.icr.io/<my_namespace>/hello-world:1 .
```

✓ **Tip:** Docker must be running on your computer or the `docker` commands fail. IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).

2. Log your local Docker daemon into IBM Cloud Container Registry by running the `ibmcloud cr login` command:

```
$ ibmcloud cr login
```

✓ **Tip:** If you have a problem when you try to log in, see [Why can't I log in to Container Registry?](#) for assistance.

3. Push the image by running the following command:

```
$ docker push us.icr.io/<my_namespace>/hello-world:1
```

4. Confirm that your image uploaded successfully by running the following command:

```
$ ibmcloud cr images
```

✓ *API keys* and [grant access to IBM Cloud Container Registry resources](#) by using IAM.

Deploy a container that uses your image

Now that you have an image that is stored in IBM Cloud Container Registry, you can run a container on IBM Cloud Kubernetes Service that uses that image, see [Deploying apps with the CLI](#).

Throughout this tutorial, replace `<my_cluster>` with the name of your free Kubernetes cluster.

1. Run the following command:

```
$ ibmcloud ks cluster-config <my_cluster> --export
```

This command produces an `export` command that sets the `KUBECONFIG` environment variable.

2. Run the `export` command, which was generated by the previous command, by copying and pasting it.
3. Update the following line in the `hello-world.yaml` file by replacing `<my_namespace>` with your namespace:

```
$ image: us.icr.io/<my_namespace>/hello-world:1
```

4. Run your image as a deployment and expose it by creating a service that is accessed through the IP address of the worker node:

```
$ kubectl apply -f hello-world.yaml
```

5. Find the port that is used on the worker node by examining your new service by running the following command:

```
$ kubectl describe service hello-world
```

Note the number on the `NodePort:` line. Throughout this tutorial, use the number to replace the variable `<node_port>`.

6. In the output of the following command, note the public IP address. Throughout this tutorial, replace the variable `<public_ip>` with the IP address:

```
$ ibmcloud ks workers <my_cluster>
```

7. Access your service by running the following command. You can also use a web browser.

```
$ curl <public_ip>:<node_port>
```

If you see "Hello, world!", you're good to go.

Step 2: Secure your images and clusters

When you push an image to a namespace, the image is automatically scanned by [Vulnerability Advisor](#) to find [potential vulnerabilities](#). If vulnerabilities are found, instructions are provided to help fix the reported vulnerabilities.

To demonstrate these features, you must push an intentionally vulnerable image.



Note: Images are continually updated and new CVEs are discovered. As a result, you might see more vulnerabilities present in your image. If so, fix those vulnerabilities by using the information that is provided by Vulnerability Advisor.

View the vulnerability report for your image

When a vulnerability is found in one of your images, a [report](#) is produced that gives you more information about the vulnerability and the steps to resolve the vulnerability.

1. Build and push a vulnerable image:
 1. Build a vulnerable image by running the following command:

```
$ docker build -t us.icr.io/<my_namespace>/hello-world:2 -f Dockerfile-vulnerable .
```

2. Push the vulnerable image by running the following command:

```
$ docker push us.icr.io/<my_namespace>/hello-world:2
```

You can read the Dockerfile to better understand how this image was made vulnerable. In short, a Debian base image is used, and the `apt` package is rolled back to a version that is vulnerable to `CVE-2019-3462`.

2. List your images, and take note of the `SECURITY STATUS` column by running the following command:

```
$ ibmcloud cr images --va
```

This column conveys the number of issues present in your image. Because the number isn't zero, this image is vulnerable.

3. Run the `ibmcloud cr vulnerability-assessment` (alias `ibmcloud cr va`) command to get more information about the vulnerability:

```
$ ibmcloud cr va us.icr.io/<my_namespace>/hello-world:1
```

Among other things, this output includes the ID of the vulnerability (if applicable), the affected package, and the steps to resolve the issue.

Enforce security in your cluster

Despite the vulnerability that is present in your image, you are still able to deploy a container to your cluster by using this image, which you might not want. By using [Portieris](#), you can enforce security in several ways. For example, you can prevent vulnerable images from being used in deployments to your cluster.

1. [Install Portieris](#).
2. The [Portieris default policies](#) are too restrictive for this tutorial because they involve [image signing](#). Therefore, you must create custom policies. View the `security.yaml` file in the [GitHub repository](#), and read about customizing [policies](#) to understand this file's contents. In short, this policy requires all images in your namespace to have no issues reported by Vulnerability Advisor.
3. Update the following line in the `security.yaml` file by replacing `<my_namespace>` with your namespace:

```
$ - name: us.icr.io/<my_namespace>/*
```

4. Apply the custom policies:

```
$ kubectl apply -f security.yaml
```

5. To update `hello-world.yaml` so that it references your vulnerable image, change the tag from `1` to `2` as shown here:

```
$ image: us.icr.io/<my_namespace>/hello-world:2
```

6. Try to patch the existing deployment by running the following command:

```
$ kubectl apply -f hello-world.yaml
```

You see the following error message:

```
Deny "us.icr.io/<my_namespace>/hello-world:2", the Vulnerability Advisor image scan assessment found issues with the container image that are not exempted. Refer to your image vulnerability report for more details by using the `ibmcloud cr va` command.
```



Note: The Vulnerability Advisor verdict is subject to any [exemption policies](#) that you create. If you want to use an image that Vulnerability Advisor considers vulnerable, you can exempt one, or more vulnerabilities so that Vulnerability Advisor doesn't consider them in its verdict. You can see whether an issue is exempted by looking at the `Policy Status` column in the output of the `ibmcloud cr va` command, and you can also list your exemptions by running the [ibmcloud cr exemption-list](#) command.

Resolve vulnerabilities in your image

Vulnerability Advisor provides steps to resolve each vulnerability that is present in an image. The steps are displayed in the `How To Resolve` column in the output of the `ibmcloud cr va` command. If you follow the steps, you can resolve the issues in your image.

Because CVEs are frequently discovered and patched, this Dockerfile includes a contrived vulnerability that is introduced by rolling a package back to a known vulnerable version. Therefore, to fix the vulnerability, you can comment out the line that rolls back `apt`.

1. To prevent `apt` from being rolled back, comment out the following line in `Dockerfile-vulnerable` by putting a number sign (`#`) at the beginning of the line as shown here:

```
$ # RUN apt-get install --allow-downgrades -y apt=1.4.8
```

2. Build and push the image again:

1. Build the image again by running the following command:

```
$ docker build -t us.icr.io/<my_namespace>/hello-world:2 -f Dockerfile-vulnerable .
```

2. Push the image again by running the following command:

```
$ docker push us.icr.io/<my_namespace>/hello-world:2
```

3. Wait for the scan to complete and then run the following command to ensure that no issues are present in the image:

```
$ ibmcloud cr images --va
```

4. To patch the deployment, run the following command:


```
$ kubectl apply -f hello-world.yaml
```

5. Wait for the deployment to complete. To check whether the deployment is complete, run the following command:

```
$ kubectl rollout status deployment hello-world
```

This deployment succeeds, and you can access your service and see "Hello, world!" displayed.

6. Delete the deployment and the service before proceeding:

```
$ kubectl delete -f hello-world.yaml
```

Step 3: Deploying to nondefault Kubernetes namespaces

An IBM Cloud Kubernetes Service cluster can automatically pull images from IBM Cloud Container Registry to the `default` Kubernetes namespace. However, if you want to deploy to namespaces other than `default`, you must take extra steps.



Tip: Kubernetes and IBM Cloud Container Registry namespaces are different. For more information about IBM Cloud Container Registry namespaces, see [Registry namespace](#). For more information about Kubernetes namespaces, see [Namespaces](#).

1. In your cluster, create a Kubernetes namespace called `test`:

```
$ kubectl create namespace test
```

2. To deploy your deployment and service into this Kubernetes namespace, in the `hello-world.yaml` file change the `metadata.namespace` fields for both the deployment and the service from `default` to `test`. This snippet shows the `metadata.namespace` field in context:

```
metadata:
  name: hello-world
  namespace: test
```

3. Apply the configuration.

1. Apply the configuration with Portieris that is still enabled in your cluster by running the following command:

```
$ kubectl apply -f hello-world.yaml
```

Because Portieris is still enabled in your cluster, your deployment fails immediately, and you see the following message:

```
Error from server: error when creating "hello-world.yaml": admission webhook
"va.hooks.securityenforcement.admission.cloud.ibm.com" denied the request:
Deny "us.icr.io/<my_namespace>/hello-world:2", no valid ImagePullSecret defined for us.icr.io
```

This error is because Portieris determines that this deployment can't succeed because the `test` namespace is unable to pull images from your IBM Cloud Container Registry namespace. The `default` Kubernetes namespace in an IBM Cloud Kubernetes Service cluster comes preconfigured with [image pull secrets](#) to pull images from IBM Cloud Container Registry. However, these secrets aren't present in your new namespace.

2. Apply the configuration after Portieris is removed from your cluster.

1. [Remove Portieris](#).

2. Apply the configuration by running the following command:

```
$ kubectl apply -f hello-world.yaml
```

The `kubectl apply` command completes successfully. However, when you inspect the deployment's `pod` by running the `kubectl describe pod <pod_name> -n test` command, where `<pod_name>` is the name of the pod, the events log indicates that the cluster isn't authorized to pull the image.



Tip: You can find the pod name by running `kubectl get pod -n test`.

4. You must [set up an image pull secret](#) in your namespace so that you can deploy containers to that namespace. Several options are available, but this tutorial follows the steps to [copy an image pull secret](#) to the `test` namespace. Rather than copying all the `icr.io` secrets, you can copy the `us.icr.io` secret because your image is in that local registry. The following command copies the `default-us-icr-io` secret to the `test`

namespace, giving it the name `test-us-icr-io`:

```
$ kubectl get secret default-us-icr-io -o yaml | sed 's/default/test/g' | kubectl -n test create -f -
```

- Two options are available to [use the image pull secret](#). This tutorial uses the option to refer to the image pull secret in the deployment `.yaml` file by populating the `spec.imagePullSecrets` field. The following snippet shows the required lines in context; you must add the final two lines:

```
spec:
  containers:
  - name: hello-world
    image: us.icr.io/<my_namespace>/hello-world:1
    imagePullPolicy: Always
    imagePullSecrets:
    - name: test-us-icr-io
```

- Delete your deployment and reapply the configuration:

1. Delete your deployment by running the following command:

```
$ kubectl delete -f hello-world.yaml
```

2. Reapply the configuration by running the following command:

```
$ kubectl apply -f hello-world.yaml
```

This time the command succeeds, and you can access your container by using a `curl` command or a web browser.

Granting access to Container Registry resources tutorial

Use this tutorial to find out how to grant access to your resources by configuring IBM Cloud® Identity and Access Management (IAM) for IBM Cloud® Container Registry.

All accounts require IAM access policies. To set up and manage IAM access policies, see [Defining IAM access policies](#).

For more information about how to use IAM to manage access to your resources, see [Managing access to resources](#).

Before you begin

Before you begin, you must complete the following tasks:

- Complete the instructions in [Getting started with IBM Cloud Container Registry](#).
- Ensure that you have the most recent version of the **container-registry** CLI plug-in for the IBM Cloud CLI, see [Updating the container-registry CLI plug-in](#).
- Ensure that you have access to two [IBM Cloud accounts](#) that you can use for this tutorial, one for User A and one for User B, each must use a unique email address. You work in your own account, User A, and invite another user, User B, to use your account. You can choose to create a second IBM Cloud account, or you can work with a colleague that has an IBM Cloud account.
- Ensure that you have the correct access permissions for adding and removing *namespaces*, see [Access roles for configuring IBM Cloud Container Registry](#).

Step 1: Authorize a user to configure the registry

Add a second user to your account and grant them the ability to configure IBM Cloud Container Registry.

1. Add User B to User A's account.

1. Log in to User A's account, by running the following command.

```
$ ibmcloud login
```

2. Invite User B to access User A's account by running the following command, where `<user.b@example.com>` is User B's email address.

```
$ ibmcloud account user-invite <user.b@example.com>
```

3. Get User A's Account ID by running the following command.

```
$ ibmcloud target
```

Make a note of the Account ID that is in the parentheses () in the Account row.

2. Prove that User B can target User A's account but can't do anything with IBM Cloud Container Registry yet.

1. Log in as User B and target User A's account by running the following command, where `<YourAccountID>` is User A's Account ID.

```
$ ibmcloud login -c <YourAccountID>
```

2. Try to edit your registry quota to 4 GB of traffic by running the following command.

```
$ ibmcloud cr quota-set --traffic=4000
```

The command fails because User B doesn't have the correct access.

3. Grant User B the Manager role so that User B can configure IBM Cloud Container Registry.

1. Log back in to your account as yourself, User A, by running the following command.

```
$ ibmcloud login
```

2. Create a policy that grants the Manager role to User B by running the following command.

```
$ ibmcloud iam user-policy-create <user.b@example.com> --service-name container-registry --roles Manager
```

4. Prove that User B can now change quotas in User A's account.

1. Log in as User B, targeting User A's account by running the following command.

```
$ ibmcloud login -c <YourAccountID>
```

2. Try to edit your registry quota to 4 GB of traffic by running the following command.

```
$ ibmcloud cr quota-set --traffic=4000
```

It works because User B has the correct type of access.

3. Now change the quota back by running the following command.

```
$ ibmcloud cr quota-set --traffic=5120
```

5. Clean up.

1. Log back in to your account as yourself, User A, by running the following command.

```
$ ibmcloud login
```

2. List the policies for User B, find the policy that you created by running the following command, and note the ID.

```
$ ibmcloud iam user-policies <user.b@example.com>
```

3. Delete the policy by running the following command, where `<Policy_ID>` is your Policy ID.

```
$ ibmcloud iam user-policy-delete <user.b@example.com> <Policy_ID>
```

Step 2: Authorize a user to access specific namespaces

Create some *namespaces* with sample images, and grant access to them. You create policies to grant different roles to each namespace, and show what effect that has.


1. Create three new namespaces in User A's account. These namespaces must be unique across the region, so choose your own namespace names, but this tutorial uses `namespace_a`, `namespace_b` and `namespace_c` as examples.


1. Log in as User A, by running the following command.

```
$ ibmcloud login
```

2. Create `namespace_a` by running the following command.

```
$ ibmcloud cr namespace-add namespace_a
```

 **Tip:** The namespace must be unique across all IBM Cloud accounts in the same region. Namespaces must have 4 - 30 characters, and contain lowercase letters, numbers, hyphens (-), and underscores (_) only. Namespaces must start and end with a letter or number.

 **Tip:** If you have a problem when you try to add a namespace, see [Why can't I add a namespace?](#) for assistance.

3. Create `namespace_b` by running the following command.

```
$ ibmcloud cr namespace-add namespace_b
```

4. Create `namespace_c` by running the following command.

```
$ ibmcloud cr namespace-add namespace_c
```

2. Prove that User B can't see anything.

1. Log in as User B, targeting User A's account by running the following command.

```
$ ibmcloud login -c <YourAccountID>
```

2. Try to list the namespaces as User B by running the following command.

```
$ ibmcloud cr namespaces
```

It returns an empty list because User B doesn't have access to any namespaces.

3. Create policies to grant User B the ability to interact with the namespaces by running the following command.

1. Log in as User A's account by running the following command.

```
$ ibmcloud login
```

2. Check that at least three namespaces are listed by running the following command.

```
$ ibmcloud cr namespaces
```

The three namespaces that you created in this tutorial (`namespace_a` , `namespace_b` , and `namespace_c`) are shown. If you do not see these namespaces, repeat the instructions to create them again.

3. Create a policy that grants the Reader role on `namespace_b` to User B by running the following command, where `<cloud_region>` is the name of your IBM Cloud region, for example `us-south` .

```
$ ibmcloud iam user-policy-create <user.b@example.com> --service-name container-registry --region <cloud_region> --resource-type namespace --resource namespace_b --roles Reader
```



Tip: To see the names of the IBM Cloud regions, run the [ibmcloud regions](#) command.

4. Create a second policy that grants the Reader and Writer roles on `namespace_c` to User B by running the following command.

```
$ ibmcloud iam user-policy-create <user.b@example.com> --service-name container-registry --region <cloud_region> --resource-type namespace --resource namespace_c --roles Reader,Writer
```



Tip: This command adds two roles to the same resource in the same policy.

4. Push images into `namespace_a` and `namespace_b` .

1. Pull the `hello-world` image by running the following command.

```
$ docker pull hello-world
```

2. Tag the image to `namespace_a` by running the following command, where `<registry_region>` is the name of your [IBM Cloud Container Registry region](#), for example `us-south` .

```
$ docker tag hello-world <registry_region>.icr.io/namespace_a/hello-world
```

3. Tag the image to `namespace_b` by running the following command.

```
$ docker tag hello-world <registry_region>.icr.io/namespace_b/hello-world
```

4. Log in to IBM Cloud Container Registry by running the [ibmcloud cr login](#) command.

```
$ ibmcloud cr login
```



Tip: IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).



Tip: If you have a problem when you try to log in, see [Why can't I log in to Container Registry?](#) for assistance.

5. Push the image to `namespace_a` by running the following command.

```
$ docker push <registry_region>.icr.io/namespace_a/hello-world
```

6. Push the image to `namespace_b` by running the following command.

```
$ docker push <registry_region>.icr.io/namespace_b/hello-world
```

5. Prove that User B can interact with `namespace_b` and `namespace_c`, but not `namespace_a`.

1. Log in as User B by running the following command.

```
$ ibmcloud login -c <YourAccountID>
```

2. Show that User B can see `namespace_b` and `namespace_c`, but not `namespace_a` because User B doesn't have access to `namespace_a`, by running the following command.

```
$ ibmcloud cr namespaces
```


3. List your images by running the following command.

```
$ ibmcloud cr images
```

The image in `namespace_b` is shown in the list, but the image in `namespace_a` doesn't, because User B doesn't have access to `namespace_a`.

4. Log in to IBM Cloud Container Registry by running the following command.

```
$ ibmcloud cr login
```

 **Tip:** IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).

5. Pull the image by running the following command.

```
$ docker pull <registry_region>.icr.io/namespace_b/hello-world
```

6. Push the image to `namespace_b` by running the following command.

```
$ docker push <registry_region>.icr.io/namespace_b/hello-world
```

This command fails because User B doesn't have the Writer role in `namespace_b`.

7. Tag the image with `namespace_c` by running the following command.

```
$ docker tag hello-world <registry_region>.icr.io/namespace_c/hello-world
```

8. Push the image to `namespace_c` by running the following command.

```
$ docker push <registry_region>.icr.io/namespace_c/hello-world
```

The command works because User B has the Writer role in `namespace_c`.

9. Pull from `namespace_c` by running the following command.

```
$ docker pull <registry_region>.icr.io/namespace_c/hello-world
```

The command works because User B has the Reader role in `namespace_c`.

6. Clean up:

1. Log back in to User A's account by running the following command.

```
$ ibmcloud login
```

2. List the policies for User B by running the following command.

```
$ ibmcloud iam user-policies <user.b@example.com>
```

Find the policies that you created and note the Policy IDs.

3. Delete the policies that you created by running the following command, where `<Policy_ID>` is the Policy ID.

```
$ ibmcloud iam user-policy-delete <user.b@example.com> <Policy_ID>
```

Step 3: Create a service ID and grant access to a resource

Configure a service ID and grant it access to your IBM Cloud Container Registry namespace.

1. Set up a service ID with access to IBM Cloud Container Registry and create an `API key` for it.

1. Log in to User A's account by running the following command.

```
$ ibmcloud login
```

2. Create a service ID named `cr-roles-tutorial` with the description `"Created during the access control tutorial for Container Registry"` by running the following command.

```
$ ibmcloud iam service-id-create cr-roles-tutorial --description "Created during the access control tutorial for Container Registry"
```

3. Create a service policy for the service ID that grants the Reader role on `namespace_a` by running the following command.

```
$ ibmcloud iam service-policy-create cr-roles-tutorial --service-name container-registry --region <cloud_region> --resource-type namespace --resource namespace_a --roles Reader
```

4. Create a second service policy that grants the Writer role on `namespace_b` by running the following command.

```
$ ibmcloud iam service-policy-create cr-roles-tutorial --service-name container-registry --region <cloud_region> --resource-type namespace --resource namespace_b --roles Writer
```

5. Create an API key for the service ID by running the following command.

```
$ ibmcloud iam service-api-key-create cr-roles-tutorial-apikey cr-roles-tutorial
```

2. Use Docker to log in with the service ID API key, where `<API_Key>` is your API key, and interact with the registry.

1. Log in to IBM Cloud Container Registry by running the following command.

```
$ docker login -u iamapikey -p <API_Key> <registry_region>.icr.io
```



Tip: IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces in automation](#).

2. Pull your image by running the following command.

```
$ docker pull <registry_region>.icr.io/namespace_a/hello-world
```

3. Push your image to `namespace_a` by running the following command.

```
$ docker push <registry_region>.icr.io/namespace_a/hello-world
```

This command doesn't work because the user doesn't have the Writer role in `namespace_a`.

4. Push your image to `namespace_b` by running the following command.

```
$ docker push <registry_region>.icr.io/namespace_b/hello-world
```

This command works because the user has the Writer role in `namespace_b`.

3. Clean up:

1. Log back in to IBM Cloud Container Registry as User A.

```
$ ibmcloud cr login
```

✓ **Tip:** IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).

2. List your service policies by running the following command.

```
$ ibmcloud iam service-policies cr-roles-tutorial
```

Note your Policy IDs.

3. Delete your service policies by running the following command for each policy.

```
$ ibmcloud iam service-policy-delete cr-roles-tutorial <Policy_ID>
```

4. Delete your service ID by running the following command.

```
$ ibmcloud iam service-id-delete cr-roles-tutorial
```

Step 4: Cleaning up your account

Remove the resources that you created in previous sections to leave your account as it was at the start of this tutorial.

1. Log in to User A's account by running the following command.

```
$ ibmcloud login
```

2. Delete `namespace_a`, `namespace_b`, and `namespace_c` by running the following commands.

```
$ ibmcloud cr namespace-rm namespace_a
```

```
$ ibmcloud cr namespace-rm namespace_b
```


```
$ ibmcloud cr namespace-rm namespace_c
```

3. Remove User B from your account by running the following command.

```
$ ibmcloud account user-remove <user.b@example.com>
```


Solution tutorials

Moving a VM based app to Kubernetes

 **Tip:** This tutorial may incur costs. Use the [Cost Estimator](#) to generate a cost estimate based on your projected usage.

This tutorial walks you through the process of moving a VM based app to a Kubernetes cluster by using Kubernetes Service. [Kubernetes Service](#) delivers powerful tools by combining container and Kubernetes technologies, an intuitive user experience, and built-in security and isolation to automate the deployment, operation, scaling, and monitoring of containerized apps in a cluster of compute hosts.

The lessons in this tutorial include concepts for how to take an existing app, containerize the app, and deploy the app to a Kubernetes cluster. To containerize your VM based app, you can choose between the following options.

1. Identify components of a large monolith app that can be separated into their own microservice. You can containerize these microservices and deploy them to a Kubernetes cluster.
2. Containerize the entire app and deploy the app to a Kubernetes cluster.

Depending on the type of app that you have, the steps to migrate your app might vary. You can use this tutorial to learn about the general steps that you have to take and things that you have to consider before migrating your app.

Objectives

- Understand how to identify microservices in a VM based app and learn how to map components between VMs and Kubernetes.
- Learn how to containerize a VM based app.
- Learn how to deploy the container to a Kubernetes cluster in Kubernetes Service.
- Put everything learned to practice, run the **JPetStore** app in your cluster.

Architecture

Traditional app architecture with VMs

The following diagram shows an example of a traditional app architecture that is based on virtual machines.



Figure 1. Architecture diagram of the tutorial

1. The user sends a request to the public endpoint of the app. The public endpoint is represented by a load balancer service that load balances incoming network traffic between available app server instances.
2. The load balancer selects one of the healthy app server instances that run on a VM and forwards the request. App files, such as the app code, configuration files, and dependencies are stored on the app server VM.
3. The app server stores app data in a MySQL database that runs on a database VM.

Containerized architecture

The following diagram shows an example of a modern container architecture that runs in a Kubernetes cluster.



Diagram of a modern container architecture

1. The user sends a request to the public endpoint of the app. The public endpoint is represented by an Ingress application load balancer (ALB) that load balances incoming network traffic across app pods in the cluster. The ALB is a collection of rules that allow inbound network traffic to a publicly exposed app.
2. The ALB forwards the request to one of the available app pods in the cluster. App pods run on worker nodes that can be a virtual or physical machine.
3. App pods store data in persistent volumes. Persistent volumes can be used to share data between app instances or worker nodes.
4. App pods store data in an IBM Cloud database service. You can run your own database inside the Kubernetes cluster, but using a managed database-as-a-service (DBaaS) is usually easier to configure and provides built-in backups and scaling. You can find many different types of databases in the [IBM Cloud catalog](#).

VMs, containers, and Kubernetes

Kubernetes Service provides the capability to run containerized apps in Kubernetes clusters and delivers the following tools and functions:

- Intuitive user experience and powerful tools.
- Built-in security and isolation to enable rapid delivery of secure applications.
- Cloud services that include cognitive capabilities from IBM® Watson™.
- Ability to manage dedicated cluster resources for both stateless applications and stateful workloads.

Virtual machines vs containers

VMs, traditional apps run on native hardware. A single app does not typically use the full resources of a single compute host. Most organizations try to run

multiple apps on a single compute host to avoid wasting resources. You could run multiple copies of the same app, but to provide isolation, you can use VMs to run multiple app instances (VMs) on the same hardware. These VMs have full operating system stacks that make them relatively large and inefficient due to duplication both at runtime and on disk.

Containers are a standard way to package apps and all their dependencies so that you can seamlessly move the apps between environments. Unlike virtual machines, containers do not bundle the operating system. Only the app code, runtime, system tools, libraries, and settings are packaged inside containers. Containers are more lightweight, portable, and efficient than virtual machines.

In addition, containers allow you to share the host OS. This reduces duplication while still providing the isolation. Containers also allow you to drop unneeded files such as system libraries and binaries to save space and reduce your attack surface. Read more on virtual machines and containers [here](#).

Kubernetes orchestration

[Kubernetes](#) is a container orchestrator to manage the lifecycle of containerized apps in a cluster of worker nodes. Your apps might need many other resources to run, such as volumes, networks, and secrets which will help you connect to other cloud services, and secure keys. Kubernetes helps you to add these resources to your app. The key paradigm of Kubernetes is its declarative model. The user provides the desired state and Kubernetes attempts to conform to, and then maintains the described state.

This [self-paced workshop](#) can help you to get your first hands-on experience with Kubernetes. Additionally, check out the Kubernetes [concepts](#) documentation page to learn more about the concepts of Kubernetes.

What IBM's doing for you

By using Kubernetes clusters with IBM Cloud Kubernetes Service, you get the following benefits:

- Multiple data centers where you can deploy your clusters.
- Support for ingress and load balancer networking options.
- Dynamic persistent volume support.
- Highly available, IBM-managed Kubernetes masters.

Sizing clusters

As you design your cluster architecture, you want to balance costs against availability, reliability, complexity, and recovery. Kubernetes clusters in IBM Cloud Kubernetes Service provide architectural options based on the needs of your apps. With a bit of planning, you can get the most out of your cloud resources without over-architecting or over-spending. Even if you over or underestimate, you can easily scale up or down your cluster, by changing either the number or size of worker nodes.

To run a production app in the cloud by using Kubernetes, consider the following items:

1. Do you expect traffic from a specific geographic location? If yes, select the location that is physically closest to you for best performance.
2. How many replicas of your cluster do you want for higher availability? A good starting point might be three clusters, one for development, one for testing and one for production. Check out the [Best practices for organizing users, teams, applications](#) solution guide for creating multiple environments.
3. What [hardware](#) do you need for the worker nodes? Virtual machines or bare metal?
4. How many worker nodes do you need? This depends highly on the apps scale, the more nodes you have the more resilient your app will be.
5. How many replicas should you have for higher availability? Deploy replica clusters in multiple locations to make your app more available and protect the app from being down due to a location failure.
6. Which is the minimal set of resources your app needs to startup? You might want to test your app for the amount of memory and CPU it requires to run. Your worker node should then have enough resources to deploy and start the app. Make sure to then set resource quotas as part of the pod specifications. This setting is what Kubernetes uses to select (or schedule) a worker node that has enough capacity to support the request. Estimate how many pods will run on the worker node and the resource requirements for those pods. At a minimum, your worker node must be large enough to support one pod for the app.
7. When to increase the number of worker nodes? You can monitor the cluster usage and increase nodes when needed. See this tutorial to understand how to [analyze logs and monitor the health of Kubernetes applications](#).
8. Do you need redundant, reliable storage? If yes, create a persistent volume claim for NFS storage or bind an IBM Cloud database service to your pod.
9. Do you need to deploy a cluster on [Virtual Private Cloud infrastructure](#) or in [Classic infrastructure](#)? VPC gives you the security of a private cloud environment with the dynamic scalability of a public cloud.

To make the above more specific, let's assume you want to run a production web application in the cloud and expect a medium to high load of traffic. Let's explore what resources you would need:

1. Setup three clusters, one for development, one for testing and one for production.
2. The development and testing clusters can start with minimum RAM and CPU option (for example 2 CPU's, 4GB of RAM and one worker node for each cluster).
3. For the production cluster, you might want to have more resources for performance, high availability, and resiliency. We might choose a dedicated or even a bare metal option and have at least 4 CPU's, 16GB of RAM, and two workers nodes.

Decide what Database option to use

With Kubernetes, you have two options for handling databases:

1. You can run your database inside the Kubernetes cluster, to do that you would need to create a microservice to run the database. For example, if you are using a MySQL database, you would need to complete the following steps:

- Create a MySQL Dockerfile, see an example [MySQL Dockerfile](#) here.
 - You would need to use secrets to store the database credential. See example of this [here](#).
 - You would need a **deployment.yaml** file with the configuration of your database to deployed to Kubernetes. See example of this [here](#).
- The second option would be to use the managed database-as-a-service (DBaaS) option. This option is usually easier to configure and provides built-in backups and scaling. You can find many different types of databases in the [IBM Cloud catalog](#). To use this option, you would need to do the following:
 - Create a managed database-as-a-service (DBaaS) from the [IBM Cloud catalog](#).
 - Store database credentials inside a secret. You will learn more on secrets in the "Store credentials in Kubernetes secrets" section.
 - Use the database-as-a-service (DBaaS) in your application.

Decide where to store application files

Kubernetes Service provides several options to store and share data across pods. Not all storage options offer the same level of persistence and availability in disaster situations.

Non-persistent data storage

Containers and pods are, by design, short-lived and can fail unexpectedly. You can store data in the local file system of a container. Data inside a container cannot be shared with other containers or pods and is lost when the container crashes or is removed.

Learn how to create persistent data storage for your app

You can persist app data and container data on [NFS file storage](#) or [block storage](#) by using native Kubernetes persistent volumes.

To provision NFS file storage or block storage, you must request storage for your pod by creating a persistent volume claim (PVC). In your PVC, you can choose from predefined storage classes that define the type of storage, storage size in gigabytes, IOPS, the data retention policy, and the read and write permissions for your storage. A PVC dynamically provisions a persistent volume (PV) that represents an actual storage device in IBM Cloud. You can mount the PVC to your pod to read from and write to the PV. Data that is stored in PVs is available, even if the container crashes, or the pod reschedules. The NFS file storage and block storage that backs the PV is clustered by IBM to provide high availability for your data.

To learn how to create a PVC, follow the steps covered in the [Kubernetes Service storage documentation](#).

Learn how to move existing data to persistent storage

To copy data from your local machine to your persistent storage, you must mount the PVC to a pod. Then, you can copy data from your local machine to the persistent volume in your pod.

- To copy data, first, you would need to create a configuration that looks like something like this:

```
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: mypvc
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/mnt/data"
          name: task-pv-storage
```

- Then, to copy data from your local machine to the pod you would use a command like this:

```
$ kubectl cp <local_filepath>/<filename> <namespace>/<pod>:<pod_filepath>
```

- Copy data from a pod in your cluster to your local machine:

```
$ kubectl cp <namespace>/<pod>:<pod_filepath>/<filename> <local_filepath>/<filename>
```

Set up backups for persistent storage

File shares and block storage are provisioned into the same location as your cluster. The storage itself is hosted on clustered servers by IBM to provide high availability. However, file shares and block storage are not backed up automatically and might be inaccessible if the entire location fails. To protect your

data from being lost or damaged, you can set up periodic backups, which you can use to restore your data when needed.

For more information, see [Planning for storage](#) options for NFS file storage and block storage.

Prepare your code

Apply the 12-factor principles

The [twelve-factor app](#) is a methodology for building cloud native apps. When you want to containerize an app, move this app to the cloud, and orchestrate the app with Kubernetes, it is important to understand and apply some of these principles. Some of these principles are required in IBM Cloud.

The following key principles are required:

- **Codebase** - All source code and configuration files are tracked inside a version control system (for example a GIT repository), this is required if using DevOps pipeline for deployment.
- **Build, release, run** - The 12-factor app uses strict separation between the build, release, and run stages. This can be automated with an integrated DevOps delivery pipeline to build and test the app before deploying it to the cluster. Check out the [Continuous Deployment to Kubernetes tutorial](#) to learn how to set up a continuous integration and delivery pipeline. It covers the set up of source control, build, test and deploy stages and shows you how to add integrations such as security scanners, notifications, and analytics.
- **Config** - All configuration information is stored in environment variables. No service credentials are hard-coded within the app code. To store credentials, you can use Kubernetes secrets. More on credentials covered below.

Store credentials in Kubernetes secrets

It's never good practice to store credentials within the app code. Instead, Kubernetes provides so-called ["secrets"](#) that hold sensitive information, such as passwords, OAuth tokens, or SSH keys. Kubernetes secrets are encrypted by default which makes secrets a safer and a more flexible option to store sensitive data than to store this data verbatim in a `pod` definition or in a container image.

One way of using secrets in Kubernetes is by doing something like this:

1. Create a file called `cloud-secrets.txt` and store the service credentials of any cloud service inside it.

```
{
  "url": "<SERVICE_URL>",
  "api_key": <API_Key>
}
```

2. Then, create a Kubernetes secret by running a command below and verify that the secret is created by using `kubecttl get secrets` after running the command below:

```
$ kubecttl create secret generic cloud-service-secret --from-file=cloud-secrets.txt=./cloud-secrets.txt
```

Containerize your app

To containerize your app, you must create a container image.

An image is created from a [Dockerfile](#), which is a file that contains instructions and commands to build the image. A Dockerfile might reference build artifacts in its instructions that are stored separately, such as an app, the app's configuration, and its dependencies.

To build your own Dockerfile for your existing app, you might use the following commands:

- FROM - Choose a parent image to define the container runtime.
- ADD/COPY - Copy the content of a directory into the container.
- WORKDIR - Set the working directory inside the container.
- RUN - Install software packages that the apps needs during runtime.
- EXPOSE - Make a port available outside the container.
- ENV NAME - Define environment variables.
- CMD - Define commands that run when the container launches.

Images are typically stored in a registry that can either be accessible by the public (public registry) or set up with limited access for a small group of users (private registry). Public registries, such as Docker Hub, can be used to get started with Docker and Kubernetes to create your first containerized app in a cluster. But when it comes to enterprise apps, use a private registry, like the one provided in IBM Cloud Container Registry to protect your images from being used and changed by unauthorized users.

To containerize an app and store it in IBM Cloud Container Registry:

1. You would need to create a Dockerfile, below is an example of a Dockerfile.

```
# Build JPetStore war
FROM openjdk:8 as builder
COPY . /src
WORKDIR /src
```

```
RUN ./build.sh all

# Use WebSphere Liberty base image from the Docker Store
FROM websphere-liberty:latest

# Copy war from build stage and server.xml into image
COPY --from=builder /src/dist/jpetstore.war /opt/ibm/wlp/usr/servers/defaultServer/apps/
COPY --from=builder /src/server.xml /opt/ibm/wlp/usr/servers/defaultServer/
RUN mkdir -p /config/lib/global
COPY lib/mysql-connector-java-3.0.17-ga-bin.jar /config/lib/global
```

2. Once a Dockerfile is created, next you would need to build the container image and push it to IBM Cloud Container Registry. You can build a container using a command like:

```
$ docker build . -t <image_name>
docker push <image_name>
```

Deploy your app to a Kubernetes cluster

After a container image is built and pushed to the cloud, next you need to deploy to your Kubernetes cluster. To do that you would need to create a deployment.yaml file.

Learn how to create a Kubernetes deployment yaml file

To create Kubernetes deployment.yaml files, you would need to do something like this:

1. Create a deployment.yaml file, here is an example of a [deployment YAML](#) file.
2. In your deployment.yaml file, you can define [resource quotas](#) for your containers to specify how much CPU and memory each container needs to properly start. If containers have resource quotas specified, the Kubernetes scheduler can make better decisions about the worker node where to place your pods on.
3. Next, you can use below commands to create and view the deployment and services created:

```
$ kubectl create -f <filepath/deployment.yaml>
kubectl get deployments
kubectl get services
kubectl get pods
```

Summary

In this tutorial, you learned the following:

- The differences between VMs, containers, and Kubernetes.
- How to define clusters for different environment types (dev, test, and production).
- How to handle data storage and the importance of persistent data storage.
- Apply the 12-factor principles to your app and use secrets for credentials in Kubernetes.
- Build container images and push them to IBM Cloud Container Registry.
- Create Kubernetes deployment files and deploy the container image to Kubernetes.

Put everything learned to practice, run the JPetStore app in your cluster

To put everything you've learned in practice, follow the [demonstration](#) to run the **JPetStore** app on your cluster and apply the concepts learned. The JPetStore app has some extended functionality to allow you to extend an app in Kubernetes by running image classification as a separate microservice.

Related Content

- [Get started](#) with Kubernetes and Kubernetes Service.
- Kubernetes Service labs on [GitHub](#).
- Kubernetes main [docs](#).
- [Persistent storage](#) in Kubernetes Service.
- [Best practices solution guide](#) for organizing users, teams and apps.
- [Analyze logs and monitor application health](#).
- Set up [continuous integration and delivery pipeline](#) for containerized apps that run in Kubernetes.
- Use [multiple clusters across multiple locations](#) for high availability.
- Re-platform applications to Kubernetes using [Konveyor Move2Kube](#).

Resilient and secure multi-region Kubernetes clusters with IBM Cloud Internet Services



Tip: This tutorial may incur costs. Use the [Cost Estimator](#) to generate a cost estimate based on your projected usage.

Users are less likely to experience downtime when an application is designed with resiliency in mind. When implementing a solution with Kubernetes Service, you benefit from built-in capabilities, like load balancing and isolation, increased resiliency against potential failures with hosts, networks, or apps. By creating multiple clusters and if an outage occurs with one cluster, users can still access an app that is also deployed in another cluster. With multiple clusters in different locations, users can also access the closest cluster and reduce network latency. For additional resiliency, you have the option to also select the multi-zone clusters, meaning your nodes are deployed across multiple zones within a location.

This tutorial highlights how Cloud Internet Services (CIS), a uniform platform to configure and manage the Domain Name System (DNS), Global Load Balancing (GLB), Web Application Firewall (WAF), and protection against Distributed Denial of Service (DDoS) for internet applications, can be integrated with Kubernetes clusters to support this scenario and to deliver a secure and resilient solution across multiple locations.

Objectives

- Deploy an application on multiple Kubernetes clusters in different locations.
- Distribute traffic across multiple clusters with a Global Load Balancer.
- Route users to the closest cluster.
- Protect your application from security threats.
- Increase application performance with caching.



Figure 1. Architecture diagram of the tutorial

1. The developer builds a Docker image for the application.
2. The image is pushed to a Container Registry.
3. The application is deployed to Kubernetes clusters in Dallas and London.
4. End-users access the application.
5. IBM Cloud Internet Services is configured to intercept requests to the application and to distribute the load across the clusters. In addition, DDoS Protection and Web Application Firewall are enabled to protect the application from common threats. Optionally assets like images, CSS files are cached.

Before you begin

This tutorial requires:

- IBM Cloud CLI,
 - IBM Cloud Kubernetes Service plugin (**kubernetes-service**),
- **kubectl** to interact with Kubernetes clusters,

You will find instructions to download and install these tools for your operating environment in the [Getting started with solution tutorials](#) guide.

In addition, make sure you:

- own a custom domain so you can configure the DNS for this domain to point to IBM Cloud Internet Services name servers.
- and [understand the basics of Kubernetes](#).

Step 1: Deploy an application to one location

This tutorial deploys a Kubernetes application to clusters in multiple locations. You will start with one location, Dallas, and then repeat these steps for London.

Create a Kubernetes cluster

A minimal cluster with one (1) zone, one (1) worker node and the smallest available size (**Flavor**) is sufficient for this tutorial.

When creating the Kubernetes cluster below:

1. Set **Cluster name** to **my-us-cluster**.
2. Locate in **North America** and **Dallas**

Open the [Kubernetes clusters](#) and click **Create cluster**. See the documentation referenced below for more details based on the cluster type. Summary:

- Click **Standard tier cluster**
- For Kubernetes on VPC infrastructure see reference documentation [Creating VPC clusters](#).
 - Click **Create VPC**:
 - Enter a **name** for the VPC.
 - Chose the same resource group as the cluster.
 - Click **Create**.
 - Attach a Public Gateway to each of the subnets that you create:
 - Navigate to the [Virtual private clouds](#).

- Click the previously created VPC used for the cluster.
 - Scroll down to subnets section and click a subnet.
 - In the **Public Gateway** section, click **Detached** to change the state to **Attached**.
 - Click the browser **back** button to return to the VPC details page.
 - Repeat the previous three steps to attach a public gateway to each subnet.
- For Kubernetes on Classic infrastructure see reference documentation [Creating classic cluster](#).
- Choose a resource group.
- Uncheck all zones except one.
- Scale down to 1 **Worker nodes per zone**.
- Choose the smallest **Worker Pool flavor**.
- Enter a **Cluster name**.
- Click **Create**.

While the cluster is getting ready, you are going to prepare the application.

Deploy the application to the Kubernetes cluster

The cluster should be ready. You can check its status in the [Kubernetes Service](#) console.

1. Gain access to your cluster as described on the Access tab of your cluster. Something like:

```
$ MYCLUSTER=my-us-cluster
ibmcloud ks cluster config --cluster $MYCLUSTER
```

2. Create the deployment using a pre-built image of the application. The application source code can be found in this [GitHub repository](#).

```
$ kubectl create deploy hello-world-deployment --image=icr.io/solution-tutorials/tutorial-scalable-webapp-kubernetes
```

Example output: `deployment "hello-world-deployment" created`.

3. Make the application accessible within the cluster by creating a service:

```
$ kubectl expose deployment/hello-world-deployment --type=ClusterIP --port=80 --name=hello-world-service --target-port=3000
```

It returns message like `service "hello-world-service" exposed`. To see the services:

```
$ kubectl get services
```

4. Run the application in the cluster with two replicas:

```
$ kubectl scale deployment hello-world-deployment --replicas=2
```

5. You can check the status of the deployment with the following command:

```
$ kubectl get pods
```

Get the Ingress Subdomain assigned to the cluster

When a Kubernetes cluster is created, it gets assigned an Ingress subdomain (for example, *my-us-cluster.us-south.containers.appdomain.cloud*) and a public Application Load Balancer IP address.

1. Retrieve the Ingress subdomain of the cluster:

```
$ ibmcloud ks cluster get --cluster $MYCLUSTER
```

Look for the **Ingress Subdomain** value.

2. Make note of this information for a later step.

This tutorial uses the Ingress Subdomain to configure the Global Load Balancer. You could also replace the Ingress Subdomain with the public Application Load Balancer, ALB of the cluster. An `<IngressSubdomain>` looks something like `my-us-cluster-e7f2ca73139645ddf61a8702003a483a-0000.us-south.containers.appdomain.cloud`

Configure the Ingress for your DNS subdomain

It will be required to have your own DNS domain name and a global load balancer subdomain will be created below: `<glb_name>`.

`<your_domain_name>`. Something like `hello-world-service.example.com` `<glb_name> = hello-world-service` and `<your_domain_name> = example.com`

1. Create the file `glb-ingress.yaml` and replace the placeholders with their respective values:

```
$ apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: <glb-name>
  annotations:
    kubernetes.io/ingress.class: "public-iks-k8s-nginx"
spec:
  rules:
  - host: <glb-name>.<your_domain_name>
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: hello-world-service
            port:
              number: 80
```

2. Add the ingress instance:

```
$ kubectl apply -f glb-ingress.yaml
```

It can take a few minutes before ingress becomes available as indicated by a value in the ADDRESS column in the command:

```
$ kubectl get ingress
```

3. Now test by configuring the curl **Host** http header with your DNS subdomain name to override the default of `<IngressSubdomain>`:

```
$ curl --header 'Host: <glb_name>.<your_domain_name>' <IngressSubdomain>/hostname
```

The curl command would look something like this: `curl --header 'Host: hello-world-service.ibmom.com' my-us-cluster-e7f2ca73139645ddf61a8702003a483a-0000.us-south.containers.appdomain.cloud/hostname`

Step 2: And then to another location

Repeat the steps from above for the London location with the following replacements:

- In [Create a Kubernetes cluster](#) replace:
 - the cluster name **my-us-cluster** with **my-uk-cluster**;
 - the location from **North America** and **Dallas** with **Europe** and **London**.
- In the [Deploy the application to the Kubernetes cluster](#) replace:
 - Replace the MYCLUSTER= **my-us-cluster** with **my-uk-cluster**
- [Configure the Ingress for the DNS subdomain](#)

Step 3: Configure multi-location load-balancing

Your application is now running in two clusters but it is missing one component for the users to access either clusters transparently from a single entry point.

In this section, you will configure IBM Cloud Internet Services (CIS) to distribute the load between the two clusters. CIS is a one stop-shop service providing *Global Load Balancer (GLB)*, *Caching*, *Web Application Firewall (WAF)* and *Page rule* to secure your applications while ensuring the reliability and performance for your Cloud applications.

To configure a global load balancer, you will need:

- to point a custom domain to CIS name servers,
- to retrieve the Ingress Subdomain of the Kubernetes clusters,
- to configure health checks to validate the availability of your application,
- and to define origin pools pointing to the clusters.

Register a custom domain with IBM Cloud Internet Services

The first step is to create an instance of CIS and to point your custom domain to CIS name servers.

1. If you do not own a domain, you can buy one from a registrar.

2. Navigate to [IBM Cloud Internet Services](#) in the IBM Cloud catalog.
3. Set the service name, and click **Create** to create an instance of the service.
4. When the service instance is provisioned, click on **Add domain**.
5. Enter your domain name and click **Next**.
6. Setup your DNS records is an optional step and can be skipped for this tutorial. click on **Next**
7. When the name servers are assigned, configure your registrar or domain name provider to use the name servers listed.
8. After you've configured your registrar or the DNS provider, it may require up to 24 hours for the changes to take effect.



Tip: When the domain's status on the Overview page changes from *Pending* to *Active*, you can use the `dig <your_domain_name> ns` command to verify that the new name servers have taken effect.

Configure Health Check for the Global Load Balancer

Health Checks monitor responses to HTTP/HTTPS requests from origin pools on a set interval. They are used with origin pools to determine if the pools are still running properly.

1. In the IBM Cloud Internet Services dashboard, use the left navigation menu to select **Reliability > Global Load Balancers**.
2. Select the **Health checks** tab and click **Create**.
 1. Set **Name** to **hello-world-service**
 2. Set **Monitor Type** to **HTTP**.
 3. Set **Port** to **80**.
 4. Set **Path** to **/**.
 5. In the **Configure request headers (optional)** add Header name: **Host** and Value: `<glb_name>.<your_domain_name>`
 6. Click **Create**.



Tip: When building your own applications, you could define a dedicated health endpoint such as `/healthz` where you would report the application state.

Define Origin Pools

A pool is a group of origin servers that traffic is intelligently routed to when attached to a GLB. With clusters in the United Kingdom and United States, you can define location-based pools and configure CIS to redirect users to the closest clusters based on the geographical location of the user requests.

One pool for the cluster in Dallas

1. Select the **Origin pools** tab and click **Create**.
2. Set **Name** to **US**
3. Set **Origin Name** to **us-cluster**
4. Set **Origin Address** to the kubernetes service `<IngressSubdomain>` printed by `ibmcloud ks cluster get --cluster $MYCLUSTER` for the US cluster
5. Set **Health check** to the one created in the previous section
6. Set **Health Check Region** to **Western North America**
7. Click **Save**

One pool for the cluster in London

1. Select the **Origin pools** tab and click **Create**.
2. Set **Name** to **UK**
3. Set **Origin Name** to **uk-cluster**
4. Set **Origin Address** to the kubernetes service `<IngressSubdomain>` printed by `ibmcloud ks cluster get --cluster $MYCLUSTER` for the UK cluster
5. Set **Health check** to the one created in the previous section
6. Set **Health Check Region** to **Western Europe**
7. Click **Save**

And one pool with both clusters

1. Select the **Origin pools** tab and click **Create**.
2. Set **Name** to **All**
3. Add two origins:
 1. one with **Origin Name** set to **us-cluster** and the **Origin Address** set to `<IngressSubdomain>` in Dallas
 2. one with **Origin Name** set to **uk-cluster** and the **Origin Address** set to `<IngressSubdomain>` in London

- 4. Set **Health Check Region** to **Eastern North America**
- 5. Set **Health check** to the one created in the previous section
- 6. Click **Save**

Create the Global Load Balancer

With the origin pools defined, you can complete the configuration of the load balancer.

- 1. Select the **Load balancers** tab and click **Create**.
- 2. Enter a name, `<glb_name>`, under **Name** for the Global Load Balancer. This name will also be part of your universal application URL (`http://<glb_name>.<your_domain_name>`), regardless of the location.
- 3. Under **Geo routes**, click **Add route**
 - 1. Select **Default** from the **Region** drop down
 - 2. Select the pool **All**
 - 3. Click Add

Repeat the process to create the following:

Region	Origin Pool
Default	All
Western Europe	UK
Eastern Europe	UK
Northeast Asia	UK
Southeast Asia	UK
Western North America	US
Eastern North America	US

List of Geo routes to create

With this configuration, users in Europe and in Asia will be redirected to the cluster in London, users in US to the Dallas cluster. When a request does not match any of the defined route, it will be redirected to the **All origin pool**.

- 4. Click **Create**


At this stage, you have successfully configured a Global Load Balancer with Kubernetes clusters across multiple locations. You can access the GLB URL `http://<glb_name>.<your_domain_name>/hostname` to view your application. Based on your location, you are redirected to the closest cluster or a cluster from the default pool if CIS was not able to map your IP address to a specific location.

Step 4: Secure the application

Turn the Web Application Firewall On

The Web Application Firewall(WAF) protects your web application against ISO Layer 7 attacks. Usually, it is combined with grouped rule-sets, these rule-sets aim to protect against vulnerabilities in the application by filtering out malicious traffic.

- 1. In the IBM Cloud Internet Services dashboard, navigate to **Security**, then on the **WAF**.
- 2. Ensure the WAF is **On**.
- 3. Click **OWASP Rule Set**. From this page, you can review the **OWASP Core Rule Set** and individually enable or disable rules. When a rule is enabled, if an incoming request triggers the rule, the global threat score will be increased. The **Sensitivity** setting will decide whether an **Action** is triggered for the request.
 - 1. Leave default OWASP rule sets as it is.
 - 2. Set **Sensitivity** to **Low**.
 - 3. Set **Action** to **Simulate** to log all the events.
- 4. Click **CIS Rule Set**. This page shows additional rules based on common technology stacks for hosting websites.

**Tip:** For a secured connection with HTTPS, you can either obtain a certificate from [Let's Encrypt](#) as described in the following [IBM Cloud® blog](#) or through [IBM Cloud Secrets Manager](#).

Increase performance and protect from Denial of Service attacks

A distributed denial of service ([DDoS](#)) attack is a malicious attempt to disrupt normal traffic of a server, service, or network by overwhelming the target or its surrounding infrastructure with a flood of internet traffic. CIS is equipped to protect your domain from DDoS.

- 1. In the CIS dashboard, select **Reliability > Global Load Balancer**.
- 2. Locate the GLB you created in the **Load Balancers** table.
- 3. Enable the Security and Performance features in the **Proxy** column:



Your GLB is now protected. An immediate benefit is that the origin IP addresses of your clusters will be hidden from the clients. If CIS detects a threat for an upcoming request, the user may see a screen like this one before being redirected to your application:



In addition, you can now control what content gets cached by CIS and how long it stays cached. Go to **Performance > Caching** to define the global caching level and the browser expiration. You can customize the global security and caching rules with **Page Rules**. Page Rules enable fine-grained configuration using specific domain paths. As example with Page Rules, you could decide to cache all contents under **/assets** for **3 days**:



Step 5: Remove resources

Remove Kubernetes Cluster resources

- 1. Remove the Ingress, you can do so by running the following command:

```
$ kubectl delete -f glb-ingress.yaml
```

- 2. Remove the service, you can do so by running the following command:

```
$ kubectl delete service hello-world-service
```

- 3. Remove the deployment, you can do so by running the following command:

```
$ kubectl delete deployment hello-world-deployment
```

- 4. Delete the clusters if you created them specifically for this tutorial.


Remove CIS resources

- 1. Remove the GLB.
- 2. Remove the origin pools.
- 3. Remove the health checks.
- 4. Update the DNS for your custom domain.
- 5. Delete the CIS instance if you created it specifically for this tutorial.

Related content

- [IBM Cloud Internet Services](#)
- [Manage your IBM CIS for optimal security](#)
- [Kubernetes Service](#)
- [Building containers from images](#)
- [Best practice to secure traffic and internet application via CIS](#)
- [Improving App Availability with Multizone Clusters](#)

Continuous Deployment to Kubernetes

 **Tip:** This tutorial may incur costs. Use the [Cost Estimator](#) to generate a cost estimate based on your projected usage.

This tutorial walks you through the process setting up a continuous integration and delivery pipeline for containerized applications running on the Kubernetes Service. You will learn how to set up source control, then build, test and deploy the code to different deployment stages. Next, you will add

integrations to other services like Slack notifications.

Objectives

- Create development and production Kubernetes clusters.
- Create a sample application, run it locally and push it to a Git repository.
- Configure the DevOps delivery pipeline to connect to your Git repository, build and deploy the sample app to dev/prod environments.
- Explore and integrate the app to use Slack notifications.



Figure 1. Architecture diagram of the tutorial

1. The code is pushed to a private Git repository.
2. The pipeline picks up changes in Git and builds container image.
3. The container image is uploaded to registry. The app is deployed to the Development environment.
4. Once changes are validated, the app is deployed to the Production environment.
5. Notifications are sent to Slack to track the deployment activities.

Before you begin

This tutorial requires:

- [set up a registry namespace](#)
- and [understand the basics of Kubernetes](#).

Step 1: Create development Kubernetes cluster

Kubernetes Service delivers powerful tools by combining Docker and Kubernetes technologies, an intuitive user experience, and built-in security and isolation to automate the deployment, operation, scaling, and monitoring of containerized apps in a cluster of compute hosts.

A minimal cluster with one (1) zone, one (1) worker node and the smallest available size (**Flavor**) is sufficient for this tutorial. The name `mycluster` will be used in this tutorial.

Open the [Kubernetes clusters](#) and click **Create cluster**. See the documentation referenced below for more details based on the cluster type. Summary:

- Click **Standard tier cluster**
- For Kubernetes on VPC infrastructure see reference documentation [Creating VPC clusters](#).
 - Click **Create VPC**:
 - Enter a **name** for the VPC.
 - Chose the same resource group as the cluster.
 - Click **Create**.
 - Attach a Public Gateway to each of the subnets that you create:
 - Navigate to the [Virtual private clouds](#).
 - Click the previously created VPC used for the cluster.
 - Scroll down to subnets section and click a subnet.
 - In the **Public Gateway** section, click **Detached** to change the state to **Attached**.
 - Click the browser **back** button to return to the VPC details page.
 - Repeat the previous three steps to attach a public gateway to each subnet.
- For Kubernetes on Classic infrastructure see reference documentation [Creating classic cluster](#).
- Choose a resource group.
- Uncheck all zones except one.
- Scale down to 1 **Worker nodes per zone**.
- Choose the smallest **Worker Pool flavor**.
- Enter a **Cluster name**.
- Click **Create**.

Note: Do not proceed until your workers are ready.

Step 2: Create a sample application

1. From the [IBM Cloud console](#), use the left side menu option and select [DevOps](#).
2. Click **Create toolchain**.
3. In the **search box** type **kubernetes** as a filter.
4. Click on the **Develop a Kubernetes app with Helm** tile.
5. Enter a unique **Toolchain Name** for the toolchain such as `<your-initials>-mynodestarter-toolchain` and select a resource group.
6. Enter a unique **Repository Name** for the repository such as `<your-initials>-mynodestarter-repository`.
7. Click on the **Delivery Pipeline** tab.
8. Provide an IBM Cloud API Key. If you don't have one, create by clicking on **New**.
9. Select a region and your cluster from the list.

10. Make sure to set the cluster namespace to **default** and click **Create**.

In a new browser tab open the [Kubernetes clusters](#):

- Click on your cluster.
- Locate the **Ingress subdomain** field and click the copy button. The ingress subdomain will be used in the next section.

The toolchain will build your application and deploy it to the cluster.

1. Once the pipeline is created, click the pipeline under **Delivery Pipelines**.
2. Wait for the DEPLOY stage to complete. The **Check health** job will fail for VPC clusters.
3. Click on the settings cog in the DEPLOY stage and click **Configure stage**.
 - Click the **Environment properties** tab at the top.
 - Click **Add property**, click **Text property**.
 - Enter **Name**: HELM_UPGRADE_EXTRA_ARGS.
 - Enter **Value**: --set ingress.enabled=true,ingress.hosts={dev.INGRESS_SUBDOMAIN} something like: --set ingress.enabled=true,ingress.hosts={dev.vpc-e7f2ca73139645ddf61a8702003a483a-0000.us-south.containers.appdomain.cloud}.
4. Click Save.
5. Click the **play** button on DEPLOY stage.
6. In a browser tab paste the dev.INGRESS_SUBDOMAIN of the cluster. The message from the application should be returned: **Welcome to IBM Cloud DevOps with Docker, Kubernetes and Helm Charts. Lets go use the Continuous Delivery Service**.
7. Switch back to the pipeline tab.

Step 3: Modify the application and deploy the updates

1. Follow the breadcrumbs on the upper left of the screen and click on the first entry after of **<your-initials>-mynodestarter** after **Toolchains**.
2. Click the link under the **Repositories** tile, a new browser tab will open to the repository.
3. Click on the **utils.js** file and then click on **Open in Web IDE/Edit** drop down and choose **EDIT** then click **EDIT**.
4. Make a simple change, for example change "Welcome to" to something else.
5. Enter a commit message: *my first changes* and click on **Commit changes**.
6. Return to the previous tab showing the toolchain.
7. Click on the **Delivery Pipeline** tile named **ci-pipeline**.
8. Notice a new **BUILD** has started.
9. Wait for the **DEPLOY** stage to complete.
10. After the DEPLOY stage completes open a new tab and paste in the dev.INGRESS_SUBDOMAIN of the cluster.

If you don't see your application updating, confirm all the steps passed and review their respective logs.

Note: If you prefer to work locally for making and viewing updates to the application. Once your changes are pushed to the repository they will also trigger a build in the **Delivery Pipeline**.

Step 4: Deploy to a production environment

In this section, you will complete the deployment pipeline by deploying the application to development and production environments respectively.

There are [different options](#) to handle the deployment of an application to multiple environments. In this tutorial, you will deploy the application to two different namespaces.

1. Go to the toolchain you created earlier and click the **Delivery Pipeline** tile.
2. Rename the **DEPLOY** stage to **Deploy dev** by clicking on the settings icon, then **Configure Stage**.



3. To save the changes scroll down and click **Save**.
4. Clone the **Deploy dev** stage (settings icon > Clone Stage) and name the cloned stage as **Deploy prod**.
5. On the **Input** panel change the **stage trigger** to **Run jobs only when this stage is run manually**.
6. In **Environment properties** panel, set **CLUSTER_NAMESPACE** to **production**. Update the **HELM_UPGRADE_EXTRA_ARGS** property **Value**: --set ingress.enabled=true,ingress.hosts={**prod**.INGRESS_SUBDOMAIN}
7. **Save** the stage.
8. Click the **Play** button on the **Deploy prod** stage just created.

You now have the full deployment setup. To deploy from dev to production, you must manually run the **Deploy prod** stage. This is a simplification process stage over a more advanced scenario where you would include unit tests and integration tests as part of the pipeline.



You now have the full deployment setup. To deploy from dev to production, you manually run the **Run Pipeline**. This is a simplification process stage over a more advanced scenario where you would include unit tests, integration tests and automated deployment as part of the pipeline.

Step 5: Setup Slack notifications

1. For **Slack webhook**, follow the steps in this [link](#). You need to login with your Slack credentials and provide an existing channel name or create a new one. Copy the **Webhook URL** for later use.
2. Go back to view the list of [toolchains](#) and select your toolchain, then click on **Add**.
3. Search for Slack in the search box or scroll down to see **Slack**. Click to see the configuration page.



Configure the Slack integration

4. Once the Incoming webhook integration is added, copy the **Webhook URL** captured earlier and paste under **Slack webhook**.
5. The Slack channel is the channel name you provided while creating a webhook integration above.
6. **Slack team name** is the team-name(first part) of team-name.slack.com. for example, kube is the team name in kube.slack.com
7. Click **Create Integration**. A new tile will be added to your toolchain.



Toolchain with new Slack integration

8. From now on, whenever your toolchain executes, you should see Slack notifications in the channel you configured.



Slack app with notification

Step 6: Remove resources

In this step, you will clean up the resources to remove what you created above.

- Delete the Git repository.
 - Back to the toolchain, click the link under the **Repositories** tile, a new browser tab will open to the repository.
 - In the git repository: select **Settings** on the left then **General** scroll down and click **Advanced Expand** then scroll down and click **Delete Project**.
- Delete the toolchain.
- Delete the images from the [Container Registry](#).
- Delete the cluster.
- Delete the Slack app.

Expand the Tutorial

Do you want to learn more? Here are some ideas of what you can do next:


- [Analyze logs and monitor application health](#).
- Add a 3rd environment dedicated to testing.


Related Content

- End to end Kubernetes solution guide, [moving VM based apps to Kubernetes](#).
- [Security](#) for IBM Cloud Kubernetes Service.
- Toolchain [integrations](#).
- Analyze logs and monitor [application health](#).

Setting up the Container Registry CLI and namespace

To manage your Docker images in IBM Cloud® Container Registry, you must install the `container-registry` CLI plug-in and create a *namespace* in a *resource group*.

 **Important:** Do not put personal information in your container images, namespace names, description fields, or in any image configuration data (for example, image names or image labels).

 **Tip:** To add and remove namespaces, you must have the Manager role at the account level, see [Access roles for configuring IBM Cloud Container Registry](#).

Before you begin, install the IBM Cloud CLI, see [Getting started with the IBM Cloud CLI](#).

Installing the `container-registry` CLI plug-in


Install the `container-registry` CLI plug-in so that you can use the command line to manage your namespaces and Docker images in IBM Cloud Container Registry.

1. Install the `container-registry` CLI plug-in by running the following command:

```
$ ibmcloud plugin install container-registry
```

For more information about installing plug-ins, see [Extending IBM Cloud CLI with plug-ins](#).

2. Optional: [Configure your Docker client to run commands without root permissions](#). If you do not do this step, you must run `ibmcloud login`, `ibmcloud cr login`, `docker pull`, and `docker push` commands with `sudo` or as root.

 **Tip:** IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).

You can now [set up your own namespace](#) in IBM Cloud Container Registry.

Updating the `container-registry` CLI plug-in

You might want to update the `container-registry` CLI plug-in periodically to use new features.

Updating `container-registry` CLI plug-in version 1.0

To update version 1.0 of the Container Registry CLI, run the following command:

```
$ ibmcloud plugin update container-registry
```

Updating `container-registry` CLI plug-in version 0.1

To update version 0.1 of the Container Registry CLI, run the following command, where `<version_number>` is the number of the version of the CLI.

 **Deprecated:** Version 0.1 of the Container Registry CLI is deprecated, see [All releases of Container Registry plug-in 0.1 are deprecated](#).

```
$ ibmcloud plugin install container-registry -v <version_number>
```

For example, to update the CLI to version 0.1.584, run the following command:

```
$ ibmcloud plugin install container-registry -v 0.1.584
```

Uninstalling the `container-registry` CLI plug-in

If you no longer need the `container-registry` CLI plug-in, you can uninstall it.

To uninstall the `container-registry` CLI plug-in, run the following command:


```
$ ibmcloud plugin uninstall container-registry
```

Planning namespaces


IBM Cloud Container Registry provides a multi-tenant private image *registry* that is hosted and managed by IBM. You can store and share your Docker images in this registry by setting up a registry *namespace*.

Namespaces are created in a *resource group* that you specify so that you can configure access to resources within the namespace at the [resource group](#) level. If you don't specify a resource group, and a resource group isn't targeted, the default resource group is used. However, you can still set permissions for the namespace at the account level or in the namespace itself. If you don't specify a resource group, and a resource group isn't targeted, the default resource group is used.


If you have an older namespace that isn't in a resource group, you can assign it to a resource group and then set permissions for that namespace at the resource group level. For more information about resource groups, see [Assigning existing namespaces to resource groups](#).

 **Tip:** Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console.

You can set up multiple namespaces, for example, to have separate repositories for your production and staging environments. If you want to use the registry in multiple IBM Cloud regions, you must set up a namespace for each region. Namespace names are unique within regions. You can use the same namespace name for each region, unless someone else already has a namespace with that name in that region.


 **Note:** You can have 100 namespaces in each region.

To work with the IBM-provided public images only, you do not need to set up a namespace.

 **Tip:** If you're unsure whether a namespace is already set for your account, run the `ibmcloud cr namespace-list` command with the `-v` option to retrieve existing namespace information.

Consider the following rules when you choose a namespace:

- Your namespace must be unique across all IBM Cloud accounts in the same region.
- Your namespace must have 4 - 30 characters.
- Your namespace must start and end with a letter or number.
- Your namespace must contain lowercase letters, numbers, hyphens (-), and underscores (_) only.

 **Important:** Do not put personal information in your namespace names.

After you set your first namespace, you are assigned to the free IBM Cloud Container Registry service plan unless you [upgrade your plan](#).

User permissions for working with namespaces

You can control which users can work with namespaces by using IAM roles.

- To add, assign, and remove namespaces, you must have the Manager role in the IBM Cloud Container Registry service at the account level, see [Access roles for configuring IBM Cloud Container Registry](#).
 - To add and assign namespaces, you must also have the Viewer platform role for the resource group in which you want to create the namespace. To assign the Viewer role for a resource group to a user, run the following `ibmcloud iam user-policy-create` command, where `<user>` is the name of the user and `<resource_group_id>` is the resource group ID:

```
$ ibmcloud iam user-policy-create <user> --roles Viewer --resource-type resource-group --resource <resource_group_id>
```

- To view and analyze namespaces, you must have the Reader or Manager role in the IBM Cloud Container Registry service, see [Access roles for using IBM Cloud Container Registry](#).

For more information about user roles, [Defining IAM access policies](#).

Setting up a namespace

You must create a namespace to store your Docker images in IBM Cloud Container Registry.


Before you begin, complete the following tasks:

- [Install the IBM Cloud CLI and the container-registry CLI plug-in](#).
- [Plan how to use and name your registry namespaces](#).

To create a namespace, see [Set up a namespace](#). Namespaces are created in the resource group that you specify so that you can configure access to resources within the namespace at the resource group level. If you don't specify a resource group, and a resource group isn't targeted, the default resource

group is used. For more information about resource groups, see [Managing resource groups](#).


Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console.

 **Tip:** The namespace must be unique across all IBM Cloud accounts in the same region. Namespaces must have 4 - 30 characters, and contain lowercase letters, numbers, hyphens (-), and underscores (_) only. Namespaces must start and end with a letter or number.

You can now [push Docker images to your namespace in IBM Cloud Container Registry](#) and share these images with other users in your account. To control access to namespaces in IBM Cloud IAM, see [Creating policies](#).

Assigning existing namespaces to resource groups

Namespaces created in version 0.1.484 of the CLI or earlier and in the IBM Cloud console before 29 July 2020, aren't assigned to resource groups. If you have a namespace that isn't assigned to a resource group, you can assign the namespace to a resource group and then set permissions for that namespace at the resource group level.

 **Note:** You can assign a namespace to a resource group only once. When a namespace is in a resource group, you can't move it to another resource group.

You can assign an existing namespace to a resource group by using the [ibmcloud cr namespace-assign](#) command. To find out which namespaces are assigned to resource groups and which are unassigned, run the [ibmcloud cr namespace-list](#) command with the `-v` option.

Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console.

 **Tip:** If the namespaces don't all show in the **Resource list** page, see [Why don't all my namespaces show in the Resource list?](#) for assistance.

For more information about resource groups, see [Creating a resource group](#).

To assign an existing namespace to a resource group, complete the following steps:

1. Log in to IBM Cloud.

```
$ ibmcloud login
```

2. To find the namespace, list the available namespaces.

```
$ ibmcloud cr namespace-list -v
```

3. Assign the namespace to a resource group.

Replace `<my_resource_group>` with the name or ID of the resource group and `<my_namespace>` with the name of the namespace.

```
$ ibmcloud cr namespace-assign -g <my_resource_group> <my_namespace>
```

Removing namespaces

If you no longer require a registry namespace, you can remove the namespace from your IBM Cloud account.


1. Log in to IBM Cloud.

```
$ ibmcloud login
```

2. List available namespaces.

```
$ ibmcloud cr namespace-list
```

3. Remove a namespace.

 **Important:** When you remove a namespace, any images that are stored in that namespace are also deleted. This action cannot be undone.

Replace `<my_namespace>` with the namespace that you want to remove.

```
$ ibmcloud cr namespace-rm <my_namespace>
```

After you delete a namespace, it might take a few minutes before that namespace becomes available again to reuse.

Setting up Container Registry as a private registry on Red Hat OpenShift

You can add value to your Red Hat® OpenShift® Container Platform clusters by using IBM Cloud® Container Registry even where an internal registry is already provided.

For example, if you have multiple clusters, Container Registry integrates conveniently with Red Hat OpenShift Container Platform clusters so that you can build, share, synchronize, and scan image assets across clusters. For more information, see [Choosing an image registry solution](#).

You can set up Container Registry to work with the internal registry of [Red Hat OpenShift on IBM Cloud](#) or [other Red Hat OpenShift Container Platform providers](#).

Set up Red Hat OpenShift on IBM Cloud to use Container Registry

By default, your Red Hat OpenShift on IBM Cloud clusters are set up with an internal registry that stores images locally in your cluster. The clusters are also set up with image pull secrets in the `default` project to pull images that you store in your private Container Registry repositories.

You can use either registry separately or in combination. When you set up the Red Hat OpenShift on IBM Cloud internal registry to import images from Container Registry, you get the advantage of a private registry that is common to multiple clusters. Another benefit is that copies of the pulled images from Container Registry are stored locally on the cluster, therefore reducing latency and external traffic, but you are subject to storage limitations.

To set up your Red Hat OpenShift on IBM Cloud clusters to use the internal registry in combination with Container Registry, see the following topics in the Red Hat OpenShift on IBM Cloud documentation:

- [Importing images from IBM Cloud Container Registry into the internal registry image stream](#).
- [Setting up builds in the internal registry to push images to IBM Cloud Container Registry](#).

Set up Red Hat OpenShift Container Platform to use Container Registry

To set up Red Hat OpenShift Container Platform, you must create secrets that have the credentials to access Container Registry so that you can perform the following actions.

- **Pull** Create image pull secrets to [pull images](#) from Container Registry to your Red Hat OpenShift cluster. For example, you might deploy an app that uses an image in a private registry.
- **Push** Create image push secrets to [push images](#) from your Red Hat OpenShift cluster to a repository in Container Registry. For example, you might set up a continuous delivery pipeline that builds an image to a private registry instead of the internal registry.
- **Both** Create a secret that can pull images from and push images to Container Registry. For example, you might set up a continuous delivery pipeline that builds an image to a private registry so that your team can pull the most recent image across multiple clusters.

Set up the Red Hat OpenShift Container Platform internal registry to pull from Container Registry

To configure Red Hat OpenShift Container Platform to pull from Container Registry, you must complete the following steps:

1. [Set up image pull secrets to IBM Cloud Container Registry](#) for each project that you want to pull images in.
2. [Configure Red Hat OpenShift Container Platform to use the image pull secrets](#) by adding the secrets to a service account in each project or by referring to the secret in your `pod` deployment. You are only required to add the secret to the projects that you want to pull to.

Set up the Red Hat OpenShift Container Platform build to push images to Container Registry

If you want to push application images from Red Hat OpenShift Container Platform to Container Registry, you must edit the Red Hat OpenShift Container Platform build configuration to point at Container Registry, where `myregistry.mycompany.io` is `<region_domain_name>.icr.io`. For more information about Container Registry regions and domain names, see [Regions](#).

For instructions, see [Setting up builds in the internal registry to push images to IBM Cloud Container Registry](#).

Adding images to your namespace

You can securely store and share Docker images with other users by adding images to your `namespace` in IBM Cloud® Container Registry.

Every image that you want to add to your namespace must exist on your local computer first. You can either download (pull) an image from another repository to your local computer, or build your own image from a *Dockerfile* by using the Docker `build` command. To add an image to your namespace, you must upload (push) the local image to your namespace in IBM Cloud Container Registry.

Important: Do not put personal information in your container images, namespace names, description fields, or in any image configuration data (for example, image names or image labels).

Pulling images from another registry

You can pull (download) an image from any private or public *registry* source, and then tag it for later use in IBM Cloud Container Registry.

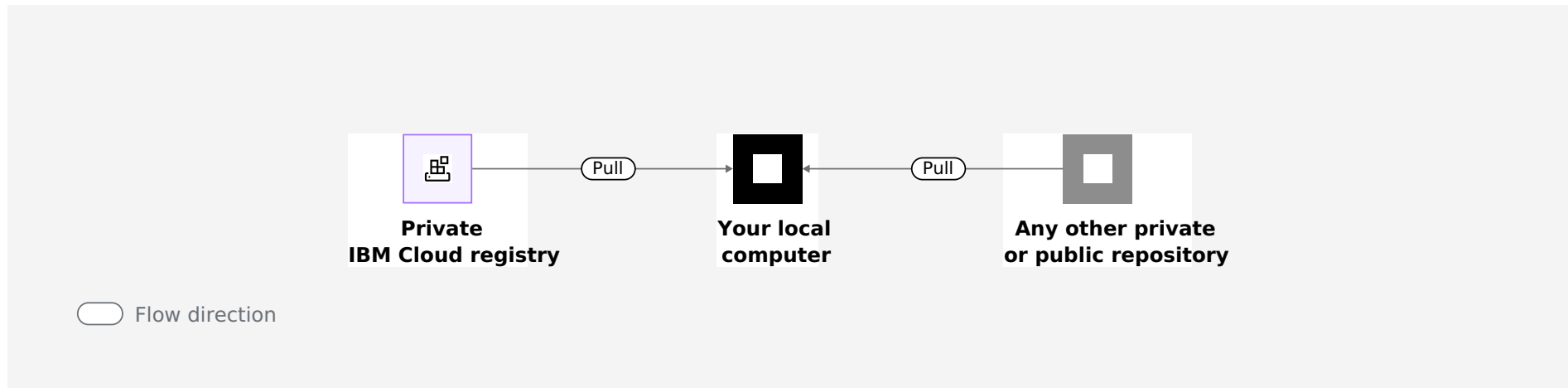


Figure 1. Pulling images from another registry

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with images in your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- [Make sure that you can run Docker commands without root permissions](#). If your Docker client is set up to require root permissions, you must run `ibmcloud login`, `ibmcloud cr login`, `docker pull`, and `docker push` commands with `sudo`.

If you change your permissions to run Docker commands without root privileges, you must run the `ibmcloud login` command again.

1. Download the image, see [Pull an image](#) in the Getting Started documentation.

Tip: If you get an `unauthorized: authentication required` or a `denied: requested access to the resource is denied` message, run the `ibmcloud cr login` command.

After you pull an image and `tag` it for your `namespace`, you can upload (push) the image from your local computer to your namespace.

Tip: If you deploy a workload that pulls an image from Container Registry and your pods fail with an `ImagePullBackOff` status, see [Why do images fail to pull from registry with ImagePullBackOff or authorization errors?](#) for assistance.

Pushing Docker images to your namespace

You can push (upload) an image to your namespace in IBM Cloud Container Registry to store and share your image with other users.

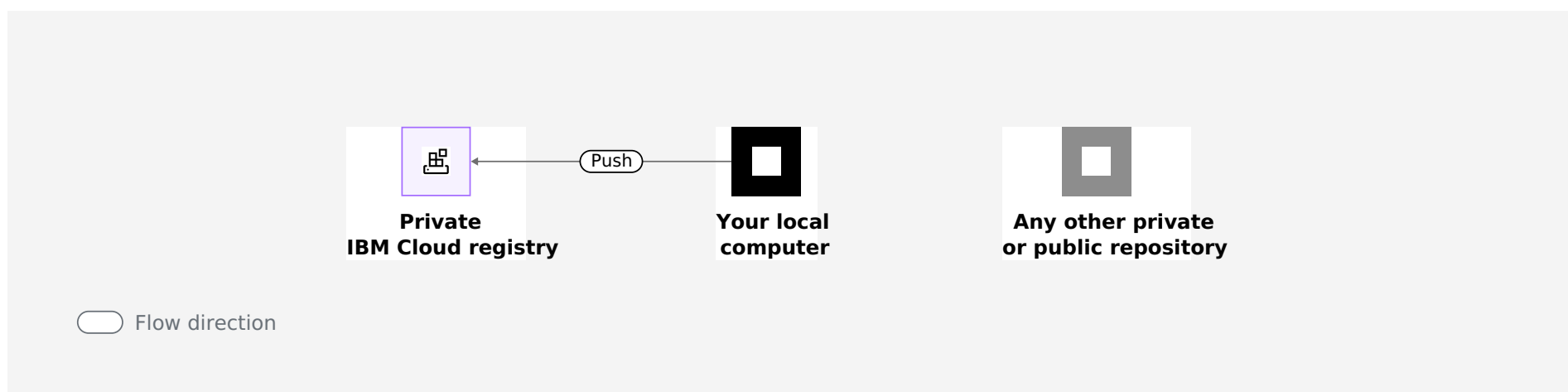


Figure 2. Pushing Docker images to your namespace

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with images in your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- [Pull](#) or [build](#) an image on your local computer and tag the image with your namespace information.
- [Make sure that you can run Docker commands without root permissions](#). If your Docker client is set up to require root permissions, you must run `ibmcloud login`, `ibmcloud cr login`, `docker pull`, and `docker push` commands with `sudo`.

If you change your permissions to run Docker commands without root privileges, you must run the `ibmcloud login` command again.

✓ **Tip:** IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces interactively](#).

To upload (push) an image, complete the following steps:

1. Log in to the CLI by running the `ibmcloud cr login` command.

```
$ ibmcloud cr login
```

✓ **Tip:** You must log in if you pull an image from your private IBM Cloud Container Registry.

✓ **Tip:** If you have a problem when you try to log in, see [Why can't I log in to Container Registry?](#) for assistance.

2. To view all namespaces that are available in your account, run the `ibmcloud cr namespace-list` command.
3. [Upload the image to your namespace](#).

✓ **Tip:** If you get an `unauthorized: authentication required` or a `denied: requested access to the resource is denied` message, run the `ibmcloud cr login` command.

After you push your image to IBM Cloud Container Registry, you can do one of the following tasks.

- [Manage security with Vulnerability Advisor](#) to find information about potential security issues and vulnerabilities.
- [Create a cluster and use this image to deploy a container](#) to the cluster in IBM Cloud Kubernetes Service.

Copying images between registries

You can pull an image from a registry in one region and push it to a registry in another region so that you can share the image with users in both regions.

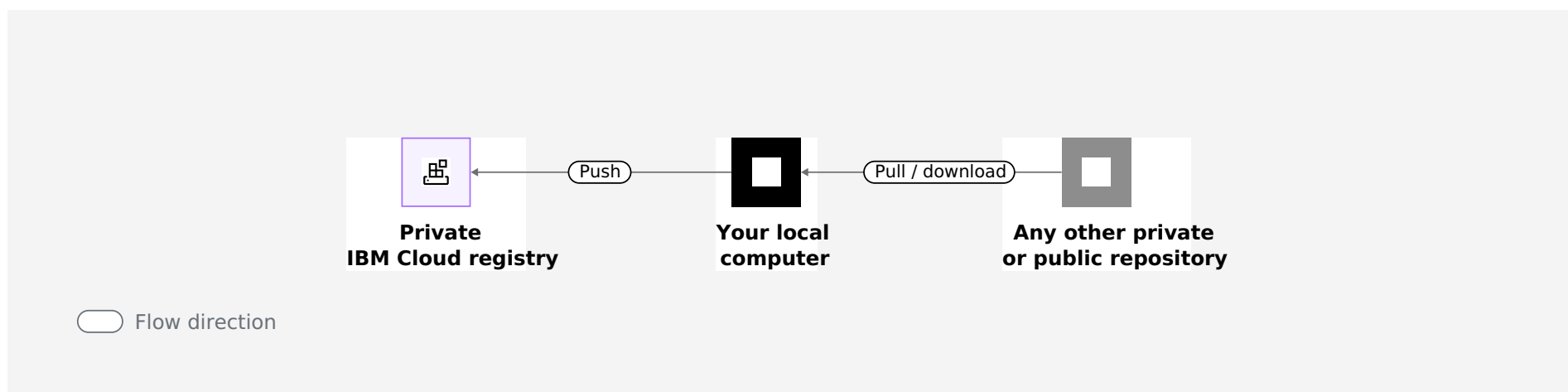


Figure 3. Copying images between registries

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with images in your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- [Make sure that you can run Docker commands without root permissions](#). If your Docker client is set up to require root permissions, you must run `ibmcloud login`, `ibmcloud cr login`, `docker pull`, and `docker push` commands with `sudo`.

If you change your permissions to run Docker commands without root privileges, you must run the `ibmcloud login` command again.

To copy an image between two registries, complete the following steps:

1. [Pull an image from a registry](#).
2. [Push the image to another registry](#). Make sure that you use the correct `domain name` for the new region you're targeting.


After you copy your image, you can do one of the following tasks.

- [Manage image security with Vulnerability Advisor](#) to find information about potential security issues and vulnerabilities.
- [Create a cluster and use this image to deploy a container](#) to the cluster in IBM Cloud Kubernetes Service.

Creating images that refer to a source image

Create an image by using the `ibmcloud cr image-tag` command.

In the region that you are logged in to, create an image in IBM Cloud Container Registry that refers to an existing image in the same region. This action is supported for source images that are created by using supported versions of Docker Engine, see [Support for Docker](#).

 **Tip:** New images that are created by using this mechanism do not retain signatures. If you require the new image to be signed, do not use this mechanism.

Before you begin, complete the following tasks.


- [Install the CLI](#) to work with images in your namespace.
- Ensure that you have access to a private namespace in IBM Cloud Container Registry that contains a source image to which you want to refer another image.

To create an image from a source image, complete the following steps.

1. Log in to the CLI by running the `ibmcloud cr login` command.

```
$ ibmcloud cr login
```

2. Run the following command to add the new reference, where `SOURCE_IMAGE` is the name of your source image and `TARGET_IMAGE` is the name of your target image. The source and target images must be in the same region. `SOURCE_IMAGE` must be in the format `repository:tag` or `repository@digest` and `TARGET_IMAGE` must be in the format `repository:tag`, for example, `us.icr.io/namespace/image:latest`.

 **Tip:** To find the names of your images, run `ibmcloud cr image-list`. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`. To identify your image by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (`@`) symbol to create the image name in the format `repository@digest`. If the list images command times out, see [Why is it timing out when I list images?](#) for assistance.

```
$ ibmcloud cr image-tag [SOURCE_IMAGE] [TARGET_IMAGE]
```

3. Verify that the new image was created by running the following command, and check that the image is shown in the list with the same image [digest](#) as the source image.

```
$ ibmcloud cr image-list
```

Building Docker images to use them with your namespace

You can build a Docker image directly in IBM Cloud or create your own Docker image on your local computer and upload (push) it to your namespace in IBM Cloud Container Registry.

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with images in your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- [Make sure that you can run Docker commands without root permissions](#). If your Docker client is set up to require root permissions, you must run `ibmcloud login`, `ibmcloud cr login`, `docker pull`, and `docker push` commands with `sudo`.

If you change your permissions to run Docker commands without root privileges, you must run the `ibmcloud login` command again.

A Docker image is the basis for every container that you create. An image is created from a Dockerfile, which is a file that contains instructions to build the image. A Dockerfile might reference build artifacts in its instructions that are stored separately, such as an app, the configuration of the app, and its dependencies.

If you want to take advantage of IBM Cloud compute resources and internet connection or Docker is not installed on your workstation, build your image directly in IBM Cloud. If you need to access resources in your build that are on servers that are behind your firewall, build your image locally.

To build your own Docker image, complete the following steps:

1. Create a local directory where you want to store the build context. The build context contains your Dockerfile and related build artifacts, such as the app code. Navigate to this directory in a command-line window.
2. Create a Dockerfile.

1. Create a Dockerfile in your local directory.

```
$ touch Dockerfile
```

2. Use a text editor to open the Dockerfile. At a minimum, you must add the base image to build your image from. Replace `<source_image>` and `<tag>` with the image repository and tag that you want to use. If you're using an image from another private registry, define the full path to the image in IBM Cloud Container Registry.

```
$ FROM <source_image>:<tag>
```

For example, to create a Dockerfile that is based on the public IBM WebSphere Application Server Liberty (`ibm/liberty`) image, use the following command.

```
$ FROM icr.io/ibm/liberty:latest
LABEL description="This is my test Dockerfile"
EXPOSE 9080
```

This example adds a label to the image metadata and exposes port 9080. For more Dockerfile instructions that you can use, see the [Dockerfile Reference](#).

3. Decide on a name for your image. The image name must be in the following format, where `<my_namespace>` is your namespace information, `<repo_name>` is the name of your repository, and `<tag>` is the version that you want to use for your image:

```
$ <region>.icr.io/<my_namespace>/<repo_name>:<tag>
```



Tip: To find your namespace, run the `ibmcloud cr namespace-list` command.

4. Take note of the path to the directory that contains your Dockerfile. If you run the commands in the following steps while your working directory is set to where your build context is stored, you can replace `<directory>` with a period (.).
5. Build and test your image locally before you push it to IBM Cloud.

1. Build the image from your Dockerfile on your local computer and tag it with your image name, where `<image_name>` is the name of your image and `<directory>` is the path to the directory.

```
$ docker build -t <image_name> <directory>
```

2. Optional: Test your image on your local computer before you push it to your namespace.

```
$ docker run <image_name>
```

Replace `<image_name>` with the name of your image.

3. After you create your image and tag it for your namespace, [you can push your image to your namespace in IBM Cloud Container Registry](#).

To use Vulnerability Advisor to check the security of your image, see [Managing image security with Vulnerability Advisor](#).

Pushing images by using an API key

Create a service ID that uses an *API key* to push images to IBM Cloud Container Registry.


Complete the following steps:

1. Create a service ID, see [Creating and working with service IDs](#).
2. Create a policy that gives the service ID permission to access the registry, for example, Administrator and Manager roles, see [Managing IAM access for Container Registry](#).
3. Create an API key, see [Creating an API key for a service ID](#).
4. Use the API key to log in to registry so that you can push images to the registry, see [Automating access to IBM Cloud Container Registry](#).
5. Push your images, see [Pushing Docker images to your namespace](#).

You can now use clusters to pull the images, see [Building containers from images](#).

Removing tags from images in your private repository


You can remove a [tag](#), or tags, from an image in your private IBM Cloud repository, and leave the underlying image and any other tags in place by using the `ibmcloud cr image-untag` command.

 **Tip:** If multiple tags exist for the same image [digest](#) within a repository and you want to remove the underlying image and all its tags, see [Deleting images from your private IBM Cloud repository](#).

To remove a tag, or tags, by using the CLI, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. To remove a tag, run the following command, where `IMAGE` is the name of the image that you want to remove, in the format `repository:tag`. If a tag is not specified in the image name, the command fails. You can delete the tags for multiple images by listing each private IBM Cloud registry path in the command with a space between each path.

```
$ ibmcloud cr image-untag IMAGE
```

 **Tip:** To find the names of your images, run `ibmcloud cr image-list`. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`.

3. Verify that the tag was removed by running the following command, and check that the tag does not show in the list.


```
$ ibmcloud cr image-list
```


 **Tip:** If the list images command times out, see [Why is it timing out when I list images?](#) for assistance.


Deleting images from your private repository

You can delete unwanted images from your private IBM Cloud repository by using either the IBM Cloud console or the CLI.

If you want to delete a private repository and its associated images, see [Deleting a private repository and any associated images](#).


 **Important:** Deleting an image that is being used by an existing deployment might cause scale-up, reschedule, or both, to fail.


 **Tip:** If you want to restore a deleted image, you can list the contents of the trash by running the `ibmcloud cr trash-list` command and restore a selected image by running the `ibmcloud cr image-restore` command.

 **Tip:** Where multiple tags exist for the same image digest within a repository, the `ibmcloud cr image-rm` command removes the underlying image and all its tags. If the same image exists in a different repository or namespace, the copy of the image is not removed. If you want to remove a tag from an image and leave the underlying image and any other tags in place, see [Removing tags from images in your private repository](#) command.

Deleting images from your private repository in the CLI

You can delete unwanted images and all their tags from your private IBM Cloud repository by using the CLI.

 **Important:** Deleting an image that is being used by an existing deployment might cause scale-up, reschedule, or both, to fail.

 **Tip:** If you want to restore a deleted image, you can list the contents of the trash by running the `ibmcloud cr trash-list` command and restore a selected image by running the `ibmcloud cr image-restore` command.

To delete an image by using the CLI, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. To delete an image, run the following command, where `IMAGE` is the name of the image that you want to remove, in the format `repository@digest` or `repository:tag`. If a tag is not specified in the image name, the image tagged `latest` is deleted by default. You can delete multiple images by listing each private IBM Cloud registry path in the command with a space between each path.

```
$ ibmcloud cr image-rm IMAGE
```

✓ **Tip:** To find the names of your images, run `ibmcloud cr image-list`. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`. To identify your image by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (`@`) symbol to create the image name in the format `repository@digest`. If the list images command times out, see [Why is it timing out when I list images?](#) for assistance.

3. Verify that the image was deleted by running the following command, and check that the image does not show in the list.

```
$ ibmcloud cr image-list
```

Deleting images from your private repository in the console

You can delete unwanted images and all their tags from your private IBM Cloud image repository by using the IBM Cloud console.



Important: Deleting an image that is being used by an existing deployment might cause scale-up, reschedule, or both, to fail.



Tip: If you want to restore a deleted image, you can list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore a selected image by running the [ibmcloud cr image-restore](#) command.

To delete an image by using the IBM Cloud console, complete the following steps:

1. Log in to the IBM Cloud console <https://cloud.ibm.com/login> with your IBMid.
2. If you have multiple IBM Cloud accounts, from the account menu, select the account and region that you want to use.
3. Click the **Navigation menu** icon, then click **Container Registry**.
4. Click **Images**. A list of your images is displayed.
5. In the row that contains the image that you want to delete, select the checkbox.
6. Click **Delete Image**.

Listing images in the trash

You can list deleted images that are in the trash and see when they expire.

To find out which images are in the trash, you can use the [ibmcloud cr trash-list](#) command. Images are stored in the trash for 30 days.

To list the images in the trash, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. List the images in the trash by running the following command:

```
$ ibmcloud cr trash-list
```

3. List only the images in the trash for the namespace that you're interested in by running the following command, where `<namespace>` is your namespace:

```
$ ibmcloud cr trash-list --restrict <namespace>
```

Restoring images

You can restore images from the trash. Deleted images are stored in the trash for 30 days.

You can restore an image from the trash by running the [ibmcloud cr image-restore](#) command. To find out which images are in the trash, run the [ibmcloud cr trash-list](#) command.

You can restore images by running the [ibmcloud cr image-restore](#) command. You can use the following options:

- `<repo>@<digest>`, which restores the digest and all its tags in the repository that aren't already in the live repository, see [Restoring images by digest](#).
- `<repo>:<tag>`, which restores the tag, see [Restoring images by tag](#).

Restoring images by digest

When you restore an image by digest, the digest is copied from the trash into your live repository, and all the tags for the digest in the repository are

restored. The digest continues to show in the trash because a copy is restored.

To restore an image by digest from the trash, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. List the images in the trash by running the following command:

```
$ ibmcloud cr trash-list
```

A table is displayed that shows the items in the trash. The table shows the digest, the days until expiry, and the tags for that digest.

3. Note the digest for the image that you want to restore.
4. Run the following command to restore the image to your repository. Where `<dns>` is the [domain name](#), `<namespace>` is the [namespace](#), `<repo>` is the [repository](#), and `<digest>` is the [digest](#) of the image that you want to restore.

```
$ ibmcloud cr image-restore <dns>/<namespace>/<repo>@<digest>
```



Tip: If some tags aren't restored, see [Why aren't all the tags restored when I restore by digest?](#) for assistance.



Tip: In your live repository, you can pull the image by digest. If you run the `ibmcloud cr image-digests` command, the image shows in the output.

Restoring images by tag

When you restore an image by tag, only that specific tag is moved out of the trash into your live repository.

To restore an image by tag from the trash, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. List the images in the trash by running the following command:

```
$ ibmcloud cr trash-list
```

A table is displayed that shows the items in the trash. The table shows the digest, the days until expiry, and the tags for that digest.

3. For the image that you want to restore, make a note of the digest up to, but not including, the at sign (`@`). This part of the digest is `<dns>/<namespace>/<repo>`, where `<dns>` is the domain name, `<namespace>` is the namespace, and `<repo>` is the repository.
4. For the image that you want to restore, make a note of the tag, `<tag>` .
5. Run the following command to restore the image to your repository, where `<dns>/<namespace>/<repo>` is the name of the image that you want to restore and `<tag>` is the tag.

```
$ ibmcloud cr image-restore <dns>/<namespace>/<repo>:<tag>
```

In your live repository, you can pull the image by tag.



Tip: If you get an error when you're restoring an image that says that the tagged image exists, see [Why do I get an error when I'm restoring an image?](#) for assistance.



Tip: If you run the `ibmcloud cr trash-list` command, the digest and any other tags show in the output, but the tag is no longer displayed.

Deleting a private repository and any associated images

You can delete private repositories that are no longer required, and any associated images, by using the IBM Cloud console.



Important: When you delete a repository, all images in that repository are deleted. This action can't be undone.



Tip: Before you begin, you must back up any images that you want to keep.

To delete a private repository by using the IBM Cloud console, complete the following steps:

1. Log in to the IBM Cloud console <https://cloud.ibm.com/login> with your IBMid.
2. If you have multiple IBM Cloud accounts, from the account menu, select the account and region that you want to use.
3. Click the **Navigation menu** icon, then click **Container Registry**.
4. Click **Repositories**. A list of your private repositories is displayed.
5. In the row that contains the private repository that you want to delete, select the checkbox.



Important: Ensure that the correct repository is selected because this action can't be undone.

6. Click **Delete Repository**.

Using Helm charts

You can securely store and share Helm charts with other users in IBM Cloud® Container Registry.

OCI support for Helm charts

The [Open Container Initiative \(OCI\)](#) released [Open Container Initiative Distribution Specification v1.0.0](#) in September 2021. This specification supports other artifact types in addition to container images. One of the [artifact types that is supported](#) is the [Helm chart](#).

[Helm v3.8.0](#) provides support to store and work with charts in OCI registries, as an alternative to [Helm chart repositories](#). For more information, see the [Helm Registries documentation](#).

Adding Helm charts to your namespace

You can securely store and share Helm charts with other users by adding charts to your `namespace` in IBM Cloud Container Registry.

Every Helm chart that you want to add to your namespace must exist on your local computer first. You can either download (pull) a chart from another repository to your local computer, or build your own chart by using the [helm create](#) command. To add a chart to your namespace, you must upload (push) the local chart to your namespace in IBM Cloud Container Registry.

Important: Do not put personal information in your charts (for example, in namespace names or description fields) or in any chart or chart configuration data (for example, chart names or chart labels).

Pulling charts from another registry or Helm repository

You can pull (download) a chart from any private or public `registry` source or Helm repository, and then tag it for later use in IBM Cloud Container Registry.

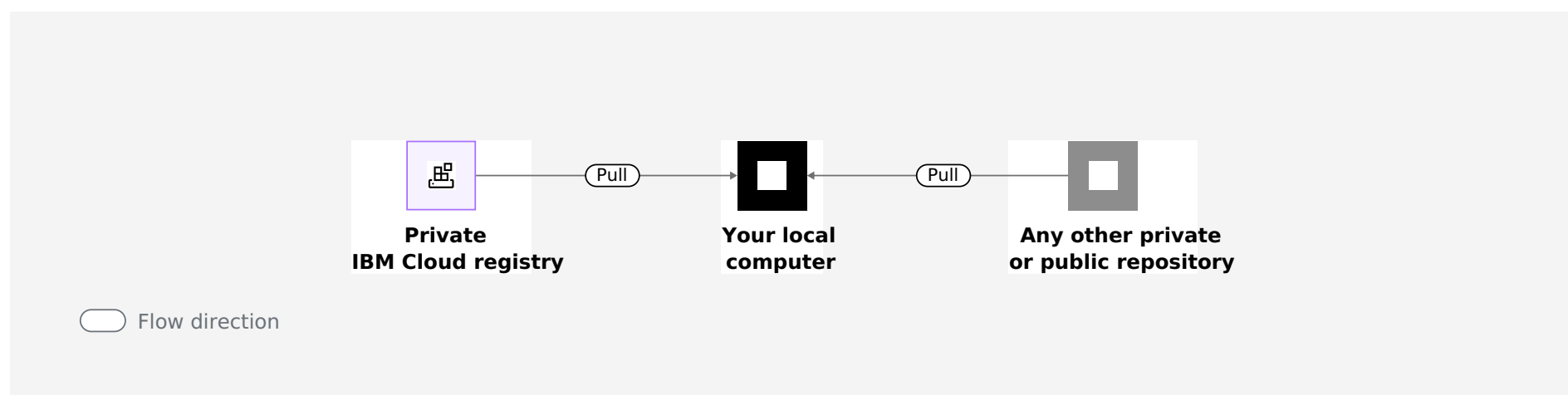


Figure 1. Pulling charts from another registry

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- Install the latest release of [Helm CLI](#) to work with charts.

1. Download the Helm chart to your local computer.

- Download the Helm chart from the OCI registry:

```
$ helm pull oci://<registry>/<my_namespace>/<chart_name> --version <chart_version>
```

Example, where `<registry>` is `localhost:5000`, `my_namespace` is `helm-charts`, `chart_name` is `mychart`, and `<chart_version>` is `0.1.0`:

```
$ helm pull oci://localhost:5000/helm-charts/mychart --version 0.1.0
```

- Download the Helm chart from a Helm repository:

```
$ helm pull <chart URL | repo/chartname> --version <chart_version>
```

Example, where `<repo/chartname>` is `ibm-charts/ibm-istio` and `<chart_version>` is `1.2.2`.

Tip: You can add the repo alias by using the [helm repo add](#) command.


```
$ helm pull ibm-charts/ibm-istio --version 1.2.2
```

✓ **Tip:** If you get an `unauthorized: authentication required` or a `denied: requested access to the resource is denied` message, run the `ibmcloud cr login` command.

After you pull a chart for your [namespace](#), you can upload (push) the chart from your local computer to your namespace.

Pushing Helm charts to your namespace

You can push (upload) a chart to your namespace in IBM Cloud Container Registry to store and share your chart with other users.

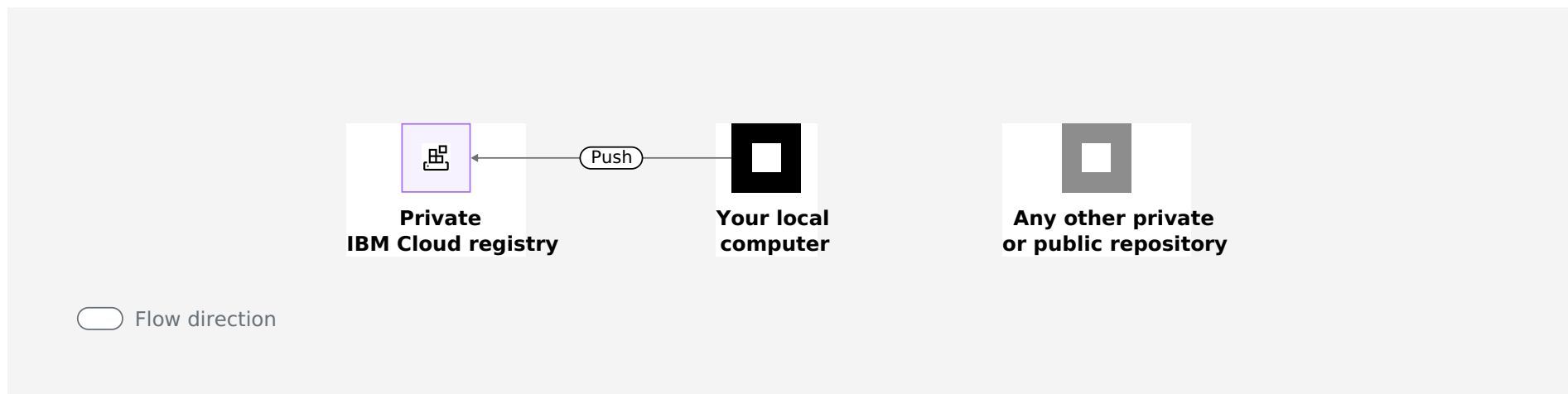


Figure 2. Pushing charts to your namespace

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- Install the latest release of [Helm CLI](#) to work with charts.
- [Pull](#) or [create](#) a chart on your local computer. If you create a chart, you must save the chart as an archive by using the [helm package](#) command.

To upload (push) a chart, complete the following steps:

1. Log in to the CLI by running the `ibmcloud cr login` command.

```
$ ibmcloud cr login
```

✓ **Tip:** You must log in if you pull a chart from your private IBM Cloud Container Registry.

2. To view all namespaces that are available in your account, run the `ibmcloud cr namespace-list` command.
3. Upload the chart to your namespace.

```
$ helm push <my_chart_package> oci://<region>.icr.io/<my_namespace>
```

Example, where `<my_chart_package>` is `mychart-0.1.0.tgz`, `region` is `uk`, and `<my_namespace>` is `helm-charts`:

```
$ helm push mychart-0.1.0.tgz oci://uk.icr.io/helm-charts
```

✓ **Tip:** If you get an `unauthorized: authentication required` or a `denied: requested access to the resource is denied` message, run the `ibmcloud cr login` command.

After you push your chart to IBM Cloud Container Registry, you can [install the Helm chart to the cluster in IBM Cloud Kubernetes Service](#).

Copying charts between registries

You can pull a chart from a registry in one region and push it to a registry in another region so that you can share the chart with users in both regions.

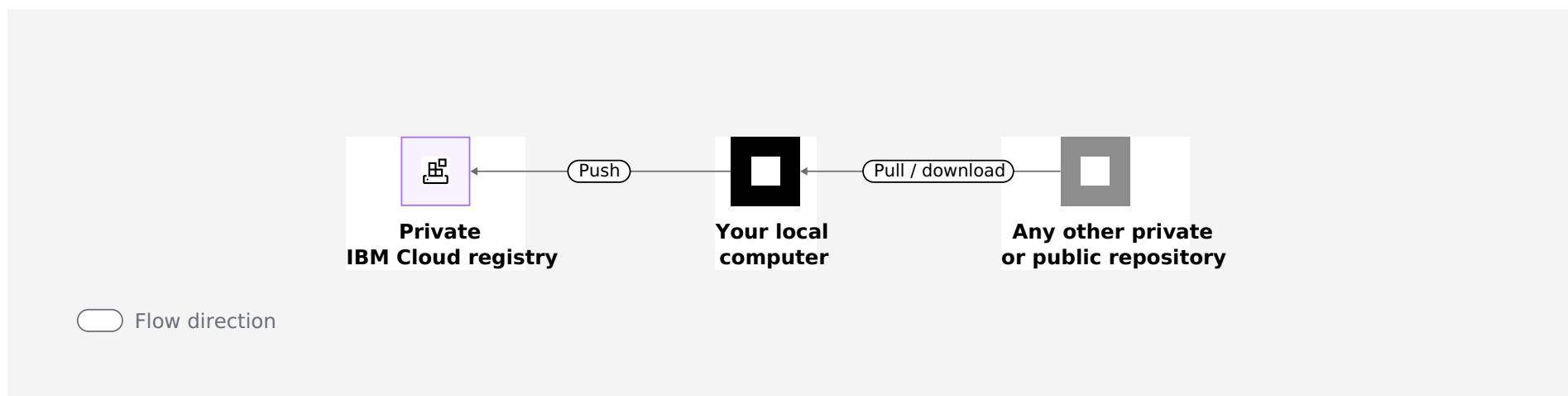


Figure 3. Copying charts between registries

Before you begin, complete the following tasks.

- [Install the CLI](#) to work with your namespace.
- [Set up your own namespace in IBM Cloud Container Registry](#).
- Install the latest release of [Helm CLI](#) to work with charts.

To copy a chart between two registries, complete the following steps:

1. [Pull a chart from a registry](#).
2. [Push the chart to another registry](#). Make sure that you use the correct domain name for the new region that you are targeting.

After you copy your chart, you can [install the Helm chart to the cluster in IBM Cloud Kubernetes Service](#).

Installing a Helm chart to the cluster

You can install a Helm chart to the cluster in IBM Cloud Kubernetes Service directly from the registry. Follow the instructions in the Helm chart `README`, and use the full registry reference to the chart and chart version for installation.

```
$ helm install <release_name> oci://<region>.icr.io/<my_namespace>/<chart_name> --version <chart_version>
```

Example, where `<release_name>` is `myrelease`, `region` is `uk`, `<my_namespace>` is `helm-charts`, `<chart_name>` is `mychart`, and `<chart_version>` is `0.1.0`:

```
$ helm install myrelease oci://uk.icr.io/helm-charts/mychart --version 0.1.0
```

Deleting charts from your private repository

You can delete unwanted charts from your private IBM Cloud [repository](#) by using either the IBM Cloud console or the CLI.

If you want to delete a private repository and its associated charts, see [Deleting a private repository and any associated charts](#).

Important: Deleting a chart that is being used by an existing deployment might cause a Helm upgrade, rollback, or delete to fail.

Tip: If you want to restore a deleted chart, you can list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore a selected chart by running the [ibmcloud cr image-restore](#) command. You can use these commands because Helm charts are a supported artifact type in OCI.

Tip: If `tags` exist for the same chart digest within a repository, the [ibmcloud cr image-rm](#) command removes the underlying chart and all its tags. If the same chart exists in a different repository or namespace, that copy of the chart is not removed.

Tip: A tag must always match the chart's semantic version, which means that a `latest` tag isn't used.

Deleting charts from your private repository in the CLI

You can delete unwanted charts and all their tags from your private IBM Cloud repository by using the CLI.


Important: Deleting a chart that is being used by an existing deployment might cause a Helm upgrade, rollback, or delete to fail.

Tip: If you want to restore a deleted chart, you can list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore a selected chart by running the [ibmcloud cr image-restore](#) command.

To delete a chart by using the CLI, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. To delete a chart, run the following command, where `CHART` is the name of the chart that you want to remove, in the format `repository@digest` or `repository:tag`. Unlike images, a tag must be specified because the `latest` tag doesn't exist because a tag must always match the chart's semantic version. You can delete multiple charts by listing each private IBM Cloud registry path in the command with a space between each path.

```
$ ibmcloud cr image-rm CHART
```


 **Tip:** To find the names of your charts, run `ibmcloud cr image-list`. The registry stores different artifact types that include Helm charts and container images. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`. To identify your chart by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (`@`) symbol to create the image name in the format `repository@digest`.


3. Verify that the chart was deleted by running the following command, and check that the chart does not show in the list.

```
$ ibmcloud cr image-list
```

Deleting charts from your private repository in the console

You can delete unwanted charts and all their tags from your private IBM Cloud repository by using the IBM Cloud console.

 **Important:** Deleting a chart that is being used by an existing deployment might cause a Helm upgrade, rollback, or delete to fail.

 **Tip:** If you want to restore a deleted chart, you can list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore a selected chart by running the [ibmcloud cr image-restore](#) command.

To delete a chart by using the IBM Cloud console, complete the following steps.

1. Log in to the IBM Cloud console <https://cloud.ibm.com/login> with your IBMid.
2. If you have multiple IBM Cloud accounts, select the account and region that you want to use from the account menu.
3. Click the **Navigation menu** icon, then click **Container Registry**.
4. Click **Images**. A list of your charts (and images if they exist) is displayed. The registry stores different artifact types that include Helm charts and container images.
5. In the row that contains the chart that you want to delete, select the checkbox.
6. Click **Delete Image**.

Listing charts in the trash

You can list deleted charts that are in the trash and see when they expire.

To find out which charts are in the trash, you can use the [ibmcloud cr trash-list](#) command. Charts are stored in the trash for 30 days.

To list the charts in the trash, complete the following steps.

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. List the charts in the trash by running the following command.

```
$ ibmcloud cr trash-list
```

3. List only the charts in the trash for the namespace that you are interested in by running the following command, where `<namespace>` is your namespace.

```
$ ibmcloud cr trash-list --restrict <namespace>
```

Restoring charts

You can restore charts from the trash. Deleted charts are stored in the trash for 30 days.

You can restore a chart from the trash by running the [ibmcloud cr image-restore](#) command. To find out which charts are in the trash, run the

`ibmcloud cr trash-list` command.

You can restore charts by running the `ibmcloud cr image-restore` command. You can use the following options:

- `<repo>@<digest>`, which restores the digest and all its tags in the repository that aren't already in the live repository, see [Restoring charts by digest](#).
- `<repo>:<tag>`, which restores the tag, see [Restoring charts by tag](#).

Restoring charts by digest

When you restore a chart by digest, the digest is moved from the trash into your live repository, and all the tags for that digest in the repository are restored.

To restore a chart by digest from the trash, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. List the charts in the trash by running the following command.

```
$ ibmcloud cr trash-list
```

A table is displayed that shows the items in the trash. The table shows the digest, the days until expiry, and the tags for that digest.

3. Note the digest for the chart that you want to restore.
4. Run the following command to restore the chart to your repository. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, and `<digest>` is the digest of the chart that you want to restore.

```
$ ibmcloud cr image-restore <dns>/<namespace>/<repo>@<digest>
```

✓ **Tip:** If some tags aren't restored, see [Why aren't all the tags restored when I restore by digest](#).

✓ **Tip:** In your live repository, you can pull the chart by digest. If you run the `ibmcloud cr image-digests` command, the chart shows in the output.

Restoring charts by tag

When you restore a chart by tag, only that specific tag is moved out of the trash into your live repository.

To restore a chart by tag from the trash, complete the following steps.

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. List the charts in the trash by running the following command.

```
$ ibmcloud cr trash-list
```

A table is displayed that shows the items in the trash. The table shows the digest, the days until expiry, and the tags for that digest.

3. For the chart that you want to restore, make a note of the digest up to, but not including, the at sign (`@`). This part of the digest is `<dns>/<namespace>/<repo>`, where `<dns>` is the domain name, `<namespace>` is the namespace, and `<repo>` is the repository.
4. For the chart that you want to restore, make a note of the tag, `<tag>`.
5. Run the following command to restore the chart to your repository, where `<dns>/<namespace>/<repo>` is the name of the chart that you want to restore and `<tag>` is the tag.

```
$ ibmcloud cr image-restore <dns>/<namespace>/<repo>:<tag>
```

In your live repository, you can pull the chart by tag.

✓ **Tip:** If you run the `ibmcloud cr trash-list` command, the digest and any other tags show in the output, but the tag is no longer displayed.

Deleting a private repository and any associated charts

You can delete private repositories that are no longer required, and any associated charts, by using the IBM Cloud console.



Important: When you delete a repository, all charts in that repository are deleted. This action can't be undone.



Tip: Before you begin, you must back up any charts that you want to keep.

To delete a private repository by using the IBM Cloud console, complete the following steps.

1. Log in to the IBM Cloud console <https://cloud.ibm.com/login> with your IBMid.
2. If you have multiple IBM Cloud accounts, select the account and region that you want to use from the account menu.
3. Click the **Navigation menu** icon, then click **Container Registry**.
4. Click **Repositories**. A list of your private repositories is displayed.
5. In the row that contains the private repository that you want to delete, select the checkbox.



Important: Ensure that the correct repository is selected because this action can't be undone.

6. Click **Delete Repository**.

Cleaning up your namespaces

You can clean up your *namespace* by choosing to retain only the most recent images in each repository in that namespace in IBM Cloud® Container Registry.

You can also choose whether to delete or retain your untagged images.

You can detect and delete old images from all the repositories in a namespace by running a one-off command, `ibmcloud cr retention-run`, or by setting a scheduled policy by running the `ibmcloud cr retention-policy-set` command. You can choose the number of images that you want to keep in each repository in a namespace, all other images are automatically deleted. Both options keep the most recent images. The age of the image is determined by when the image was created, not when it was pushed to the *registry*. The number of images that are kept is the same for each repository in that namespace.

When you run the `ibmcloud cr retention-run` and `ibmcloud cr retention-policy-set` commands, a list of images to delete is shown, and you must confirm that you want to delete those images. After you run the `ibmcloud cr retention-policy-set` command the first time, the policy runs automatically and deletes any images that meet the criteria that are specified in the policy. Deleted images are stored in the trash for 30 days.

If you want to check what's in the trash, run the `ibmcloud cr trash-list` command. You can restore images from the trash by running the `ibmcloud cr image-restore` command.

If you want to check your policies, you can run the `ibmcloud cr retention-policy-list` command.

If you want to cancel a policy, [update the retention policy so that it keeps all your images](#).

You can also clean up your namespace by [deleting your untagged images](#).

Planning retention

The `ibmcloud cr retention-run` and `ibmcloud cr retention-policy-set` commands operate on a per-namespace basis. If you have multiple namespaces in your pipeline, you can apply different retention criteria for each namespace to best suit your requirements.

Consider a typical delivery pipeline with development, staging, and production environments. As code is delivered, continuous integration and continuous deployment pushes images into the registry and then deploys them straight to your development environment. After testing, some builds from development are promoted to staging, and then potentially onto production. In this scenario, the rate of image change is fastest in development and slowest in production. If all your environments pull images from the same namespace, it can be difficult to choose an appropriate number of images to retain due to this difference in velocity.

A good approach is to deliver all images into a development namespace, for example, `project-development`, and then to use the `ibmcloud cr image-tag` command to tag the image into a different namespace when it is promoted to a higher stage of the pipeline. In the previous example, you can have three namespaces: development, `project-development`, staging, `project-staging`, and production, `project-production`. When you are promoting from development to staging, images are tagged from the `project-development` namespace into the `project-staging` namespace, and the images from the `project-staging` namespace are used for deployment. Similarly, when you are promoting from staging to production, images are tagged from the `project-staging` namespace into the `project-production` namespace, and the `project-production` namespace images are used in the production deployment.

You gain the following advantages by using this technique:


- You can choose different retention settings for development, staging, and production namespaces.
- You minimize the chances of accidentally removing an image that might be in use in your staging or production environments, when compared with using a single namespace for all images.
- You can use different [IAM](#) policies. For example, you can have more restrictive access to production images.
- You can sign production images, but leave development and staging images unsigned.


Clean up your namespaces to keep a set number of images

Use the `ibmcloud cr retention-run` command to clean up a namespace by retaining a specified number of images for each repository within a namespace in IBM Cloud Container Registry. All other images in the namespace are deleted.

You can choose whether to exclude untagged images from the clean-up process.

The `ibmcloud cr retention-run` command lists the images to delete and gives you the option to cancel before deletion.

 **Tip:** Where an image, within a repository, is referenced by multiple tags, that image is counted only once. Newest images are retained. Age is determined by when the image was created, not when it was pushed to the registry.

 **Tip:** If you want to restore a deleted image, you can list the contents of the trash by running the `ibmcloud cr trash-list` command and restore a selected image by running the `ibmcloud cr image-restore` command.

To reduce the number of images in each repository within your namespace by using the CLI, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. Choose the registry in which you want to clean up your images by running the following command and selecting the appropriate region:

```
$ ibmcloud cr region-set
```

3. To retain the most recent images and delete the others, run one the following commands:
 - If you want to clean up both tagged and untagged images, run the following command:


```
$ ibmcloud cr retention-run --images <image_count> <namespace>
```

Where `<image_count>` is the number of images that you want to retain for each repository within your namespace, `<namespace>`.

- If you want to clean up tagged images only and retain all untagged images, run the following command:


```
$ ibmcloud cr retention-run --retain-untagged --images <image_count> <namespace>
```

Where `<image_count>` is the number of images that you want to retain for each repository within your namespace, `<namespace>`.

 **Tip:** If an image that you're expecting to see doesn't show in the list that is produced, see [Why doesn't the retention command show all the images?](#) for assistance.

4. Verify that the images were deleted by running the following command, and check that the images do not show in the list.

```
$ ibmcloud cr image-list
```


 **Tip:** If listing images times out, see [Why is it timing out when I list images?](#) for assistance.


Set a retention policy for your namespaces

You can set a retention policy for your namespaces to retain only images that meet your criteria. The retention policy runs automatically to clean up your namespaces.

You can choose whether to exclude untagged images from the clean-up process.

You can use the `ibmcloud cr retention-policy-set` command to set a policy that retains a specified number of images for each repository within a namespace in IBM Cloud Container Registry. All other images in the namespace are deleted and moved to the trash. When you set a policy it runs immediately, then it runs daily. You can set only one policy in each namespace.

 **Tip:** Where an image, within a repository, is referenced by multiple tags, that image is counted only once. Newest images are retained. Age is determined by when the image was created, not when it was pushed to the registry.

 **Tip:** If you delete an image in error, you can restore the image by using the `ibmcloud cr trash-list` and `ibmcloud cr image-restore` commands.

To set a policy and immediately move your deleted images to the trash, complete the following steps:

1. Log in to IBM Cloud by running the `ibmcloud login` command.
2. Choose the registry in which you want to clean up your images by running the following command and selecting the appropriate region:

```
$ ibmcloud cr region-set
```

3. To set a policy that retains the most recent images and deletes the others, run one of the following commands:
 - If you want to clean up both tagged and untagged images, run the following command:

```
$ ibmcloud cr retention-policy-set --images <image_count> <namespace>
```

Where `<image_count>` is the number of images that you want to retain for each repository within your namespace, `<namespace>`.

A list of images to delete is displayed.

- If you want to clean up tagged images only and retain all untagged images, run the following command:

```
$ ibmcloud cr retention-policy-set --retain-untagged --images <image_count> <namespace>
```

Where `<image_count>` is the number of images that you want to retain for each repository within your namespace, `<namespace>`.

A list of images to delete is displayed.

4. Review the list of images. To run the policy and delete the images, confirm that you want to set the policy.



Tip: If you don't want to delete those images, choose `No`. The policy is not set and the images are not deleted.

5. Verify that the images were deleted by running the following command, and check that the images show in the list.

```
$ ibmcloud cr trash-list
```

6. Verify that the policy is set by running the `ibmcloud cr retention-policy-list` command, and check that the policy that you set for the namespace retains the required number of images. If you set the policy to retain all untagged images, ensure that the **Retain all untagged** column has the value `true`.

```
$ ibmcloud cr retention-policy-list
```

Update a retention policy to keep all your images

All namespaces have a default policy that keeps all images. You can return a policy to the default state.

You can use the `ibmcloud cr retention-policy-set` command to set the policy back to the default state by running the following command, where `<namespace>` is your namespace:

```
$ ibmcloud cr retention-policy-set --images All <namespace>
```

Clean up your namespaces by deleting untagged images

You can clean up your namespace and reduce your bills by deleting your [untagged](#) images in the namespace and optionally output the results in JSON format.

If you want to delete your untagged images and output the results in JSON format, run the following `ibmcloud cr image-prune-untagged` command, where `<namespace>` is your namespace:

```
$ ibmcloud cr image-prune-untagged [--force | -f [--output json]] --restrict <namespace>
```

Managing quota limits for storage and pull traffic

You can limit the amount of storage and pull traffic that can be used in your IBM Cloud account by setting and managing custom quota limits in IBM Cloud® Container Registry.

Setting quota limits for storing and pulling images

You can limit the amount of storage and pull traffic to your private images by setting your own quota limits.

When you upgrade to the IBM Cloud Container Registry standard plan, you benefit from unlimited amount of storage and pull traffic to your private images. To avoid exceeding your preferred payment level, you can set individual quotas for the amount of storage and pull traffic. Quota limits are applied to all *namespaces* that you set up in IBM Cloud Container Registry. If you're using the free service plan, you can also set custom quotas within your free amount of storage and pull traffic.

To set a quota, complete the following steps.

1. Log in to IBM Cloud.

```
$ ibmcloud login
```

2. Review your current quota limits for storage and pull traffic.

```
$ ibmcloud cr quota
```


Your output looks similar to the following example.

```
Getting quotas and usage for the current month, for account '<account_owner> Account'...

QUOTA      LIMIT    USED
Pull traffic 5.1 GB   0 B
Storage     512 MB   511 MB

OK
```

3. Change the quota limit for storage and pull traffic. To change the pull traffic usage, specify the **traffic** option, and replace `<traffic_quota>` with the value in megabytes that you want to set for the pull traffic quota. If you want to change the amount of storage in your account, specify the **storage** option, and replace `<storage_quota>` with the value in megabytes that you want to set.

 **Tip:** If you are on the free plan, you cannot set your quota to an amount that exceeds the free tier. The free tier allowance for storage is 512 MB and traffic is 5120 MB.

```
$ ibmcloud cr quota-set --traffic <traffic_quota> --storage <storage_quota>
```

Example to set your quota limit for storage to 600 megabytes, and the pull traffic to 7000 megabytes:

```
$ ibmcloud cr quota-set --storage 600 --traffic 7000
```

Reviewing quota limits and usage

You can review your quota limits and check your current storage and pull traffic usage for your account.

1. Log in to IBM Cloud.

```
$ ibmcloud login
```

2. Review your current quota limits for storage and pull traffic.

```
$ ibmcloud cr quota
```

Your output looks similar to the following example.

```
Getting quotas and usage for the current month, for account '<account_owner> Account'...

QUOTA      LIMIT    USED
Pull traffic 5.1 GB   0 B
Storage     512 MB   511 MB
```

OK

Staying within quota limits

If you exceed the quota limits that are set for your IBM Cloud account, you can free up storage and change your service plan or quota limits so that you can continue pushing and pulling images to and from your namespace.



Note: From 1 February 2022, both [tagged](#) and [untagged](#) images are charged for.

To free up image storage in your IBM Cloud account, complete the following steps.



Note: Depending on the size of the image, it might take a while for the image to be removed and for the storage to be available.

1. Find the names of the images that you want to remove.

- To list only tagged images, run the [ibmcloud cr image-list](#) command. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag` . If the listing images command times out, see [Why is it timing out when I list images?](#) for assistance.

```
$ ibmcloud cr image-list
```

- To list both tagged and untagged images, you must list the images by [digest](#) in all your namespaces of your IBM Cloud account. To list the images by digest, run the [ibmcloud cr image-digests](#) command. Combine the content of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (`@`) symbol to create the image name in the format `repository@digest` .

```
$ ibmcloud cr image-digests
```

2. You can remove images individually, collectively, or by using retention policies.

- To remove images individually from your namespace, use the [ibmcloud cr image-rm](#) command. Replace `<image_name>` with the name of the image that you want to remove. The name must be in the format `repository@digest` or `repository:tag` . If a tag is not specified in the image name, the image that is tagged `latest` is deleted by default. Deleted images are stored in the trash for 30 days. Images that are in the trash don't count toward your quota.



Note: You can remove both tagged and untagged images by using the format `repository@digest` . You can remove only tagged images by using the format `repository:tag` .

```
$ ibmcloud cr image-rm <image_name>
```



Tip: Where multiple tags exist for the same image digest within a repository, the [ibmcloud cr image-rm](#) command removes the underlying image and all its tags. If the same image exists in a different repository or namespace, that copy of the image is not removed. If you want to remove a tag from an image and leave the underlying image and any other tags in place, see [Removing tags from images in your private repository](#) command.

- To remove untagged images collectively from your namespace, use the [ibmcloud cr image-prune-untagged](#) command, see [Clean up your namespaces by deleting untagged images](#).
- To use retention policies, see [Cleaning up your namespaces](#).

3. Review your storage quota usage.

```
$ ibmcloud cr quota
```

4. To reduce your pull traffic usage, you must wait until the next billing period.

To continue pulling images from your namespaces, choose between the following options.

- Wait until the next billing cycle starts.
- If you have a free plan, [upgrade to the standard service plan](#).
- If you already have a standard plan, [set new quota limits for the pull traffic](#).


Managing user access


Accessing Container Registry

To access your IBM Cloud® Container Registry namespaces so that you can push and pull images, use IBM Cloud® Identity and Access Management (IAM).

All accounts require IAM access policies. To set up and manage IAM access policies, see [Defining IAM access policies](#).

Access to IBM Cloud Container Registry is either [automated](#), which typically uses [API keys](#), or [interactive](#), which typically uses bearer tokens.

 **Tip:** If you have an IAM access policy, but you are getting `Access denied` errors, see [Why am I getting Access denied errors?](#) for assistance.

 **Tip:** If you want to use your container images in Kubernetes deployments, see [Using an image pull secret to access images in other IBM Cloud accounts or external private registries from nondefault Kubernetes namespaces](#).

Accessing your namespaces in automation

You can use service ID API keys to automate the pushing and pulling of container images to and from your namespaces.

API keys are linked to user IDs or service IDs in your account and you can use them across IBM Cloud®. You can use an API key in the CLI or as part of automation to authenticate as your user or service identity. A [user API key](#) is associated with a user and their access policies. A [service ID API key](#) has its own access policies. You can have several service IDs with different fine grained policies so that your automation is granted specific and limited capabilities.

When you create an IBM Cloud Kubernetes Service or Red Hat® OpenShift® on IBM Cloud® cluster, the cluster is created with an IBM Cloud IAM service ID that is given an IAM Reader service access policy to IBM Cloud Container Registry. The service ID credentials are authenticated in a nonexpiring service ID API key that is stored in image pull secrets in your cluster. The image pull secrets are added to the `default` Kubernetes namespace and to the list of secrets in the `default` service account for this Kubernetes namespace. If you require more service ID API keys or the service ID API key is missing, you can [create a service ID API key manually](#).

You can use service ID API keys in the following places:

- IBM Cloud Kubernetes Service or Red Hat OpenShift on IBM Cloud clusters. When you create IBM Cloud Kubernetes Service and Red Hat OpenShift on IBM Cloud clusters, one service ID is automatically created for each cluster. If you want more than one service ID, you can create them manually.
- Kubernetes and Red Hat® OpenShift® clusters that aren't on IBM Cloud. You must create your own service ID, API key, and pull secret.
- Docker CLI and other clients. You must create your own service ID and API key.

Creating a service ID API key manually

Create a service ID API key that you can use to log in to the registry.

To create a service ID API key, complete the following steps:

1. Create a service ID, see [ibmcloud iam service-id-create](#).
2. Assign service policies to the service ID to control the level of access that is allowed when the service ID is used to authenticate with IBM Cloud Container Registry, see [Managing access to resources](#).
3. Create a service ID API key, see [Managing service ID API keys](#) and [ibmcloud iam service-api-key-create](#).

Creating a user API key manually

Create a user API key that you can use to log in to the registry.

If you create a user API key, the user's access policies are used.

To create a user API key, see [Managing user API keys](#) and [ibmcloud iam api-key-create](#).

Using client software to authenticate in automation

Use an API key to log in to the registry by using common clients.

Clients require an API key and a domain, replace `<apikey>` with your API key and `<registry_domain>` with the domain of the registry where your namespaces are set up.

Region	<registry_domain>
global	icr.io

ap-north	jp.icr.io
ap-south	au.icr.io
br-sao	br.icr.io
ca-tor	ca.icr.io
eu-central	de.icr.io
jp-osa	jp2.icr.io
uk-south	uk.icr.io
us-south	us.icr.io

Table 1. Registry domains

For more information about how to use IBM Cloud Container Registry in a Continuous Delivery pipeline, see [Using a private image registry](#).

Examples of how to authenticate automatically with the registry are provided for the following clients:

- [Buildah](#)
- [Docker](#)
- [Podman](#)
- [Skopeo](#)

Using Buildah to authenticate with the registry

You can use Buildah to authenticate with the registry so that you can push and pull images to and from the registry.

Use the API key and [domain](#) to log in to the registry by running the following Buildah command, replace `<apikey>` with the API key and `<registry_domain>` with the domain:

```
$ buildah login -u iamapikey -p <apikey> <registry_domain>
```

Using Docker to authenticate with the registry

You can use Docker to authenticate with the registry so that you can push and pull images to and from the registry.

Use the API key and [domain](#) to log in to the registry by running the following Docker command, replace `<apikey>` with the API key and `<registry_domain>` with the domain:

```
$ docker login -u iamapikey -p <apikey> <registry_domain>
```

Using Podman to authenticate with the registry

You can use Podman to authenticate with the registry so that you can push and pull images to and from the registry.

Use the API key and [domain](#) to log in to the registry by running the following Podman command, replace `<apikey>` with the API key and `<registry_domain>` with the domain:

```
$ podman login -u iamapikey -p <apikey> <registry_domain>
```

Using Skopeo to authenticate with the registry

You can use Skopeo to authenticate with the registry so that you can push and pull images to and from the registry.

For example, you can use the following Skopeo command to pull an image from Docker Hub and push it to your namespace. Replace `<registry_domain>` with the name of your [domain](#), `<namespace>` with your namespace, and `<apikey>` with your API key:

```
$ skopeo --insecure-policy --override-os linux copy docker://busybox:latest  
docker://<registry_domain>/<namespace>/busybox:latest --dest-creds iamapikey:<apikey>
```

Accessing your namespaces interactively

You can use bearer tokens and refresh tokens to push and pull images to and from your namespaces interactively.

Examples of how to access your namespaces interactively are provided for the following clients:

- [Buildah](#)
- [Docker](#)
- [Podman](#)
- [Skopeo](#)

Using Buildah to access your namespace

Log in to the registry by using the Buildah CLI.

You can use the Buildah CLI to log in to the registry by using a bearer token, replace `<registry_domain>` with the [domain](#):

```
$ ibmcloud iam oauth-tokens | sed -ne '/IAM token/s/.*/p' | buildah login -u iambearer --password-stdin <registry_domain>
```

Using Docker to access your namespace

Log in to the registry by using the Docker CLI.

You can use the Docker CLI to log in to the registry by using a refresh token in the [IBM Cloud CLI](#):

```
$ ibmcloud cr login --client docker
```

You can use the Docker CLI to log in to the registry by using a bearer token:

1. Generate a bearer token by using [ibmcloud iam oauth-tokens](#).
2. Log in to the registry by using the `docker login` command. Replace `<bearer_token>` with your bearer token and `<registry_domain>` with the [domain](#):

```
$ docker login -u iambearer --password <bearer_token> <registry_domain>
```

Using Podman to access your namespace

Log in to the registry and pull an image by using the CLI, where `<image_name>` is the name of the image.

```
$ ibmcloud cr login --client podman
```

```
$ podman pull <image_name>
```

Using Skopeo to access your namespace

Log in to the registry by using the Skopeo CLI.

You can use the Skopeo CLI to log in to the registry by using a bearer token, replace `<registry_domain>` with the [domain](#):

```
$ ibmcloud iam oauth-tokens | sed -ne '/IAM token/s/.*/p' | skopeo login -u iambearer --password-stdin <registry_domain>
```

Accessing your namespaces programmatically

Use your own code to access to your namespaces in IBM Cloud Container Registry.

Most users can use the [ibmcloud cr login](#) command to simplify `docker login`, but if you are implementing automation or you are using a different client, you might want to authenticate manually. You must present a username and password. In IBM Cloud Container Registry, the username indicates the type of secret that is presented in the password.

The following usernames are valid:

- **iambearer** The password contains an IAM *access token*. This type of authentication is short lived, but can be derived from all types of IAM identity. For example, from [ibmcloud iam oauth-tokens](#).
- **iamrefresh** The password contains an IAM refresh token that is used internally by the registry to generate an IAM access token. This type of authentication is longer lived. This authentication type is used by the `ibmcloud cr login` command.
- **iamapikey** The password is an IAM API key that is used internally by the registry to generate an IAM access token. This type of authentication is the preferred type for automation. You can use a user API key or a service ID API key. For more information, see [Accessing your namespaces in automation](#).

Managing IAM access for Container Registry

Access to IBM Cloud® Container Registry for users in your account is controlled by IBM Cloud® Identity and Access Management (IAM).

Every user that accesses the IBM Cloud Container Registry service in your account must be assigned an IAM *access policy* with an IAM role. A user can also be a member of an [access group](#) with assigned IAM access policies that grant an IAM role. Review the following roles, actions, and more to help determine the best way to assign access to Container Registry.

For more information about IAM, see [How IBM Cloud Identity and Access Management works](#).

 **Tip:** Try out the tutorial [Granting access to Container Registry resources tutorial](#).

Access policies

The IAM access policy that you assign to users in your account determines the actions that a user can perform within the context of the service or specific instance that you select. The allowable actions are customized and defined by Container Registry as operations that are allowed to be performed on the service. Each action is mapped to an [IAM platform or service role](#) that you can assign to a user.

Policies enable access to be granted at different levels. Some options include the following access levels:

- Access to the service in your account
- Access to a specific resource within the service
- Access to all IAM-enabled services in your account
- Access to resources within a resource group

If you want to restrict user access to one or more *namespaces* for an ID that you are using for automation, use an IAM service ID. For more information about service IDs, see [Creating and working with service IDs](#).

You can set permissions so that you can configure access to resources within a namespace at the *resource group* level. For more information, see [User permissions for working with namespaces](#).

For more information about enabling policies for Container Registry, see [Defining IAM access policies](#).

Assign roles

After you define the scope of the IAM access policy, you assign a role.

If a specific role and its actions don't fit the use case that you're looking to address, you can [create a custom role](#) and pick the actions to include.

Review the following tables that outline the actions that each role allows within the Container Registry service.

- [Platform management roles](#) enable users to perform tasks on service resources at the platform level, for example, assign user access to the service, create or delete instances, and bind instances to applications.
- [Service access roles](#) enable users access to Container Registry and the ability to call the Container Registry API.

For more information about the exact actions that are mapped to each role, see [IAM roles and actions for Container Registry](#).

For more information about assigning user roles in the UI, see [Managing access to resources](#).

Context-based restrictions

Container Registry also supports context-based restrictions. You can use context-based restrictions to define and enforce access restrictions for IBM Cloud resources based on the network location of access requests. These restrictions work with traditional IAM policies, which are based on identity, to provide an extra layer of protection.

For more information, see [What are context-based restrictions](#).

For an example of how to set up context-based restrictions, see the context-based restrictions tutorial [Leveraging context-based restrictions to secure your resources](#).

When you set up context-based restrictions, the restrictions apply to everything for the selected service in the account unless you select a subset of resources. To set up your own context-based restrictions for Container Registry, when you're creating a rule, in the **Select your resources** section, select **Container Registry**. Container Registry supports the following subset of resources: `resource type = namespace` and `resource id = YOUR_IMAGE_NAMESPACE`, where `YOUR_IMAGE_NAMESPACE` is the namespace of your image.

For example, if your image is in the format `uk.icr.io/<my_project>/<my_image>:latest`, where `<my_project>` is the name of your project and `<my_image>` is the name of the image, the attribute types are as shown in the following table.

Attribute type	Operator	Value
----------------	----------	-------

Region	string equals	London
Resource Type	string equals	namespace
Resource Name	string equals	<my_project>

Table 1. Example attribute types

The **Resource Name** value is a namespace, as shown by the `ibmcloud cr namespace-list` command.

Platform management roles

The following table details actions that are mapped to platform management roles. Platform management roles enable users to perform tasks on service resources at the platform level, for example assign user access for the service, and create or delete service IDs.

Platform management roles	Description of actions	Example actions
Viewer	Not supported	
Editor	Not supported	
Operator	Not supported	
Administrator	Configure access for other users. Apply pull secrets to clusters.	For more information about assigning user roles in the UI, see Managing access to resources . To create clusters in IBM Cloud Kubernetes Service that have pull secrets to access images in Container Registry, you must have the Administrator role. To use the <code>ibmcloud ks cluster pull-secret apply</code> command to configure the pull secrets for an existing cluster, you must have the Administrator role. For more information, see Preparing to create clusters .

Table 2. IAM user roles and actions

Service access roles

The following table details actions that are mapped to service access roles. Service access roles give users access to Container Registry as well as the ability to call the Container Registry API.

Service access role	Description of actions	Example actions
Reader	The Reader role can view information.	View, inspect, and pull images. View and analyze namespaces. View quotas. View vulnerability reports. View image signatures. View retention policies. View the contents of the trash. View the contents of the manifest for an image. List Vulnerability Advisor security exemption policies and types of security exemptions.

Writer	The Writer role can edit information.	Push, delete, and restore images. View quotas. Sign images. Set and run retention policies. Delete all untagged images in your Container Registry account.
Manager	The Manager role can perform all actions.	View, inspect, pull, push, delete, and restore images. View, add, analyze, and remove namespaces. Assign namespaces to resource groups. View and set quotas. View vulnerability reports. View and create image signatures. Review and change pricing plans. Enable IAM access policy enforcement. List, add, and remove Vulnerability Advisor security issue exemption policies. List types of security exemptions. Set and run retention policies. View the contents of the trash. Restore images. View the contents of the manifest for an image. Prevent or allow image pulls or pushes over public network connections for your account. Check whether the use of public connections is prevented for image pushes or pulls in your account. Delete all untagged images in your Container Registry account.

Table 3. IAM service access roles and actions

For the following Container Registry commands, you must have at least one of the specified roles as shown in the following tables. To create a policy that allows access to Container Registry, you must create a policy where the following criteria apply.

- The service name is `container-registry`.
- The service instance is empty.
- The region is the region that you want to grant access to, or is empty to give access to all regions.

Access roles for configuring Container Registry

To grant a user permission to configure Container Registry in your account, you must create a policy that grants one or more of the roles in the following table. When you create your policy, you must not specify a `resource type` or `resource`. Policies for configuring Container Registry must not be set at a resource group level.

For example, run the following `ibmcloud iam user-policy-create` command. Where `<user_email>` is the user's email address, `<region>` is the region, and `<roles>` is the role, or roles, that you want the user to have.

```
$ ibmcloud iam user-policy-create <user_email> --service-name container-registry --region <region> --roles <roles>
```

The following table details actions that are mapped to operations on the service and to the service access roles for configuring Container Registry.

Action	Operation on service	Role
<code>container-registry.auth.get</code>	ibmcloud cr private-only Check whether the use of public connections is prevented for image pushes or pulls in your account.	Manager

container-registry.auth.set	ibmcloud cr iam-policies-enable Enable IAM access policy enforcement. ibmcloud cr private-only Prevent or allow image pulls or pushes over public network connections for your account.	Manager
container-registry.exemption.list	ibmcloud cr exemption-list List your Vulnerability Advisor exemption policies for security issues. ibmcloud cr exemption-types List the types of security issues that you can exempt.	Reader, Manager
container-registry.exemption.manager	ibmcloud cr exemption-add Create a Vulnerability Advisor exemption policy for a security issue. ibmcloud cr exemption-rm Delete a Vulnerability Advisor exemption policy for a security issue.	Manager
container-registry.namespace.create	ibmcloud cr namespace-add Create a namespace. ibmcloud cr namespace-assign Assign a namespace to a resource group.	Manager
container-registry.namespace.delete	ibmcloud cr namespace-rm Remove a namespace.	Manager
container-registry.plan.get	ibmcloud cr plan Display your pricing plan.	Manager
container-registry.plan.set	ibmcloud cr plan-upgrade Upgrade to the standard plan.	Manager
container-registry.quota.get	ibmcloud cr quota Display your current quotas for traffic and storage, and usage information against those quotas.	Reader, Writer, Manager
container-registry.quota.set	ibmcloud cr quota-set Modify the specified quota.	Manager
container-registry.settings.get	ibmcloud cr platform-metrics Get registry service settings for the targeted account, such as whether platform metrics are enabled.	Reader, Writer, Manager
container-registry.settings.set	ibmcloud cr platform-metrics Update registry service settings for the targeted account, such as enabling platform metrics.	Manager

Table 4. Service actions and operations for configuring Container Registry

Access roles for using Container Registry

To grant a user permission to access Container Registry content in your account, you must create a policy that grants one or more of the roles in the following table. When you create your policy, you can restrict access to a specific namespace by specifying the resource type `namespace` and the namespace name as the resource. If you don't specify a `resource-type` and a `resource`, the policy grants access to all resources in the account. Alternatively, if your namespace is within a resource group, permission can be granted by using an IAM access policy on that resource group.

For example, use the following command to create a user policy. Where `<user_email>` is the user's email address, `<region>` is the region, `<roles>` is the role, or roles, that you want the user to have, and `<namespace_name>` is the name of the namespace.

```
$ ibmcloud iam user-policy-create <user_email> --service-name container-registry --region <region> --roles <roles> [--resource-type namespace --resource <namespace_name>]
```

The following table details actions that are mapped to operations on the service and to the service access roles for using Container Registry.

Action	Operation on service	Role	Status
--------	----------------------	------	--------

container-registry.image.delete	<p>docker trust revoke Delete the signature for a container image.</p> <p>ibmcloud cr image-prune-untagged Delete all untagged images in your Container Registry account.</p> <p>ibmcloud cr image-rm Delete one or more container images.</p> <p>ibmcloud cr image-untag Remove a tag, or tags, from each specified container image in Container Registry.</p> <p>ibmcloud cr retention-policy-set Set a policy to clean up your namespaces by retaining only container images that meet your criteria.</p> <p>ibmcloud cr retention-run Clean up your namespaces by retaining only container images that meet your criteria.</p>	Writer, Manager
container-registry.image.inspect	<p>ibmcloud cr image-inspect Display details about a specific container image.</p> <p>ibmcloud cr manifest-inspect View the contents of the manifest for an image.</p>	Reader, Manager
container-registry.image.list	<p>ibmcloud cr image-digests List all your container images, including untagged images.</p> <p>ibmcloud cr image-list List your tagged container images.</p> <p>ibmcloud cr image-prune-untagged Delete all untagged images in your Container Registry account.</p> <p>ibmcloud cr trash-list Display the container images that are in the trash.</p>	Reader, Manager
container-registry.image.pull	<p>docker pull Pull a container image.</p> <p>docker trust inspect Inspect the signature for a container image.</p> <p>ibmcloud cr image-tag Create a container image that refers to a source image.</p> <p>ibmcloud cr vulnerability-assessment View a vulnerability assessment report for your container image.</p>	Reader, Writer, Manager
container-registry.image.push	<p>docker push Push a container image.</p> <p>docker trust sign Sign a container image.</p> <p>ibmcloud cr image-restore Restore a deleted container image from the trash.</p> <p>ibmcloud cr image-tag Create a container image that refers to a source image.</p>	Writer, Manager
container-registry.namespace.list	<p>ibmcloud cr namespace-list List your namespaces.</p>	Reader, Manager

<code>container-registry.retention.analyze</code>	ibmcloud cr retention-policy-set Set a policy to clean up your namespaces by retaining only container images that meet your criteria. ibmcloud cr retention-run Clean up your namespaces by retaining only container images that meet your criteria.	Reader, Manager To run <code>ibmcloud cr retention-run</code> and <code>ibmcloud cr retention-policy-set</code> you must have Manager, or both Reader and Writer.
<code>container-registry.retention.get</code>	View the image retention policy for a namespace by using the API, see IBM Cloud Container Registry API .	Reader, Manager
<code>container-registry.retention.set</code>	ibmcloud cr retention-policy-set Set a policy to clean up your namespaces by retaining only container images that meet your criteria.	Writer, Manager
<code>container-registry.retention.list</code>	ibmcloud cr retention-policy-list List the image retention policies for your account.	Reader, Manager

Table 5. Service actions and operations for using Container Registry

Assigning access to Container Registry in the console

You can use one of the following options to assign access in the console:

- Access policies per user. You can manage access policies per user from the **Manage > Access (IAM) > Users** page in the console. For more information about the steps to assign IAM access, see [Managing access to resources](#).
- Access groups. Access groups are used to streamline access management by assigning access to a group once, then you can add or remove users as required from the group to control their access. You manage access groups and their access from the **Manage > Access (IAM) > Access groups** page in the console. For more information, see [Assigning access to a group in the console](#).

Assigning access to Container Registry in the CLI

For step-by-step instructions for assigning, removing, and reviewing access, see [Assigning access to resources by using the CLI](#). The following example shows a command for assigning the `Manager` role for Container Registry:

✓ **Tip:** Use `container-registry` for the service name.

```
$ ibmcloud iam user-policy-create <user@example.com> --service-name container-registry --roles Manager
```

Assigning access to Container Registry by using the API

For step-by-step instructions for assigning, removing, and reviewing access, see [Assigning access to resources by using the API](#) or [Create a policy](#) in the API docs. Role cloud resource names (CRN) in the following table are used to assign access with the API.

Role name	Role CRN
Administrator	<code>crn:v1:bluemix:public:container-registry:::serviceRole:Administrator</code>
Reader	<code>crn:v1:bluemix:public:container-registry:::serviceRole:Reader</code>
Writer	<code>crn:v1:bluemix:public:container-registry:::serviceRole:Writer</code>
Manager	<code>crn:v1:bluemix:public:container-registry:::serviceRole:Manager</code>

Table 6. Role ID values for API use

The following example is for assigning the `Manager` role for Container Registry:

✓ **Tip:** Use `container-registry` for the service name, and refer to the Role ID values table to ensure that you're using the correct value for the CRN.

Curl

```
curl -X POST 'https://iam.cloud.ibm.com/v1/policies' -H 'Authorization: Bearer $TOKEN' -H 'Content-Type: application/json' -d '{
  "type": "access",
  "description": "Manager role for Container Registry",
  "subjects": [
    {
      "attributes": [
        {
          "name": "iam_id",
          "value": "IBMid-123453user"
        }
      ]
    }
  ],
  "roles": [
    {
      "role_id": "crn:v1:bluemix:public:container-registry::::serviceRole:Manager"
    }
  ],
  "resources": [
    {
      "attributes": [
        {
          "name": "accountId",
          "value": "$ACCOUNT_ID"
        },
        {
          "name": "serviceName",
          "value": "container-registry"
        }
      ]
    }
  ]
}
```

Java

```
SubjectAttribute subjectAttribute = new SubjectAttribute.Builder()
    .name("iam_id")
    .value("IBMid-123453user")
    .build();

PolicySubject policySubjects = new PolicySubject.Builder()
    .addAttributes(subjectAttribute)
    .build();

PolicyRole policyRoles = new PolicyRole.Builder()
    .roleId("crn:v1:bluemix:public:container-registry::::serviceRole:Manager")
    .build();

ResourceAttribute accountIdResourceAttribute = new ResourceAttribute.Builder()
    .name("accountId")
    .value("ACCOUNT_ID")
    .operator("stringEquals")
    .build();

ResourceAttribute serviceNameResourceAttribute = new ResourceAttribute.Builder()
    .name("serviceName")
    .value("container-registry")
    .operator("stringEquals")
    .build();

PolicyResource policyResources = new PolicyResource.Builder()
    .addAttributes(accountIdResourceAttribute)
    .addAttributes(serviceNameResourceAttribute)
    .build();

CreatePolicyOptions options = new CreatePolicyOptions.Builder()
    .type("access")
    .subjects(Arrays.asList(policySubjects))
    .roles(Arrays.asList(policyRoles))
    .resources(Arrays.asList(policyResources))
    .build();

Response<Policy> response = service.createPolicy(options).execute();
Policy policy = response.getResult();

System.out.println(policy);
```

Node

```
const policySubjects = [
  {
    attributes: [
      {
        name: 'iam_id',
        value: 'IBMid-123453user',
      },
    ],
  },
];
const policyRoles = [
  {
    role_id: 'crn:v1:bluemix:public:container-registry:::serviceRole:Manager',
  },
];
const accountIdResourceAttribute = {
  name: 'accountId',
  value: 'ACCOUNT_ID',
  operator: 'stringEquals',
};
const serviceNameResourceAttribute = {
  name: 'serviceName',
  value: 'container-registry',
  operator: 'stringEquals',
};
const policyResources = [
  {
    attributes: [accountIdResourceAttribute, serviceNameResourceAttribute]
  },
];
const params = {
  type: 'access',
  subjects: policySubjects,
  roles: policyRoles,
  resources: policyResources,
};

iamPolicyManagementService.createPolicy(params)
  .then(res => {
    examplePolicyId = res.result.id;
    console.log(JSON.stringify(res.result, null, 2));
  })
  .catch(err => {
    console.warn(err)
  });
```

Python

```
policy_subjects = PolicySubject(
    attributes=[SubjectAttribute(name='iam_id', value='IBMid-123453user')])
policy_roles = PolicyRole(
    role_id='crn:v1:bluemix:public:container-registry:::serviceRole:Manager')
account_id_resource_attribute = ResourceAttribute(
    name='accountId', value='ACCOUNT_ID')
service_name_resource_attribute = ResourceAttribute(
    name='serviceName', value='container-registry')
policy_resources = PolicyResource(
    attributes=[account_id_resource_attribute,
                service_name_resource_attribute])

policy = iam_policy_management_service.create_policy(
    type='access',
    subjects=[policy_subjects],
    roles=[policy_roles],
    resources=[policy_resources]
).get_result()

print(json.dumps(policy, indent=2))
```

Go

```
subjectAttribute := &iampolicymanagementv1.SubjectAttribute{
    Name:  core.StringPtr("iam_id"),
    Value: core.StringPtr("IBMid-123453user"),
```



```

}
policySubjects := &iampolicymanagementv1.PolicySubject{
    Attributes: []iampolicymanagementv1.SubjectAttribute{*subjectAttribute},
}
policyRoles := &iampolicymanagementv1.PolicyRole{
    RoleID: core.StringPtr("crn:v1:bluemix:public:container-registry:::serviceRole:Manager"),
}
accountIDResourceAttribute := &iampolicymanagementv1.ResourceAttribute{
    Name:      core.StringPtr("accountId"),
    Value:     core.StringPtr("ACCOUNT_ID"),
    Operator:  core.StringPtr("stringEquals"),
}
serviceNameResourceAttribute := &iampolicymanagementv1.ResourceAttribute{
    Name:      core.StringPtr("serviceName"),
    Value:     core.StringPtr("container-registry"),
    Operator:  core.StringPtr("stringEquals"),
}
policyResources := &iampolicymanagementv1.PolicyResource{
    Attributes: []iampolicymanagementv1.ResourceAttribute{
        *accountIDResourceAttribute, *serviceNameResourceAttribute
    }
}


options := iamPolicyManagementService.NewCreatePolicyOptions(
    "access",
    []iampolicymanagementv1.PolicySubject{*policySubjects},
    []iampolicymanagementv1.PolicyRole{*policyRoles},
    []iampolicymanagementv1.PolicyResource{*policyResources},
)

policy, response, err := iamPolicyManagementService.CreatePolicy(options)
if err != nil {
    panic(err)
}
b, _ := json.MarshalIndent(policy, "", " ")
fmt.Println(string(b))

```

Assigning access to Container Registry by using Terraform

The following example is for assigning the `Manager` role for Container Registry:

 **Tip:** Use `container-registry` for the service name.

```

resource "ibm_iam_user_policy" "policy" {
    ibm_id = "test@example.com"
    roles  = ["Manager"]

    resources {
        service = "container-registry"
    }
}

```

For more information, see [ibm_iam_user_policy](#) in the Terraform documentation.

Defining IAM access policies for Container Registry

As an administrator, you can define IBM Cloud® Identity and Access Management (IAM) access policies to create different levels of access for different users in IBM Cloud® Container Registry. For example, you can authorize some users to view quotas and other users to set quotas.

You must define IAM *access policies* for every user that works with IBM Cloud Container Registry. The scope of an IAM access policy is based on the user's role or roles that determine the actions that they are allowed to do. Some roles are predefined, but custom roles can be defined.

To find out more about IAM access policies, see [IBM Cloud IAM roles](#).

You can assign Container Registry namespaces to a [resource group](#) and scope access policies to that group, see [Planning namespaces](#). However, you can still define access policies that are scoped to individual Container Registry namespaces or to all namespaces that are owned by the account.

Creating policies

Before you begin, complete the following tasks:

- Decide on the roles that each user needs and on which resources in IBM Cloud Container Registry, see [IAM roles](#). You can create multiple policies, for example, you can grant write access on a resource but grant read access only on another resource. Policies are additive, which means that a global read policy and a resource-scoped write policy grants both read and write access on that resource.

- [Invite users to an account.](#)



Tip: If you want users to create clusters in IBM Cloud Kubernetes Service, ensure that you assign the IBM Cloud Container Registry Administrator role to those users, and don't assign a resource group. For more information, see [Preparing to create clusters](#).

To create policies for IBM Cloud Container Registry, the service name field must be `container-registry`.

If you want to access resources, you must assign roles to users or service IDs. If you want to grant access to everything, don't specify a resource type or a resource. If you want to grant access to a specific namespace, specify the resource type as `namespace` and use the namespace name as the resource.

- To create a policy for users, see [Managing access to resources](#).
- To create a policy for service IDs, run the `ibmcloud iam service-policy-create` command or use the IBM Cloud console to bind roles to your service IDs. To create policies, you must have the Administrator role. You automatically have the Administrator role on your own account. For more information, see [Creating and working with service IDs](#) and [Managing access to resources](#).



Tip: For an example of useful access policies for IBM Cloud Container Registry, see [Granting access to Container Registry resources tutorial](#).

Managing image security with Vulnerability Advisor

Vulnerability Advisor is provided as part of IBM Cloud® Container Registry. Vulnerability Advisor checks the security status of container images that are provided by IBM, third parties, or added to your organization's registry namespace.

Vulnerability Advisor provides security management for [IBM Cloud Container Registry](#). Vulnerability Advisor generates a security status report that includes suggested fixes and best practices. Vulnerability Advisor is available in two versions: version 3 and version 4. Version 4 uses new architecture and a different scanning engine.

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

When you add an image to a namespace, the image is automatically scanned by Vulnerability Advisor to detect security issues and potential vulnerabilities. If security issues are found, instructions are provided to help fix the reported vulnerability.

Any issues that are found by Vulnerability Advisor result in a verdict that indicates that it is not advisable to deploy this image. If you choose to deploy the image, any containers that are deployed from the image include known issues that might be used to attack or otherwise compromise the container. The verdict is adjusted based on any exemptions that you specified.

Fixing the security and configuration issues that are reported by Vulnerability Advisor can help you to secure your IBM Cloud infrastructure.

You can use IBM Cloud Security and Compliance Center to monitor vulnerabilities that are detected by Vulnerability Advisor. For more information, see [Getting started with Security and Compliance Center](#).

⊖ **Deprecated:** Using Portieris to block the deployment of images with issues that are found by Vulnerability Advisor is deprecated.

About Vulnerability Advisor

Vulnerability Advisor provides functions to help you to secure your images.

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

The following functions are available in version 3:

- Scans images for issues.
- Provides an evaluation report that is based on security practices that are specific to IBM Cloud Kubernetes Service.
- Provides recommendations to secure configuration files for a subset of application types.
- Provides instructions about how to fix a reported [vulnerable package](#) or [configuration issue](#) in its reports.
- Applies exemption policies to reports at an account, [namespace](#), [repository](#), or [tag](#) level to mark when issues that are flagged do not apply to your use case.

The following functions are available in version 4:

- Scans images for issues.
- Provides an evaluation report that is based on security practices that are specific to IBM Cloud Kubernetes Service.
- Provides instructions about how to fix a reported [vulnerable package](#) in its reports.
- Applies exemption policies to reports at an account, [namespace](#), [repository](#), or [tag](#) level to mark when issues that are flagged do not apply to your use case.

The **Security status** column in the **Images** tab of the Container Registry dashboard displays the number of issues that are associated with each image. To find out more about the issues, click the link in the **Security status** column.

The Vulnerability Advisor dashboard provides an overview and assessment of the security for an image. If you want to find out more about the Vulnerability Advisor dashboard, see [Reviewing a vulnerability report](#).

📌 **Note:** Encrypted images aren't scanned by Vulnerability Advisor.

Data protection

To scan images and containers in your account for security issues, Vulnerability Advisor collects, stores, and processes the following information:

- Free-form fields, including IDs, descriptions, and image names (registry, namespace, repository name, and image tag)
- Metadata about the file modes and creation timestamps of the configuration files
- The content of system and application configuration files in images and containers
- Installed packages and libraries (including their versions)

Do not put personal information into any field or location that Vulnerability Advisor processes, as identified in the preceding list.

Scan results, aggregated at a data center level, are processed to produce anonymized metrics to operate and improve the service. In version 3, a vulnerability report (scan result) is generated when the image is pushed to the registry (and is regenerated regularly thereafter). When Vulnerability Advisor is queried, a scan result is retrieved that might be up to 5 days old. Scan results are deleted 30 days after they are generated.

In version 4, the image is indexed when it is first pushed to Container Registry registry, and that index report is stored in the database. When Vulnerability Advisor is queried, the image index report is retrieved, and a vulnerability assessment is produced. This action happens dynamically every time Vulnerability Advisor is queried. Therefore, no pregenerated scan result exists that requires deleting. However, the image index report is deleted within 30 days of the deletion of the image from the registry.

Types of vulnerabilities

Vulnerable packages

Vulnerability Advisor checks for vulnerable packages in images that are using supported operating systems and provides a link to any relevant security notices about the vulnerability.

Packages that contain known vulnerability issues are displayed in the scan results. The possible vulnerabilities are updated daily by using the published security notices for the Docker image types that are listed in the following table. Typically, for a vulnerable package to pass the scan, a later version of the package is required that includes a fix for the vulnerability. The same package can list multiple vulnerabilities, and in this case, a single package update can address multiple vulnerabilities.

Vulnerability Advisor returns vulnerabilities only when a package fix is published by the distributor. Declared vulnerabilities that aren't fixed yet, or are not going to be fixed, are not reported by Vulnerability Advisor. Therefore, if Vulnerability Advisor does not report any vulnerabilities, there might still be a risk in the image.

For version 3, the scanning of an image is triggered in one of the following ways:

- When a new image is pushed to the registry.
- When a new security notice is released for a package that is installed in the image, the image is queued for scanning, which might take some time to complete.
- While an image is tagged in the registry, it is scanned every week.

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

For version 4, the image is indexed the first time that it is pushed. Thereafter, the vulnerability assessment is calculated every time Vulnerability Advisor is queried about that image.

The following tables show the supported Docker base images that Vulnerability Advisor checks for vulnerable packages.



Note: Vulnerability Advisor supports only releases of platforms that are currently supported by the vendor of that platform.


Docker base image	Supported versions	Source of security notices
Alpine	All stable versions with vendor security support.	Git - Alpine Linux and CVE .
CentOS	Version 7	CentOS announce archives and CentOS CR announce archives .
Debian	All stable versions with vendor security support or long-term support.	Debian security announcements and Debian LTS Security Information .
GoogleContainerTools distroless	All stable versions with vendor security support.	GoogleContainerTools distroless
Red Hat® Enterprise Linux® (RHEL)	RHEL/UBI 7, RHEL/UBI 8, and RHEL/UBI 9	Red Hat Security Data API .
Ubuntu	All stable versions with vendor security support.	Ubuntu Security Notices .

Table 1. Supported Docker base images that Vulnerability Advisor 3 checks for vulnerable packages

Docker base image	Supported versions	Source of security notices
Alpine	All stable versions with vendor security support.	Alpine SecDB database .
Debian	All stable versions with vendor security support up to version 11 (bullseye). CVEs on binary packages that are associated with the Debian source package <code>linux</code> , such as <code>linux-libc-dev</code> , are not reported. Most of these binary packages are kernel and kernel modules, which are not run in container images.	Debian Oval database .
GoogleContainerTools distroless	All stable versions with vendor security support.	GoogleContainerTools distroless
Red Hat® Enterprise Linux® (RHEL)	RHEL/UBI 7, RHEL/UBI 8, and RHEL/UBI 9	Red Hat Security Data API .
Ubuntu	All stable versions with vendor security support.	Ubuntu CVE Tracker .

Table 2. Supported Docker base images that Vulnerability Advisor 4 checks for vulnerable packages

Configuration issues - version 3 only

 **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).


Configuration issues are potential security issues that are related to how an `app` is set up. Configuration issues are not supported in version 4.

Many of the reported problems can be fixed by updating your `Dockerfile`.

Images are scanned only if they are using an operating system that is supported by Vulnerability Advisor. Vulnerability Advisor checks the configuration settings for the following types of apps:

- MySQL
- NGINX
- Apache

Setting the Vulnerability Advisor version

 **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

Starting from version 1.0.0 of the Container Registry plug-in, you can choose whether to fetch results from either version 3, `v3` (the default), or version 4, `v4` , of Vulnerability Advisor for the following commands:

- `ibmcloud cr va IMAGE`, where `IMAGE` is the name of the image.
- `ibmcloud cr image-list`.
- `ibmcloud cr image-digests`.

To retrieve results from version 4 instead of version 3, run the following `ibmcloud cr va-version-set` command.

```
$ ibmcloud cr va-version-set v4
```

Alternatively, you can set an environment variable, `va_version` , and specify the Vulnerability Advisor version that you want to use. Valid values are `v3` and `v4` .

Reviewing a vulnerability report

Before you deploy an image, you can review its Vulnerability Advisor report for details about any vulnerable packages and nonsecure container or app settings.

You can also check whether the image is compliant with organizational policies.

If you don't address any discovered issues, those issues can impact the security of containers that are using that image. If you use enforcement in your container runtime environment, you might be prevented from deploying that image unless all issues are exempted by your policy.



Tip: If your image does not meet the requirements that are set by your organization's policy, you must configure the image to meet those requirements before you can deploy it. For more information about how to view and change the organization policy, see [Setting organizational exemption policies](#).

Reviewing a vulnerability report by using the console - version 3 only



Deprecated: Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

You can review the security of Docker images that are stored in your namespaces in IBM Cloud Container Registry by using the IBM Cloud console. Version 4 does not support viewing vulnerability reports in the console, but you can use the CLI or the API.

1. Log in to IBM Cloud.
2. Click the **Navigation menu** icon, then click **Container Registry**.
3. Click **Images**. A list of your images and the security status of each image is displayed in the **Images** table.
4. To see the report for the image that is tagged `latest`, click the row for that image. The **Image details** tab opens showing the data for that image. If no `latest` tag exists in the repository, the most recent image is used.
5. If the **Security status** column shows any issues, to find out about the issues, click the **Issues by type** tab. The **Vulnerabilities** and **Configuration Issues** tables open.
 - **Vulnerabilities** table. Shows the Vulnerability ID for each issue, the policy status for that issue, the affected packages and how to resolve the issue. To see more information about that issue, expand the row. A summary of that issue is displayed that contains a link to the vendor security notice for that issue. Lists packages that contain known vulnerability issues.

The list is updated daily by using published security notices for the Docker image types that are listed in [Types of vulnerabilities](#). Typically, for a vulnerable package to pass the scan, a later version of the package is required that includes a fix for the vulnerability. The same package can list multiple vulnerabilities, and in this case, a single package update can correct multiple issues. Click the security notice code to view more information about the package and for steps to update the package.
 - **Configuration issues** table. Shows the configuration issue ID for each issue, the policy status for that issue, and the security practice. To see more information about that issue, expand the row. A summary of that issue is displayed that contains a link to the security notice for that issue.

The list contains suggestions for actions that you can take to increase the security of the container and any application settings for the container that are nonsecure. Expand the row to view how to resolve the issue.
6. Complete the corrective action for each issue shown in the report, and rebuild the image.

Reviewing a vulnerability report by using the CLI

You can review the security of Docker images that are stored in your namespaces in IBM Cloud Container Registry by using the CLI.

1. List the images in your IBM Cloud account. A list of all images is returned, independent of the namespace where they are stored.

```
$ ibmcloud cr image-list
```

2. Check the status in the **SECURITY STATUS** column.
 - **No Issues** No security issues were found.
 - **<X> Issues** The number of potential security issues or vulnerabilities that are found, where **<X>** is the number of issues.
 - **Scanning** The image is being scanned and the final vulnerability status is not determined.
 - **Unsupported OS** The scan found no supported operating system (OS) distribution and no active configuration issues.
3. To view the details for the status, review the Vulnerability Advisor report:

```
$ ibmcloud cr va <region>.icr.io/<my_namespace>/<my_image>:<tag>
```

In the CLI output, you can view the following information about the configuration issues.

- **Security practice** A description of the vulnerability.
- **Corrective action** Details about how to fix the vulnerability.

Setting organizational exemption policies


If you want to manage the security of an IBM Cloud organization, you can use your policy setting to determine whether an issue is exempt or not.

You can deploy containers from any image regardless of security status.

To find out about the required permissions for working with exemptions, see [Access roles for configuring IBM Cloud Container Registry](#).


 **Deprecated:** Using Portieris to block the deployment of images with issues that are found by Vulnerability Advisor is deprecated.

Setting exemption policies by using the console

 **Note:** If you are using the IBM Cloud console, you can set a [namespace](#), [repository](#), or [tag](#) as the scope of the exemption policy. If you want to use the [digest](#) as the scope, you must use the CLI, see [Setting organizational exemption policies by using the CLI](#).

If you want to set exemptions from the policy by using the IBM Cloud console, complete the following steps:

1. Log in to IBM Cloud. You must be logged in to see Vulnerability Advisor in the IBM Cloud console.
2. Click the **Navigation menu** icon, then click **Container Registry**.
3. Click **Settings**.
4. In the **Security policy exemptions** section, click **Create**.
5. Select the issue type.
6. Enter the issue ID.

 **Tip:** You can find this information in your [vulnerability report](#). The **Vulnerability ID** column contains the ID to use for CVE or security notice issues; the **Configuration Issue ID** column contains the ID to use for configuration issues.

7. Select the registry namespace, repository, image, and tag that you want the exemption to apply to.
8. Click **Create**.

You can also edit and remove exemptions by hovering over the relevant row and clicking the **open and close list of options** icon.

Setting exemption policies by using the CLI

 **Note:** If you are using the CLI, you can set a [namespace](#), [repository](#), [digest](#), or [tag](#) as the scope of the exemption policy.

If you want to set exemptions from the policy by using the CLI, you can run the following commands:

- To create an exemption for a security issue, run the [ibmcloud cr exemption-add](#) command.
- To list your exemptions for security issues, run the [ibmcloud cr exemption-list](#) command.
- To list the types of security issues that you can exempt, run the [ibmcloud cr exemption-types](#) command.
- To delete an exemption for a security issue, run the [ibmcloud cr exemption-rm](#) command.

For more information about the commands, you can use the `--help` option when you run the command.

Setting up Terraform for Container Registry

Terraform on IBM Cloud® enables predictable and consistent provisioning of IBM Cloud services so that you can rapidly build complex, multitiered cloud environments that follow Infrastructure as Code (IaC) principles. Similar to using the IBM Cloud CLI or API and SDKs, you can automate the provisioning, update, and deletion of your IBM Cloud® Container Registry instances by using HashiCorp Configuration Language (HCL).

✓ **Tip:** Are you looking for a managed Terraform on IBM Cloud solution? Try out [IBM Cloud Schematics](#). With Schematics, you can use the Terraform scripting language that you are familiar with, but you don't need to worry about setting up and maintaining the Terraform command line and the IBM Cloud Provider plug-in. Schematics also provides pre-defined Terraform templates that you can install from the IBM Cloud catalog.

Installing Terraform and creating a Container Registry namespace

Before you begin, ensure that you have the [required access](#) to create and work with IBM Cloud Container Registry resources.

1. To install the Terraform CLI and configure the IBM Cloud Provider plug-in for Terraform, follow the [Terraform on IBM Cloud getting started tutorial](#). The plug-in abstracts the IBM Cloud APIs that are used to provision, update, or delete Container Registry resources.
2. Create a Terraform configuration file that is named `main.tf`. In this file, you add the configuration to create a Container Registry namespace and to assign a user an IAM *access policy* in Identity and Access Management (IAM) for that namespace by using HashiCorp Configuration Language (HCL). For more information, see the [Terraform Language Documentation](#).

The following example creates a namespace in the default *resource group* with a name of your choice and attaches an image retention policy to that namespace that retains 10 images. To retrieve the ID of the default resource group, the `ibm_resource_group` data source is used. Then, the user `user@ibm.com` is assigned the Manager role in the IAM access policy for the namespace for a particular region. The region is retrieved from the `terraform.tfvars` file that you created in step 1.

```
data "ibm_resource_group" "group" {
  name = "default"
}

resource "ibm_cr_namespace" "cr_namespace" {
  name = "<namespace_name>"
  resource_group_id = data.ibm_resource_group.group.id
}

resource "ibm_cr_retention_policy" "cr_retention_policy" {
  namespace = ibm_cr_namespace.cr_namespace.id
  images_per_repo = 10
}

resource "ibm_iam_user_policy" "policy" {
  ibm_id = "user@ibm.com"
  roles = ["Manager"]

  resources {
    service = "container-registry"
    resource = ibm_cr_namespace.cr_namespace.id
    resource_type = "namespace"
    region = var.region
  }
}
```



Note: Updating a namespace by using Terraform is not supported. You can use Terraform to create and remove namespaces only.

3. Initialize the Terraform CLI.

```
$ terraform init
```

4. Create a Terraform execution plan. The Terraform execution plan summarizes all the actions that need to be run to create the Container Registry namespace and IAM access policy in your account.

```
$ terraform plan
```

5. Create the Container Registry namespace and IAM access policy in IBM Cloud.

```
$ terraform apply
```

6. From the [Container Registry namespace overview page](#), verify that your namespace is created successfully.

7. Verify that the IAM access policy is successfully assigned. For more information, see [Reviewing assigned access in the console](#).

Next steps

Now that you successfully created your first Container Registry namespace with Terraform on IBM Cloud, you can choose between the following tasks:

- Learn how to [add images to your namespace](#).
- Explore other supported arguments and attributes for the [Container Registry Terraform resources and data sources](#) that were used in this example.

Enhancing security


Managing security and compliance for Container Registry

IBM Cloud® Container Registry is integrated with the Security and Compliance Center to help you manage security and compliance for your organization.

For more information about managing security and compliance, see [Getting started with Security and Compliance Center](#).


Using VPEs for VPC to privately connect to Container Registry

You can use IBM Cloud® virtual private endpoints (VPE) for Virtual Private Cloud (VPC) to connect to IBM Cloud® Container Registry from your VPC network by using the IP addresses of your choice, which are allocated from a subnetwork within your VPC.

 **Important:** Any Container Registry VPE gateways that were created before 11 November 2022 are deprecated and must be replaced by 15 December 2022. For more information, see [Changes to Container Registry VPE gateways from 11 November 2022](#).

VPEs are virtual IP interfaces that are bound to an endpoint gateway created on a per service, or service instance, basis (depending on the service operation model). The endpoint gateway is a virtualized function that scales horizontally, is redundant and highly available, and spans all *availability zones* of your VPC. Endpoint gateways enable communications from virtual server instances within your VPC and IBM Cloud service on the private backbone. VPE for VPC gives you the experience of controlling all the private addressing within your cloud. For more information, see [About virtual private endpoint gateways](#).

If you have an IBM Cloud VPC instance and want to connect the VPC instance to IBM Cloud Container Registry for your Container Registry services, you can create a VPE gateway for your VPC to access IBM Cloud Container Registry within your VPC network. Any connections to IBM Cloud Container Registry that originate from within the VPC automatically go through the Container Registry VPE gateway, if one exists. For more information, see [Getting started with Virtual Private Cloud](#).

 **Note:** When you connect to Container Registry from the IBM Cloud console, you must go through a browser in your VPC to ensure that the connection goes through the Container Registry VPE gateway.

For VPE gateways created before 11 November 2022, you must ensure that the canonical domain name for the registry [region](#) (for example, `us.icr.io` in `us-south`) resolves to the IP address of the VPE gateway. This action ensures that the image name, which starts with the hostname, is consistent. You can ensure consistency by creating container hostmap entries or configuring the `kube` Domain Name System (DNS).

For VPE gateways created after 11 November 2022, this additional configuration is not required because the domain name resolution is now handled automatically by the VPE gateway.

For more information about other IBM Cloud VPE services, see [VPE supported services](#).

Before you begin


Before you target a VPE for Container Registry, you must complete the following tasks.

- Ensure that a Virtual Private Cloud is created, see [Getting started with Virtual Private Cloud](#).
- Make a plan for your virtual private endpoints, see [Planning for virtual private endpoint gateways](#).
- Ensure that correct access controls are set for your VPE, see [Configuring ACLs and security groups for use with endpoint gateways](#).
- Understand the limitations of having a VPE, see [Virtual private endpoint limitations](#).
- Understand how to view details about a VPE, see [Viewing details of an endpoint gateway](#).

Virtual private endpoints

The table lists IBM Cloud Container Registry private endpoints that are supported from the following VPC regions:

- Dallas (`us-south`)
- Frankfurt (`eu-de`)
- London (`eu-gb`)
- Osaka (`jp-osa`)
- Sao Paulo (`br-sao`)
- Sydney (`au-syd`)
- Tokyo (`jp-tok`)
- Toronto (`ca-tor`)
- Washington (`us-east`)

 **Important:** You can create a VPE gateway for your local Container Registry service only. For VPE gateways created after 11 November 2022, you can pull images from any other Container Registry region by using the public hostnames, such as `uk.icr.io`. For VPE gateways created before 11 November 2022, if you want to connect to IBM Cloud Container Registry in another region, you must enable classic access, see [Creating a](#)

[classic access VPC](#) and use private hostnames, such as `private.uk.icr.io`.

Setting up a VPE for IBM Cloud Container Registry

When you create a VPE gateway by using the CLI or API, you must specify the *cloud resource name (CRN)* of the region that you want to connect to Container Registry. Review the following table for the available regions and CRNs to use to create your VPE gateway.

You can create VPE gateways in the following locations: `ap-north`, `ap-south`, `br-sao`, `ca-tor`, `eu-central`, `jp-osa`, `uk-south`, `us-south`, and `us-east` (global registry).

Registry region	Cloud resource name (CRN)
ap-north	crn:v1:bluemix:public:container-registry:jp-tok:::endpoint:jp.icr.io
ap-south	crn:v1:bluemix:public:container-registry:au-syd:::endpoint:au.icr.io
br-sao	crn:v1:bluemix:public:container-registry:br-sao:::endpoint:br.icr.io
ca-tor	crn:v1:bluemix:public:container-registry:ca-tor:::endpoint:ca.icr.io
eu-central	crn:v1:bluemix:public:container-registry:eu-de:::endpoint:de.icr.io
jp-osa	crn:v1:bluemix:public:container-registry:jp-osa:::endpoint:jp2.icr.io
uk-south	crn:v1:bluemix:public:container-registry:eu-gb:::endpoint:uk.icr.io
us-south	crn:v1:bluemix:public:container-registry:us-south:::endpoint:us.icr.io
Global us-east	crn:v1:bluemix:public:container-registry:us-east:::endpoint:icr.io

Table 1. Region availability and cloud resource names for connecting Container Registry over private IBM Cloud networks

For VPE gateways that were created before 11 November 2022, if you want to connect to IBM Cloud Container Registry in another region, you must use hostnames, such as `private.uk.icr.io`. For more information about private Container Registry networks, see [Securing your connection to Container Registry](#).

For VPE gateways that are created after 11 November 2022, you can pull images from any other Container Registry region by using the public hostnames, such as `uk.icr.io`.

Configuring an endpoint gateway

To configure a VPE gateway, complete the following steps:

1. List the available services, including IBM Cloud infrastructure services available (by default) for all VPC users. For more information, see [VPE supported services](#).
2. [Create an endpoint gateway](#) for IBM Cloud Container Registry that you want to be privately available to the VPC. To create the VPE gateway by using the CLI, run the following command, where `<CRN>` is the CRN of the target region as shown in Table 1.

```
$ ibmcloud is endpoint-gateway-create --target <CRN> --vpc-id <VPC-ID> --name myname
```

3. [Bind a reserved IP address](#) to the endpoint gateway.
4. View the created VPE gateways associated with the IBM Cloud Container Registry. For more information, see [Viewing details of an endpoint gateway](#).

Now your virtual server instances in the VPC can access your IBM Cloud Container Registry instance privately through it.

Securing your connection to Container Registry

You can use private network connections to securely route your data in IBM Cloud® Container Registry.

If you use cloud-based services for production workloads, you can use a secure private connection so that you ensure that you adhere to any compliance regulations. You can use IBM Cloud service endpoints to connect to IBM Cloud services over the private IBM Cloud network.

Service endpoints make it easier to securely route network traffic between different IBM Cloud services and your registry over the private IBM Cloud network. This network routing ensures that your data doesn't go over the public internet.

To learn more about IBM Cloud service endpoints, see [Secure access to services by using service endpoints](#).

Using private network connections

You must set up your account with the correct authority so that you can set up a private network connection for pushing and pulling images.

IBM Cloud Container Registry is a multi-tenant offering and you are, therefore, not required to create your own service endpoints to connect to the registry over private connections.

Enabling service endpoint support for the account

To connect to IBM Cloud services over a private network, you must meet the following criteria.

- You must be able to access the classic infrastructure.
- You must enable [virtual routing and forwarding](#) (VRF) and connectivity to service endpoints for your account.
- You must also have a billable account.

To enable your IBM Cloud account to use virtual routing and forwarding (VRF) and service endpoints, see [Enabling VRF and service endpoints](#).

Considerations for private network connections

- Private domain names aren't used by the `container-registry` CLI plug-in. If you push an image on the private domain name, and then use the CLI on a public connection to run the `ibmcloud cr images` command, the image is listed with all the other images that have the same public domain name.
- If you have private IBM Cloud Kubernetes Service clusters, you aren't required to change anything. IBM Cloud Kubernetes Service already modifies the connection, which is referenced by the public domain name, to use a private connection. This behavior is unchanged.
 - If you use IBM Cloud Kubernetes Service, you can optionally reference your images from private clusters by using the private registry domain name. However, you must create more `imagePullSecrets` for the extra domain names, see [Understanding how to authorize your cluster to pull images from a registry](#).
- Pull traffic is not charged for image pulls that use private connections.
- If you use [image signing](#) and want to use the private domain names, you must re-sign any images that you already have because the domain name forms part of the signed artifact.

Pushing and pulling images

Private connections are available so that you can push and pull images. You use the same `icr.io` domain name, with the prefix `private.`, see [Regions](#).



Note: You can't use private connections for image management operations by using the IBM Cloud Container Registry CLI.

Run the `docker login` command to authenticate with your registry. Replace `<apikey>` with your *API key* and `<private_registry_domain>` with the private domain name where your namespaces are set up. The private domain names are described in [Regions](#).

```
$ docker login -u iamapikey -p <apikey> <private_registry_domain>
```



Tip: IBM Cloud Container Registry supports other clients as well as Docker. To log in by using other clients, see [Accessing your namespaces in automation](#).

For more information, see [Automating access to IBM Cloud Container Registry](#).

Enforcing access to your account over a private network

You can prevent or allow image pulls or pushes over public network connections for your account by using the `ibmcloud cr private-only` command.

You can also use this command to check whether the use of private connections is set for your account.



Important: After you enable the use of private connections on your account, any attempts to pull and push images or access signatures over the public network are rejected.



Tip: Because the use of private connections doesn't apply to the management API, you can still use the CLI over a public connection.

- To prevent image pulls or pushes over public network connections for your account, run the following command.

```
$ ibmcloud cr private-only --enable
```

- To reinstate image pulls or pushes over public network connections for your account, run the following command.

```
$ ibmcloud cr private-only --disable
```

- To check whether the use of public connections is prevented for image pushes or pulls in your account, run the following command.

```
$ ibmcloud cr private-only --status
```

Encrypting images for content confidentiality

You can protect the confidentiality of your IBM Cloud® Container Registry images, and ensure that hosts that aren't trusted can't run the images.

Create an encrypted image so that people without the *private key* can't access the content. Create the encrypted image by using an RSA public-private key pair to encrypt and decrypt the image. A public key is not a secret and anyone can use it to encrypt an image. A private key is a secret, and only users that have that private key can use it to decrypt the image.

Encryption is supported in IBM Cloud Container Registry and complies with the following standards:

- [opencontainers/image-spec](#)
- [opencontainers/artifacts](#)

For more information about encrypting images, see [Advancing container image security with encrypted container images](#) and [Advancing image security and compliance through Container Image Encryption](#).

Before you begin

- Complete the instructions in [Getting started with IBM Cloud Container Registry](#).
- Ensure that you have the most recent version of the **container-registry** CLI plug-in for the IBM Cloud CLI, see [Updating the container-registry CLI plug-in](#).
- Ensure that you have a private namespace to push your encrypted image to, see [Set up a namespace](#).
- Install [Buildah version 1.15](#), or later, so that you can build encrypted images. Buildah works in the Linux® environment only. As an alternative to Linux®, you can use a virtual machine or Docker image to run Buildah to build images.
- Install [Podman](#).

Step 1: Create the public-private key pair

Create a public-private key pair by using OpenSSL commands.

1. Create a work directory, for example, `<user_keys>`, in which to store the keys and change to that directory:

```
$ mkdir <user_keys>; cd <user_keys>
```

2. Use OpenSSL to create a private key, where `<user>` is the name for your key's identity:

```
$ openssl genrsa --out <user>Private.pem
```

3. Create a public key:

```
$ openssl rsa -in <user>Private.pem -pubout -out <user>Pub.pem
```



Tip: To use the private key in production, you must safely store and protect the private key in a suitable store. You might also want to manage the public key in the same way. For more information, see [Storing keys](#).

4. List the keys to ensure that they are created:

```
$ ls -l
```

5. To use this key pair to encrypt images, display the public key by running the following `cat` command:

```
$ cat <user>Pub.pem
```

You get a response similar to the following output:


```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA8Ny7dCWQ8PdqlddYSwk
QOCB3lUEZVEyj9StX3jnISF/rxIsUZzJfb0rQN0fGkm+1sCCtltgQdztTjito8Fh
DGflqQBSmV40XP3iZnNUJDrHuAol463Z/BuxxFXL3ry6rTosLGfrRwdQjxp8RSsn
WyII02rmcqXZYe4SCtiMjMejLLTIDWLIMdYL3d6hA4DpgDLoh6EPmhKMOVwRt5b0
ew5eMLcDuq6But0M5yv4zYVHNrajY41NK+abSlFb6wzMg2AUDiC/MxV1LRq6mpyZ
GJllx3LS1M1j7fD03pmh/M0X7yD/4RgHwFaW4/4CQBw3fyxr0v0pZzZay+o
-----END PUBLIC KEY-----
```

Step 2: Encrypt the image

Encrypt the image by using the public key and then build a container image by using a *Dockerfile*.

1. Go to the directory where you store your apps, for example, `<my_app>`.

```
$ cd <my_app>
```

2. Create the Dockerfile by running the following command:

```
$ cat << EOF >> Dockerfile
FROM nginx:latest
RUN echo "some secret" > /secret-file
EOF
```

3. Use Buildah to create an unencrypted image by running the following command, where `<namespace>` is your namespace:

```
$ buildah bud -t us.icr.io/<namespace>/<my_app> .
```

`us.icr.io/<namespace>/<my_app>` is committed to the local image store.

4. Encrypt the image by using the public key and upload the image to the registry by running the following commands and by specifying the JSON Web Encryption (`jwe`) protocol to encrypt the image, where `<user_keys>/<user>Pub.pem` is the encryption key.

```
$ buildah push --encryption-key jwe:..<user_keys>/<user>Pub.pem us.icr.io/<namespace>/<my_app>
```



API key, you can supply the `--creds <user_name>` option, where `<user_name>` is the username. If you use the `--creds <user_name>` the option, when requested, type in the password of the registry credential.

You get a response that informs you that the manifest is written to the image destination, which is the registry.

5. Check in your registry to make sure that the image is there.

Step 3: Pull and decrypt the image

Pull the image from the registry and decrypt it by using the private key.

1. To ensure that you are pulling from the registry and that you are not using the local cache, remove the image locally:

```
$ buildah rmi -f us.icr.io/<namespace>/<my_app>
```

2. (Optional) You can try to pull the image without providing the decryption key to confirm that the image can't be decrypted:

```
$ buildah pull us.icr.io/<namespace>/<my_app>
```

The output contains a message similar to the following message:

```
...<truncated>...
Error decrypting layer sha256:ab4ea03582e08a8e8fc35b778cc6f1a1fa797469fa9cc61cee85f703b316bb12: missing
private key needed for decryption
ERROR exit status 125
```

3. Use Buildah to pull the image with the decryption key, where `<user_keys>/<user>Private.pem` is the decryption key and `us.icr.io/<namespace>/<my_app>` is the registry:

```
$ buildah pull --decryption-key ../<user_keys>/<user>Private.pem us.icr.io/<namespace>/<my_app>
```


The encrypted image is retrieved from the registry, decrypted, and stored in the local image store.

4. Confirm that the image is stored by running Podman:

```
$ podman run -it us.icr.io/<namespace>/<my_app> /bin/bash
```

Storing keys

To use the private key in production, you must safely store and protect the private key. You might also want to manage the public key in the same way to control who can build images. You can use [IBM Key Protect](#) to store and protect your keys.

IBM Key Protect stores symmetric keys rather than the asymmetric PKI keys that are used for image encryption. You can add your keys separately as two IBM Key Protect standard keys by using the dashboard, CLI, or API. IBM Key Protect requires that only Base64 data is imported. To obtain pure Base64 data, you can encode the PEM files by running `"openssl enc -base64 -A -in <user>Private.pem -out <user>Private.b64"` before you load the Base64 content, and reverse this action to obtain the usable key again by running `"openssl enc -base64 -A -d -in <user>Private..b64 -out <user>Private.pem"`.

For more information about how to use IBM Key Protect to store and protect your keys, see [Bringing your encryption keys to the cloud](#) and [Importing your own keys](#).

As an alternative, you can protect your keys in your own store by using an IBM Key Protect root key to wrap them. This action means that you must unwrap them again by using IBM Key Protect and the valid root key.

For example, to wrap keys by using the CLI, run the command `"ibmcloud kp key wrap <root_key_id> -p <base64 encoded image key>"` and to unwrap keys, run the command `"ibmcloud kp key unwrap <root_key_id> -p <base64 cyphertext>"`, where `<root_key_id>` is the ID of the root key that you are using.

For more information about how you can protect your keys in your own store by using an IBM Key Protect root key to wrap them, see [Wrapping keys](#).



Note: Encrypted images are not scanned by Vulnerability Advisor.

Next steps

Run your encrypted image in a Red Hat® OpenShift® on IBM Cloud® cluster by using the [Image Key Synchronizer cluster add-on](#).

Managing your data in Container Registry

Information about your data and how it is stored in IBM Cloud® Container Registry.

The IBM Cloud platform provides layered security controls across network and infrastructure. IBM Cloud provides a group of security services that can be used by application developers to secure their mobile and web apps. For more information, see [How do I know that my data is safe?](#)

How your data is stored

Image data

Image data is stored in IBM Cloud Object Storage, which is encrypted at rest, and encrypted in transit between IBM Cloud Container Registry and IBM Cloud Object Storage. For more information about IBM Cloud Object Storage, see [About IBM Cloud Object Storage](#).

Data that is stored in IBM Cloud Container Registry is backed up regularly. For more information, see [High availability and disaster recovery](#).

Scanning data

To scan images and containers in your account for security issues, IBM Cloud Container Registry collects, stores, and processes the following information:

- Free-form fields, including IDs, descriptions, and image names ([registry](#), [namespace](#), [repository](#) name, and image [tag](#))
- Metadata about the file modes and creation timestamps of the configuration files
- The content of system and application configuration files in images and containers
- Installed packages and libraries (including their versions)

Scan results, aggregated at a data center level, are processed to produce anonymized metrics to operate and improve the service. Scan results are deleted 30 days after they are generated. For more information, see [Data protection](#).



Important: Do not put personal information into any field or location that IBM Cloud Container Registry processes, as identified in the preceding list.

Deleting your data

You can delete your IBM Cloud Container Registry namespaces, images, and private repositories.

Deleting the service

When an IBM Cloud account is canceled or removed, the resource instances that track the account's usage of IBM Cloud Container Registry are deleted. As part of that action all namespaces, and the images that they contain, are removed in accordance with the data retention policy.

The IBM Cloud Container Registry data retention policy describes how long your data is stored after you delete the service. The data retention policy is included in the IBM Cloud Container Registry service description, which you can find in the Service Description for IBM Cloud in the [IBM Cloud Terms and Notices](#).

Deleting namespaces

If you no longer require a registry namespace, you can remove the namespace from your IBM Cloud account. Deleting a namespace removes all images, trash, and trust information that is contained in the namespace. For more information, see [Removing namespaces](#).

Deleting images

You can delete unwanted images from your private repository by using either the IBM Cloud console or the CLI. For more information, see [Deleting images from your private repository](#).

You can clean up your namespace by choosing to retain only the most recent images in each repository in that namespace in IBM Cloud Container Registry. You can detect and delete old images from all the repositories in a namespace by running a one-off command, `ibmcloud cr retention-run`, or by setting a scheduled policy by running the `ibmcloud cr retention-policy-set` command. For more information, see [Cleaning up your namespaces](#).

Deleting private repositories

You can delete private repositories that are no longer required, and any associated images, by using the IBM Cloud console. For more information, see [Deleting a private repository and any associated images](#).

Restoring deleted data

You can restore images from the trash by [digest](#) or by [tag](#).

You can restore an image from the trash by running the `ibmcloud cr image-restore` command. To find out which images are in the trash, run the `ibmcloud cr trash-list` command. Images are stored in the trash for 30 days. For more information, see [Restoring images](#).

Signing images for trusted content

IBM Cloud® Container Registry provides trusted content technology so that you can sign images to ensure the integrity of images in your registry namespace.

By pulling and pushing signed images, you can verify that your images were pushed by the correct party, such as your continuous integration (CI) tools.

You can use Red Hat® signatures to sign your images.

Signing images by using Red Hat signatures

You can use various tools to create [Red Hat signatures](#) for your images. You can store your signed images for trusted content by using the Red Hat signatures extension API, which is supported by IBM Cloud Container Registry.

You can use the following tools to create Red Hat signatures:

- [Skopeo](#)
- [Podman](#)
- [Red Hat OpenShift CLI](#)

Using Skopeo to sign images

To use [Skopeo](#) to sign your images, you must create a private [GNU Privacy Guard \(GnuPG or GPG\)](#) identity and then run the `skopeo` command.



Note: The following example doesn't include Skopeo authentication.

1. To create a GnuPG identity, run the following command.

```
$ gpg --generate-key
```

2. Push and sign the image at the same time by using the GnuPG identity to sign the image. Where `<your_email>` is the email address that you used

to sign up for GnuPG, `<repository:tag>` is your repository and tag, and `<image>` is the name of your image in the format `<region><namespace><repository:tag>`, where `<region>` is the name of your region and `<namespace>` is the name of your namespace.

✓ **Tip:** To find the names of your images, run `ibmcloud cr image-list`. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`. If the list images command times out, see [Why is it timing out when I list images?](#) for assistance.

```
$ skopeo --insecure-policy copy --sign-by <your_email> docker-daemon:<repository:tag> docker://<image>
```

For example, where `user@email.com` is your GnuPG email address, `bluebird:build1` is your repository and tag, and `us.icr.io/birds/bluebird:build1` is the name of your image.

```
$ skopeo --insecure-policy copy --sign-by user@email.com docker-daemon:bluebird:build1
docker://us.icr.io/birds/bluebird:build1
```

On macOS, if you get the error `Error copying image to the remote destination: Error writing signatures: mkdir /var/lib/containers/sigstore: permission denied`, override the internal default for registry configuration so that the correct signature storage is used by running the command with the `--registries.d` option.

```
$ skopeo --registries.d . --insecure-policy copy --sign-by user@email.com docker-
daemon:us.icr.io/birds/bluebird:build1 docker://us.icr.io/birds/bluebird:build1
```

✓ **Tip:** On Linux® and macOS, the default configuration for the tools is to store the signatures locally. Storing signatures locally can lead to signature verification failure because the signature is not in the registry. To fix this problem, you can modify or delete the configuration file. On Linux®, the configuration is saved in `/etc/containers/registries.d/default.yaml`. On macOS, the configuration file is saved in `/usr/local/etc/containers/registries.d/default.yaml`. On macOS, when Skopeo is installed by using the [Homebrew](#) package manager, the configuration file might be at `/opt/homebrew/Cellar/etc/containers/registries.d/default.yaml` for Apple silicon, or `/usr/local/Cellar/etc/containers/registries.d/default.yaml` for Intel.

Using Podman to sign images

You can use Podman to sign images. For more information, see [Podman](#).

Signing images by using the Red Hat OpenShift CLI

You can sign your images by using the Red Hat OpenShift CLI. For more information, see [Red Hat OpenShift CLI](#). The Red Hat® OpenShift® CLI uses the `oc` command.

Using IAM IP address access restrictions

You can enable IAM IP address access restrictions when you're using IBM Cloud® Container Registry.

To enable IAM IP address access restrictions, you must ensure that the Cloud Identity and Access Management (IAM) access list is configured so that the Container Registry OAuth service can still function. The OAuth service is used to authenticate image pulls and pushes in Container Registry.

You must ensure that the IP addresses of any computers that can originate pulls and pushes are added to the IAM IP address access list, see [Allowing specific IP addresses](#).

Granting access if you are using a public network

If you're using IBM Cloud Container Registry over a public network, you must ensure that the Public IP addresses of any computers that can originate pulls and pushes are added to the IAM IP allowlist.

Granting access if you are using a private network

If you're using IBM Cloud Container Registry in one of the following scenarios, you must add the private IP addresses of any computers that can originate pulls and pushes to the allowlist.

- You're using one of the `private.*` domains, for example, `private.us.icr.io`.
- You're using an IBM Cloud Kubernetes Service cluster in a [configuration](#) that automatically talks to the registry over a private connection.


Enforcing container image security by using Portieris

You can use [Portieris](#) to enforce image security policies in IBM Cloud® Container Registry.

Portieris is a Kubernetes admission controller that verifies your container images before you deploy them to your cluster in IBM Cloud Kubernetes Service.

You can use Portieris to enforce policies on image signatures. If an image doesn't meet your policy requirements, the resource that contains the pod is not deployed to your cluster.

 **Deprecated:** Using Portieris to block the deployment of images with issues that are found by Vulnerability Advisor is deprecated.

 **Tip:** If Portieris is deployed and the cluster workers are showing as working correctly, but nothing is scheduled, see [Why don't my pods restart after my workers were down?](#) for assistance.

Portieris is supported on Red Hat® OpenShift®.

Installing Portieris in your cluster

Install Portieris in your cluster.

To install Portieris in your IBM Cloud Kubernetes Service or Red Hat OpenShift on IBM Cloud cluster, see [Enabling image security enforcement in your cluster](#). If you use this installation, it is deployed and maintained for you and it runs in the control plane, which gives higher availability.

If you prefer to install Portieris directly, or you aren't using IBM Cloud Kubernetes Service or Red Hat OpenShift on IBM Cloud, see [Installing Portieris](#).

Portieris policies

Portieris has two types of policy. Image policy resources and cluster image policy resources. You can override the default Portieris policies.

For more information about Portieris policies, see [Portieris policies](#).

Uninstalling Portieris

How to uninstall Portieris.

[Uninstall Portieris](#).

Observability

Monitoring metrics for Container Registry

You can use IBM Cloud® Monitoring to monitor platform metrics of IBM Cloud® Container Registry usage for your account and to create alerts based on these metrics.

Platform metrics for Container Registry must be enabled in each Container Registry region that you want to monitor, see [Enabling metrics for Container Registry](#).

Enabling metrics for Container Registry



Note: You must enable Container Registry metrics in each region that you want to see metrics.

You can create a Monitoring instance in the region that you want to monitor and enable platform metrics for it. Alternatively, you can enable platform metrics on an existing Monitoring instance in that region.

Complete the following steps to create and configure platform metrics for Container Registry.

1. Create and configure an IBM Cloud Monitoring instance that is configured with platform metrics in the region that you want to monitor, see [Getting started with IBM Cloud Monitoring](#).

For more information about the locations where Container Registry is enabled for monitoring, see [Locations of platform metrics](#).

2. Log in to IBM Cloud.

```
$ ibmcloud login
```

3. Target the [region](#) where you want to enable metrics by running the `ibmcloud cr region-set` command. Replace `<region>` with the name of the [region](#).

```
$ ibmcloud cr region-set <region>
```

4. Check whether metrics are already enabled for your account by running the following `ibmcloud cr platform-metrics` command. This command also displays the registry that the result applies to.

```
$ ibmcloud cr platform-metrics --status
```

5. Enable platform metrics by running the following `ibmcloud cr platform-metrics` command. It can take up to 30 minutes for your metrics to become effective.

```
$ ibmcloud cr platform-metrics --enable
```

If you want to target a different [region](#), run the `ibmcloud cr region-set` command.

Locations of platform metrics

You can configure one monitoring instance in each region to collect platform metrics for IBM Cloud Container Registry. The following tables list the locations where metrics can be collected if you enable collection of Container Registry service metrics in that region.

Locations in Americas	Platform metrics available
Dallas (us-south)	Yes
Sao Paulo (br-sao)	Yes
Toronto (ca-tor)	Yes

Table 1. The automatic collection of Container Registry service metrics in Americas locations

Locations in Asia Pacific	Platform metrics available
Osaka (jp-osa)	Yes

Sydney (au-syd)	Yes
-----------------	-----

Tokyo (jp-tok)	Yes
----------------	-----

Table 2. The automatic collection of Container Registry service metrics in Asia Pacific locations

Locations in Europe	Platform metrics available
Frankfurt (eu-de)	Yes
London (eu-gb)	Yes

Table 3. The automatic collection of Container Registry service metrics in Europe locations

Location for Global	Platform metrics available
Global	Yes

Table 4. The automatic collection of Container Registry service metrics for Global

For more information about where to see Container Registry metrics, see [Where to look for metrics](#).

For more information about the locations where IBM Cloud services are enabled to send metrics to IBM Cloud Monitoring, see [IBM Cloud services by location](#).

Where to look for metrics

Monitoring metrics

The [region](#) in which a Container Registry or a Vulnerability Advisor metric is available corresponds to the region of the Container Registry that generated the metric.

The following table shows the location of Monitoring metrics.

Region for your account's registry	Domain name of your registry	Location of Monitoring metrics
ap-north	jp.icr.io	Tokyo (jp-tok)
ap-south	au.icr.io	Sydney (au-syd)
br-sao	br.icr.io	Sao Paulo (br-sao)
ca-tor	ca.icr.io	Toronto (ca-tor)
eu-central	de.icr.io	Frankfurt (eu-de)
jp-osa	jp2.icr.io	Osaka (jp-osa)
uk-south	uk.icr.io	London (eu-gb)
us-south	us.icr.io	Dallas (us-south)


Table 5. Location of Monitoring metrics

The following table shows the location of global registry Monitoring metrics.

Registry	Global registry	Location of IBM Cloud Activity Tracker metrics
Global	icr.io	Washington (us-east)

Table 6. Location of global registry Monitoring metrics

Viewing metrics

 **Important:** To monitor Container Registry metrics, you must start the Monitoring UI instance that is enabled for platform metrics in the region

where you're using Container Registry.

Starting the Monitoring UI from the Observability page

To start the Monitoring UI from the Observability page, complete the following steps:

1. [Start the Monitoring UI](#).
2. Select **DASHBOARDS**.
3. In the **Default Dashboards** section, expand **IBM**.
4. Choose the Container Registry dashboard from the list. Available dashboards are **Container Registry Usage** and **Container Registry Quota Usage**. For more information about predefined dashboards, see [Predefined dashboards](#).

Next, change the scope or make a copy of the Default dashboard so that you can monitor your account in Container Registry. For more information, see [Working with dashboards](#).

Predefined dashboards

The following table outlines the predefined monitoring dashboards that you can use to monitor Container Registry metrics.

Dashboard name	Description	Default dashboard
Container Registry Usage	A dashboard that you can use to visualize the traffic usage and storage usage. Traffic usage is the sum of bytes from image pulls from your Container Registry namespaces in the current billing period. Storage usage is the sum of bytes of images in your Container Registry namespaces.	Yes
Container Registry Quota Usage	A dashboard that you can use to visualize the traffic usage and storage usage and compare the data to your quotas, if set. Visible only to those accounts that have finite quotas. The Container Registry Quota Usage dashboard is available only if you enable metrics and you have both a storage and a traffic quota set.	

Table 7. Predefined dashboards



Important: The predefined dashboards can't be changed. You can copy any predefined dashboard so that you can change it to suit your requirements. For more information, see [Working with dashboards](#).



Note: When you start your dashboard, some metrics might display a **Data Load Error** warning icon. This warning is because more time is required to create the data. When data is available, the warning sign goes away, and the metric is populated. You might also need to change the resolution period.

Platform metrics

Metric Name	Information
Pull Traffic	The account's pull traffic in the current month.
Pull Traffic Quota	The account's pull traffic quota.
Storage Quota	The account's storage quota.
Storage	The account's storage usage.

Table 8. Metrics available by plan names

Pull Traffic

The account's pull traffic in the current month.

Metadata	Description
Metric Name	ibm_containerregistry_pull_traffic
Metric Type	gauge

Value Type	byte
------------	------

Table 9. Pull Traffic metric metadata

Pull Traffic Quota

The account's pull traffic quota.

Metadata	Description
Metric Name	ibm_containerregistry_pull_traffic_quota
Metric Type	gauge
Value Type	byte

Table 10. Pull Traffic Quota metric metadata

Storage Quota

The account's storage quota.

Metadata	Description
Metric Name	ibm_containerregistry_storage_quota
Metric Type	gauge
Value Type	byte

Table 11. Storage Quota metric metadata

Storage

The account's storage usage.

Metadata	Description
Metric Name	ibm_containerregistry_storage
Metric Type	gauge
Value Type	byte

Table 12. Storage metric metadata

Auditing events for Container Registry

Use IBM Cloud® Activity Tracker to track how users and applications interact with the IBM Cloud® Container Registry service in IBM Cloud.

The IBM Cloud Activity Tracker service records user-initiated activities that change the state of a service in the IBM Cloud. For more information, see [IBM Cloud Activity Tracker](#).

Locations of service events

You can track how users and applications interact with the IBM Cloud Container Registry service. The following tables list the locations where the automatic collection of Container Registry service events is enabled.

Locations in Americas	Service events available
Dallas (us-south)	Yes
Sao Paulo (br-sao)	Yes

Toronto (ca-tor)	Yes
------------------	-----

Table 1. The automatic collection of Container Registry service events in Americas locations	
Locations in Asia Pacific	Service events available
Osaka (jp-osa)	Yes
Sydney (au-syd)	Yes
Tokyo (jp-tok)	Yes

Table 2. The automatic collection of Container Registry service events in Asia Pacific locations	
Locations in Europe	Service events available
Frankfurt (eu-de)	Yes
London (eu-gb)	Yes

Table 3. The automatic collection of Container Registry service events in Europe locations	
Location for Global	Service events available
Global	Yes

Table 4. The automatic collection of Container Registry service events for Global

For more information about where to see Container Registry events, see [Where to look for events](#).

For more information about the locations where IBM Cloud services are enabled to automatically collect events, see [IBM Cloud services that generate Activity Tracker events by location](#).

Where to look for events

IBM Cloud Activity Tracker events

The [region](#) in which a Container Registry or a Vulnerability Advisor event is available corresponds to the region of the Container Registry that generated the event, except for `ap-south`. Events for `ap-south` show in `Tokyo (jp-tok)`.

The following table shows the location of IBM Cloud Activity Tracker events.

Region for your account's registry	Domain name of your registry	Location of IBM Cloud Activity Tracker events
ap-north	jp.icr.io	Tokyo (jp-tok)
ap-south	au.icr.io	Tokyo (jp-tok)
br-sao	br.icr.io	Sao Paulo (br-sao)
ca-tor	ca.icr.io	Toronto (ca-tor)
eu-central	de.icr.io	Frankfurt (eu-de)
jp-osa	jp2.icr.io	Osaka (jp-osa)
uk-south	uk.icr.io	London (eu-gb)
us-south	us.icr.io	Dallas (us-south)

Table 5. Location of IBM Cloud Activity Tracker events

The following table shows the location of global registry IBM Cloud Activity Tracker events.

Registry	Global registry	Location of IBM Cloud Activity Tracker events
----------	-----------------	---

Global	icr.io	Dallas (us-south)
--------	--------	-------------------

Table 6. Location of global registry IBM Cloud Activity Tracker events

API methods

The following tables list the API methods that generate an event when they are called.

Actions that generate events for authorization

Action	Description	Data Event
container-registry.auth.get	Check whether the use of public connections is prevented for image pushes or pulls in your account.	
container-registry.auth.set	Prevent or allow image pulls or pushes over public network connections for your account.	

Table 7. Actions that generate events for your authorization

Actions that generate events for images

Action	Description	Data Event
container-registry.image.bulkdelete	Delete multiple images from Container Registry. If the image is signed, the signature is deleted as well.	True
container-registry.image.delete	Delete an image from Container Registry. If the image is signed, the signature is deleted as well.	True
container-registry.image.inspect	Display details about an image.	
container-registry.image.list	List the images in your IBM account.	
container-registry.image.pull	Pull an image from Container Registry.	True
container-registry.image.push	Push an image to Container Registry.	True
container-registry.image.tag	Add a tag that refers to a pre-existing Container Registry image.	
container-registry.image.untag	Remove a tag, or tags, from each specified image in Container Registry.	
container-registry.manifest.inspect	View the contents of the manifest for an image.	

Table 8. Actions that generate events for images

Actions that generate events for namespaces

Action	Description	Data Event
container-registry.namespace.create	Create a namespace in Container Registry. Assign a Container Registry namespace to a resource group.	
container-registry.namespace.delete	Delete a namespace from Container Registry.	
container-registry.namespace.list	List the Container Registry namespaces in your IBM account.	

Table 9. Actions that generate events for namespaces

Actions that generate events for plans

Action	Description	Data Event
container-registry.plan.get	Display information about the current pricing plan.	
container-registry.plan.set	Upgrade to the standard plan.	

Table 10. Actions that generate events for plans

Actions that generate events for quotas

Action	Description	Data Event
container-registry.quota.get	Display the current quotas for traffic and storage, and the usage information against those quotas.	
container-registry.quota.set	Modify the quotas. Quota settings must be managed separately for your account in each registry instance. You can set quota limits for storage in your free or standard plan.	

Table 11. Actions that generate events for quotas

Actions that generate events for retention policies

Action	Description	Data Event
container-registry.retention.analyze	List the images that are deleted if you apply a specific retention policy.	
container-registry.retention.list	List the image retention policies for your account.	
container-registry.retention.set	Set a policy to retain images in a namespace in Container Registry by applying specified criteria.	

Table 12. Actions that generate events for retention policies

Actions that generate events for settings

Action	Description	Data Event
container-registry.settings.get	Get registry service settings for the targeted account, such as whether platform metrics are enabled.	
container-registry.settings.set	Update registry service settings for the targeted account, such as enabling platform metrics.	

Table 13. Actions that generate events for settings

Actions that generate events for signing images

Action	Description	Data Event
container-registry.signature.delete	Delete a signature from an image in Container Registry.	True
container-registry.signature.read	Read a signature from an image in Container Registry.	True
container-registry.signature.write	Write a signature to an image in Container Registry.	True

Table 14. Actions that generate events for signing images

Actions that generate events for trash

Action	Description	Data Event
<code>container-registry.trash.list</code>	Display all the images in the trash in your IBM Cloud account.	
<code>container-registry.trash.restore</code>	Restore a deleted image from the trash. If the deleted image is signed, the signature is restored too.	

Table 15. Actions that generate events for trash

Actions that generate events for vulnerabilities

Action	Description	Data Event
<code>container-registry.account-vulnerability-report.list</code>	View the Vulnerability Advisor reports for images in your Container Registry account. For more information about request data, see Request data for the account vulnerability report .	
<code>container-registry.account-vulnerability-status.list</code>	View Vulnerability Advisor security status for images in your Container Registry account. For more information about request data, see Request data for the account vulnerability status .	
<code>container-registry.image-vulnerability-report.read</code>	View the Vulnerability Advisor report for an image in Container Registry. For more information about request and response data, see Request and response data for the vulnerability report .	
<code>container-registry.image-vulnerability-status.read</code>	View the Vulnerability Advisor security status for an image in Container Registry. For more information about request and response data, see Request and response data for the vulnerability status .	

Table 16. Actions that generate events for vulnerabilities

Actions that generate events for exemption policies

Action	Description	Data Event
<code>container-registry.exemption.create</code>	Create a Vulnerability Advisor exemption.	
<code>container-registry.exemption.delete</code>	Delete a Vulnerability Advisor exemption.	

Table 17. Actions that generate events for Vulnerability Advisor exemption policies

Analyzing Activity Tracker events

The following fields are populated as described, depending on how you populate the request:

- `target.name` shows the image name and, if you request an image name with a tag, a tag. If you request an image name by digest, the digest is shown instead of the tag because the digest might have many tags.
- `target.id` shows the image name by digest to represent a searchable unique ID for the image, unless the request is for an image with a tag and the request fails before the digest is discovered. To see all the events for this digest across all tags, you can search by `target.id`.
- `target.resourceGroupId` shows the resource group ID that is associated with a namespace and its resources. For more information, see [Set up a namespace](#).



Note: Earlier namespaces that aren't migrated to IAM don't have a resource group, therefore this field is not available.

Request data for vulnerability events

Get the data for vulnerability events in Container Registry.

Request data for the account vulnerability report

Get the vulnerability assessment (`container-registry.account-vulnerability-report.list`) for the list of registry images that belong to a specific account.

The following table lists the fields that are available through the `requestData` field in events with the action `container-registry.account-vulnerability-report.list`.

Custom Event Fields	Type	Description
<code>requestData.RequestParameters.repository</code>	String	The name of the repository that you want to see image vulnerability assessments for. For example, <code>us.icr.io/namespace/image</code> .
<code>requestData.RequestParameters.includeIBM</code>	String	When set to true , the returned list contains IBM public images and the account images. If not set, or set to false , the list contains only the account images.
<code>requestData.RequestParameters.includePrivate</code>	String	When set to false , the returned list does not contain the private account images. If not set, or set to true , the list contains the private account images.

Table 18. Custom event fields for Container Registry account vulnerability reports list

For more information about the action `container-registry.account-vulnerability-report.list`, see [Get the vulnerability assessment for all images](#) in the API documentation.

Request data for the account vulnerability status

Get the vulnerability assessment status (`container-registry.account-vulnerability-status.list`) for the list of registry images that belong to a specific account.

The following table lists the fields that are available through the `requestData` field in events with the action `container-registry.account-vulnerability-status.list`.

Custom Event Fields	Type	Description
<code>requestData.RequestParameters.repository</code>	String	The name of the repository that you want to see image vulnerability assessments for. For example, <code>us.icr.io/namespace/image</code> .
<code>requestData.RequestParameters.includeIBM</code>	String	When set to true , the returned list contains IBM public images and the account images. If not set, or set to false , the list contains only the account images.
<code>requestData.RequestParameters.includePrivate</code>	String	When set to false , the returned list does not contain the private account images. If not set, or set to true , the list contains the private account images.

Table 19. Custom event fields for Container Registry account vulnerability status list

For more information about the action `container-registry.account-vulnerability-status.list`, see [Get vulnerability assessment status for all images](#) in the API documentation.

Request and response data for the vulnerability report

Get the vulnerability assessment (`container-registry.image-vulnerability-report.read`) for a registry image.

The following table lists the fields that are available through the `requestData` and `responseData` fields in events with the action `container-registry.image-vulnerability-report.read`.

Custom Event Fields	Type	Description
<code>requestData.RequestParameters.name</code>	String	The name of the image. For example, <code>us.icr.io/namespace/repository:tag</code> . The following constraint applies: The value must match the regular expression <code>.*</code> .

<code>responseData.id</code>	String	The unique ID of the report.
<code>responseData.status</code>	String	<p>The following values for the overall vulnerability assessment status are available:</p> <div> <div>OK</div> <div>WARN</div> <div>FAIL</div> <div>UNSUPPORTED</div> <div>INCOMPLETE</div> <div>UNSCANNED</div> </div> <p>For more information about these status codes, see Vulnerability report status codes in the API documentation.</p>

Table 20. Custom event fields for Container Registry image vulnerability reports read

For more information, see [Get vulnerability assessment status](#) in the API documentation.

Request and response data for the vulnerability status

Get the overall vulnerability status (`container-registry.image-vulnerability-status.read`) for a registry image.

The following table lists the fields that are available through the `requestData` and `responseData` fields in events with the action `container-registry.image-vulnerability-status.read`.

Custom Event Fields	Type	Description
<code>requestData.RequestParameters.name</code>	String	<p>The name of the image. For example, <code>us.icr.io/namespace/repository:tag</code>.</p> <p>The following constraint applies: The value must match the regular expression <code>.*</code>.</p>
<code>responseData.status</code>	String	<p>The following values for the overall vulnerability assessment status are available:</p> <div> <div>OK</div> <div>WARN</div> <div>FAIL</div> <div>UNSUPPORTED</div> <div>INCOMPLETE</div> <div>UNSCANNED</div> </div> <p>For more information about these status codes, see Vulnerability report status codes in the API documentation.</p>

Table 21. Custom event fields for Container Registry image vulnerability status read

For more information, see [Get vulnerability status](#) in the API documentation.

Request data for image signing events

Get the data for events about image signing in Container Registry.

The following table lists the fields that are available through the `requestData` field in events with the following actions:

- `container-registry.signature.delete`
- `container-registry.signature.read`
- `container-registry.signature.write`

Custom Event Fields	Type	Description
<code>requestData.RequestParameters.repository</code>	String	The name of the repository for which you want to see image signing reports. For example, <code>us.icr.io/namespace/image</code> .

<code>requestData.RequestParameters.signatureMethod</code>	String	Displays the technology that is used to sign the image, such as Red Hat Signing .
<code>requestData.RequestParameters.signatureObject</code>	String	Specifies the object type upon which a signing operation is performed, for example, <code>image</code> .

Table 22. Custom event fields for Container Registry signing

Analyzing logs for Container Registry

IBM Cloud® Container Registry generates platform services logs that are displayed in your logging instances.

For more information about how to configure logging instances to receive platform services logs, see [Configuring IBM Cloud service logs](#).

Most of the time when you work with IBM Cloud Container Registry you are pushing, pulling, or managing images. These interactions output results that either you or your automation, tools, or runtime receive; IBM Cloud Container Registry doesn't generate platform logs for them. IBM Cloud Activity Tracker provides a comprehensive list of events for auditing these interactions, see [Auditing events for Container Registry](#).

IBM Cloud Container Registry generates platform services logs that are displayed in your logging instances for the following specific cases:

- When IBM Cloud Container Registry runs scheduled retention policies for namespaces in your account.
- When IBM Cloud Container Registry calculates your account's registry traffic and average storage usage per hour for a region.

Locations of platform services logs

IBM Cloud Container Registry generates platform logs that are displayed in your logging instances. The following tables list the locations where the automatic collection of Container Registry service logs is enabled.

Locations in Americas	Are platform services logs available?
Dallas (us-south)	Yes
Sao Paulo (br-sao)	Yes
Toronto (ca-tor)	Yes

Table 1. The automatic collection of Container Registry service logs in the Americas locations

Locations in Asia Pacific	Are platform services logs available?
Osaka (jp-osa)	Yes
Sydney (au-syd)	Yes
Tokyo (jp-tok)	Yes

Table 2. The automatic collection of Container Registry service logs in the Asia Pacific locations

Locations in Europe	Are platform services logs available?
Frankfurt (eu-de)	Yes
London (eu-gb)	Yes

Table 3. The automatic collection of Container Registry service logs in the Europe locations

Location for Global	Are platform services logs available?
Global	Yes

Table 4. The automatic collection of Container Registry service logs for Global

For more information about where to see Container Registry service logs, see [Where to look for IBM Log Analysis logs](#).

For more information about the locations where IBM Cloud services are enabled to send platform logs to IBM Log Analysis, see [IBM Cloud services that generate IBM Log Analysis logs by location](#).

Where to look for IBM Log Analysis logs

The [region](#) in which an IBM Cloud Container Registry or a Vulnerability Advisor log entry is available corresponds to the region of the IBM Cloud Container Registry that generated the log, except for `ap-south`. Logs for `ap-south` show in `Tokyo (jp-tok)`.

The following table shows the location of IBM Log Analysis logs.

Region for your account's registry	Domain name of your registry	Location of IBM Log Analysis logs
<code>ap-south</code>	<code>au.icr.io</code>	<code>Tokyo (jp-tok)</code>
<code>ap-north</code>	<code>jp.icr.io</code>	<code>Tokyo (jp-tok)</code>
<code>br-sao</code>	<code>br.icr.io</code>	<code>Sao Paulo (br-sao)</code>
<code>ca-tor</code>	<code>ca.icr.io</code>	<code>Toronto (ca-tor)</code>
<code>eu-central</code>	<code>de.icr.io</code>	<code>Frankfurt (eu-de)</code>
<code>jp-osa</code>	<code>jp2.icr.io</code>	<code>Osaka (jp-osa)</code>
<code>uk-south</code>	<code>uk.icr.io</code>	<code>London (eu-gb)</code>
<code>us-south</code>	<code>us.icr.io</code>	<code>Dallas (us-south)</code>

Table 5. Location of IBM Log Analysis logs

The following table shows the location of global registry IBM Log Analysis logs.

Registry	Global registry	Location of IBM Log Analysis logs
Global	<code>icr.io</code>	<code>Dallas (us-south)</code>

Table 6. Location of global registry IBM Log Analysis logs

IAM and Activity Tracker actions by API method

When you use IBM Cloud® Container Registry through the command line or console, the service calls application programming interface (API) methods to complete your requests.

You might need certain permissions to call these API methods, and you can track the requests that you make with an IBM Cloud Activity Tracker instance.

Review the following list of Cloud Identity and Access Management (IAM) actions and IBM Cloud Activity Tracker events that correspond to each API method in Container Registry.

For more information, see the following topics:

- [Container Registry API documentation](#)
- [Vulnerability Advisor for Container Registry API documentation](#)
- [Auditing events for Container Registry](#)
- [Managing IAM access for Container Registry](#)

Container Registry

Review the following account API methods, their required actions in IBM Cloud IAM, and the events that are sent to IBM Cloud Activity Tracker for Container Registry.

Authorization API methods

Action	Method	IAM ACTION	AT ACTION
Get authorization options for the targeted account.	<code>GET /api/v1/auth</code>	<code>container-registry.auth.get</code>	<code>container-registry.auth.get</code>
Update authorization options for the targeted account.	<code>PATCH /api/v1/auth</code>	<code>container-registry.auth.set</code>	<code>container-registry.auth.set</code>

Table 1. Auth

Image API methods

Action	Method	IAM ACTION	AT ACTION
List images.	GET /api/v1/images	container-registry.image.list	container-registry.image.list
Delete more than one image.	POST /api/v1/images/bulkdelete	container-registry.image.delete	container-registry.image.bulkdelete
List images by digest.	POST /api/v1/images/digests	container-registry.image.list	container-registry.image.list
Create tag.	POST /api/v1/images/tags	container-registry.image.pull container-registry.image.push	container-registry.image.tag
Delete image.	DELETE /api/v1/images/{image}	container-registry.image.delete	container-registry.image.delete
Inspect an image.	GET /api/v1/images/{image}/json	container-registry.image.inspect	container-registry.image.inspect
Get image manifest.	GET /api/v1/images/{image}/manifest	container-registry.image.inspect	container-registry.manifest.inspect

Table 2. Images

Message API methods

Action	Method	IAM ACTION	AT ACTION
Return any published system messages.	GET /api/v1/messages		

Table 3. Messages

Namespace API methods

Action	Method	IAM ACTION	AT ACTION
List namespaces.	GET /api/v1/namespaces	container-registry.namespace.list	container-registry.namespace.list
Detailed namespace list.	GET /api/v1/namespaces/details	container-registry.namespace.list	container-registry.namespace.list
Create namespace.	PUT /api/v1/namespaces/{namespace}	container-registry.namespace.create	container-registry.namespace.create
Assign namespace.	PATCH /api/v1/namespaces/{namespace}	container-registry.namespace.create	container-registry.namespace.update
Delete namespace.	DELETE /api/v1/namespaces/{namespace}	container-registry.namespace.delete	container-registry.namespace.delete

Table 4. Namespaces

Plan API methods

Action	Method	IAM ACTION	AT ACTION
Get plans for the targeted account.	GET /api/v1/plans	container-registry.plan.get	container-registry.plan.get

Update plans for the targeted account.	PATCH <code>/api/v1/plans</code>	<code>container-registry.plan.set</code>	<code>container-registry.plan.set</code>
--	-------------------------------------	--	--

Table 5. Plans

Quota API methods

Action	Method	IAM ACTION	AT ACTION
Get quotas for the targeted account.	GET <code>/api/v1/quotas</code>	<code>container-registry.quota.get</code>	<code>container-registry.quota.get</code>
Update quotas for the targeted account.	PATCH <code>/api/v1/quotas</code>	<code>container-registry.quota.set</code>	<code>container-registry.quota.set</code>

Table 6. Quotas

Retention API methods

Action	Method	IAM ACTION	AT ACTION
List retention policies for all namespaces in the targeted IBM Cloud account.	GET <code>/api/v1/retentions</code>	<code>container-registry.retention.list</code>	<code>container-registry.retention.list</code>
Set the retention policy for the specified namespace.	POST <code>/api/v1/retentions</code>	<code>container-registry.retention.set</code>	<code>container-registry.retention.set</code>
Analyze a retention policy, and get a list of what would be deleted by it.	POST <code>/api/v1/retentions/analyze</code>	<code>container-registry.retention.analyze</code>	<code>container-registry.retention.analyze</code>
Get the retention policy for the specified namespace.	GET <code>/api/v1/retentions/{namespace}</code>	<code>container-registry.retention.get</code>	<code>container-registry.retention.get</code>

Table 7. Retentions

Settings API methods

Action	Method	IAM ACTION	AT ACTION
Get registry service settings for the targeted account, such as whether platform metrics are enabled.	GET <code>/api/v1/settings</code>	<code>container-registry.settings.get</code>	<code>container-registry.settings.get</code>
Update registry service settings for the targeted account, such as enabling platform metrics.	PATCH <code>/api/v1/settings</code>	<code>container-registry.settings.set</code>	<code>container-registry.settings.set</code>

Table 8. Settings

Tag API methods

Action	Method	IAM ACTION	AT ACTION
Delete tag.	DELETE <code>/api/v1/tags/{image}</code>	<code>container-registry.image.delete</code>	<code>container-registry.image.untag</code>

Table 9. Tags

Trash API methods

Action	Method	IAM ACTION	AT ACTION
--------	--------	------------	-----------

List images in the trash.	GET /api/v1/trash	container-registry.image.delete	container-registry.trash.list
Restore a digest and all associated tags.	POST /api/v1/trash/{digest}/restoretags	container-registry.image.push	container-registry.trash.restore
Restore deleted image.	POST /api/v1/trash/{image}/restore	container-registry.image.push	container-registry.trash.restore

Table 10. Trash

Vulnerability Advisor API methods

Report API methods

Action	Method	IAM ACTION	AT ACTION
Get the vulnerability assessment for all images.	GET /va/api/v3/report/account	container-registry.exemption.list	container-registry.account-vulnerability-report.list
Get vulnerability assessment status for all images.	GET /va/api/v3/report/account/status	container-registry.exemption.list	container-registry.account-vulnerability-status.list
Get vulnerability status.	GET /va/api/v3/report/image/status/{name}	container-registry.exemption.list	container-registry.image-vulnerability-status.read
Get vulnerability assessment status.	GET /va/api/v3/report/image/{name}	container-registry.exemption.list	container-registry.image-vulnerability-report.read

Table 11. Report

Exemption API methods

Action	Method	IAM ACTION	AT ACTION
List account-level exemptions.	GET /va/api/v3/exempt/image	container-registry.exemption.list	
Get an account-level exemption.	GET /va/api/v3/exempt/image/issue/{issueType}/{issueID}	container-registry.exemption.list	
Create or update an account-level exemption.	POST /va/api/v3/exempt/image/issue/{issueType}/{issueID}	container-registry.exemption.manager	container-registry.exempt
Delete an account-level exemption.	DELETE /va/api/v3/exempt/image/issue/{issueType}/{issueID}	container-registry.exemption.manager	container-registry.exempt
List resource exemptions.	GET /va/api/v3/exempt/image/{resource}	container-registry.exemption.list	

Get details of a resource exemption.	GET /va/api/v3/exempt/image/{resource}/issue/{issueType}/{issueID}	container-registry.exemption.list	
Create or update a resource exemption.	POST /va/api/v3/exempt/image/{resource}/issue/{issueType}/{issueID}	container-registry.exemption.manager	container-registry.exempt
Delete a resource exemption.	DELETE /va/api/v3/exempt/image/{resource}/issue/{issueType}/{issueID}	container-registry.exemption.manager	container-registry.exempt
List the types of exemption.	GET /va/api/v3/exempt/types		
List all exemptions.	GET /va/api/v3/exemptions/account	container-registry.exemption.list	
Delete all exemptions.	POST /va/api/v3/exemptions/deleteAll	container-registry.exemption.manager	container-registry.exempt
List image exemptions.	GET /va/api/v3/exemptions/image/{resource}	container-registry.exemption.list	
List exemptions for images.	POST /va/api/v3/exemptions/images	container-registry.exemption.list	

Table 12. Exemption

Container Registry CLI

IBM Cloud Container Registry CLI

You can use the IBM Cloud® Container Registry CLI, which is provided in the `container-registry` CLI plug-in, to manage your *registry* and its resources for your IBM Cloud account.

Prerequisites

Before you can use the Container Registry CLI, you must complete the following prerequisites.

1. Install the `ibmcloud` CLI plug-in, see [Getting started with the IBM Cloud CLI](#).
2. Install the `container-registry` CLI plug-in, see [Installing the container-registry CLI plug-in](#).
3. Log in to IBM Cloud with the `ibmcloud login` command to generate an *access token* and authenticate your session so that you can run commands in the CLI.

Notes

To find out how to use the Container Registry CLI, see [Getting started with IBM Cloud Container Registry](#).

You're notified on the command line when updates to the `ibmcloud` CLI and `container-registry` CLI plug-ins are available. Ensure that you keep your CLIs up to date so that you can use all the available commands and options. If you want to view the current version of your `container-registry` CLI plug-in, run the `ibmcloud plugin list` command.

For more information about the IAM platform and service access roles that are required for some Container Registry commands, see [Managing IAM access for Container Registry](#).



Important: Do not put personal information in your container images, namespace names, description fields, or in any image configuration data (for example, image names or image labels).



Tip: If Container Registry commands fail with an error that says that they're not registered commands, see [Why do cr commands fail saying they're not registered?](#) for assistance. If the commands fail saying that you're not logged in, see [Why do commands fail saying that I'm not logged in?](#) for assistance.

`ibmcloud cr api`

Returns the details about the registry API endpoint that the commands are run against.

```
ibmcloud cr api
```

Prerequisites

None

`ibmcloud cr exemption-add`

Create an exemption for a security issue. You can create an exemption for a security issue that applies to different scopes. The scope can be the account, [namespace](#), [repository](#), [digest](#), or [tag](#).



Tip: You can identify the images in the scope by using either the tag or the digest. You can reference the image by digest `<dns>/<namespace>/<repo>@<digest>`, which affects the digest and all its tags in the same repository, or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag. To list all images, including [untagged](#) images, run the [ibmcloud cr image-digests](#) command.

```
ibmcloud cr exemption-add --scope SCOPE --issue-type ISSUE_TYPE --issue-id ISSUE_ID [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

`--scope SCOPE`

To set your account as the scope, use `"*"` as the value.

To set a namespace, repository, digest, or tag as the scope, enter the value in one of the following formats:

- `namespace`
- `namespace/repository`
- `namespace/repository:tag`
- `namespace/repository@digest`

`--issue-type ISSUE_TYPE`

The type of security issue that you want to exempt. To find valid issue types, run `ibmcloud cr exemption-types`.

`--issue-id ISSUE_ID`

The ID of the security issue that you want to exempt. To find an issue ID, run `ibmcloud cr va <image>`, where `<image>` is the name of your image, and use the relevant value from either the **Vulnerability ID** or **Configuration Issue ID** column.

`--output json, -o json`

(Optional) Outputs the list in JSON format.

Examples

Create a CVE exemption for CVE with ID `CVE-2018-17929` for all images in the `us.icr.io/birds/bluebird` repository.

```
$ ibmcloud cr exemption-add --scope us.icr.io/birds/bluebird --issue-type cve --issue-id CVE-2018-17929
```

Create an account-wide CVE exemption for CVE with ID `CVE-2018-17929`.

```
$ ibmcloud cr exemption-add --scope "*" --issue-type cve --issue-id CVE-2018-17929
```

Create a configuration issue exemption for issue `application_configuration:nginx.ssl_protocols` for a single image with the tag `us.icr.io/birds/bluebird:1`.

```
$ ibmcloud cr exemption-add --scope us.icr.io/birds/bluebird:1 --issue-type configuration --issue-id application_configuration:nginx.ssl_protocols
```

Create a configuration issue exemption for issue `application_configuration:nginx.ssl_protocols` for a single image with the digest `us.icr.io/birds/bluebird@sha256:101010101010`.

```
$ ibmcloud cr exemption-add --scope us.icr.io/birds/bluebird@sha256:101010101010 --issue-type configuration --issue-id application_configuration:nginx.ssl_protocols
```

`ibmcloud cr exemption-list (ibmcloud cr exemptions)`

List your exemptions for security issues.



Tip: You can identify the images in the scope by using either the [tag](#) or the [digest](#). You can reference the image by digest `<dns>/<namespace>/<repo>@<digest>`, which affects the digest and all its tags in the same repository, or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag. To list all images, including [untagged](#) images, run the [ibmcloud cr image-digests](#) command.

```
ibmcloud cr exemption-list [--scope SCOPE] [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

`--scope SCOPE`

(Optional) List only the exemptions that apply to this scope.

To set a namespace, repository, digest, or tag as the scope, enter the value in one of the following formats:

- `namespace`
- `namespace/repository`
- `namespace/repository:tag`
- `namespace/repository@digest`

`--output json, -o json`

(Optional) Outputs the list in JSON format.

Examples

List all your exemptions for security issues that apply to images in the `birds/bluebird` repository. The output includes exemptions that are account-wide, exemptions that are scoped to the `birds` namespace, and exemptions that are scoped to the `birds/bluebird` repository. The output doesn't include any exemptions that are scoped to specific tags within the `birds/bluebird` repository.

```
$ ibmcloud cr exemption-list --scope birds/bluebird
```

List all your exemptions for security issues that apply to images in the `birds/bluebird@sha256:101010101010` digest. The output includes exemptions that are account-wide, exemptions that are scoped to the `birds` namespace, and exemptions that are scoped to the `birds/bluebird` repository and to the `birds/bluebird@sha256:101010101010` digest. The output doesn't include any exemptions that are scoped to specific tags within the `birds/bluebird` repository.

```
$ ibmcloud cr exemption-list --scope birds/bluebird@sha256:101010101010
```

`ibmcloud cr exemption-rm`

Delete an exemption for a security issue. To view your existing exemptions, run `ibmcloud cr exemption-list`.



Tip: You can identify the images in the scope by using either the [tag](#) or the [digest](#). You can reference the image by digest `<dns>/<namespace>/<repo>@<digest>`, which affects the digest and all its tags in the same repository, or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag. To list all images, including [untagged](#) images, run the [ibmcloud cr image-digests](#) command.

```
ibmcloud cr exemption-rm --scope SCOPE --issue-type ISSUE_TYPE --issue-id ISSUE_ID
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

`--scope SCOPE`

To set your account as the scope, use `"*"` as the value.

To set a namespace, repository, digest, or tag as the scope, enter the value in one of the following formats:

- `namespace`
- `namespace/repository`
- `namespace/repository:tag`
- `namespace/repository@digest`

`--issue-type ISSUE_TYPE`

The issue type of the exemption for the security issue that you want to remove. To find the issue types for your exemptions, run `ibmcloud cr exemption-list`.

`--issue-id ISSUE_ID`

The ID of the exemption for the security issue that you want to remove. To find the issue IDs for your exemptions, run `ibmcloud cr exemption-`

```
list
```

Examples

Delete a CVE exemption for CVE with ID `CVE-2018-17929` for all images in the `us.icr.io/birds/bluebird` repository.

```
$ ibmcloud cr exemption-rm --scope us.icr.io/birds/bluebird --issue-type cve --issue-id CVE-2018-17929
```

Delete an account-wide CVE exemption for CVE with ID `CVE-2018-17929`.

```
$ ibmcloud cr exemption-rm --scope "*" --issue-type cve --issue-id CVE-2018-17929
```

Delete a configuration issue exemption for issue `application_configuration:nginx.ssl_protocols` for a single image with the tag `us.icr.io/birds/bluebird:1`.

```
$ ibmcloud cr exemption-rm --scope us.icr.io/birds/bluebird:1 --issue-type configuration --issue-id application_configuration:nginx.ssl_protocols
```

Delete a configuration issue exemption for issue `application_configuration:nginx.ssl_protocols` for a single image with the digest `us.icr.io/birds/bluebird@sha256:101010101010`.

```
$ ibmcloud cr exemption-rm --scope us.icr.io/birds/bluebird@sha256:101010101010 --issue-type configuration --issue-id application_configuration:nginx.ssl_protocols
```

```
ibmcloud cr exemption-types
```

Lists the types of security issues that you can exempt.

```
ibmcloud cr exemption-types [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

```
--output json, -o json
```

(Optional) Outputs the list in JSON format.

```
ibmcloud cr iam-policies-enable
```



Important: From 5 July 2022, all accounts require Cloud Identity and Access Management (IAM) access policies. If you started to use IBM Cloud Container Registry before the availability of [IAM API key policies in Container Registry](#) in February 2019, you must ensure that you are using IAM access policies to manage access to the Container Registry service. For more information, see [IAM access policies are required from 5 July 2022](#).

If you're using IAM authentication, this command enables fine-grained authorization. For more information, see [Managing IAM access for Container Registry](#) and [Defining IAM access policies](#).

```
ibmcloud cr iam-policies-enable
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

```
ibmcloud cr iam-policies-status
```

Displays the IAM access policy status of the targeted IBM Cloud Container Registry account. For more information, see [Managing IAM access for Container Registry](#) and [Defining IAM access policies](#).



Important: From 5 July 2022, all accounts require IBM Cloud® Identity and Access Management (IAM) access policies. If you started to use IBM


Cloud Container Registry before the availability of [IAM API key policies in Container Registry](#) in February 2019, you must ensure that you are using IAM access policies to manage access to the Container Registry service. For more information, see [IAM access policies are required from 5 July 2022](#).


```
ibmcloud cr iam-policies-status
```


`ibmcloud cr image-digests (ibmcloud cr digests)`


Lists all images, including [untagged](#) images, in your IBM Cloud account. This command returns the [digest](#) in its long format. When you're using the digest to identify an image, always use the long format.

If you want to list tagged images only, run the [ibmcloud cr image-list](#) command.

 **Tip:** You can refer to an image by using a combination of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (`@`) symbol to create the image name in the format `repository@digest` . You can also refer to the image name by using a combination of the content of the **Repository** column (`repository`) and one of the tags in the **Tags** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag` .

 **Note:** From version 1.0.0 of the container-registry plug-in, you can choose whether to run the `ibmcloud cr vulnerability-assessment` command in either version 3 (the default) or version 4 of Vulnerability Advisor. If you want to run Vulnerability Advisor in a different version, see [Setting the Vulnerability Advisor version](#). For more information about the versions of Vulnerability Advisor, see [About Vulnerability Advisor](#).

 **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

 **Deprecated:** From 24 July 2023, the `--json` option is deprecated and is replaced by the `--output json` option.

```
ibmcloud cr image-digests [--format FORMAT | --quiet | -q | --output json | -o json] [--restrict RESTRICTION] [-include-ibm] [--no-va] [--va]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

`--format FORMAT`

(Optional) Format the output elements by using a Go template. For more information, see [Formatting and filtering the Container Registry CLI output](#).

`--quiet, -q`

(Optional) Each image is listed in the format: `repository@digest`

`--output json, -o json`

(Optional) Outputs the list in JSON format.

`--restrict RESTRICTION`

(Optional) Limit the output to display only images in the specified namespace or repository.

`--include-ibm`

(Optional) Includes IBM-provided public images in the output. By default only private images are listed. You can view IBM-provided images in the global registry only.

`--no-va`

(Optional) Excludes the security status (Vulnerability Advisor) results from the output. If you don't need the security status results as part of your `ibmcloud cr image-digests` output, you can use this option to increase performance.

`--va`

(Optional) Includes the Vulnerability Advisor security status results in the output. Use this option to ensure that you are ready for [IBM Cloud Container Registry CLI plug-in version 1.0.0](#). You can use the `--va` option with the `--restrict` option to receive just the information that you require.

Example

Display all the images in the `birds` namespace, including untagged images, in the format `repository@digest`.

```
$ ibmcloud cr image-digests --restrict birds --quiet
```

`ibmcloud cr image-inspect`

Displays details about a specific image. You can reference the image that you want to inspect either by [digest](#), `repository@digest`, or by [tag](#), `repository:tag`.

```
ibmcloud cr image-inspect [--format FORMAT] IMAGE [IMAGE...]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

`--format FORMAT`

(Optional) Format the output elements by using a Go template. For more information, see [Formatting and filtering the Container Registry CLI output](#).

IMAGE

The name of the image for which you want to get a report. You can inspect multiple images by listing each image in the command with a space between each name.

You can identify images by using either the digest `<dns>/<namespace>/<repo>@<digest>` or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag.

To find the names of your images, run one of the following commands:

- To identify your image by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (@) symbol to create the image name in the format `repository@digest`.
- To identify your image by tag, run the `ibmcloud cr image-list` command. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (:) to create the image name in the format `repository:tag`. If a tag is not specified in the image name, the image that is tagged `latest` is deleted by default.

Example

Display details about the exposed ports for the image `us.icr.io/birds/bluebird:1` by using the following formatting directive.

```
$ ibmcloud cr image-inspect --format "{{ .Config.ExposedPorts }}" us.icr.io/birds/bluebird:1
```

`ibmcloud cr image-list (ibmcloud cr images)`

Displays all tagged images in your IBM Cloud account. If you want to list all your images, including untagged images, run the [ibmcloud cr image-digests](#) command. By default, the `ibmcloud cr image-list` command returns the [digest](#) for the images in a truncated format. The `ibmcloud cr image-digests` command returns the long format of the digest.



Note: When you're using the digest to identify an image, always use the long format.



Tip: The image name is the combination of the content of the **Repository** and **Tag** columns in the format: `repository:tag`



Note: From version 1.0.0 of the container-registry plug-in, you can choose whether to run the `ibmcloud cr vulnerability-assessment` command in either version 3 (the default) or version 4 of Vulnerability Advisor. If you want to run Vulnerability Advisor in a different version, see [Setting the Vulnerability Advisor version](#). For more information about the versions of Vulnerability Advisor, see [About Vulnerability Advisor](#).



Deprecated: Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update](#)

[Vulnerability Advisor to version 4 by 19 June 2023.](#)



Tip: If the command to list images times out, see [Why is it timing out when I list images?](#) for assistance.

```
ibmcloud cr image-list [--format FORMAT] [--quiet | -q ] [--restrict RESTRICTION] [--include-ibm] [--no-trunc]
[--show-type] [--no-va] [--va] [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

`--format FORMAT`

(Optional) Format the output elements by using a Go template. For more information, see [Formatting and filtering the Container Registry CLI output](#).

`--quiet, -q`

(Optional) Each image is listed in the format: `repository:tag`

`--restrict RESTRICTION`

(Optional) Limit the output to display only images in the specified namespace or repository.

`--include-ibm`

(Optional) Includes IBM-provided public images in the output. By default only private images are listed. You can view IBM-provided images in the global registry only.

`--no-trunc`

(Optional) Returns the image digest in its long format.

`--show-type`

(Optional) Displays the image manifest type.

`--no-va`

(Optional) Excludes the security status (Vulnerability Advisor) results from the output. If you don't need the security status results as part of your `ibmcloud cr image-list` output, you can use this option to increase performance.

`--va`

(Optional) Includes the Vulnerability Advisor security status results in the output. Use this option to ensure that you are ready for [IBM Cloud Container Registry CLI plug-in version 1.0.0](#). You can use the `--va` option with the `--restrict` option to receive just the information that you require.

`--output json, -o json`

(Optional) Outputs the list in JSON format.

Example

Display the images in the `birds` namespace in the format `repository:tag`, without truncating the image digests.

```
$ ibmcloud cr image-list --restrict birds --quiet --no-trunc
```

`ibmcloud cr image-prune-untagged`

Delete all [untagged](#) images in your IBM Cloud Container Registry account.



Deprecated: From 24 July 2023, the `--json` option is deprecated and is replaced by the `--output json` option.

```
ibmcloud cr image-prune-untagged [--force | -f [--output json | -o json]] --restrict RESTRICTION
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

`--force, -f`

(Optional) Force the command to run with no user prompts.

`--output json, -o json`

(Optional) Outputs JSON that contains the results of cleaning up your untagged images. This option must be used with `--force`.

`--restrict`

(Optional) Limit the clean up to only untagged images in the specified namespace or repository.

Example

Delete all untagged images that are in the `birds` namespace and output the results in JSON format.

```
$ ibmcloud cr image-prune-untagged [--force | -f [--json]] --restrict birds
```

`ibmcloud cr image-restore`

Restore a deleted image from the trash. You can choose to restore by [tag](#) or by [digest](#). If you restore by digest, the digest and all its tags in the same repository are restored. To find out what is in the trash, run the [ibmcloud cr trash-list](#) command.



Tip: If you get an error when you're restoring an image that says that the tagged image exists, see [Why do I get an error when I'm restoring an image?](#) for assistance.



Tip: If you're restoring an image by digest, but some tags aren't restored, see [Why aren't all the tags restored when I restore by digest?](#) for assistance.

```
ibmcloud cr image-restore IMAGE
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

IMAGE

The name of the image that you want to restore from the trash.

To find the names of your images in the trash, run the `ibmcloud cr trash-list` command.

You can identify images by using either the tag or the digest. The image to restore can be referenced by digest `<dns>/<namespace>/<repo>@<digest>`, which restores the digest and all its tags in the same repository, or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag.

Images are stored in the trash for 30 days.

Example

To restore the image `us.icr.io/birds/bluebird:1`, run the following command.

```
$ ibmcloud cr image-restore us.icr.io/birds/bluebird:1
```

For more information about how to use the `ibmcloud cr image-restore` command, see [Restoring images](#).

`ibmcloud cr image-rm`

Delete one or more specified images from Container Registry. You can reference the image that you want to delete either by [digest](#) or by [tag](#).

✓ **Tip:** Where multiple tags exist for the same image digest within a repository, the `ibmcloud cr image-rm` command removes the underlying image and all its tags. If the same image exists in a different repository or namespace, that copy of the image is not removed. If you want to remove a tag from an image and leave the underlying image and any other tags in place, use the [ibmcloud cr image-untag](#) command.

✓ **Tip:** If you want to restore a deleted image, you can list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore a selected image by running the [ibmcloud cr image-restore](#) command.

```
ibmcloud cr image-rm IMAGE [IMAGE...]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

IMAGE

The name of the image that you want to delete. You can delete multiple images at the same time by listing each image in the command with a space between each name. You can identify images by using either the digest `<dns>/<namespace>/<repo>@<digest>` or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag.

Images are stored in the trash for 30 days.

To find the names of your images, run one of the following commands:

- To identify your image by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (**repository**) and the **Digest** column (**digest**) separated by an at (@) symbol to create the image name in the format `repository@digest`.
- To identify your image by tag, run the `ibmcloud cr image-list` command. Combine the content of the **Repository** column (**repository**) and **Tag** column (**tag**) separated by a colon (:) to create the image name in the format `repository:tag`. If a tag is not specified in the image name, the image that is tagged **latest** is deleted by default.

Example

Delete the image `us.icr.io/birds/bluebird:1`.

```
$ ibmcloud cr image-rm us.icr.io/birds/bluebird:1
```

ibmcloud cr image-tag

Add a [tag](#) that you specify in the command to an existing image, copy the tag to another repository, or copy the tag to a repository in a different namespace. When you copy a tag, any Red Hat® signatures for its [digest](#) are also copied. The target image, `TARGET_IMAGE`, is the new image and the source image, `SOURCE_IMAGE`, is the existing image in IBM Cloud Container Registry. The source and target images must be in the same region. You can reference the source image that you want to tag by either digest, `repository@digest`, or by tag, `repository:tag`. You must reference the target image by tag.

You can identify source images by using either the digest `<dns>/<namespace>/<repo>@<digest>` or by tag `<dns>/<namespace>/<repo>:<tag>`. You must reference the target image by tag, `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag.

To find the names of your images, use one of the following alternatives:

- To identify your image by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (**repository**) and the **Digest** column (**digest**) separated by an at (@) symbol to create the image name in the format `repository@digest`.
- To identify your image by tag, run the `ibmcloud cr image-list` command. Combine the content of the **Repository** column (**repository**) and **Tag** column (**tag**) separated by a colon (:) to create the image name in the format `repository:tag`.

✓ **Tip:** If you get a manifest error when you try to tag your image, see [Why do I get a manifest type error when I tag my image?](#), [Why do I get a manifest version error?](#), and [Why do I get a manifest list invalid error?](#) for assistance.

```
ibmcloud cr image-tag [SOURCE_IMAGE] [TARGET_IMAGE]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

SOURCE_IMAGE

The name of the source image. You can identify source images by using either the digest `<dns>/<namespace>/<repo>@<digest>` or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag.

TARGET_IMAGE

The name of the target image. `TARGET_IMAGE` must be in the format `repository:tag`, for example, `us.icr.io/namespace/image:latest`.

Examples

Add another tag reference, `latest`, to the image `us.icr.io/birds/bluebird:1`.

```
$ ibmcloud cr image-tag us.icr.io/birds/bluebird:1 us.icr.io/birds/bluebird:latest
```

Copy the image `us.icr.io/birds/bluebird:peck` to another repository in the same namespace `birds/pigeon`.

```
$ ibmcloud cr image-tag us.icr.io/birds/bluebird:peck us.icr.io/birds/pigeon:peck
```

Copy the image `us.icr.io/birds/bluebird:peck` to another namespace `animals` to which you have access.

```
$ ibmcloud cr image-tag us.icr.io/birds/bluebird:peck us.icr.io/animals/dog:bark
```

ibmcloud cr image-untag

Remove a [tag](#), or tags, from each specified image in IBM Cloud Container Registry.



Tip: To remove a specific tag from an image and leave the underlying image and any other tags in place, use the `ibmcloud cr image-untag` command. If you want to delete the underlying image, and all its tags, use the [ibmcloud cr image-rm](#) command instead.

```
ibmcloud cr image-untag IMAGE [IMAGE...]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

IMAGE

The name of the image for which you want to remove the tag. You can delete the tag from multiple images at the same time by listing each image in the command with a space between each name. `IMAGE` must be in the format `repository:tag`, for example, `us.icr.io/namespace/image:latest`.

To find the names of your images, run `ibmcloud cr image-list`. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`. If a tag is not specified in the image name, the command fails.

Example

Remove the tag `1` from the image `us.icr.io/birds/bluebird:1`.

```
$ ibmcloud cr image-untag us.icr.io/birds/bluebird:1
```

`ibmcloud cr info`

Displays the name and the account of the registry that you are logged in to.

```
ibmcloud cr info
```

Prerequisites

None

`ibmcloud cr login`

Log the local Docker or Podman client in to IBM Cloud Container Registry.

This command is required if you want to run the `push` or `pull` commands for the registry. If you want to run other `ibmcloud cr` commands, you're not required to log in to Container Registry.

```
ibmcloud cr login [--client CLIENT]
```

Container Registry supports other clients as well as Docker and Podman. To log in by using other clients, see [Accessing your namespaces interactively](#).



Tip: If you have a problem when you try to log in, see [Why can't I log in to Container Registry?](#) for assistance. If you're using a Mac and you have a problem when you try to log in, see [Why is Docker login on my Mac failing?](#) for assistance.



Tip: Logging in to Container Registry by using the `ibmcloud cr login` command is subject to IAM login session limits. If your login expires, see [Why does the Container Registry login keep expiring?](#) for assistance.

Prerequisites

None

Command options

`CLIENT`

(Optional) Select the client that you want to log in. Valid values are `docker` and `podman`. If this option is not used and Docker is installed, the default is `docker`; if Docker is not installed, the default is `podman`.

Example

To log in to the registry with Podman, run the following command.

```
$ ibmcloud cr login --client podman
```

`ibmcloud cr manifest-inspect`

View the contents of the [manifest](#) for an image. You can reference the image that you want to inspect either by digest or by tag.

```
ibmcloud cr manifest-inspect [--quiet | -q ] IMAGE
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

`IMAGE`

The name of the image for which you want to inspect the manifest. You can identify images by using either the digest `<dns>/<namespace>/<repo>@<digest>` or by tag `<dns>/<namespace>/<repo>:<tag>`. Where `<dns>` is the domain name, `<namespace>` is the namespace, `<repo>` is the repository, `<digest>` is the digest, and `<tag>` is the tag.

To find the names of your images, run one of the following commands:

- To identify your image by digest, run the `ibmcloud cr image-digests` command. Combine the content of the **Repository** column (**repository**) and the **Digest** column (**digest**) separated by an at (@) symbol to create the image name in the format `repository@digest`.
- To identify your image by tag, run the `ibmcloud cr image-list` command. Combine the content of the **Repository** column (**repository**) and **Tag** column (**tag**) separated by a colon (:) to create the image name in the format `repository:tag`.

`--quiet, -q`

(Optional) Reduces the output to display essential elements only.

Example

To view the contents of the manifest for the image `us.icr.io/birds/bluebird:1`, run the following command.

```
$ ibmcloud cr manifest-inspect us.icr.io/birds/bluebird:1
```

`ibmcloud cr namespace-add`

Choose a name for your *namespace* and add it to your IBM Cloud account.

You can create a namespace in a *resource group* of your choice by using one of the following options.

- Before you create the namespace, run the `ibmcloud target -g <resource_group>` command, where `<resource_group>` is the resource group.
- Specify the required resource group by using the `-g` option on the `ibmcloud cr namespace-add` command.

If you create a namespace in a resource group, you can configure access to resources within the namespace at the [resource group](#) level. However, you can still set permissions for the namespace at the account level or in the namespace itself. If you don't specify a resource group, and a resource group isn't targeted, the default resource group is used.

If you have an older namespace that is not in a resource group, you can assign it to a resource group, see [ibmcloud cr namespace-assign](#).

 **Tip:** Namespaces that are assigned to a resource group show in the **Resource list** page of the IBM Cloud console.

```
ibmcloud cr namespace-add [-g (RESOURCE_GROUP_NAME | RESOURCE_GROUP_ID)] NAMESPACE
```

For more information about resource groups, see [Creating a resource group](#).

 **Tip:** If you have a problem when you try to add a namespace, see [Why can't I add a namespace?](#) for assistance.


Prerequisites

To find out about the required permissions, see [Platform management roles](#) and [Access roles for configuring Container Registry](#).

Command options

`NAMESPACE`

The namespace that you want to add. The namespace must be unique across all IBM Cloud accounts in the same region. Namespaces must have 4 - 30 characters, and contain lowercase letters, numbers, hyphens (-), and underscores (_) only. Namespaces must start and end with a letter or number.

 **Important:** Do not put personal information in your namespace names.

`-g (RESOURCE_GROUP_NAME | RESOURCE_GROUP_ID)`

(Optional) Specify the name or ID of the resource group to which you want to add the namespace. If you don't set this option, the targeted resource group is used. If you don't set this option and a resource group is not targeted, the default resource group for the account is used.

Example

Create a namespace with the name `birds` and add it to the resource group `beaks`.

```
$ ibmcloud cr namespace-add -g beaks birds
```

ibmcloud cr namespace-assign

Namespaces created in version 0.1.484 of the Container Registry CLI or earlier, or in the IBM Cloud console before 29 July 2020 are not assigned to *resource groups*. You can assign an unassigned namespace to a resource group for your IBM Cloud account. If you assign a namespace to a resource group, you can configure access to resources within the namespace at the [resource group](#) level. If you don't specify a resource group, and a resource group isn't targeted, the command fails.



Note: You can assign a namespace to a resource group only once. When a namespace is in a resource group, you can't move it to another resource group.



Tip: To find out which namespaces are assigned to resource groups and which are unassigned, run the [ibmcloud cr namespace-list](#) command with the `-v` option. Namespaces that are assigned to a resource group also show in the **Resource list** page of the IBM Cloud console.

```
ibmcloud cr namespace-assign -g (RESOURCE_GROUP_NAME | RESOURCE_GROUP_ID) NAMESPACE
```

For more information about resource groups, see [Creating a resource group](#).

Prerequisites

To find out about the required permissions, see [Platform management roles](#) and [Access roles for configuring Container Registry](#).

Command options

`-g (RESOURCE_GROUP_NAME | RESOURCE_GROUP_ID)`

(Optional) Specify the name or ID of the resource group to which you want to assign the namespace. If you don't set this option, the targeted resource group is used.

`NAMESPACE`

The namespace that you want to assign to a resource group.

Example

Assign a namespace with the name `birds` to the resource group `beaks`.

```
$ ibmcloud cr namespace-assign -g beaks birds
```

```
ibmcloud cr namespace-list (ibmcloud cr namespaces)
```

Displays all *namespaces* that are owned by your IBM Cloud account. You can use this command to list your namespaces so that you can verify which namespaces are assigned to *resource groups*, and which namespaces are unassigned. Namespaces that are assigned to a resource group also show in the **Resource list** page of the IBM Cloud console.

```
ibmcloud cr namespace-list [--verbose | -v] [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

`--verbose, -v`

(Optional) List all the namespaces and include information about the resource group and the creation date of the namespace.

`--output json, -o json`

(Optional) Outputs the list in JSON format.

Example

View a list of all your namespaces, including information about resource groups and creation dates.

```
$ ibmcloud cr namespace-list -v
```

```
ibmcloud cr namespace-rm
```

Removes a *namespace* from your IBM Cloud account. Images in this namespace are deleted when the namespace is removed.

```
ibmcloud cr namespace-rm NAMESPACE [--force | -f]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

NAMESPACE

The namespace that you want to remove.

--force, -f

(Optional) Force the command to run with no user prompts.

Example

Remove the namespace `birds`.

```
$ ibmcloud cr namespace-rm birds
```

```
ibmcloud cr plan
```

Displays your pricing plan for the registry region that you're targeting.

```
ibmcloud cr plan [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

--output json, -o json

(Optional) Outputs the list in JSON format.

```
ibmcloud cr plan-upgrade
```

Upgrades you to the standard plan for the registry region that you're targeting.

```
ibmcloud cr plan-upgrade [PLAN]
```

For more information about plans, see [Registry plans](#).

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

PLAN

(Optional) The name of the pricing plan that you want to upgrade to. If **PLAN** is not specified, the default is **standard**.

Example

Upgrade to the standard pricing plan.

```
$ ibmcloud cr plan-upgrade standard
```

ibmcloud cr platform-metrics

You can use the command to enable and disable platform metrics. You can also use it to find out whether you have platform metrics set up on your account for the registry region that you're targeting.



Important: If you want to view the platform metrics for IBM Cloud Container Registry, you must opt in by running the `ibmcloud cr platform-metrics` command.



Tip: You must specify one of the command options or the command fails with an error.

```
ibmcloud cr platform-metrics --enable | --disable | --status
```

For more information about the platform metrics that you can view in Container Registry, see [Monitoring metrics for IBM Cloud Container Registry](#).

Prerequisites

- You must set up IBM Cloud Monitoring, see [Getting started tutorial for IBM Cloud Monitoring](#).
- [Enable your IBM Cloud Monitoring instance for platform metrics](#).
- To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

--enable

(Optional) Enable the setting for your account.

--disable

(Optional) Disable the setting for your account.

--status

(Optional) Display whether the setting is enabled for your account.

Example

Enable platform metrics for your account.

```
$ ibmcloud cr platform-metrics --enable
```

ibmcloud cr private-only

Prevent image pulls or pushes over public network connections for your account for the registry region that you're targeting. You must specify one of the command options or the command fails with an error.

```
ibmcloud cr private-only --enable | --disable | --status
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

--enable

(Optional) Prevent image pulls or pushes over public network connections for your account.

--disable

(Optional) Reinstate image pulls or pushes over public network connections for your account.

--status

(Optional) Check whether the use of public connections is prevented for image pushes or pulls in your account.

Example

Prevent image pulls or pushes over public network connections for your account.

```
$ ibmcloud cr private-only --enable
```

ibmcloud cr quota

Displays your current quotas for traffic and storage, and usage information against those quotas for the registry region that you're targeting.

```
ibmcloud cr quota [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

--output json, -o json

(Optional) Outputs the list in JSON format.

ibmcloud cr quota-set

Modify the specified quota for the registry region that you're targeting.

```
ibmcloud cr quota-set [--traffic TRAFFIC] [--storage STORAGE]
```

Prerequisites

To find out about the required permissions, see [Access roles for configuring Container Registry](#).

Command options

--traffic TRAFFIC

(Optional) Changes your traffic quota to the specified value in megabytes. The operation fails if you are not authorized to set traffic, or if you set a value that exceeds your current pricing plan.

--storage STORAGE

(Optional) Changes your storage quota to the specified value in megabytes. The operation fails if you are not authorized to set storage quotas, or if you set a value that exceeds your current pricing plan.

Example

Set your quota limit for pull traffic to 7000 megabytes and storage to 600 megabytes.

```
$ ibmcloud cr quota-set --traffic 7000 --storage 600
```

ibmcloud cr region

Displays the targeted region and the registry.

```
ibmcloud cr region
```

For more information, see [Regions](#).

Prerequisites

None

```
ibmcloud cr region-set
```

Set a target region for the IBM Cloud Container Registry commands. To list the available regions, run the command with no options.

```
ibmcloud cr region-set [REGION]
```

Prerequisites

None

Command options

REGION

(Optional) The name of your target region, for example, **us-south**. For more information, see [Regions](#).

Example

Target the US South region.

```
$ ibmcloud cr region-set us-south
```

```
ibmcloud cr retention-policy-list
```

List the image retention policies for your account. Image retention policies retain the specified number of images for each repository within a namespace in IBM Cloud Container Registry. All other images in the namespace are deleted. You can also see whether the option to retain all untagged images applies to the policy.



Tip: Where an image within a repository is referenced by multiple tags, that image is counted only once. Newest images are retained. Age is determined by when the image was created, not when it was pushed to the registry.

```
ibmcloud cr retention-policy-list [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

```
--output json, -o json
```

(Optional) Outputs the list in JSON format.

Example

List the retention policies in your account.


```
$ ibmcloud cr retention-policy-list
```


For more information about how to use the `ibmcloud cr retention-policy-list` command, see [Retaining images](#).

ibmcloud cr retention-policy-set

Set a policy to retain the specified number of images for each repository within a namespace in IBM Cloud Container Registry. All other images in the namespace are deleted. When you set a policy it runs interactively, then it runs daily. You can set only one policy in each namespace.

You can choose whether to exclude all untagged images from the total number of images that you decide to retain.

 **Tip:** Where an image, within a repository, is referenced by multiple tags, that image is counted only once. Newest images are retained. Age is determined by when the image was created, not when it was pushed to the registry.

 **Tip:** If a retention policy deletes an image that you want to keep, you can restore the image. To identify the image, list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore the selected image by running the [ibmcloud cr image-restore](#) command.

If you want to cancel a retention policy, see [Update a retention policy to keep all your images](#).

```
ibmcloud cr retention-policy-set [--retain-untagged] --images IMAGECOUNT NAMESPACE
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

NAMESPACE

The namespace for which you want to create a policy.

--retain-untagged

(Optional) Retain all untagged images when the retention policy is being processed. Only tagged images are analyzed and, if the images don't meet the criteria, they are deleted. If the option isn't specified, all tagged and untagged images are analyzed and, if the images don't meet the criteria, they are deleted.

--images

Determines how many images to keep within each repository in the specified namespace. The newest images are retained. The age of images is determined by their build date. **IMAGECOUNT** is the number of images that you want to retain in each repository for the namespace. To return a policy to the default state that keeps all the images, set **IMAGECOUNT** to **All**.

Examples

Set a policy that retains the newest 20 images within each repository in the namespace `birds`.

```
$ ibmcloud cr retention-policy-set --images 20 birds
```

Set the policy back to the default state so that you keep all your images in the namespace `birds`.


```
$ ibmcloud cr retention-policy-set --images All birds
```


For more information about how to use the `ibmcloud cr retention-policy-set` command, see [Retaining images](#).

ibmcloud cr retention-run

Cleans up a namespace by retaining a specified number of images for each repository within a namespace in IBM Cloud Container Registry. All other images in the namespace are deleted.

You can choose whether to exclude all untagged images from the total number of images that you decide to retain.

 **Tip:** Where an image, within a repository, is referenced by multiple tags, that image is counted only once. Newest images are retained. Age is determined by when the image was created, not when it was pushed to the registry.

 **Tip:** If you want to restore a deleted image, you can list the contents of the trash by running the [ibmcloud cr trash-list](#) command and restore a selected image by running the [ibmcloud cr image-restore](#) command.



Tip: If an image that you're expecting to see doesn't show in the list that is produced, see [Why doesn't the retention command show all the images?](#) for assistance.



Deprecated: From 24 July 2023, the `--json` option is deprecated and is replaced by the `--output json` option.

```
ibmcloud cr retention-run [--force | -f [--output json | -o json]] [--retain-untagged] --images IMAGECOUNT NAMESPACE
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

NAMESPACE

The namespace that you want to clean up.

--force, -f

(Optional) Force the command to run with no user prompts.

--output json, -o json

(Optional) Outputs JSON that contains the results of cleaning your namespace. This option must be used with `--force`.

--retain-untagged

(Optional) Retain all untagged images when the retention policy is being processed. Only tagged images are analyzed and, if the images don't meet the criteria, they are deleted. If the option isn't specified, all tagged and untagged images are analyzed and, if the images don't meet the criteria, they are deleted.

--images

Determines how many images to keep within each repository in the specified namespace. The newest images are retained. The age of images is determined by their build date. **IMAGECOUNT** is the number of images that you want to retain in each repository for the namespace.

Example

Retain the newest 20 images within each repository, in the namespace `birds`.

```
$ ibmcloud cr retention-run --images 20 birds
```

For more information about how to use the `ibmcloud cr retention-run` command, see [Retaining images](#).

ibmcloud cr trash-list

Displays all images in the trash in your IBM Cloud account. You can also see the number of remaining days until the image is removed from the trash. The number of remaining days until removal is rounded up. For example, if the time until removal is 2 hours, it shows as 1 day. Images remain in the trash for 30 days after they are deleted from your live repository.

If you want to restore an image from the trash, run the [ibmcloud cr image-restore](#) command, see [Restoring images](#).



Deprecated: From 24 July 2023, the `--json` option is deprecated and is replaced by the `--output json` option.

```
ibmcloud cr trash-list [--restrict NAMESPACE] [--output json | -o json]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

--restrict NAMESPACE

(Optional) Limit the output to display only images in the specified namespace.

```
--output json, -o json
```

(Optional) Outputs JSON that contains the details of the contents of the trash.

Example

Display the images that are in the trash in the `birds` namespace.

```
$ ibmcloud cr trash-list --restrict birds
```

```
ibmcloud cr va-version
```

Find out which version of Vulnerability Advisor you're using.

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

```
ibmcloud cr va-version
```

Prerequisites

None.

```
ibmcloud cr va-version-set
```

Set the version of Vulnerability Advisor. You can set the version to either `v3`, the default, or `v4`. If you want to use version 4 temporarily, see [Setting the Vulnerability Advisor version](#).

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

```
ibmcloud cr va-version-set VERSION
```

Prerequisites

None.

Command options

VERSION

The version of Vulnerability Advisor that you want to use. Valid values are `v3` and `v4`.


Example

To set the Vulnerability version to version 4, run the following command:

```
$ ibmcloud cr va-version-set v4
```

```
ibmcloud cr vulnerability-assessment (ibmcloud cr va)
```

View a vulnerability assessment report for your images.

 **Note:** From version 1.0.0 of the container-registry plug-in, you can choose whether to run the `ibmcloud cr vulnerability-assessment` command in either version 3 (the default) or version 4 of Vulnerability Advisor. If you want to run Vulnerability Advisor in a different version, see [Setting the Vulnerability Advisor version](#). For more information about the versions of Vulnerability Advisor, see [About Vulnerability Advisor](#).

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

```
ibmcloud cr vulnerability-assessment [--extended | -e] [--vulnerabilities | -v] [--configuration-issues | -c] [-output FORMAT | -o FORMAT] IMAGE [IMAGE...]
```

Prerequisites

To find out about the required permissions, see [Access roles for using Container Registry](#).

Command options

IMAGE

The name of the image for which you want to get a report. The report states whether the image has any known package vulnerabilities. You can request reports for multiple images at the same time by listing each image in the command with a space between each name.

To find the names of your images, run `ibmcloud cr image-list`. Combine the content of the **Repository** column (`repository`) and **Tag** column (`tag`) separated by a colon (`:`) to create the image name in the format `repository:tag`. If a tag is not specified in the image name, the report assesses the image that is tagged `latest`.

For information about supported Docker base images, see [Vulnerable packages](#).

For more information, see [Managing image security with Vulnerability Advisor](#).

--extended, -e

(Optional) The command output shows additional information about fixes for vulnerable packages.

--vulnerabilities, -v

(Optional) The command output is restricted to show vulnerabilities only.

--configuration-issues, -c

(Optional) The command output is restricted to show configuration issues only.

--output FORMAT, -o FORMAT

(Optional) The command output is returned in the chosen format. The default format is `text`.

The following formats are supported:

- `text`
- `json`

Examples

View a standard vulnerability assessment report for your image.

```
$ ibmcloud cr vulnerability-assessment us.icr.io/birds/bluebird:1
```

View a vulnerability assessment report for your image `us.icr.io/birds/bluebird:1` in JSON format, showing vulnerabilities only.

```
$ ibmcloud cr vulnerability-assessment --vulnerabilities --output json us.icr.io/birds/bluebird:1
```

Formatting and filtering the CLI output

You can format and filter the IBM Cloud® Container Registry CLI output for supported IBM Cloud Container Registry commands.

By default, the CLI output is displayed in a human-readable format. However, this view might limit your ability to use the output, particularly if the command is run programmatically. For example, in the `ibmcloud cr image-list` CLI output, you might want to sort the `Size` field by numerical size, but the command returns a string description of the size. The `container-registry` CLI plug-in provides the format option that you can use to apply a Go template to the CLI output. The Go template is a feature of the [Go programming language](#) that you can use to customize the CLI output.

You can alter the CLI output by applying the format option in two different ways:

- Format data in your CLI output. For example, change the **Created** field output from UNIX® time to standard time.
- Filter data in your CLI output. For example, filter by details of the image to display a specific subset of images by using the Go template `if gt` condition.

You can use the format option with the following IBM Cloud Container Registry commands. Click a command to view a list of available fields and their data types.

- [ibmcloud cr image-digests](#) command
- [ibmcloud cr image-list](#) command
- [ibmcloud cr image-inspect](#) command

The following code examples demonstrate how you might use the formatting and filtering options.

- Run the following `ibmcloud cr image-digests` command to display all untagged images referenced by their digests.

```
$ ibmcloud cr image-digests --format '{{if not .Tags}}{{.Repository}}@{{.Digest}}{{end}}'
```

The following message is an example of the output from the command


```
example-<region>.icr.io/user1/my_first_repo@<digest1>
example-<region>.icr.io/user1/my_first_repo@<digest2>
example-<region>.icr.io/user1/my_first_repo@<digest3>
```

- Run the following `ibmcloud cr image-list` command to display repository, tag, and security status of all tagged images that have a size over 1 MB.

```
$ ibmcloud cr image-list --format "{{ if gt .Size 1000000 }}{{ .Repository }}:{{ .Tag }} {{
.SecurityStatus.Status }}"
```

The following message is an example of the output from the command:

```
example-<region>.icr.io/user1/my_first_repo:latest No Issues
example-<region>.icr.io/user1/my_second_repo:1 2 Issues
example-<region>.icr.io/user1/my_second_repo:test1 1 Issue
example-<region>.icr.io/user1/my_second_repo_2:test2 7 Issues
```

 **Tip:** If the listing images command times out, see [Why is it timing out when I list images?](#) for assistance.

- Run the following `ibmcloud cr image-inspect` command to display where IBM Documentation is hosted for a specified IBM public image.

```
$ ibmcloud cr image-inspect ibm_public_image --format "{{ .ContainerConfig.Labels }}"
```

The following message is an example of the output from the command:

```
map[doc.url:/docs/images/docker_image_ibm_public_image/ibm_public_image_starter.html]
```

- Run the following `ibmcloud cr image-inspect` command to display the exposed ports for a specified image.

```
$ ibmcloud cr image-inspect ibm_public_image --format "{{ .Config.ExposedPorts }}"
```

The following message is an example of the output from the command:

```
map[9080/tcp: 9443/tcp:]
```

Go template options for `ibmcloud cr image-digests`

Review the following table to find available Go template options and data types for the [ibmcloud cr image-digests](#) command.

Field	Type	Description
<code>Created</code>	Integer (64 bit)	Displays when the image was created, expressed in number of seconds in UNIX time.
<code>Digest</code>	String	Displays the unique identifier for an image.
<code>ManifestType</code>	String	Displays the image manifest type.
<code>Repository</code>	String	Displays the repository of the image.

SecurityStatus	Object	Displays the vulnerability status for the image. You can filter and format the following values: <ul style="list-style-type: none">Status stringIssueCount intExemptionCount int The possible statuses are described in Reviewing a vulnerability report by using the CLI .
Size	Integer (64 bit)	Displays the size of the image in bytes.
Tags	Array of strings	Displays the tags for the image.

Table 1. Available fields and data types in the Container Registry command to list image digests

Go template options for `ibmcloud cr image-list`

Review the following table to find available Go template options and data types for the `ibmcloud cr image-list` command.

Field	Type	Description
Created	Integer (64 bit)	Displays when the image was created, expressed in number of seconds in UNIX time.
Digest	String	Displays the unique identifier for an image.
ManifestType	String	Displays the image manifest type.
Namespace	String	Displays the namespace where the image is stored.
Repository	String	Displays the repository of the image.
SecurityStatus	Object	Displays the vulnerability status for the image. You can filter and format the following values: <ul style="list-style-type: none">Status stringIssueCount intExemptionCount int The possible statuses are described in Reviewing a vulnerability report by using the CLI .
Size	Integer (64 bit)	Displays the size of the image in bytes.
Tag	String	Displays the tag for the image.

Table 2. Available fields and data types in the Container Registry command to list images

Go template options for `ibmcloud cr image-inspect`

Review the following table to find available Go template options and data types for the `ibmcloud cr image-inspect` command.

Field	Type	Description
Architecture	String	Displays the processor architecture that was used to build this image, and that is required to run the image.
Author	String	Displays the author of the image.
Comment	String	Displays the description of the image.
Config	Object	Displays configuration metadata for the image. For more information, see Config field details .
Container	String	Displays the ID of the container that created the image.

ContainerConfig	Object	Displays the default configuration for containers that are started from this image. For more information, see Config field details .
Created	String	Displays the UNIX timestamp when the image was created.
DockerVersion	String	Displays the Docker version that was used to build this image.
ID	String	Displays the unique identifier for an image.
Os	String	Displays the operating system family that was used to build this image, and that is required to run the image.
OsVersion	String	Displays the version of the operating system that was used to build this image.
Parent	String	Displays the ID of the parent image that was used to build this image.
RootFS	Object	Displays metadata that describes the root file system for the image. For more information, see RootFS field details .
Size	Integer (64 bit)	Displays the size of the image in bytes.
VirtualSize	Integer (64 bit)	Displays the sum of the size of each layer in the image in bytes.

Table 3. Available fields and data types in the Container Registry command to inspect images

Config field details

Field	Type	Description
ArgsEscaped	Boolean	Displays true if the command is already escaped (Windows® specific).
AttachStderr	Boolean	Displays <i>true</i> if the standard error stream is attached to the container and <i>false</i> if not.
AttachStdin	Boolean	Displays <i>true</i> if the standard input stream is attached to the container and <i>false</i> if not.
AttachStdout	Boolean	Displays <i>true</i> if the standard output stream is attached to the container and <i>false</i> if not.
Cmd	Array of strings	Describes the commands and arguments that are passed to a container to run when the container is started.
Domainname	String	Displays the fully qualified domain name of the container.
Entrypoint	Array of strings	Describes the command that is run when the container starts.
Env	Array of strings	Displays the list of environment variables in the form of key-value pairs.
ExposedPorts	Key-value map	Displays the list of exposed ports in the format <code>[123: , 456:]</code> .
Healthcheck	Object	Describes how to check that the container is working correctly. For more information, see Healthcheck field details .
Hostname	String	Displays the hostname of the container.
Image	String	Displays the name of the image that was passed by the operator.

Labels	Key-value map	Displays the list of labels that were added to the image as key-value pairs.
MacAddress	String	Displays the MAC address that is assigned to the container.
NetworkDisabled	Boolean	Displays <i>true</i> if the networking is disabled for the container and <i>false</i> if the networking is enabled for the container.
OnBuild	Array of strings	Displays the ONBUILD metadata that was defined on the image Dockerfile.
OpenStdin	Boolean	Displays <i>true</i> if the standard input stream is opened and <i>false</i> if the standard input stream is closed.
Shell	Array of strings	Displays the shell-form of RUN, CMD, ENTRYPOINT .
StdinOnce	Boolean	Displays <i>true</i> if the standard input stream is closed after the attached client disconnects and <i>false</i> if the standard input stream stays open.
StopSignal	String	Describes the UNIX® stop signal to send when to stop the container.
StopTimeout	Integer	Displays the timeout in seconds to stop a container.
Tty	Boolean	Displays <i>true</i> if a pseudo-tty is allocated to the container and <i>false</i> if not.
User	String	Displays the user that runs commands inside the container where the image is used.
Volumes	Key-Value map	Displays the list of volume mounts that are mounted to a container.
WorkingDir	String	Displays the working directory inside the container where specified commands are run.

Table 4. Available fields and data types in Config

Healthcheck field details

Field	Type	Description
Interval	Integer (64 bit)	Displays the time to wait between two health checks in nanoseconds.
Retries	Integer	Displays the number of consecutive failures that are needed to consider a container as not working correctly.
Test	Array of strings	Displays how to run the health check test. The following options are available. <ul style="list-style-type: none"> <code>{}</code> inherit the health check. <code>"NONE"</code> the health check is disabled. <code>"CMD", args...</code> exec arguments directly. <code>"CMD-SHELL", command</code> run the command with the system's default shell.
Timeout	Integer (64 bit)	Displays the time to wait, in nanoseconds, before the health check fails.

Table 5. Available fields and data types in Healthcheck

RootFS field details

Option	Type	Description
BaseLayer	String	Displays the descriptor for the base layer in the image.
Layers	Array of strings	Displays the descriptors of each image layer.

Type	String	Displays the type of file system.
------	--------	-----------------------------------

Table 6. Available fields and data types in RootFS

Container Registry CLI change log

In this change log, you can learn about the most recent changes, improvements, and updates for the IBM Cloud® Container Registry CLI.

For more information about how to update the Container Registry CLI, see [Updating the container-registry CLI plug-in](#).

 **Deprecated:** Version 0.1 of the Container Registry CLI is deprecated, see [All releases of Container Registry plug-in 0.1 are deprecated](#) .

Version 1.2.2

Version 1.2.2 of the CLI was released on 24 July 2023.

This release has the following changes:

- Added a JSON output option, `--output json` or `-o json`, to several commands.
- The `--json` option is deprecated and is replaced with the `--output json` option.
- Updated translations.

For more information about the commands for which the JSON format option is available, see [JSON output option added to several IBM Cloud Container Registry commands](#).

Version 1.1.0

Version 1.1.0 of the CLI was released on 7 July 2023.

This release has the following changes:

- Fixes a Vulnerability Advisor versioning defect that affected some commands.

Version 1.0.11

Version 1.0.11 of the CLI was released on 19 June 2023.

This release has the following changes:

- The backup default Vulnerability Advisor version is now version 4.
- Vulnerability remediations.
- Updated translations.

Version 1.0.8

Version 1.0.8 of the CLI was released on 5 April 2023.

This release has the following changes:

- Vulnerability remediations.
- Updated translations.
- Updated error messages.

Version 0.1.587

Version 0.1.587 of the CLI was released on 26 January 2023.

This release has the following changes:

- Vulnerability remediations.

Version 1.0.6

Version 1.0.6 of the CLI was released on 25 January 2023.

This release has the following changes:

- Vulnerability remediations.

Version 1.0.5

Version 1.0.5 of the CLI was released on 5 December 2022.

This release has the following changes:

- Vulnerability remediations.
- Updated translations.
- You can install and uninstall the container-registry plug-in by using the `cr` alias.

Version 0.1.585

Version 0.1.585 of the CLI was released on 5 December 2022.

This release has the following changes:

- Vulnerability remediations.

Version 1.0.2

Version 1.0.2 of the CLI was released on 19 October 2022.

This release has the following changes:

- Minor bug fixes.
- Vulnerability remediations.
- Updated translations.

Version 0.1.584

Version 0.1.584 of the CLI was released on 19 October 2022.

This release has the following changes:

- Updated translations.

Version 1.0.1

Version 1.0.1 of the CLI was released on 23 September 2022.

This release has the following changes:

- Vulnerability remediations.
- Updated translations.

Version 0.1.583

Version 0.1.583 of the CLI was released on 23 September 2022.

This release has the following changes:

- Vulnerability remediations.
- Updated translations.

Version 1.0.0

Version 1.0.0 of the CLI was released on 15 September 2022.

This release has the following changes:

- Vulnerability Advisor v4 is now available, see [Vulnerability Advisor 4 is available from Container Registry plug-in 1.0.0](#).
- Image and digest list output no longer includes security status by default. You can either add the `--va` argument to include security status for all the listed images, or you can use the `ibmcloud cr va` command to query security status for an individual image.

Version 0.1.582

Version 0.1.582 of the CLI was released on 15 September 2022.

This release has the following changes:

- Vulnerability remediations.

High availability and disaster recovery

High availability for Container Registry

The IBM Cloud® Container Registry service is a highly available, regional, service.

High availability (HA) is a core discipline in an IT infrastructure to keep your apps up and running, even after a partial or full site failure. The main purpose of high availability is to eliminate potential points of failures in an IT infrastructure.

- In each supported region, traffic is load balanced across registry infrastructure in multiple availability zones, with no single point of failure.
- Data that is stored in IBM Cloud Container Registry is replicated over the availability zones and it is also backed up in another region regularly.
- If you're worried about the availability of your images if an entire region is unavailable, you can choose to push your images to multiple registry regions.

You might also choose to push your images to multiple registry regions in case you accidentally delete or overwrite your images.

For more information about regions, see [Regions](#).

For more information about service availability, see [Service Level Agreements](#).

Ownership of responsibilities

To find out more about responsibility ownership for using IBM Cloud products between IBM and the customer, see [Shared responsibilities for IBM Cloud products](#).

For more information about your responsibilities when you are using IBM Cloud Container Registry, see [Shared responsibilities for IBM Cloud Container Registry](#).

What level of availability do I need?

You can achieve high availability on different levels in your IT infrastructure and within different components of your cluster. The level of availability that is appropriate for you depends on several factors, such as your business requirements, the service level agreements (SLAs) that you have with your customers, and the resources that you want to expend.

What level of availability does IBM Cloud offer?

The level of availability that you set up for your cluster impacts your coverage under the IBM Cloud high availability service level agreement terms.

Service level objectives (SLO) describe the design points that the IBM Cloud services are engineered to meet. IBM Cloud Container Registry is designed to achieve the following availability target.

Availability target	Target Value
Availability %	99.999

Table 1. SLO for Container Registry

The SLO is not a warranty and IBM does not issue credits for failure to meet an objective. Refer to the SLAs for commitments and credits that are issued for failure to meet any committed SLAs. For a summary of all service level objectives, see [IBM Cloud service level objectives](#).

Locations for service availability

For more information about service availability within regions and data centers, see [Service and infrastructure availability by location](#).

Frequently asked questions about high availability

Review the following FAQs about high availability.

Does IBM Cloud replicate the service?

IBM doesn't make replicas of your data available in any region other than the region where you stored it. High availability is achieved by running the service in three data centers in each region.

Are users required to replicate the service?

You're not required to replicate your data into another region, but you can do it yourself by using tools such as [skopeo copy](#).

Business continuity and disaster recovery for Container Registry

Find out about the business continuity and disaster recovery strategy for IBM Cloud® Container Registry.

Disaster recovery involves a set of policies, tools, and procedures for returning a system, an application, or an entire data center to full operation after a catastrophic interruption. It includes procedures for copying and storing an installed system's essential data in a secure location, and for recovering that data to restore normalcy of operation.

Your responsibilities when you're using Container Registry

For more information about your responsibilities when you're using IBM Cloud Container Registry, see [Shared responsibilities for IBM Cloud Container Registry](#).

Disaster recovery strategy

IBM Cloud has *business continuity* plans in place to provide for the recovery of services within hours if a disaster occurs. You're responsible for your data backup and associated recovery of your content.

Container Registry provides mechanisms to protect your data and restore service functions. Business continuity plans are in place to achieve targeted *recovery point objective* (RPO) and *recovery time objective* (RTO) for the service. The following table outlines the targets for Container Registry.

Disaster recovery objective	Target Value
Recovery point objective (RPO)	48 hours
Recovery time objective (RTO)	24 hours

Table 1. RPO and RTO for Container Registry

Locations for service availability

For more information about service availability within regions and data centers, see [Service and infrastructure availability by location](#).

Frequently asked questions about disaster recovery

Review the following FAQs about disaster recovery.

Does the service replicate the data?

All customer data in IBM Cloud Container Registry is replicated and backed up. Backups include service and policy settings and image data, but not vulnerability results, which can be reconstructed. All data, including vulnerability results, is replicated within each region so that the loss of a single availability zone is tolerated transparently. Regular point-in-time backups are used by IBM to restore the content if the data is corrupted. Extra backups are created in other regions with compatible privacy policies that are used by IBM to restore the service in a disaster situation.

The following table shows the backup locations.

Environment	Active location	Backup location
ap-north	jp-tok	au-syd
ap-south	au-syd	jp-tok
br-sao	br-sao	us-south
ca-tor	ca-tor	us-east (service and policy settings) ca-mon (images)
eu-de	eu-de	eu-gb
eu-gb	eu-gb	eu-de
global	us-east	us-south
jp-osa	jp-osa	jp-tok

us-south	us-south	us-east
----------	----------	---------

Table 2. Backup locations

What data is backed up or replicated?

The image data, service settings, and policy settings are backed up by IBM.

Are users required to replicate the data?

You are not expected to replicate your images. However, you can create a service instance in another IBM Cloud Container Registry region. You can also choose from a range of tools, including pushing to multiple locations from your development pipeline, and the use of replication tools, such as [skopeo](#) [copy](#). IBM doesn't replicate service instances.

Your responsibilities when you are using Container Registry

Learn about the management responsibilities and terms and conditions that you have when you use IBM Cloud® Container Registry.

For a high-level view of the service types in IBM Cloud and the breakdown of responsibilities between the customer and IBM for each type, see [Shared responsibilities for IBM Cloud offerings](#).

Review the following sections for the specific responsibilities for you and for IBM when you use IBM Cloud Container Registry. For the overall terms of use, see [IBM Cloud Terms and Notices](#).

Incident and operations management

Incident and operations management includes tasks such as monitoring, event management, high availability, problem determination, recovery, and full state backup and recovery.

Task	IBM Responsibilities	Your Responsibilities
Data incidents.	It is the responsibility of IBM to inform you if your data is lost.	
Ensure that the application is available.	It is the responsibility of IBM to inform you if the application is not available.	
Track events.	It is the responsibility of IBM to ensure that IBM Cloud Activity Tracker is tracking events.	It is your responsibility to monitor events by using IBM Cloud Activity Tracker to ensure that your application is being accessed only by users with the correct authority. It is also your responsibility to ensure that an Activity Tracker instance is set up to receive events. For more information, see Auditing events for Container Registry .

Table 1. Responsibilities for incident and operations

Change management

Change management includes tasks such as deployment, configuration, upgrades, patching, configuration changes, and deletion.

Task	IBM Responsibilities	Your Responsibilities
Provisioning.	It is the responsibility of IBM to provision the service.	
Deprovisioning.	It is the responsibility of IBM to deprovision the service.	
Update package versions.		It is your responsibility to update package versions inside container images. You can use Vulnerability Advisor to identify the required updates. For more information, see Managing image security with Vulnerability Advisor .

Table 2. Responsibilities for change management

Identity and access management

Identity and access management includes tasks such as authentication, authorization, access control policies, and approving, granting, and revoking access.

Task	IBM Responsibilities	Your Responsibilities
Authentication	It is the responsibility of IBM to implement authentication.	

Process access policies	It is the responsibility of IBM to ensure that the policies are processed.	
Set up access policies		It is your responsibility to set up access policies. For more information, see Creating policies .
Access to back-end resources	It is the responsibility of IBM to access to back-end resources.	
Access to namespaces		It is your responsibility to set up access to namespaces. For more information, see Automating access to IBM Cloud Container Registry .

Table 3. Responsibilities for identity and access management

Security and regulation compliance

Security and regulation compliance includes tasks such as security controls implementation and compliance certification.

Task	IBM Responsibilities	Your Responsibilities
Secure confidential information.		It is your responsibility to make sure that no confidential information is put into your images.
Ensure that the service instance is secure.	It is the responsibility of IBM to ensure the security of the service instance.	

Table 4. Responsibilities for security and regulation compliance

Disaster recovery

Disaster recovery includes tasks such as providing dependencies on disaster recovery sites, provision disaster recovery environments, data and configuration backup, replicating data and configuration to the disaster recovery environment, and fail over on disaster events. For more information, see [High availability for Container Registry](#) and [Business continuity and disaster recovery for Container Registry](#).

Task	IBM Responsibilities	Your Responsibilities
Copy your data to another region.		It is your responsibility to copy the data to another region. For more information, see Regions .
Restore the contents of the data in a single region.	It is the responsibility of IBM to restore the contents of the data in a single region.	

Table 5. Responsibilities for disaster recovery

Troubleshooting Container Registry

Answers to common troubleshooting questions about how to use IBM Cloud® Container Registry.

Troubleshooting topics

The following troubleshooting topics are available to help you:

Troubleshooting CLI login

Troubleshoot logging in problems.

- [Why can't I log in to Container Registry?](#)
- [Why does the Container Registry login keep expiring?](#)
- [Why do commands fail saying that I'm not logged in?](#)
- [Why do `cr` commands fail saying they're not registered?](#)
- [Why is Docker login on my Mac failing?](#)

Troubleshooting pull and push errors

Troubleshoot pull and push problems.

- [Why can't I push or pull a Docker image?](#)
- [Why is pulling images so slow?](#)
- [Why am I getting **Authorization required** errors?](#)
 - [Why am I getting an **Unauthorized** error when I'm using Code Engine?](#)
- [Why am I getting **Access denied** errors?](#)
 - [Why am I getting **Access denied** errors for a resource?](#)
 - [Why am I getting **Access denied** errors about insufficient scope?](#)
 - [Why am I getting **Access denied** errors about my quota?](#)
 - [Why am I getting **Access denied** errors over a private network?](#)
 - [Why am I getting a **Forbidden** error when I'm using Code Engine?](#)
 - [Why do images fail to pull from registry with ImagePullBackOff or authorization errors?](#)

Troubleshooting CLI commands

Troubleshoot CLI command problems.

- [Why can't I add a namespace?](#)
- [When I create a namespace, why aren't I authorized to access the specified resource?](#)
- [Why can't I find my image or my namespace?](#)
- [Why don't all my namespaces show in the Resource list?](#)
- [Why is it timing out when I list images?](#)
- [Why can't I pull the newest image by using the **latest** tag?](#)
- [Why do all the tags get deleted when I delete an image?](#)
- [Why doesn't the retention command show all the images?](#)
- [Why do I get an error when I'm restoring an image?](#)
- [Why aren't all the tags restored when I restore by digest?](#)
- [Why do I get a manifest unknown error?](#)
- [Why do I get a manifest type error when I tag my image?](#)
- [Why do I get a manifest version error?](#)
- [Why do I get a manifest list invalid error?](#)

Troubleshooting networking

Troubleshoot networking problems.

- [Why can't I access the registry through a custom firewall?](#)
- [Why can't I connect to Container Registry?](#)

Troubleshooting Portieris

Troubleshoot Portieris problems.

- [Why don't my pods restart after my workers were down?](#)

Troubleshooting Container Registry

Answers to common troubleshooting questions about how to use IBM Cloud® Container Registry.

Troubleshooting topics

The following troubleshooting topics are available to help you:

Troubleshooting CLI login

Troubleshoot logging in problems.

- [Why can't I log in to Container Registry?](#)
- [Why does the Container Registry login keep expiring?](#)
- [Why do commands fail saying that I'm not logged in?](#)
- [Why do **cr** commands fail saying they're not registered?](#)
- [Why is Docker login on my Mac failing?](#)

Troubleshooting pull and push errors

Troubleshoot pull and push problems.

- [Why can't I push or pull a Docker image?](#)

- [Why is pulling images so slow?](#)
- [Why am I getting **Authorization required** errors?](#)
 - [Why am I getting an **Unauthorized** error when I'm using Code Engine?](#)
- [Why am I getting **Access denied** errors?](#)
 - [Why am I getting **Access denied** errors for a resource?](#)
 - [Why am I getting **Access denied** errors about insufficient scope?](#)
 - [Why am I getting **Access denied** errors about my quota?](#)
 - [Why am I getting **Access denied** errors over a private network?](#)
 - [Why am I getting a **Forbidden** error when I'm using Code Engine?](#)
 - [Why do images fail to pull from registry with ImagePullBackOff or authorization errors?](#)

Troubleshooting CLI commands

Troubleshoot CLI command problems.

- [Why can't I add a namespace?](#)
- [When I create a namespace, why aren't I authorized to access the specified resource?](#)
- [Why can't I find my image or my namespace?](#)
- [Why don't all my namespaces show in the Resource list?](#)
- [Why is it timing out when I list images?](#)
- [Why can't I pull the newest image by using the **latest** tag?](#)
- [Why do all the tags get deleted when I delete an image?](#)
- [Why doesn't the retention command show all the images?](#)
- [Why do I get an error when I'm restoring an image?](#)
- [Why aren't all the tags restored when I restore by digest?](#)
- [Why do I get a manifest unknown error?](#)
- [Why do I get a manifest type error when I tag my image?](#)
- [Why do I get a manifest version error?](#)
- [Why do I get a manifest list invalid error?](#)

Troubleshooting networking

Troubleshoot networking problems.

- [Why can't I access the registry through a custom firewall?](#)
- [Why can't I connect to Container Registry?](#)

Troubleshooting Portieris

Troubleshoot Portieris problems.

- [Why don't my pods restart after my workers were down?](#)

Troubleshooting CLI login

Why can't I log in to Container Registry?

Logging in to IBM Cloud® Container Registry fails.

What's happening

The [ibmcloud cr login](#) command fails.

Why it's happening

The following alternatives are possible causes:

- The **container-registry** CLI plug-in is out of date and needs updating.
- Docker is not installed on your local computer, or is not running.
- Your IBM Cloud login credentials expired.

How to fix it

You can fix this problem in the following ways:

- Upgrade to the most recent version of the **container-registry** CLI plug-in, see [Updating the container-registry CLI plug-in](#).
- Ensure that Docker is installed on your computer. If it is already installed, restart the Docker daemon.
- Rerun the **ibmcloud login** command to refresh your IBM Cloud login credentials.

Why does the Container Registry login keep expiring?

Logging in to IBM Cloud® Container Registry by using the `ibmcloud cr login` command is subject to IAM login session limits.

What's happening

Pushes and pulls to the registry fail after a period of inactivity. The following example is a typical scenario:

1. You log in by using the `ibmcloud cr login` command, and then you push or pull an image.
2. You don't run any other commands for a length of time that is longer than the IAM session inactivity limit.
3. Without logging in again, you try to push or pull an image and the command fails.

Why it's happening

When you log in by using the `ibmcloud cr login` command, you are subject to the IAM session limits, see [Setting limits for login sessions](#).

How to fix it

You can mitigate the issue by taking one of the following actions:

- Log in by using an IAM [API key](#) because this key is not subject to the IAM session expiry, see [Using Docker to authenticate with an API key](#).
- Change the IAM login session limits to better suit your needs, see [Setting limits for login sessions](#). The longer you set the limits, the longer your IBM Cloud Container Registry login lasts for.

Why do commands fail saying that I'm not logged in?

You are logged in to IBM Cloud but you can't run any commands in IBM Cloud® Container Registry. You get the following message: `FAILED You are not logged in to IBM Cloud`.

What's happening

All `ibmcloud cr` commands fail with the message: `FAILED You are not logged in to IBM Cloud`.

Why it's happening

The `container-registry` CLI plug-in is out of date and needs updating.

How to fix it

Upgrade to the most recent version of the `container-registry` CLI plug-in, see [Updating the container-registry CLI plug-in](#).

Why do commands fail saying they're not registered?

You can't run any IBM Cloud® Container Registry `ibmcloud cr` commands. The commands are failing with the message: `'cr' is not a registered command. See 'ibmcloud help'`.

What's happening

You're trying to run an `ibmcloud cr` command, but you receive an error message similar to one of the following error messages:

```
ibmcloud cr login
'cr' is not a registered command. See 'ibmcloud help'.
```

```
ibmcloud cr namespace
'cr' is not a registered command. See 'ibmcloud help'.
```

Why it's happening

The `container-registry` CLI plug-in is not installed.

How to fix it

Install the `container-registry` CLI plug-in, see [Installing the container-registry CLI plug-in](#).

Why is Docker login on my Mac failing?

When you are using IBM Cloud® Container Registry, Docker login fails on a Mac with the following message `Error saving credentials: error storing credentials - err: exit status 1, out: 'The user name or passphrase you entered is not correct'`.

What's happening

You receive the following error message when you try to run the [ibmcloud cr login](#) command on a Mac:

```
Error saving credentials: error storing credentials - err: exit status 1, out:
'The user name or passphrase you entered is not correct.'
```

Why it's happening

Docker for Mac has a problem that prevents your credentials from being stored in the macOS keychain.

How to fix it

You might be able to resolve the problem by restarting your Mac. If the restart doesn't solve the problem, you can disable the storage of logins in your Mac keychain:

1. In your menu, click the **Docker** icon, select **Preferences**.
2. Clear the **Securely store Docker logins in macOS keychain** checkbox.

Troubleshooting pull and push errors

Why can't I push or pull a Docker image?

When you're using IBM Cloud® Container Registry, pushing or pulling a Docker image fails. You might receive various messages, for example, about being over quota or invalid credentials.

What's happening

When you run commands to push or pull Docker images, you receive an error message. The error message varies depending on the root cause. The following error messages are potential error messages that you might receive:

```
You have exceeded your storage quota. Delete one or more images,
or review your storage quota and pricing plan
```

```
You have exceeded your pull traffic quota for the current month.
Review your pull traffic quota and pricing plan
```

```
unauthorized: authentication required
```

```
denied: requested access to the resource is denied
```

```
unauthorized: The login credentials are not valid, or your IBM Cloud account is not active.
```

Why it's happening

The following alternatives are possible causes:

- Docker is not installed.
- The Docker client is not logged in to IBM Cloud Container Registry.
- Your IBM Cloud *access token* expired.
- You exceeded the quota limit for storage or pull traffic that is set for your IBM Cloud account.
- You're on a free plan and you need to upgrade to a standard plan.

How to fix it

You can fix this problem in the following ways:

- [Ensure that Docker is installed on your computer](#).
- Check your Docker installation path.
- Log in to IBM Cloud by running **ibmcloud login**. Then, log in to the IBM Cloud Container Registry CLI by running [ibmcloud cr login](#).
- [Review quota limits and usage](#). For more information, see [Staying within quota limits](#).
- Upgrade to a standard plan. For more information, see [Upgrading your service plan](#).

Why is pulling images slow?

Pulling an image is slow from IBM Cloud® Container Registry.

What's happening

When you try to pull an image, it takes a long time.

Why it's happening

The following alternatives are possible causes:

- You are using a large image.
- You are pulling your image over a long distance.
- You have a poor connection.

How to fix it

You can fix this problem in the following ways:

- Make your image smaller.
- Store and pull your images in the same location.
- If you're using IBM Cloud Kubernetes Service or Red Hat® OpenShift® Container Platform, you can pull images over the private network to get a faster speed.

Why am I getting `Authorization required` errors?

You are trying to access IBM Cloud® Container Registry but are getting `Authorization required` errors.

What's happening

When you try to access Container Registry, you get one of the following messages.

- `Authorization required.`
- `You were not authorized to complete this operation.`
- `An error occurred when authenticating your request.`
- `Status code 401 Unauthorized` You might see this message if you are using IBM Cloud Code Engine. For more information, see [Why am I getting an Unauthorized error when I'm using Code Engine?](#)

Why it's happening

The following alternatives are possible causes:

- You attempted to log in to Container Registry with an invalid API key.
- You attempted to access Container Registry without logging in.
- A client attempted to access Container Registry without a bearer token.
- A client attempted to access Container Registry with an expired OAuth token.
- You attempted to authenticate against the Container Registry API with an invalid API key.
- You attempted to authenticate against the Container Registry API with an invalid Account ID.
- You logged in to a different region of Container Registry.

How to fix it

You can fix this problem in the following ways:

- Check the information about logging a client into Container Registry, see [Push images to your namespace](#) in [Getting started with Container Registry](#).
- Create and use a valid IAM API key to log a client, such as Docker, in to Container Registry with username `iamapikey` and the API key as your password. For more information, see [Managing user API keys](#).
- Use the `ibmcloud` CLI or IAM API to retrieve a valid OAuth token to authenticate against the Container Registry API. For more information, see [IBM Cloud Container Registry API - Authentication](#).
- When you authenticate against the Container Registry API, ensure that you use a valid Account ID.
- When you access Container Registry by using automation, set up a service ID and API key. For more information, see [Accessing Container Registry](#).
- If your image is in a different region of Container Registry, you must log in to IBM Cloud in that region by running the `ibmcloud cr region-set` and `ibmcloud cr login` commands. For more information, see [Targeting a local region](#).

Why am I getting an `Unauthorized` error when I'm using Code Engine?

You are trying to access IBM Cloud® Container Registry but are getting an `Unauthorized` error.

What's happening

When you try to access Container Registry, you get the following message. You might see this error message if you are using IBM Cloud Code Engine.

```
Status code 401 Unauthorized
```

Why it's happening

You are trying to access Container Registry by using IBM Cloud Code Engine and you don't have the correct credentials.

How to fix it

If you are accessing Container Registry through Code Engine, confirm that Code Engine is using a pull secret with a valid API key. For more information, see [Add registry access to Code Engine](#).

Why am I getting Access denied errors?

You have a valid IAM API key or OAuth token, but you still get **Access denied** errors in IBM Cloud® Container Registry.

When you try to access Container Registry, you get one of the following messages.

You are not authorized to access the specified resource.

For more information, see [Why am I getting Access denied errors for a resource](#).

Insufficient scope

You might see this message if you are trying to access Container Registry by using a client such as Docker. For more information, see [Why am I getting Access denied errors about insufficient scope](#).

Your account has exceeded its pull traffic quota for the current month.

For more information, see [Why am I getting Access denied errors about my quota](#).

Your account has exceeded its image storage quota for the current month.

For more information, see [Why am I getting Access denied errors about my quota](#).

You must access this account over the private network.

For more information, see [Why am I getting Access denied errors over a private network](#).

Status code 403 Forbidden

You might see this message if you are using IBM Cloud Code Engine. For more information, see [Why am I getting a Forbidden error when I'm using Code Engine](#).

ImagePullBackoff

You might see this status when you start containers in Kubernetes. For more information, see [Why do images fail to pull from registry with ImagePullBackOff or authorization errors](#).

Why am I getting Access denied errors for a resource?

You have a valid IAM API key or OAuth token, but you still get **Access denied** errors for a specific resource in IBM Cloud® Container Registry.

What's happening

When you try to access Container Registry, you get the following message:

You are not authorized to access the specified resource.

Why it's happening

The following alternatives are possible causes:

- The API key that is used to access Container Registry has insufficient permissions.
- Context-based restriction rules are in place.

How to fix it

You can fix this problem in the following ways:

- Confirm that the API key that you are using has suitable permissions for the resource that you are trying to access. Contact the owner of the resource for help. For more information, see [Managing IAM access](#).
- Check whether context-based restriction rules are in place. If so, these rules prevent you from accessing resources outside the defined allowed contexts. Adjust the allowed context or rerun your pull from within an allowed context. For more information, see [Context-based restrictions](#).



Tip: To confirm whether a context-based restriction rule caused the **Access denied** error, check IBM Cloud Activity Tracker for the resource that is being accessed. For more information, see [Monitoring context-based restrictions](#).

Why am I getting Access denied errors about insufficient scope?

You have a valid IAM API key or OAuth token, but you still get **Access denied** errors about insufficient scope in IBM Cloud® Container Registry.

What's happening

When you try to access Container Registry, you get the following message:

Insufficient scope

Why it's happening

You might see this message if you are trying to access Container Registry by using a client such as Docker. The following alternatives are possible causes:

- The API key that is used to access Container Registry has insufficient permissions.
- Context-based restriction rules are in place.

How to fix it

You can fix this problem in the following ways:

- Confirm that the API key that you are using has suitable permissions for the resource that you are trying to access. Contact the owner of the resource for help. For more information, see [Managing IAM access](#).
- Check whether context-based restriction rules are in place. If so, these rules prevent you from accessing resources outside the defined allowed contexts. Adjust the allowed context or rerun your pull from within an allowed context. For more information, see [Context-based restrictions](#).



Tip: To confirm whether a context-based restriction rule caused the **Access denied** error, check IBM Cloud Activity Tracker for the resource that is being accessed. For more information, see [Monitoring context-based restrictions](#).

Why am I getting Access denied errors about my quota?

You have a valid IAM API key or OAuth token, but you still get **Access denied** errors about your quota in IBM Cloud® Container Registry.

What's happening

When you try to access Container Registry, you get one of the following messages.

- **Your account has exceeded its pull traffic quota for the current month.**
- **Your account has exceeded its image storage quota for the current month.**

Why it's happening

You exceeded your image storage quota or pull traffic quota.

How to fix it

Review your quota limits and increase them as necessary. However, increasing your quota does increase your costs. For more information, see [Managing quota limits for storage and pull traffic](#).

Why am I getting Access denied errors over a private network?

You have a valid IAM API key or OAuth token, but you still get **Access denied** errors over a private network in IBM Cloud® Container Registry.

What's happening

When you try to access Container Registry, you get the following message:

You must access this account over the private network.

Why it's happening

You attempted to access Container Registry over the public network, but the account is set to allow only private network traffic.

How to fix it

Check the account settings for private only traffic. If private-only traffic is enabled, ensure that you are accessing Container Registry from the private network. For more information, see [Using private network connections](#).

Why am I getting a Forbidden error when I'm using Code Engine?

You are trying to access IBM Cloud® Container Registry but you are getting a **Forbidden** error.

What’s happening

When you try to access Container Registry, you get the following message. You might see this error message if you are using IBM Cloud Code Engine.

```
Status code 403 Forbidden
```

Why it’s happening

No token is sent, which means that you are trying to access a private namespace but you didn't send any credentials.

How to fix it

If you are accessing Container Registry through Code Engine, confirm that Code Engine is using a pull secret with a valid API key. For more information, see [Add registry access to Code Engine](#).

Why do images fail to pull from registry with `ImagePullBackOff` or authorization errors?

Virtual Private Cloud Classic infrastructure

What’s happening

When you deploy a workload that pulls an image from IBM Cloud Container Registry, your pods fail with an `ImagePullBackOff` status.

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
<pod_name>	0/1	ImagePullBackOff	0	2m

When you describe the pod, you see authentication errors similar to the following.

```
$ kubectl describe pod <pod_name>
```

```
Failed to pull image "<region>.icr.io/<namespace>/<image>:<tag>" ... unauthorized: authentication required
Failed to pull image "<region>.icr.io/<namespace>/<image>:<tag>" ... 401 Unauthorized
```

```
Failed to pull image "registry.ng.bluemix.net/<namespace>/<image>:<tag>" ... unauthorized: authentication required
Failed to pull image "registry.ng.bluemix.net/<namespace>/<image>:<tag>" ... 401 Unauthorized
```

Why it’s happening

Your cluster uses an API key that is stored in an [image pull secret](#) to authorize the cluster to pull images from IBM Cloud Container Registry.

By default, new clusters have image pull secrets that use API keys so that the cluster can pull images from any regional `icr.io` registry for containers that are deployed to the `default` Kubernetes namespace.

1. Verify that you use the correct name and tag of the image in your deployment YAML file.

```
$ ibmcloud cr images
```

2. Check your [pull traffic and storage quota](#). If the limit is reached, free up used storage or ask your registry administrator to increase the quota.

```
$ ibmcloud cr quota
```

3. Get the pod configuration file of a failing pod, and look for the `imagePullSecrets` section.

```
$ kubectl get pod <pod_name> -o yaml
```

Example output

```
...
imagePullSecrets:
- name: all-icr-io
...
```

4. If no image pull secrets are listed, set up the image pull secret in your namespace.

1. Verify that the `default` namespace has `icr-io` image pull secrets for each regional registry that you want to use. If no `icr-io` secrets are listed in the namespace, [use the `ibmcloud ks cluster pull-secret apply --cluster <cluster_name_or_ID>` command](#) to

create the image pull secrets in the `default` namespace.

```
$ kubectl get secrets -n default | grep "icr-io"
```

2. [Copy the `all-icr-io` image pull secret from the `default` Kubernetes namespace to the namespace where you want to deploy your workload.](#)
3. [Add the image pull secret to the service account for this Kubernetes namespace](#) so that all pods in the namespace can use the image pull secret credentials.
5. If image pull secrets are listed in the pod, determine what type of credentials you use to access IBM Cloud Container Registry.
 - If the secret has `icr` in the name, you use an API key to authenticate with the `icr.io` domain names. Continue with [Troubleshooting image pull secrets that use API keys](#).
 - If you have both types of secrets, then you use both authentication methods. Going forward, use the `icr.io` domain names in your deployment YAMLs for the container image. Continue with [Troubleshooting image pull secrets that use API keys](#).

Troubleshooting image pull secrets that use API keys

If your pod configuration has an image pull secret that uses an API key, check that the API key credentials are set up correctly.



Note: The following steps assume that the API key stores the credentials of a service ID. If you set up your image pull secret to use an API key of an individual user, you must verify that user's IBM Cloud IAM permissions and credentials.

1. Find the service ID that API key uses for the image pull secret by reviewing the **Description**. The service ID that is created with the cluster is named `cluster-<cluster_ID>` and is used in the `default` Kubernetes namespace. If you created another service ID such as to access a different Kubernetes namespace or to modify IBM Cloud IAM permissions, you customized the description.

```
$ ibmcloud iam service-ids
```

Example output

UUID	Name	Created At	Last Updated	Description
Locked				
ServiceId-aa11...	<service_ID_name>	2019-02-01T19:01+0000	2019-02-01T19:01+0000	ID for <cluster_name>
false				
ServiceId-bb22...	<service_ID_name>	2019-02-01T19:01+0000	2019-02-01T19:01+0000	Service ID for IBM
				Cloud Container Registry in Kubernetes cluster <cluster_name> namespace <namespace>
false				

2. Verify that the service ID is assigned at least an IBM Cloud IAM **Reader** [service access role policy for IBM Cloud Container Registry](#). If the service ID does not have the **Reader** service access role, [edit the IAM policies](#). If the policies are correct, continue with the next step to see if the credentials are valid.

```
$ ibmcloud iam service-policies <service_ID_name>
```

Example output

Policy ID:	a111a111-b22b-333c-d4dd-e555555555e5		
Roles:	Reader		
Resources:			
	Service Name	container-registry	
	Service Instance		
	Region		
	Resource Type	namespace	
	Resource	<registry_namespace>	

3. Check if the image pull secret credentials are valid.
 1. Get the image pull secret configuration. If the pod is not in the `default` namespace, include the `-n` option.

```
$ kubectl get secret <image_pull_secret_name> -o yaml [-n <namespace>]
```

2. In the output, copy the base64 encoded value of the `.dockerconfigjson` field.

```
apiVersion: v1
kind: Secret
data:
  .dockerconfigjson: eyJyZWdp...==
...
```

3. Decode the base64 string. For example, on OS X you can run the following command.

```
$ echo -n "<base64_string>" | base64 --decode
```

Example output

```
{"auths":{"<region>.icr.io":{"username":"iamapikey","password":"<password_string>","email":"<name@abc.com>","auth":"<auth_string>}}}
```

4. Compare the image pull secret regional registry domain name with the domain name that you specified in the container image. By default, new clusters have image pull secrets for each regional registry domain name for containers that run in the **default** Kubernetes namespace. However, if you modified the default settings or are using a different Kubernetes namespace, you might not have an image pull secret for the regional registry. [Copy an image pull secret](#) for the regional registry domain name.
5. Log in to the registry from your local machine by using the **username** and **password** from your image pull secret. If you can't log in, you might need to fix the service ID.

```
$ docker login -u iamapikey -p <password_string> <region>.icr.io
```

1. Re-create the cluster service ID, IBM Cloud IAM policies, API key, and image pull secrets for containers that run in the **default** Kubernetes namespace.

```
$ ibmcloud ks cluster pull-secret apply --cluster <cluster_name_or_ID>
```

2. Re-create your deployment in the **default** Kubernetes namespace. If you still see an authorization error message, repeat Steps 1-5 with the new image pull secrets. If you still can't log in, [open an IBM Cloud Support case](#).
6. If the login succeeds, pull an image locally. If the command fails with an **access denied** error, the registry account is in a different IBM Cloud account than the one your cluster is in. [Create an image pull secret to access images in the other account](#). If you can pull an image to your local machine, then your API key has correct permissions, but the API setup in your cluster is not correct.

```
$ docker pull <region>icr.io/<namespace>/<image>:<tag>
```

7. Check that the pull secret is either referenced directly from the deployment or from the service account that the deployment uses. If you still can't resolve the issue, [contact support](#).

Troubleshooting CLI commands

Why can't I add a namespace?

Setting up a namespace fails in IBM Cloud® Container Registry.

What's happening

When you run **ibmcloud cr namespace-add**, you're unable to set your entered value as the namespace.

Why it's happening

The following alternatives are possible causes:

- You entered a namespace value that is already being used by another IBM Cloud organization.
- A namespace was recently deleted and you're reusing its name. If the namespace that was deleted contained many resources, the deletion might not yet be fully processed by Container Registry.
- You used invalid characters in the namespace value.

How to fix it

You can fix this problem in the following ways:

- Follow any instructions that are in the returned error message.
- Check that you entered a valid namespace:
 - Your namespace must be unique across all IBM Cloud accounts in the same region.
 - Your namespace must be 4 - 30 characters long.
 - Your namespace must start and end with a letter or number.
 - Your namespace must contain lowercase letters, numbers, hyphens (-), and underscores (_) only.
- Choose a different value for your namespace.
- If you're re-creating a namespace that was deleted, and it contained many images, try again later.

When I create a namespace, why aren't I authorized to access the specified resource?

You aren't authorized to create a namespace in IBM Cloud® Container Registry.

What's happening

You try to create a namespace, but you receive the following error message:

```
You are not authorized to access the specified resource.
```

Why it's happening

You don't have the correct user permissions for working with namespaces. To add, assign, and remove namespaces, you must have the Manager role in the Container Registry service at the account level. If you have the Manager role on the resource group, or resource groups, it is not sufficient, the Manager role must be at the account level.

How to fix it

You must ensure that you are assigned the Manager role at the account level. For more information, see [User permissions for working with namespaces](#).

Why can't I find my image or my namespace?

Your image or your namespace is missing in IBM Cloud® Container Registry.

What's happening

When you try to find your image or your namespace, you can't find it.

Why it's happening

The following alternatives are possible causes:

- You are looking in the wrong region. Namespaces are region specific and you might be targeting the wrong region.
- The image or namespace was deleted.

How to fix it

You can fix this problem in the following ways:

- Check that you're using the correct region. To change the region, see [Regions](#).
- Check your instance of IBM Cloud Activity Tracker to see whether the image or namespace was deleted. For more information, see [Auditing events for Container Registry](#).

Why don't all my namespaces show in the Resource list?

My IBM Cloud® Container Registry namespaces don't all show up in the IBM Cloud **Resource list** page in the IBM Cloud console.

What's happening

When you view your namespaces in the IBM Cloud **Resource list** page, they don't all show up.

Why it's happening

Only namespaces that are assigned to *resource groups* show in the IBM Cloud **Resource list** page.

Namespaces created in version 0.1.485 of the Container Registry CLI or later, or in the IBM Cloud console on or after 29 July 2020, are created in the resource group that you specify. If you don't specify a resource group, and a resource group isn't targeted, the default resource group is used.

Namespaces created in version 0.1.484 of the Container Registry CLI or earlier, or in the IBM Cloud console before 29 July 2020, aren't assigned to resource groups.

How to fix it

If you want to see all your namespaces in the IBM Cloud **Resource list** page, you must assign each namespace to a resource group, see [Assigning existing namespaces to resource groups](#).

Why is it timing out when I list images?

Listing images times out in IBM Cloud® Container Registry.

What's happening

The request timed out while you attempted to list your images in the IBM Cloud console, CLI, or API.

Why it's happening

The most likely cause of the timeout is that the account has many images. The vulnerability reports can't be displayed because the targeted account contains more images than Vulnerability Advisor can process.

How to fix it

You can run the `ibmcloud cr image-list` command with the `--restrict` option to reduce the scope of the list and increase performance. Alternatively, if you don't want to fetch vulnerability reports, use the `--no-va` option. For help with managing the number of images, see [Cleaning up your namespaces](#).

Why can't I pull the newest image by using the latest tag?

You're unable to pull the most recent image by using the `latest` tag in IBM Cloud® Container Registry.

What's happening

You're trying to run the command `docker pull`, but it returned a version of your image that isn't the most recent version built.

Why it's happening

The `latest` tag is applied by default to reference an image when you run Docker commands without specifying the tag value. The `latest` tag is applied to the most recent `docker build` or `docker tag` command that was run without a tag value explicitly set. Therefore, it's possible to run `docker` commands out of order or to explicitly set tags on some images. Both scenarios cause the `latest` tag to refer to a build that isn't the most recent.

How to fix it

It is generally better to explicitly define a different sequential tag for your images every time, and not rely on the `latest` tag.

Why do all the tags get deleted when I delete an image?

You tried to delete an image from IBM Cloud® Container Registry by using the `ibmcloud cr image-rm` command and all the [tags](#) that referenced that image got deleted too.

What's happening

You deleted an image by using the `ibmcloud cr image-rm` command and all the tags within the same repository that referenced the image also got deleted.

Why it's happening

Where multiple tags exist for the same image [digest](#) within a repository, the `ibmcloud cr image-rm` command removes the underlying image and all its tags. If the same image exists in a different repository or namespace, that copy of the image is not removed.

How to fix it

If you want to remove a tag from an image, but leave the underlying image and any other tags, use the `ibmcloud cr image-untag` command. For more information, see [Removing tags from images in your private repository](#) and [Deleting images from your private repository](#).

Why doesn't the retention command show all the images?

An image doesn't show in the list that is produced by the IBM Cloud® Container Registry `ibmcloud cr retention-run` command.

What's happening

You ran the `ibmcloud cr retention-run` command and an image that you're expecting to view in the list is not displayed.

Why it's happening

You might have a [distroless](#) image. Some distroless images don't have a creation date. The `ibmcloud cr retention-run` command deletes the oldest images, and therefore requires a creation date.

How to fix it

You can delete the image manually by running the `ibmcloud cr image-rm` command, see [Deleting images from your private repository](#).

✓ **Tip:** To check the creation date of an image, you can run the `ibmcloud cr image-inspect` command. If the image doesn't have a creation date, the date is shown in the `ibmcloud cr image-inspect` output as `1970-01-01`, and the image is excluded from the results for `ibmcloud cr retention-run`.

Why do I get an error when I'm restoring an image?

You want to restore an image from the IBM Cloud® Container Registry trash, but you get a 409 error: `The tagged image already exists. You can restore this image by using the digest.`

What's happening

You receive the following error message when you run the `ibmcloud cr image-restore` command: `The tagged image already exists. You can restore this image by using the digest.`

Why it's happening

An image with the same name exists in your live repository. You can't overwrite a live image with an image that is in the trash.

How to fix it

Remove the `tag` from the existing image in your live repository by running the `ibmcloud cr image-untag` command. You can then restore the required image from the trash by running the `ibmcloud cr image-restore` command with the option `<repo>@<digest>`, which restores the digest and its tags to the live repository. For more information, see [Restoring images by digest](#).

Why aren't all the tags restored when I restore by digest?

You want to restore an image by [digest](#) from the IBM Cloud® Container Registry trash, but some [tags](#) weren't restored.

What's happening

You run the `ibmcloud cr image-restore` command, but the tags were not restored. If all the tags were unsuccessful, the digest shows in the live repository, but it is not tagged. You can see the digest if you run `ibmcloud cr image-digests`, but not if you run `ibmcloud cr image-list`.

Why it's happening

The tags that were not restored are already in your live repository. You can't overwrite a tag with a tag that is in the trash.

How to fix it

Remove the tag from the existing image in your live repository by running the `ibmcloud cr image-untag` command. You can then restore the required image from the trash by running the `ibmcloud cr image-restore` command with the option `<repo>@<digest>`, which restores the digest and its tags to the live repository. For more information, see [Restoring images by digest](#). Alternatively, you can run the `ibmcloud cr image-tag` command and use the restored digest as the source image.

✓ **Tip:** In your live repository, you can pull the image by digest. If you run the `ibmcloud cr image-digests` command, the image shows in the output.

Why do I get a manifest unknown error?

You get a `manifest unknown` error when you try to pull an image in IBM Cloud® Container Registry.

What's happening

You're trying to pull an image, but you receive the following manifest error message: `manifest unknown`

Why it's happening

The manifest does not exist in the registry.

How to fix it

To resolve the problem, try the following options:

- Check that the image name is correct.
- Check that you're pointing at the correct Container Registry region by running the `ibmcloud cr region` command.

Why do I get a manifest type error when I tag my image?

When you try to tag your image in IBM Cloud® Container Registry, you get a manifest type error: `The manifest type for this image is not supported for tagging.`

What's happening

You tried to tag your image, but you receive the following manifest error message: `The manifest type for this image is not supported for tagging.`

Why it's happening

The manifest type is not supported.

How to fix it

To resolve the problem, complete the following steps:

1. Pull the image that you tried to tag by running the following command, where `<source_image>` is your source image name:

```
$ docker pull <source_image>
```

2. Tag your local copy of the image that you pulled in the previous step by running the following command, where `<target_image>` is your new image name:

```
$ docker tag <source_image> <target_image>
```

3. Push the image that you tagged in the previous step by running the following command:

```
$ docker push <target_image>
```

Why do I get a manifest version error?

When you try to tag your image in IBM Cloud® Container Registry, you get a manifest version error: `The manifest version for this image is not supported for tagging.`

What's happening

You tried to tag your image, but you receive the following error message: `The manifest version for this image is not supported for tagging. To upgrade to a supported manifest version, pull and push this image by using Docker version 17.07 or later, then run the 'ibmcloud cr image-tag' command again.`

Why it's happening

The manifest version is not supported.

How to fix it

To resolve the problem, complete the following steps:

1. Upgrade to a supported version of Docker, see [Support for Docker](#).
2. Pull the image that you tried to tag by running the following command, where `<source_image>` is your source image name:

```
$ docker pull <source_image>
```

3. To upgrade the manifest version, push the image by running the following command:

```
$ docker push <source_image>
```

4. Tag the image by running the `ibmcloud cr image-tag` command, see [Creating new images that refer to a source image](#).

Why do I get a manifest list invalid error?

When you try to tag your manifest list or OCI Image Index in IBM Cloud® Container Registry, you get a manifest list invalid error.

What's happening

You get the following manifest list invalid error when you try to [tag](#) your [manifest](#) list or OCI Image Index in IBM Cloud Container Registry: `The manifest list or OCI index that you are tagging references an image that doesn't exist.`

Why it’s happening

Manifest lists and OCI Image Indexes contain a list of references to different images, where each image is for a different architecture. When you tag a manifest list or OCI Image Index, Container Registry tries to copy the referenced images. When you get this error message, it indicates that one of those referenced images was not found in the registry.

How to fix it

To understand how to fix this issue, you must work out what images are referenced in the manifest list or OCI Image Index by running the following command:

```
$ ibmcloud cr manifest-inspect <source_image>
```

The output of which is similar to this output:

```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
  "manifests": [
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 528,
      "digest": "sha256:865b0c35e6da393b8e80b7e3799f777572399a4cff047eb02a81fa6e7a48ed4b",
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    },
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 538,
      "digest": "sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc",
      "platform": {
        "architecture": "ppc64le",
        "os": "linux"
      }
    }
  ]
}
```

For this output to be valid, the images, with the [digests](#) from the previous output, must exist in the same repository as the manifest list or OCI index that you are trying to tag. To confirm whether one of the images is missing, you can run the following command. Replace `<src_repo>` with the name of your namespace and repository in the format `mynamespace/myrepo`.

```
$ ibmcloud cr digests --restrict <src_repo>
```

For example, if you run the following command:

```
$ ibmcloud cr digests --restrict mynamespace/myrepo
```

You get the following response:

Listing images...

Repository	Digest	Created	Size	Security status	Tags
icr.io/mynamespace/myrepo	sha256:865b0c35e6da393b8e80b7e3799f777572399a4cff047eb02a81fa6e7a48ed4b				-
Docker Image Manifest V2, Schema 2		4 days ago	1.8 MB	-	
icr.io/mynamespace/myrepo	sha256:a08e18417cec86f570be496b8bde1350dd986fc354d091b44d6a536570c26193				list
Docker Manifest List		-	433 B	-	

OK

In the previous example, you can see that only one image is in the same repository as the manifest list `sha256:865b0c35e6da393b8e80b7e3799f777572399a4cff047eb02a81fa6e7a48ed4b` and therefore, the manifest list is referencing another image `sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc` that does not currently exist in the repository.

You can resolve this issue by using one of the following options, following on from the previous example.

- If the missing image exists elsewhere in the registry, you can use the [ibmcloud cr image-tag](#) command to move the image to the same repository.
 1. To detect if the missing digest exists elsewhere in the registry, run the following command.

```
$ ibmcloud cr digests --format '{{if eq .Digest  
"sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc"}}{{.Repository}}@{{.Digest}}  
{{end}}}'  
icr.io/myrepo2/image2@sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc
```

2. If the previous command returns an image, you can copy it to the same repository as the manifest list.

```
$ ibmcloud cr image-tag  
icr.io/mynamespace/myrepo2@sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc  
icr.io/mynamespace/myrepo:ppc64le
```

- If the missing image was deleted in the last 30 days, you can restore it from the trash.

1. Detect if the image exists in trash by running the following command.

```
$ ibmcloud cr trash-list --restrict mynamespace
```

You receive the following response:

```
Listing the contents of the trash...  
  
Digest  
Days until expiry   Tags  
icr.io/mynamespace/myrepo@sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc   30  
ppc64le  
  
OK
```

2. If the image does exist in trash, you can restore it by running the following command.

```
$ ibmcloud cr image-restore  
icr.io/mynamespace/myrepo@sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc
```

You receive the following response:

```
Restoring digest  
'icr.io/mynamespace/myrepo@sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc' ...  
  
Successfully restored digest  
'icr.io/mynamespace/myrepo@sha256:1d71e323557502cc78ee6c237331a09b0c33ba59c14e5f683da3b1c6218779cc'  
Successfully restored tags: ppc64le  
  
OK
```

- If you have a local copy of the image, you can push it back to the registry. For more information, see [Pushing Docker images to your namespace](#).

Troubleshooting networking

Why can't I access the registry through a custom firewall?

Accessing IBM Cloud® Container Registry fails when you have a custom firewall.

What's happening

You set up an extra firewall in your development environment with custom settings for inbound and outbound network traffic. When you try to access IBM Cloud Container Registry, the connection fails.

Why it's happening

Your custom firewall requires certain network groups to be open for inbound and outbound network traffic to allow communication to and from the registry.

How to fix it

Allow your cluster to access infrastructure resources and services from behind a firewall, see [Allowing the cluster to access infrastructure resources and other services](#).

For INBOUND connectivity to your computer, allow incoming network traffic from the source network groups to the destination public IP address of your computer.

For OUTBOUND connectivity from your computer, use the same network groups and allow outgoing network traffic from the source public IP address of your computer to the network groups.

Why can't I connect to Container Registry?

You are unable to connect to IBM Cloud® Container Registry and you get the message `connection reset`.

What's happening

You can't connect to IBM Cloud Container Registry and you receive the following message: `connection reset`.

Why it's happening

The following alternatives are possible causes:

- Firewall or Calico rules are in place.
- The connection to Container Registry is through a proxy.
- IP allowlists are configured.
- Context-based restrictions are configured.

How to fix it

You can fix this problem in the following ways:

- Check your [firewall](#) or [Calico](#) rules.
- Connect to Container Registry without a proxy.
- Use IP addresses that are in your allowlist.
- Ensure that your [context-based restrictions](#) give you access to Container Registry.

Troubleshooting Portieris

Why don't my pods restart after my workers were down?

When you are using IBM Cloud® Container Registry, the `Pods` do not restart after your cluster workers were down.

What's happening

Portieris is deployed. The cluster workers are showing as working correctly, but nothing is scheduled.

Why it's happening

By default, Portieris adds a fail closed admission webhook. If all Portieris pods are down, the pods are not available to approve their own recovery.

How to fix it

To recover the cluster when it's in this state, you must change the webhook configuration to make it fail open instead of closed.

You must have sufficient role-based access control (RBAC) privileges to use the `GET` and `PATCH` verbs on the following resources:

- `admissionregistration.k8s.io/v1/MutatingWebhookConfiguration`
- `admissionregistration.k8s.io/v1/ValidatingWebhookConfiguration`

For more information about RBAC, see [Understanding RBAC permissions](#), [Creating custom RBAC permissions for users, groups, or service accounts](#), and [Kubernetes - Using RBAC Authorization](#).

To change the webhook configuration so that it fails open, and, when at least one Portieris pod is running, restore the webhook configuration so that it fails closed, complete the following steps:

1. Update `MutatingWebhookConfiguration` by running the following command.

```
$ kubectl edit MutatingWebhookConfiguration image-admission-config
```

Change `failurePolicy` to `Ignore`, save, and close.

2. Update `ValidatingWebhookConfiguration` by running the following command.

```
$ kubectl edit ValidatingWebhookConfiguration image-admission-config
```

Change `failurePolicy` to `Ignore`, save, and close.

3. Wait for some Portieris pods to start. If you want to check when the pods start, run the following command until you see the `STATUS` column for at

least one pod is displaying **Running**:

```
$ kubectl get po -n ibm-system -l app=ibmcloud-image-enforcement
```

4. When at least one Portieris pod is running, update **MutatingWebhookConfiguration** by running the following command.

```
$ kubectl edit MutatingWebhookConfiguration image-admission-config
```

Change **failurePolicy** to **Fail**, save, and close.

5. Update **ValidatingWebhookConfiguration** by running the following command.

```
$ kubectl edit ValidatingWebhookConfiguration image-admission-config
```

Change **failurePolicy** to **Fail**, save, and close.

Frequently asked questions about Container Registry and Vulnerability Advisor

Frequently asked questions (FAQs) about IBM Cloud® Container Registry and Vulnerability Advisor.

Frequently asked questions about Container Registry

How do you list public images?

To list public images, run the following `ibmcloud` commands to target the global registry and list the public images that are provided by IBM:

```
$ ibmcloud cr region-set global
```

```
$ ibmcloud cr images --include-ibm
```

What tools can I use to build and push images?

You can use Docker and non-Docker tools to build and push images to the registry. You can use non-Docker tools that support *OCI container image* format and protocol. To log in by using other clients, see [Accessing your namespaces interactively](#).

How many namespaces can you have?

You can have 100 registry *namespaces* in each region.

Can I rename a namespace?

You can't rename a *namespace*. If you want to change the name of the namespace, you must create a namespace with the new name and transfer its data. To transfer its data, you can copy the contents of the existing namespace into the namespace that you created.

If you don't want to transfer data manually, you can create a script for this action by using the `ibmcloud cr image-tag` command. For example, you can use the following script, where `<old_namespace>` is the existing namespace and `<new_namespace>` is the namespace that you created:

```
IMAGES=$(icr images --restrict <old_namespace> --format "{{ .Repository }}:{{ .Tag }}")

for i in $IMAGES ; do
    new=$(echo $i | sed "s|/<old_namespace>/|/<new_namespace>/|1")
    ibmcloud cr image-tag $i $new
done
```

Do images in the trash count toward my quota?

Images that are in the trash don't count toward your quota.

How do I find the image digest?

You can find the long format of the image *digest* by running one of the following commands. The digest is displayed in the **Digest** column of the CLI.



Note: When you are using the digest to identify an image, always use the long format.

- Run the `ibmcloud cr image-digests` command:

```
$ ibmcloud cr image-digests
```

- Run the `ibmcloud cr image-list` command:

```
$ ibmcloud cr image-list --no-trunc
```



Note: If you run the `ibmcloud cr image-list` command without the `--no-trunc` option, you see the truncated format of the digest.

How do I use digests to work with images?

The *digest* identifies an image by using the `sha256` hash of the *image manifest*.

To find the digests for your images, run the `ibmcloud cr image-digests` command. You can refer to an image by using a combination of the content

of the **Repository** column (`repository`) and the **Digest** column (`digest`) separated by an at (`@`) symbol to create the image name in the format `repository@digest`.

How do you use access control?

You can create IBM Cloud Identity and Access Management (IAM) policies to control access to your namespaces in IBM Cloud Container Registry. For more information, see [Granting access to IBM Cloud Container Registry resources tutorial](#) and [Managing IAM access for Container Registry](#).

How can I share an image with many users?

You can create an IBM Cloud account and invite all the users to it. They can then all have access to any *namespace* that is created in the account. You can create a subset of the users and set an IAM access policy to differentiate access at the namespace level. Users can be members of many accounts, but you can't give access outside the account, that is, you can't share a namespace to multiple accounts.

For more information, see [Defining IAM access policies](#).

Do I have any untagged images?

To find out whether you have any [untagged](#) images, list your images by running the `ibmcloud cr image-digests` command. Untagged images have a hyphen (-) in the **Tags** column.

Do I need untagged images?

If you have active containers that are running [untagged](#) images, you must retain the untagged images. If you delete untagged images that are in use, you can cause problems with scaling or automated restarts. Deleting untagged images might cause a problem in the following circumstances:

- The image was deployed by referencing the image by using the digest.
- The image reference was mutated by a webhook service, such as [Portieris](#).

What are eligible images?

If you are cleaning up images by using retention policies, only eligible images are cleaned up. Images that are always retained are distroless images that do not set a created time, such as Google distroless images and manifest lists. Images that are always retained are not eligible images.

The images that are not eligible are still displayed, but they do not count toward the total number of images that is set in the retention policy and are not removed.

What regions are available?

To find out about the regions that are available for IBM Cloud Container Registry, see [Regions](#).

Frequently asked questions about Vulnerability Advisor

How much does Vulnerability Advisor cost?

The cost of Vulnerability Advisor is built into the pricing for IBM Cloud Container Registry. For more information, see [Billing for storage and pull traffic](#).

Can images from other registries be scanned?

Vulnerability Advisor scans images from IBM Cloud Container Registry only.

How is a Vulnerability Advisor scan triggered?

For more information about how the scanning of an image is triggered, see [Vulnerable packages](#).

Why doesn't a new image scan?

If you get the vulnerability report immediately after you add the image to the *registry*, you might receive the following error:

```
BXNVA0009E: <imagename> has not been scanned. Try again later.  
If this issue persists, contact support for help;  
see https://cloud.ibm.com/docs/get-support?topic=get-support-getting-customer-support#getting-customer-support
```

You receive this message because the images are scanned asynchronously to the requests for results, and the scanning process takes a while to complete. During normal operation, the scan completes within the first few minutes after you add the image to the registry. The time that it takes to complete depends on variables like the image size and the amount of traffic that the registry is receiving.

If you get this message as part of a build pipeline and you see this error regularly, try adding some retry logic that contains a short pause.

If you still see unacceptable performance, contact support, see [Getting help and support for Container Registry](#).

How often are the security notices updated?

Security notices for Vulnerability Advisor are loaded from the vendors' operating system sites approximately every 12 hours.

Why can I see a vulnerability in Vulnerability Advisor v4 but not in v3?

Some vulnerabilities are picked up earlier by Vulnerability Advisor version 4 than by version 3 because version 4 uses a different architecture and a different scanning engine.

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

Which version of a package is installed in my image?

To determine the version of a package that is installed in your image, use the relevant package manager command for your operating system.

Alpine package manager commands

On Alpine, to determine the version of a package that is installed in your image, you can use the following commands, where `<package_name>` is the name of your package.

- To list the metadata for a specific installed package, run the following command:

```
$ apk info <package_name>
```

- To list all installed packages and their versions, run the following command:

```
$ apk list
```

Debian and Ubuntu package manager commands

On Debian and Ubuntu, to determine the version of a package that is installed in your image, you can use the following commands, where `<package_name>` is the name of your package.

- To list the metadata for a specific installed package, run either of the following commands:

```
$ apt show <package_name>
```

```
$ dpkg-query -l <package_name>
```

- To list all installed packages and their versions, run either of the following commands:

```
$ apt list
```

```
$ dpkg-query -W
```

Red Hat and CentOS package manager commands

On Red Hat® OpenShift® and CentOS, to determine the version of a package that is installed in your image, you can use the following commands, where `<package_name>` is the name of your package.

- To list the metadata for a specific installed package, run either of the following commands:

```
$ rpm -qi <package_name>
```

```
$ yum info <package_name>
```

- To list all installed packages and their versions, run either of the following commands:

```
$ rpm -qa
```

```
$ yum list installed
```

Does Vulnerability Advisor have versions?

Vulnerability Advisor is available in two versions, version 3 and version 4. For more information, see [Managing image security with Vulnerability Advisor](#).

⊖ **Deprecated:** Vulnerability Advisor version 3 is deprecated from 19 June 2023. For more information about how to update to version 4, see [Update Vulnerability Advisor to version 4 by 19 June 2023](#).

Getting help and support for Container Registry

If you experience an issue or have questions when you are using IBM Cloud® Container Registry, you can use the following resources before you open a support case.

- Review the [FAQs](#) in the product documentation.
- Review the [troubleshooting documentation](#) to troubleshoot and resolve common issues.
- Check the status of the IBM Cloud platform and resources by going to the [Status page](#).

If you still can't resolve the problem, you can open a support case. For more information, see [Creating support cases](#). And, if you're looking to provide feedback, see [Submitting feedback](#).

Site map

Getting started

[Getting started](#)

- [Before you begin](#)
- [Install the Container Registry CLI](#)
- [Set up a namespace](#)
- [Pull images from a registry to your local computer](#)
- [Tag the image](#)
- [Push images to your namespace](#)
- [Verify that the image was pushed](#)
- [Next steps in Container Registry](#)

About Container Registry

[About Container Registry](#)

- [Service plans](#)
- [Quota limits and billing](#)
 - [Billing for storage and pull traffic](#)
 - [Storage charges](#)
 - [Pull traffic charges](#)
 - [Quota limits for storage and pull traffic](#)
 - [Storage quota limits](#)
 - [Pull traffic quota limits](#)
 - [Cost of Container Registry](#)
- [Upgrading your service plan](#)
- [Terms that are used in IBM Cloud Container Registry](#)
 - [Container image](#)
 - [Digest](#)
 - [Dockerfile](#)
 - [Docker V2 container images](#)
 - [Domain name](#)
 - [Image manifest](#)
 - [OCI container images](#)
 - [Registry](#)
 - [Registry namespace](#)
 - [Repository](#)
 - [Tag](#)
 - [Untagged image](#)
- [Regions](#)
 - [Global registry](#)
 - [Targeting the global registry](#)

- [Local regions](#)
 - [Targeting a local region](#)
- [Supported clients](#)
 - [Support for Docker](#)
 - [Support for other clients](#)

Container Registry architecture and workload

[Container Registry architecture and workload](#)

- [Segmentation of data](#)
- [Private connections](#)

Public IBM images

[Public IBM images](#)

- [Accessing the public IBM images by using the CLI](#)

What are containers?

[What are containers?](#)

Notifications

[Notifications](#)

- [Notifications topics](#)

[Upcoming public networking changes for Container Registry](#)

[Update Vulnerability Advisor to version 4 by 19 June 2023](#)

- [What you need to know about this change](#)
- [What actions you must take by 19 June 2023](#)

[Changes to Container Registry VPE gateways from 11 November 2022](#)

- [What you need to know about this change](#)
- [How you benefit from this change](#)
- [Understanding if you are impacted by this change](#)
- [What actions you must take](#)

[Changes to private IP addresses from 15 December 2022](#)

- [What you need to know about this change](#)
- [What actions you need to take](#)

[Container Registry CLI stops returning security status results in lists by default from version 1.0.0](#)

- [Action required now](#)

[Container Registry private IP addresses changed on 5 July 2022](#)

- [What you need to know about this change](#)
- [How you benefit from this change](#)
- [Understanding if you are impacted by this change](#)
- [What actions you need to take](#)

[IAM access policies are required from 5 July 2022](#)

- [What are the changes?](#)

- [Check whether these changes affect you](#)
- [Prepare for the changes](#)
- [What if I did not implement the changes in time?](#)

[Minimum supported Docker version from 1 March 2022](#)

Release notes

[Release notes](#)

- [24 July 2023](#)
 - [JSON output option added to several IBM Cloud Container Registry commands](#)
 - [The `--json` option for IBM Cloud Container Registry commands is deprecated](#)
- [19 June 2023](#)
 - [Vulnerability Advisor version 3 is deprecated from 19 June 2023](#)
- [19 May 2023](#)
 - [Update Vulnerability Advisor to version 4 by 19 June 2023](#)
- [26 April 2023](#)
 - [Using Portieris to block the deployment of images with issues is deprecated.](#)
- [11 November 2022](#)
 - [Change to virtual private endpoints](#)
- [2 November 2022](#)
 - [Changes to private IP addresses from 15 December 2022](#)
- [15 September 2022](#)
 - [Container Registry plug-in 1.0.0 is available](#)
 - [All releases of Container Registry plug-in 0.1 are deprecated](#)
 - [Vulnerability Advisor 4 is available from Container Registry plug-in 1.0.0](#)
 - [New commands for setting and checking the Vulnerability Advisor version are available from Container Registry plug-in 1.0.0](#)
- [3 August 2022](#)
 - [The CLI stops returning security status results in lists by default from version 1.0.0](#)
- [8 July 2022](#)
 - [Context-based restrictions](#)
- [5 July 2022](#)
 - [Change to Container Registry private IP addresses in all regions](#)
 - [All accounts require IAM access policies](#)
- [23 June 2022](#)
 - [Change to Container Registry private IP addresses in the following regions only: `br-sao`, `ca-tor`](#)
- [20 April 2022](#)
 - [Container Registry private IP addresses are changing from 23 June 2022](#)
- [1 March 2022](#)
 - [Amendment to the minimum supported Docker version for Container Registry](#)
- [9 February 2022](#)
 - [All accounts will require IAM access policies from 5 July 2022](#)
- [2 February 2022](#)
 - [Replication of exemption policies between IBM regions is discontinued](#)

- [1 February 2022](#)
 - [Storage that is used by untagged images is being charged for](#)
- [17 January 2022](#)
 - [View IBM Cloud Activity Tracker events for Red Hat signing](#)
- [7 December 2021](#)
 - [Define configuration rules for Container Registry](#)
- [1 November 2021](#)
 - [Using Notary v1 for signing images is discontinued](#)
- [5 October 2021](#)
 - [Container Registry container builds are discontinued](#)
- [2 September 2021](#)
 - [Namespaces that are assigned to a resource group show in the Resource list page](#)
- [30 August 2021](#)
 - [New region in Brazil](#)
- [19 August 2021](#)
 - [Using registry tokens is discontinued](#)
- [9 August 2021](#)
 - [The `ibmcloud cr login` command logs you into the `<region>.icr.io` registry domain only](#)
- [8 July 2021](#)
 - [Using Notary v1 for signing images is deprecated](#)
- [21 June 2021](#)
 - [Global registry](#)
- [10 May 2021](#)
 - [New region in Canada](#)
- [4 May 2021](#)
 - [The `ibmcloud cr ppa-archive-load` command is discontinued](#)
- [18 February 2021](#)
 - [Increase the performance of the `ibmcloud cr image-list` and `ibmcloud cr image-digests` commands by using the `--no-va` option](#)
- [15 February 2021](#)
 - [New region in Japan](#)
- [19 November 2020](#)
 - [You can use Portieris to enforce container image security](#)
- [21 October 2020](#)
 - [Find out about the usage on your account by using platform metrics](#)
- [6 October 2020](#)
 - [Container Registry container builds are deprecated](#)
- [27 August 2020](#)
 - [Setting exemption policies by digest](#)
- [12 August 2020](#)
 - [Using UAA tokens is discontinued](#)
- [30 July 2020](#)
 - [New access roles are required for Vulnerability Advisor exemption policies](#)

- [29 July 2020](#)
 - [You can set permissions so that access to resources within a namespace can be configured at the resource group level](#)
- [13 July 2020](#)
 - [To work with namespaces, you must have the Manager role at the account level](#)
- [24 June 2020](#)
 - [Restoring all tags for a digest in a repository is now an option](#)
- [18 May 2020](#)
 - [Retaining untagged images is now an option when you clean up your namespaces](#)
- [30 April 2020](#)
 - [ibmcloud cr image-prune-untagged](#) command is available
- [16 April 2020](#)
 - [You can use private network connections to securely route your data in Container Registry](#)
 - [ibmcloud cr private-only](#) command is available
- [3 February 2020](#)
 - [Using Container Registry tokens is deprecated](#)
- [31 January 2020](#)
 - [ibmcloud cr image-digests](#) command is available
- [4 December 2019](#)
 - [Support for Red Hat signatures is available](#)
- [25 October 2019](#)
 - [IBM Log Analysis platform services logs are available](#)
- [14 October 2019](#)
 - [ibmcloud cr manifest-inspect](#) command is available
- [23 September 2019](#)
 - [You can create retention policies for your images](#)
 - [You can restore deleted images from the trash](#)
- [1 August 2019](#)
 - [ibmcloud cr retention-run](#) command is available
- [25 July 2019](#)
 - [IBM Cloud Activity Tracker available for Container Registry](#)
- [1 July 2019](#)
 - [ibmcloud cr token-add](#) command is no longer available
- [27 June 2019](#)
 - [Container Scanner is no longer available](#)
- [13 June 2019](#)
 - [Remove tags from images](#)
- [13 May 2019](#)
 - [End of support for Container Scanner](#)
- [2 April 2019](#)
 - [General Availability of Container Image Security Enforcement](#)
- [14 March 2019](#)
 - [IBM Cloud Activity Tracker available for Container Registry](#)

- [25 February 2019](#)
 - [New domain names](#)
 - [Adding IAM API key policies to control access to resources](#)
 - [New region in ap-north](#)
- [21 February 2019](#)
 - [Automating access to your namespaces](#)
- [8 January 2019](#)
 - [End of support for Vulnerability Advisor API version 2](#)
- [4 October 2018](#)
 - [Managing user access](#)
- [7 August 2018](#)
 - [Exemption policies available in Vulnerability Advisor](#)
- [25 July 2018](#)
 - [IBM Cloud Activity Tracker available for Vulnerability Advisor](#)
- [12 July 2018](#)
 - [Vulnerability Advisor API version 3](#)
- [31 May 2018](#)
 - [Use Helm for Passport Advantage images](#)
- [21 March 2018](#)
 - [Container Scanner](#)
- [16 March 2018](#)
 - [Container Image Security Enforcement beta](#)
- [20 February 2018](#)
 - [Trusted content](#)
- [6 November 2017](#)
 - [Global registry](#)
- [29 September 2017](#)
 - [Build Docker images](#)
- [24 August 2017](#)
 - [Graphical user interface released](#)
- [27 June 2017](#)
 - [Introducing IBM Cloud Container Registry](#)

Container Registry and Vulnerability Advisor workflow tutorial

[Container Registry and Vulnerability Advisor workflow tutorial](#)

- [Objectives](#)
- [Services used](#)
- [Before you begin](#)
- [From code to a running container](#)
 - [Create a namespace](#)
 - [Build and push an image](#)
 - [Deploy a container that uses your image](#)

- [Secure your images and clusters](#)
 - [View the vulnerability report for your image](#)
 - [Enforce security in your cluster](#)
 - [Resolve vulnerabilities in your image](#)
- [Deploying to nondefault Kubernetes namespaces](#)

Granting access to Container Registry resources tutorial

[Granting access to Container Registry resources tutorial](#)

- [Before you begin](#)
- [Authorize a user to configure the registry](#)
- [Authorize a user to access specific namespaces](#)
- [Create a service ID and grant access to a resource](#)
- [Cleaning up your account](#)

Solution tutorials

[Moving a VM based app to Kubernetes](#)

- [Objectives](#)
- [Architecture](#)
 - [Traditional app architecture with VMs](#)
 - [Containerized architecture](#)
 - [VMs, containers, and Kubernetes](#)
 - [Virtual machines vs containers](#)
 - [Kubernetes orchestration](#)
 - [What IBM's doing for you](#)
- [Sizing clusters](#)
- [Decide what Database option to use](#)
- [Decide where to store application files](#)
 - [Non-persistent data storage](#)
 - [Learn how to create persistent data storage for your app](#)
 - [Learn how to move existing data to persistent storage](#)
 - [Set up backups for persistent storage](#)
- [Prepare your code](#)
 - [Apply the 12-factor principles](#)
 - [Store credentials in Kubernetes secrets](#)
- [Containerize your app](#)
- [Deploy your app to a Kubernetes cluster](#)
 - [Learn how to create a Kubernetes deployment yaml file](#)
- [Summary](#)
- [Put everything learned to practice, run the JPetStore app in your cluster](#)
- [Related Content](#)

[Resilient and secure multi-region Kubernetes clusters with IBM Cloud Internet Services](#)

- [Objectives](#)
- [Before you begin](#)
- [Deploy an application to one location](#)
 - [Create a Kubernetes cluster](#)
 - [Deploy the application to the Kubernetes cluster](#)
 - [Get the Ingress Subdomain assigned to the cluster](#)
 - [Configure the Ingress for your DNS subdomain](#)
- [And then to another location](#)
- [Configure multi-location load-balancing](#)
 - [Register a custom domain with IBM Cloud Internet Services](#)
 - [Configure Health Check for the Global Load Balancer](#)
 - [Define Origin Pools](#)
 - [One pool for the cluster in Dallas](#)
 - [One pool for the cluster in London](#)
 - [And one pool with both clusters](#)
 - [Create the Global Load Balancer](#)
- [Secure the application](#)
 - [Turn the Web Application Firewall On](#)
 - [Increase performance and protect from Denial of Service attacks](#)
- [Remove resources](#)
 - [Remove Kubernetes Cluster resources](#)
 - [Remove CIS resources](#)
- [Related content](#)

[Continuous Deployment to Kubernetes](#)

- [Objectives](#)
- [Before you begin](#)
- [Create development Kubernetes cluster](#)
- [Create a sample application](#)
- [Modify the application and deploy the updates](#)
- [Deploy to a production environment](#)
- [Setup Slack notifications](#)
- [Remove resources](#)
- [Expand the Tutorial](#)
- [Related Content](#)

Setting up the Container Registry CLI and namespace

[Setting up the Container Registry CLI and namespace](#)

- [Installing the `container-registry` CLI plug-in](#)
- [Updating the `container-registry` CLI plug-in](#)
 - [Updating `container-registry` CLI plug-in version 1.0](#)
 - [Updating `container-registry` CLI plug-in version 0.1](#)

- [Uninstalling the `container-registry` CLI plug-in](#)
- [Planning namespaces](#)
 - [User permissions for working with namespaces](#)
- [Setting up a namespace](#)
- [Assigning existing namespaces to resource groups](#)
- [Removing namespaces](#)

Setting up Container Registry as a private registry on Red Hat OpenShift

[Setting up Container Registry as a private registry on Red Hat OpenShift](#)

- [Set up Red Hat OpenShift on IBM Cloud to use Container Registry](#)
- [Set up Red Hat OpenShift Container Platform to use Container Registry](#)
 - [Set up the Red Hat OpenShift Container Platform internal registry to pull from Container Registry](#)
 - [Set up the Red Hat OpenShift Container Platform build to push images to Container Registry](#)

Adding images to your namespace

[Adding images to your namespace](#)

- [Pulling images from another registry](#)
- [Pushing Docker images to your namespace](#)
- [Copying images between registries](#)
- [Creating images that refer to a source image](#)
- [Building Docker images to use them with your namespace](#)
- [Pushing images by using an API key](#)
- [Removing tags from images in your private repository](#)
- [Deleting images from your private repository](#)
 - [Deleting images from your private repository in the CLI](#)
 - [Deleting images from your private repository in the console](#)
- [Listing images in the trash](#)
- [Restoring images](#)
 - [Restoring images by digest](#)
 - [Restoring images by tag](#)
- [Deleting a private repository and any associated images](#)

Using Helm charts

[Using Helm charts](#)

- [OCI support for Helm charts](#)
- [Adding Helm charts to your namespace](#)
 - [Pulling charts from another registry or Helm repository](#)
 - [Pushing Helm charts to your namespace](#)
 - [Copying charts between registries](#)
 - [Installing a Helm chart to the cluster](#)
- [Deleting charts from your private repository](#)
 - [Deleting charts from your private repository in the CLI](#)

- [Deleting charts from your private repository in the console](#)
- [Listing charts in the trash](#)
- [Restoring charts](#)
 - [Restoring charts by digest](#)
 - [Restoring charts by tag](#)
- [Deleting a private repository and any associated charts](#)

Cleaning up your namespaces

[Cleaning up your namespaces](#)

- [Planning retention](#)
- [Clean up your namespaces to keep a set number of images](#)
- [Set a retention policy for your namespaces](#)
- [Update a retention policy to keep all your images](#)
- [Clean up your namespaces by deleting untagged images](#)

Managing quota limits for storage and pull traffic

[Managing quota limits for storage and pull traffic](#)

- [Setting quota limits for storing and pulling images](#)
- [Reviewing quota limits and usage](#)
- [Staying within quota limits](#)

Managing user access

[Accessing Container Registry](#)

- [Accessing your namespaces in automation](#)
 - [Creating a service ID API key manually](#)
 - [Creating a user API key manually](#)
 - [Using client software to authenticate in automation](#)
 - [Using Buildah to authenticate with the registry](#)
 - [Using Docker to authenticate with the registry](#)
 - [Using Podman to authenticate with the registry](#)
 - [Using Skopeo to authenticate with the registry](#)
- [Accessing your namespaces interactively](#)
 - [Using Buildah to access your namespace](#)
 - [Using Docker to access your namespace](#)
 - [Using Podman to access your namespace](#)
 - [Using Skopeo to access your namespace](#)
- [Accessing your namespaces programmatically](#)

[Managing IAM access](#)

- [Access policies](#)
- [Assign roles](#)
- [Context-based restrictions](#)

- [Platform management roles](#)
- [Service access roles](#)
 - [Access roles for configuring Container Registry](#)
 - [Access roles for using Container Registry](#)
- [Assigning access to Container Registry in the console](#)
- [Assigning access to Container Registry in the CLI](#)
- [Assigning access to Container Registry by using the API](#)
- [Assigning access to Container Registry by using Terraform](#)

[Defining IAM access policies](#)

- [Creating policies](#)

Managing image security with Vulnerability Advisor

[Managing image security with Vulnerability Advisor](#)

- [About Vulnerability Advisor](#)
 - [Data protection](#)
- [Types of vulnerabilities](#)
 - [Vulnerable packages](#)
 - [Configuration issues - version 3 only](#)
- [Setting the Vulnerability Advisor version](#)
- [Reviewing a vulnerability report](#)
 - [Reviewing a vulnerability report by using the console - version 3 only](#)
 - [Reviewing a vulnerability report by using the CLI](#)
- [Setting organizational exemption policies](#)
 - [Setting exemption policies by using the console](#)
 - [Setting exemption policies by using the CLI](#)

Setting up Terraform for Container Registry

[Setting up Terraform for Container Registry](#)

- [Installing Terraform and creating a Container Registry namespace](#)
- [Next steps](#)

Enhancing security

[Managing security and compliance](#)

[Using VPE for VPC to privately connect](#)

- [Before you begin](#)
- [Virtual private endpoints](#)
- [Setting up a VPE for IBM Cloud Container Registry](#)
 - [Configuring an endpoint gateway](#)

[Securing your connection to Container Registry](#)

- [Using private network connections](#)
 - [Enabling service endpoint support for the account](#)
 - [Considerations for private network connections](#)

- [Pushing and pulling images](#)
- [Enforcing access to your account over a private network](#)

[Encrypting images for content confidentiality](#)

- [Before you begin](#)
- [Create the public-private key pair](#)
- [Encrypt the image](#)
- [Pull and decrypt the image](#)
- [Storing keys](#)
- [Next steps](#)

[Managing your data](#)

- [How your data is stored](#)
 - [Image data](#)
 - [Scanning data](#)
- [Deleting your data](#)
 - [Deleting the service](#)
 - [Deleting namespaces](#)
 - [Deleting images](#)
 - [Deleting private repositories](#)
- [Restoring deleted data](#)

[Signing images for trusted content](#)

- [Signing images by using Red Hat signatures](#)
 - [Using Skopeo to sign images](#)
 - [Using Podman to sign images](#)
 - [Signing images by using the Red Hat OpenShift CLI](#)

[Using IAM IP address access restrictions](#)

- [Granting access if you are using a public network](#)
- [Granting access if you are using a private network](#)

[Enforcing container image security by using Portieris](#)

- [Installing Portieris in your cluster](#)
- [Portieris policies](#)
- [Uninstalling Portieris](#)

Observability

[Monitoring metrics](#)

- [Enabling metrics for Container Registry](#)
- [Locations of platform metrics](#)
- [Where to look for metrics](#)
 - [Monitoring metrics](#)
- [Viewing metrics](#)
 - [Starting the Monitoring UI from the Observability page](#)
- [Predefined dashboards](#)

- [Platform metrics](#)
 - [Pull Traffic](#)
 - [Pull Traffic Quota](#)
 - [Storage Quota](#)
 - [Storage](#)

[Auditing events](#)

- [Locations of service events](#)
- [Where to look for events](#)
 - [IBM Cloud Activity Tracker events](#)
- [API methods](#)
 - [Actions that generate events for authorization](#)
 - [Actions that generate events for images](#)
 - [Actions that generate events for namespaces](#)
 - [Actions that generate events for plans](#)
 - [Actions that generate events for quotas](#)
 - [Actions that generate events for retention policies](#)
 - [Actions that generate events for settings](#)
 - [Actions that generate events for signing images](#)
 - [Actions that generate events for trash](#)
 - [Actions that generate events for vulnerabilities](#)
 - [Actions that generate events for exemption policies](#)
- [Analyzing Activity Tracker events](#)
 - [Request data for vulnerability events](#)
 - [Request data for the account vulnerability report](#)
 - [Request data for the account vulnerability status](#)
 - [Request and response data for the vulnerability report](#)
 - [Request and response data for the vulnerability status](#)
 - [Request data for image signing events](#)

[Analyzing logs](#)

- [Locations of platform services logs](#)
- [Where to look for IBM Log Analysis logs](#)

[IAM and Activity Tracker actions by API method](#)

- [Container Registry](#)
 - [Authorization API methods](#)
 - [Image API methods](#)
 - [Message API methods](#)
 - [Namespace API methods](#)
 - [Plan API methods](#)
 - [Quota API methods](#)
 - [Retention API methods](#)

- [Settings API methods](#)
- [Tag API methods](#)
- [Trash API methods](#)
- [Vulnerability Advisor API methods](#)
 - [Report API methods](#)
 - [Exemption API methods](#)

Container Registry CLI

[IBM Cloud Container Registry CLI](#)

- [Prerequisites](#)
 - [Notes](#)
- [ibmcloud cr api](#)
 - [Prerequisites](#)
- [ibmcloud cr exemption-add](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Examples](#)
- [ibmcloud cr exemption-list\(ibmcloud cr exemptions\)](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Examples](#)
- [ibmcloud cr exemption-rm](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Examples](#)
- [ibmcloud cr exemption-types](#)
 - [Prerequisites](#)
 - [Command options](#)
- [ibmcloud cr iam-policies-enable](#)
 - [Prerequisites](#)
- [ibmcloud cr iam-policies-status](#)
- [ibmcloud cr image-digests\(ibmcloud cr digests\)](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr image-inspect](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr image-list\(ibmcloud cr images\)](#)
 - [Prerequisites](#)

- [Command options](#)
 - [Example](#)
- [ibmcloud cr image-prune-untagged](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr image-restore](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr image-rm](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr image-tag](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Examples](#)
- [ibmcloud cr image-untag](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr info](#)
 - [Prerequisites](#)
- [ibmcloud cr login](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr manifest-inspect](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr namespace-add](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr namespace-assign](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)

- [ibmcloud cr namespace-list\(ibmcloud cr namespaces\)](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr namespace-rm](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr plan](#)
 - [Prerequisites](#)
 - [Command options](#)
- [ibmcloud cr plan-upgrade](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr platform-metrics](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr private-only](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr quota](#)
 - [Prerequisites](#)
 - [Command options](#)
- [ibmcloud cr quota-set](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr region](#)
 - [Prerequisites](#)
- [ibmcloud cr region-set](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr retention-policy-list](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)

- [ibmcloud cr retention-policy-set](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Examples](#)
- [ibmcloud cr retention-run](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr trash-list](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr va-version](#)
 - [Prerequisites](#)
- [ibmcloud cr va-version-set](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Example](#)
- [ibmcloud cr vulnerability-assessment\(ibmcloud cr va\)](#)
 - [Prerequisites](#)
 - [Command options](#)
 - [Examples](#)

[Formatting and filtering the CLI output](#)

- [Go template options for ibmcloud cr image-digests](#)
- [Go template options for ibmcloud cr image-list](#)
- [Go template options for ibmcloud cr image-inspect](#)
 - [Config](#) field details
 - [Healthcheck](#) field details
 - [RootFS](#) field details

[Container Registry CLI change log](#)

- [Version 1.2.2](#)
- [Version 1.1.0](#)
- [Version 1.0.11](#)
- [Version 1.0.8](#)
- [Version 0.1.587](#)
- [Version 1.0.6](#)
- [Version 1.0.5](#)
- [Version 0.1.585](#)
- [Version 1.0.2](#)
- [Version 0.1.584](#)

- [Version 1.0.1](#)
- [Version 0.1.583](#)
- [Version 1.0.0](#)
- [Version 0.1.582](#)

High availability and disaster recovery

[High availability](#)

- [Ownership of responsibilities](#)
- [What level of availability do I need?](#)
- [What level of availability does IBM Cloud offer?](#)
- [Locations for service availability](#)
- [Frequently asked questions about high availability](#)
 - [Does IBM Cloud replicate the service?](#)
 - [Are users required to replicate the service?](#)

[Business continuity and disaster recovery](#)

- [Your responsibilities when you're using Container Registry](#)
- [Disaster recovery strategy](#)
- [Locations for service availability](#)
- [Frequently asked questions about disaster recovery](#)
 - [Does the service replicate the data?](#)
 - [What data is backed up or replicated?](#)
 - [Are users required to replicate the data?](#)

Your responsibilities

[Your responsibilities](#)

- [Incident and operations management](#)
- [Change management](#)
- [Identity and access management](#)
- [Security and regulation compliance](#)
- [Disaster recovery](#)

Terraform reference

[Terraform reference](#)

API reference

[IBM Cloud Container Registry API](#)

[Vulnerability Advisor 3 for IBM Cloud Container Registry API](#)

[Vulnerability Advisor 4 for IBM Cloud Container Registry API](#)

Related links

[IBM Cloud Kubernetes Service documentation](#)

[Red Hat OpenShift on IBM Cloud documentation](#)

Troubleshooting

[Troubleshooting](#)

- [Troubleshooting topics](#)
 - [Troubleshooting CLI login](#)
 - [Troubleshooting pull and push errors](#)
 - [Troubleshooting CLI commands](#)
 - [Troubleshooting networking](#)
 - [Troubleshooting Portieris](#)

Troubleshooting CLI login

[Why can't I log in to Container Registry?](#)

[Why does the Container Registry login keep expiring?](#)

[Why do commands fail saying that I'm not logged in?](#)

[Why do commands fail saying they're not registered?](#)

[Why is Docker login on my Mac failing?](#)

Troubleshooting pull and push errors

[Why can't I push or pull a Docker image?](#)

[Why is pulling images slow?](#)

[Why am I getting **Authorization required** errors?](#)

[Why am I getting an **Unauthorized** error when I'm using Code Engine?](#)

[Why am I getting **Access denied** errors?](#)

[Why am I getting **Access denied** errors for a resource?](#)

[Why am I getting **Access denied** errors about insufficient scope?](#)

[Why am I getting **Access denied** errors about my quota?](#)

[Why am I getting **Access denied** errors over a private network?](#)

[Why am I getting a **Forbidden** error when I'm using Code Engine?](#)

[Why do images fail to pull from registry with **ImagePullBackOff** or authorization errors?](#)

- [Troubleshooting image pull secrets that use API keys](#)

Troubleshooting CLI commands

[Why can't I add a namespace?](#)

[When I create a namespace, why aren't I authorized to access the specified resource?](#)

[Why can't I find my image or my namespace?](#)

[Why don't all my namespaces show in the **Resource list**?](#)

[Why is it timing out when I list images?](#)

[Why can't I pull the newest image by using the **latest** tag?](#)

[Why do all the tags get deleted when I delete an image?](#)

[Why doesn't the retention command show all the images?](#)

[Why do I get an error when I'm restoring an image?](#)

[Why aren't all the tags restored when I restore by digest?](#)

[Why do I get a manifest unknown error?](#)

[Why do I get a manifest type error when I tag my image?](#)

[Why do I get a manifest version error?](#)

[Why do I get a manifest list invalid error?](#)

Troubleshooting networking

[Why can't I access the registry through a custom firewall?](#)

[Why can't I connect to Container Registry?](#)

Troubleshooting Portieris

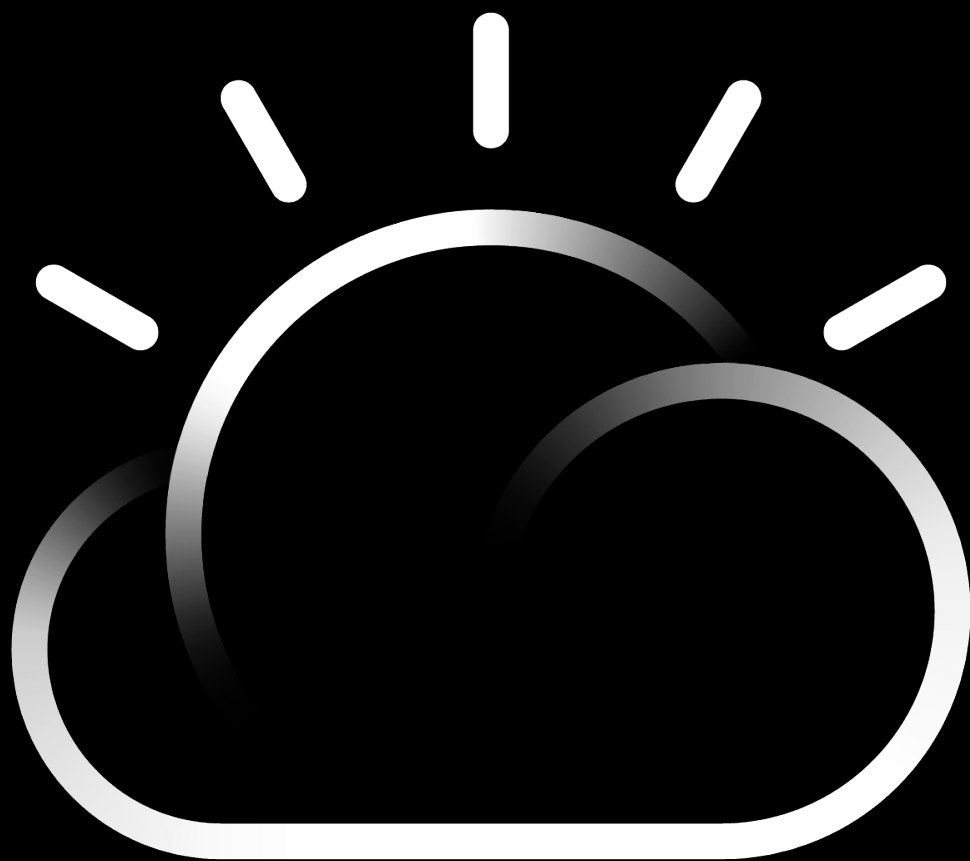
[Why don't my pods restart after my workers were down?](#)

Frequently asked questions (FAQs)

[Frequently asked questions \(FAQs\)](#)

- [Frequently asked questions about Container Registry](#)
 - [How do you list public images?](#)
 - [What tools can I use to build and push images?](#)
 - [How many namespaces can you have?](#)
 - [Can I rename a namespace?](#)
 - [Do images in the trash count toward my quota?](#)
 - [How do I find the image digest?](#)
 - [How do I use digests to work with images?](#)
 - [How do you use access control?](#)
 - [How can I share an image with many users?](#)
 - [Do I have any untagged images?](#)
 - [Do I need untagged images?](#)
 - [What are eligible images?](#)
 - [What regions are available?](#)
- [Frequently asked questions about Vulnerability Advisor](#)
 - [How much does Vulnerability Advisor cost?](#)
 - [Can images from other registries be scanned?](#)
 - [How is a Vulnerability Advisor scan triggered?](#)
 - [Why doesn't a new image scan?](#)
 - [How often are the security notices updated?](#)
 - [Why can I see a vulnerability in Vulnerability Advisor v4 but not in v3?](#)
 - [Which version of a package is installed in my image?](#)
 - [Alpine package manager commands](#)
 - [Debian and Ubuntu package manager commands](#)
 - [Red Hat and CentOS package manager commands](#)
 - [Does Vulnerability Advisor have versions?](#)

Getting help and support for Container Registry



IBM Cloud