

*Running App Connect Enterprise in
containerized environments*



Contents

Running App Connect Enterprise in containerized environments.....	1
Release notes.....	1
About this release.....	1
New features and enhancements.....	2
Known issues.....	3
Deprecated features.....	4
App Connect Enterprise components.....	4
IBM Cloud Pak for Integration.....	5
Installing IBM Cloud Pak for Integration and App Connect Enterprise.....	5
Deleting App Connect Designer or App Connect Dashboard instances.....	6
IBM App Connect Enterprise certified container.....	7
Setting up your cluster and local workstation.....	8
Setting up a namespace for deploying IBM App Connect Enterprise certified container.....	9
Installing IBM App Connect Enterprise certified container.....	10
Creating an instance of App Connect Dashboard.....	10
Creating an instance of App Connect Designer.....	13
Upgrading the Helm release for an App Connect Dashboard or App Connect Designer instance.....	16
Deleting App Connect Designer or App Connect Dashboard instances.....	16
Creating and managing flows for an API in App Connect Designer.....	17
Accessing your App Connect Designer instance.....	17
Creating flows for an API.....	21
Exporting and importing API flows.....	42
IBM Cloud Pak for Integration only: Creating and reusing assets in the Asset Repository.....	46
Deploying IBM App Connect Designer APIs to integration servers.....	53
High-level sequence for running Designer-authored APIs in integration servers.....	54
Prerequisites.....	55
Exporting an API flow to a BAR file.....	56
Overriding the ID and URL in the BAR file.....	60
Cloud Pak for Integration only: Accessing the management console.....	63
Installing the command-line tools.....	66
Logging in to your cluster from the command-line interface.....	67
Creating an IBM Cloud API key.....	67
Cloud Pak for Integration only: Creating an integration server to run your BAR file resources.....	70
App Connect Enterprise certified container only: Creating an integration server to run your BAR file resources.....	89
Invoking the deployed API.....	104
Deleting an integration server.....	105
Deleting a secret.....	106
Managing BAR files.....	107
Troubleshooting and monitoring.....	110
Index.....	115

Running App Connect Enterprise in containerized environments

App Connect Enterprise is offered as an executable package that you can deploy and run within containerized environments.

You can run App Connect Enterprise from the following offerings, which meet specific criteria for packaging and deploying containerized software:

- IBM® Cloud Pak for Integration running on Red Hat OpenShift Container Platform
- IBM App Connect Enterprise certified container, deployed on Red Hat OpenShift

The [Red Hat OpenShift](#) platform provides an environment for developing and managing containerized applications, and offers a way to deliver deployment packages for IBM software applications that are built for Kubernetes. These deployment packages are implemented as Helm charts that describe the Kubernetes resources for the applications. The Helm charts also reference Docker images that contain all the artifacts needed to run the applications.

Within your environment, an administrator can set up a Kubernetes-based *cluster* that consists of a set of machines (or *nodes*) for running containerized applications. To manage access to resources in the cluster, users and user groups (with assigned roles) can be logically grouped in teams that are mapped to dedicated *namespaces*. A team can be mapped to one or more namespaces based on resource grouping requirements.

A set of *pod security policies* are applied and bound to every namespace. When the Helm chart for an application is installed on a namespace, the configured policies are applied to any pods that the application creates for the group of containers that it needs to run in the cluster. The pod security policies establish what a pod can do or what it can access within the cluster.

Release notes

Review these release notes for information about supported versions, new features and enhancements, and known issues and workarounds.

- [“About this release” on page 1](#)
- [“New features and enhancements” on page 2](#)
- [“Known issues” on page 3](#)
- [“Deprecated features” on page 4](#)

About this release

This release relates to the following versions of these offerings:

- IBM Cloud Pak for Integration Version 2020.1.1, running on Red Hat OpenShift Container Platform Versions 4.2 and 4.3, and OpenShift Container Storage Version 4.2
- IBM App Connect Enterprise certified container Version 3.1.0, deployed on Red Hat OpenShift Versions 4.2 and 4.3

IBM App Connect Enterprise certified container contains an image of App Connect Enterprise Version 11.0.0.8.

This documentation relates to the release for these versions.

Documentation for the previous release is available in PDF format from the following download location. Download all the files to use the PDF locally.

http://public.dhe.ibm.com/software/integration/hybrid_integration/app_connect_enterprise/ACEcc

New features and enhancements

New App Connect Designer authoring environment

An App Connect Designer authoring environment is now available for non-production workloads. You can create one or more App Connect Designer instances to develop and test flows for APIs, and can enable support for local connectors only, or for both local and cloud-managed connectors. To use cloud-managed connectors, an App Connect Designer instance must be coupled with an App Connect on IBM Cloud instance that hosts the cloud connectors.

App Connect Designer provides export and import functions for sharing flows, and also lets you export a flow as a BAR file that can be deployed as an integration server in an App Connect Dashboard runtime environment.

In IBM Cloud Pak for Integration, you can also share a flow as an asset within the Asset Repository, and can use "flow" assets, which are shared by other users, to create flows within your App Connect Designer instance.

More information:

- [“App Connect Enterprise components” on page 4](#)
- [“Creating and managing flows for an API in App Connect Designer” on page 17](#)

New UI for BAR file deployment in the App Connect Dashboard

When you deploy a BAR file as an integration server, a new UI is presented for you to define the configuration. On initial entry, the UI displays the minimum set of fields that you must complete for a quick deployment, but you can expand this view to expose all the fields if you want to define a more advanced configuration with non-default values. For an alternative view of the configuration, you can also switch to a Code view.

More information:

- [Cloud Pak for Integration only: Creating an integration server to run your BAR file resources](#)
- [App Connect Enterprise certified container only: Creating an integration server to run your BAR file resources](#)
- [Deploying IBM App Connect Designer APIs to integration servers](#)

New "update" function for BAR files in the App Connect Dashboard

As part of BAR file management, you can now replace an existing BAR file with an updated file of the same name. If the existing version of the file is already deployed, any integration servers that are using that BAR file are restarted on a rolling basis to ensure that at least one replica is always running.

More information: [Managing BAR files](#)

Extended support for local connectors

In IBM Cloud Pak for Integration, 10 additional local connectors are now supported for integration servers running in an App Connect Dashboard instance.

Support for local connectors has also now been extended to:

- Integration servers that are deployed to an App Connect Dashboard instance running on IBM App Connect Enterprise certified container
- Flows that are created in the new App Connect Designer authoring environment (in both IBM Cloud Pak for Integration and IBM App Connect Enterprise certified container)

More information: [“Supported components for flows” on page 41](#)

Known issues

App Connect Designer

Error while retrieving the list of connectors

When you open the Catalog page to view the list of connectors, or when you try to add a node to a flow in the flow editor, you might occasionally see the following error:



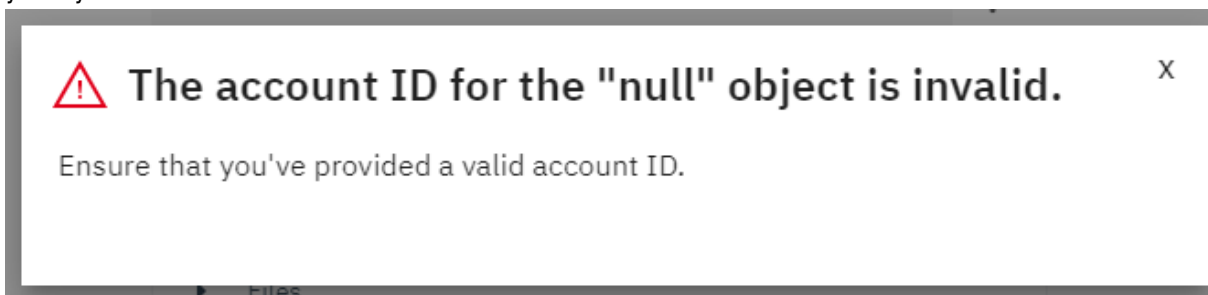
To investigate further, ask your Red Hat OpenShift administrator to check the details on the pod for your App Connect Designer instance. The pod should have a name similar to this: `designerbob-ibm-app-connect-designer-icp4i-ui-123b34f57b-1tkcb`. From the terminal for the pod, navigate to `/opt/ibm/app/logs`. The text files in this location log additional message details that can help you identify the cause.

For example, if you provided incorrect cloud account details, you might see an error such as this:

```
StatusCodeError: 400 - {"errors":[{"code":"publicapi0007","message":"The requested instance of App Connect could not be found using the provided instance ID.","more_info":"","data":{}}]}
```

Connection failure when adding a local account for Microsoft SharePoint

In the Catalog page in App Connect Designer, an incomplete set of connection fields is displayed when you try to add a local account for Microsoft SharePoint. The following error is displayed when you try to connect:



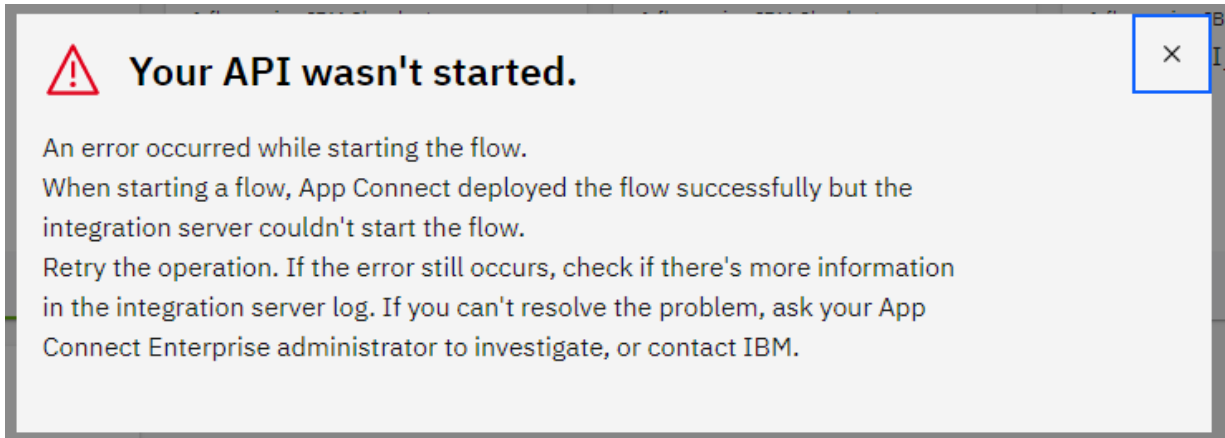
As a workaround, consider using a cloud account if both local and cloud-managed connectors are enabled.

Note: Local SharePoint accounts can be successfully added from an App Connect Dashboard instance by completing the `credentials.yaml` file for a BAR file that was exported from App Connect on IBM Cloud.

Error when trying to start a flow that was imported, or created from an asset

When you import a flow into the App Connect Designer dashboard or create one from an asset, the flow is renamed (as expected) by appending `_n` to its name if a flow with an identical name already

exists. (For example, a newly added flow is renamed Customer API_1 if a flow named Customer API already exists.) Attempting to start the renamed flow will result in this error:



As a workaround, consider renaming your existing flow before adding another flow with the same name.

Cannot upgrade the Helm release for an App Connect Designer instance

An attempted upgrade of the App Connect Designer Helm release in IBM Cloud Pak for Integration or IBM App Connect Enterprise certified container will result in an error.

As a workaround, you can delete the Helm release and then reinstall the chart.

Deprecated features

IBM Cloud Private deployment

IBM App Connect Enterprise certified container no longer supports deployment into IBM Cloud Private.

App Connect Enterprise components

The IBM Cloud Pak for Integration and IBM App Connect Enterprise certified container installations provide App Connect Enterprise environments for authoring and running integrations.

The following components are included:

- [App Connect Designer](#)
- [App Connect Dashboard](#)

App Connect Designer

App Connect Designer serves as an authoring environment for hosting non-production workloads. API developers can develop integration flows for APIs, and share them with other users or instances by using the built-in export and import functions for flows. In IBM Cloud Pak for Integration, flows can also be shared by using the Asset Repository component, which lets users add flows to the repository as assets (of type **Designer API Implementation**), and reuse shared assets of this type by creating them as flows within App Connect Designer.

Multiple instances of App Connect Designer can be created for individual or team use based on organization requirements. App Connect Designer instances can be created in the same namespace as an App Connect Dashboard instance if required.

When authoring a flow, the developer must set up accounts for each of the applications that the flow references. Based on the configuration settings enabled for flows, local connectors, or both local and cloud-managed connectors can be used to establish connections to the target apps and to run the defined

API operations. If cloud-managed connectors are enabled, access will also be required to an [App Connect on IBM Cloud instance](#) that hosts these connectors.

When a flow has been sufficiently tested locally within the authoring environment, it can be exported as a Broker archive (BAR) file that packages the API integration. The BAR file can then be uploaded to an App Connect Dashboard instance where it can be deployed as an integration server that runs the API in production systems.

To learn how to create and manage integration flows in an App Connect Designer instance, see [“Creating and managing flows for an API in App Connect Designer”](#) on page 17.

App Connect Dashboard

App Connect Dashboard provides a runtime environment for hosting production workloads. App Connect Designer and IBM App Connect Enterprise Toolkit integrations can be deployed into this environment to run as integration servers that expose APIs. These integrations are deployed from BAR files that you upload into the dashboard instance. The dashboard also supports storage and management of BAR files.

Only one App Connect Dashboard instance can be allocated per namespace.

To run Designer-authored integrations, you must set up accounts for each of the applications that are referenced in the exported flow that the BAR file packages. Based on the configuration settings enabled for integration servers in the dashboard instance, local connectors only, or both local and cloud-managed connectors can be used to establish connections and run the defined API operations. If cloud-managed connectors are enabled, access will also be required to an [App Connect on IBM Cloud instance](#) that hosts these connectors.

To learn how to deploy an App Connect Designer integration to an integration server in an App Connect Dashboard instance, see [Deploying IBM App Connect Designer APIs to integration servers](#).

IBM Cloud Pak for Integration

IBM Cloud Pak for Integration on Red Hat OpenShift Container Platform delivers a collection of product capabilities in a streamlined integration solution. App Connect Enterprise provides the *application integration* capability within the unified containerized environment.

Use this information as a reference for installing IBM Cloud Pak for Integration, and deploying and managing instances of the App Connect Enterprise capability in your cluster.

Installing IBM Cloud Pak for Integration and App Connect Enterprise

When IBM Cloud Pak for Integration is installed in a Red Hat OpenShift cluster, the App Connect Enterprise capability is installed by default.

About this task

The installation enables you to deploy the following App Connect Enterprise components as instances:

- App Connect Designer, which provides an authoring environment for developing and testing integration flows for an API
- App Connect Dashboard, which provides a runtime environment for integration servers that run App Connect Designer and IBM App Connect Enterprise Toolkit integrations

Installation and user roles:

It is anticipated that a **cluster administrator** will install IBM Cloud Pak for Integration, and a **team administrator** (or integration specialist) will deploy and manage instances of App Connect Designer and App Connect Dashboard, which are created as Helm releases.

Users with the required permissions can then develop and share flows in the authoring environment, or deploy and manage integration servers in the runtime environment.

Procedure

1. Install IBM Cloud Pak for Integration in a Red Hat OpenShift cluster. For further details, see the following information in the IBM Cloud Pak for Integration documentation:
 - [System Requirements](#)
 - [Installation](#)
2. From the IBM Cloud Pak for Integration Platform Navigator, create one or more instances of App Connect Designer. For further details, see the following information in the IBM Cloud Pak for Integration documentation:
 - [Application integration Designer deployment](#)
3. From the Platform Navigator, create one or more instances of App Connect Dashboard. For further details, see the following information in the IBM Cloud Pak for Integration documentation:
 - [Application integration Dashboard deployment](#)

What to do next

- For guidance about the use of App Connect Designer versus App Connect Dashboard, see [“App Connect Enterprise components” on page 4](#).
- To learn how you can create and manage integration flows in your App Connect Designer authoring environment, see [“Creating and managing flows for an API in App Connect Designer” on page 17](#).
- To learn how you can deploy an App Connect Designer integration to an integration server in your App Connect Dashboard runtime environment, see [Deploying IBM App Connect Designer APIs to integration servers](#).


Deleting App Connect Designer or App Connect Dashboard instances

If no longer required, you can delete an instance of App Connect Designer or App Connect Dashboard by deleting the Helm release that was created for that instance. You can delete the Helm release from the IBM Cloud Pak for Integration UI or from the command line.

Before you begin



Ensure that you have cluster administrator or team administrator authority.

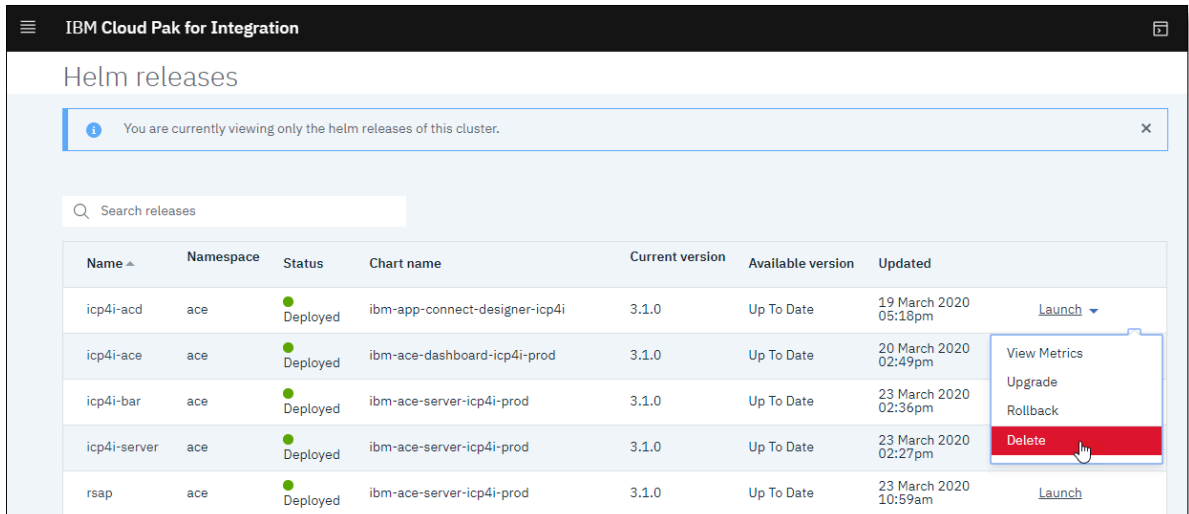
If you want to delete the Helm release from the command line, ensure that you have locally installed the set of command-line interface (CLI) tools for managing the cluster, containers, infrastructures, services, and other resources. For this task, you'll require the OpenShift Container Platform CLI (**oc**) and Helm CLI (**helm**). You can download and install the CLI tools as follows:

1. Download and install the OpenShift Container Platform CLI, as described in [Getting started with the CLI](#) in the Red Hat OpenShift documentation. Ensure that you are using the version of **oc** that's provided for the Red Hat OpenShift version on which IBM Cloud Pak for Integration is running.
2. From IBM Cloud Pak for Integration, download and install the supplied CLI tools as follows:
 - a. Open the IBM Cloud Pak menu  and then click **Cloud Pak Foundation**.
 - b. If prompted, enter credentials to log in to the management console that provides common services. (These should be the same credentials that you used to access the Platform Navigator.)
 - c. From the Cloud Pak management console, download and install the CLI tools. For more information, see [CLI tools guide](#) in the IBM Cloud Pak for Integration documentation in IBM Knowledge Center.

Procedure

Complete either of the following steps:

- From the Platform Navigator, or from an App Connect Designer or App Connect Dashboard instance, delete the Helm release as follows:
 - Open the IBM Cloud Pak menu  and then click **Cloud Pak Foundation**.
 - From the Cloud Pak management console, open the IBM Cloud Pak menu  and click **Administer** > **Helm Releases** to open the "Helm releases" page.
 - Locate the Helm release for the App Connect Designer or App Connect Dashboard instance that you want to delete. Then click the options menu (...) that appears when you hover over the row, and click **Delete**.



- Confirm the deletion (and refresh the page if necessary).
- From a command window, delete the Helm release as follows:
 - Ensure that you are logged in to your cluster:
 - Run the **oc login** command and provide the OpenShift Container Platform server URL (and optionally a token). Examples:


```
oc login https://Mycluster_hostName:8443
```

```
oc login --token=AbcdE1fgHijKLM2moP3q4rs5TU6vw7xYz --server=https://Mycluster_hostName:6443
```
 - Indicate whether to use insecure connections, and then specify a user name and password if required.
 - Run the following command, where *releaseName* is the name of the Helm release to be deleted:

```
helm delete releaseName --purge --tls
```

IBM App Connect Enterprise certified container

You can deploy an IBM App Connect Enterprise certified container on Red Hat OpenShift in order to run App Connect Enterprise in a containerized environment.

The IBM App Connect Enterprise certified container installation provides the following Helm charts to establish the starting point for using App Connect Enterprise:

- App Connect Dashboard charts:
 - `ibm-ace-dashboard-prod`: Deploys a Production Edition dashboard for BAR file management and a user interface to view and create integration servers
 - `ibm-ace-dashboard-dev`: Deploys a Developer Edition dashboard for BAR file management and a user interface to view and create integration servers

- An App Connect Designer chart:
 - `ibm-app-connect-designer`: Provides an authoring environment for creating and managing flows for APIs

Installation and user roles:

It is anticipated that a **cluster administrator** will install IBM App Connect Enterprise certified container, and a **team administrator** (or integration specialist) will deploy and manage the Helm releases of App Connect Designer and App Connect Dashboard.

Users with the required permissions can then develop and share flows in the authoring environment, or deploy and manage integration servers in the runtime environment.

For more information about App Connect Dashboard and App Connect Designer, see [“App Connect Enterprise components” on page 4](#).

Red Hat OpenShift cluster requirements

When the App Connect Dashboard is scaled, you can schedule the running pods onto different nodes. Therefore, the backing storage for the BAR files needs to support the *ReadWriteMany* access mode in Kubernetes (such as GlusterFS or NFS). For more information, see [Access modes](#) in the Kubernetes documentation. Use the **BAR files** page in the dashboard to upload BAR files that the dashboard then stores.

Setting up your cluster and local workstation

Set up your Red Hat OpenShift cluster and ensure that the required tools are installed on your local workstation.

Setting up your cluster

Set up the Red Hat OpenShift cluster where you want to install IBM App Connect Enterprise certified container. For information about setting up your cluster, see the [Red Hat Open Shift documentation](#).

Installing tools for managing the cluster, containers, and other resources

A set of tools can be used to manage the cluster, containers, and other resources. Ensure that you have these tools installed. One of the command-line interface (CLI) tools, Helm, also requires additional configuration, so review this information to verify that your Helm CLI tool is appropriately configured.

Procedure

Ensure that the following tools are installed on your local workstation:

- **OpenShift Container Platform CLI (oc)**: For installation instructions, see [Getting started with the CLI](#) in the Red Hat OpenShift documentation. Use the version of **oc** that’s provided for the Red Hat OpenShift version on which IBM App Connect Enterprise certified container will be installed.

- **Helm CLI (helm):**

1. Use these instructions as a guide, but ensure that Helm V2.13.0 is installed instead: [Getting started with Helm on OpenShift](#).

Complete only "Step 1: Create an OpenShift project for Tiller", "Step 2: Install the Helm client locally", and "Step 3: Install the Tiller server" from the article.

When you get to "Step 3: Install the Tiller server", ensure that you replace `HELM_VERSION=v2.9.0` with `HELM_VERSION=v2.13.0` in the supplied **oc process** command:

```
oc process -f https://github.com/openshift/origin/raw/master/examples/helm/tiller-template.yaml -p TILLER_NAMESPACE=${TILLER_NAMESPACE} -p HELM_VERSION=v2.13.0 | oc create -f -
```

2. Verify that the Helm client can successfully establish a connection to the cluster by running the following command:

```
helm version --tiller-namespace tiller
```

Note: Whenever you run the **helm** command, ensure that you append `--tls` to the command if TLS is enabled in the cluster.

The output should confirm that the version of **helm** running locally and on the cluster both match at 2.13.0.

- **Docker:** For installation instructions, see [Docker Documentation](#).
- Optional: **Kubernetes CLI (kubectl):** For installation instructions, see [Install and Set Up kubectl](#) in the Kubernetes documentation.

Setting up a namespace for deploying IBM App Connect Enterprise certified container

You must have a namespace in order to deploy IBM App Connect Enterprise certified container. You can set up a new namespace or use an existing namespace in your cluster.

Before you begin

Prepare your cluster and workstation as described in [“Setting up your cluster and local workstation”](#) on page 8.

Procedure

1. Ensure that you are logged in to your cluster.
2. Create a namespace (or project). You can skip this step if you have an existing namespace that you want to use instead.

```
oc new-project namespace_name
```

3. Give administration access to the Tiller namespace to service accounts in your namespace:

```
oc adm policy add-role-to-group admin system:serviceaccounts:namespace_name -n tiller
```

You can safely ignore this warning message if you see it:

```
Warning: Group 'system:serviceaccounts:namespace_name' not found
```

The preceding command creates a RoleBinding that you can view by running this command:

```
oc get rolebindings --namespace tiller
```

4. Give Tiller full administration access to your namespace:

```
oc adm policy add-cluster-role-to-user cluster-admin "system:serviceaccount:tiller:tiller" --namespace namespace_name
```

5. Create a secret in your namespace called `ibm-entitlement-key`. To create this secret, run the following sequence of commands:

```
export NAMESPACE=namespace_name
export IMAGE_PULL_SECRET=$(oc get secrets --namespace openshift-image-registry | grep --color=never registry-dockerconfig | awk '{print $1}')
oc get secrets $IMAGE_PULL_SECRET -o yaml --namespace=openshift-image-registry | sed "s:${IMAGE_PULL_SECRET}:ibm-entitlement-key:g" | sed "s:openshift-image-registry:${NAMESPACE}:g" | oc apply -f -
```

6. Install and bind the Red Hat OpenShift security context constraints (SCCs).

- For information about how to install SCCs in your namespace, see the guidance on [Installing IBM Cloud Pak SecurityContextConstraints resources to your cluster](#).
- Bind the SCC name to the namespace.

The predefined SCC name, `ibm-anyuid-scc`, has been verified for this Helm chart. If your target namespace is not bound to this SCC resource, you can bind it by running the following command:

```
oc adm policy add-scc-to-group ibm-anyuid-scc system:serviceaccounts:namespace_name
```

What to do next

Install IBM App Connect Enterprise certified container: [“Installing IBM App Connect Enterprise certified container” on page 10](#).

Installing IBM App Connect Enterprise certified container

Obtain the installation package for IBM App Connect Enterprise certified container and deploy it to Red Hat OpenShift.

Before you begin

1. To find out which versions of IBM App Connect Enterprise certified container are supported, see the release notes that are included with the Helm charts at <https://github.com/IBM/charts/blob/master/stable/ibm-ace-server-dev/RELEASENOTES.md>.
2. Set up a namespace as described in [“Setting up a namespace for deploying IBM App Connect Enterprise certified container” on page 9](#).

Procedure

1. Obtain the IBM App Connect Enterprise certified container compressed file from [IBM Passport Advantage](#).

The compressed file contains the App Connect Enterprise images and Helm charts that enable App Connect Enterprise to be deployed in a containerized environment.

2. Ensure that you are logged in to your cluster.
3. Extract the contents of the compressed file.

For example:

```
$ tar xvzf IBM-App-Connect-Enterprise-20200226-133312.tar.gz
```

4. Follow the instructions in the `README.md` file that was extracted from the compressed file. This `README.md` file explains how to upload the images to your image registry.

Creating an instance of App Connect Dashboard

You can create an instance of App Connect Dashboard within a namespace in your cluster by using a Helm chart that is supplied as part of your IBM App Connect Enterprise certified container installation. Users in this namespace can then use the dashboard to deploy BAR files as integration servers and manage the BAR files.

A single dashboard is allowed per namespace.

About this task

The following dashboard Helm charts are available:

- If you are using App Connect Enterprise (Production Edition), use the `ibm-ace-dashboard-prod` Helm chart.
- If you are using App Connect Enterprise (Developer Edition), use the `ibm-ace-dashboard-dev` Helm chart.

The documented instructions in these topics relate to the `ibm-ace-dashboard-prod` chart. For information about obtaining and using the `ibm-ace-dashboard-dev` chart, see:

- <https://github.com/IBM/charts/blob/master/stable/ibm-ace-dashboard-dev/README.md>
- <https://github.com/IBM/charts/tree/master/stable/ibm-ace-dashboard-dev>

Procedure

1. Ensure that you are logged in to your cluster.
2. From the command line, navigate to the directory that contains the extracted files from the IBM Passport Advantage package.
3. Run the following commands to install the dashboard Helm chart, where:

- `namespace_name` is the namespace where you want to deploy the chart.
- `releaseName` is a unique Helm release name for the App Connect Dashboard instance.
- If TLS is enabled in your cluster, complete the following steps:
 - a. Run the following script to create a secret that is needed when you create the dashboard instance:

```
./ibm-ace-dashboard-prod/ibm_cloud_pak/pak_extensions/createHelmSecret.sh helmcerts
```

- b. Run the following commands collectively to set environment variables and install the dashboard Helm chart:

```
export RELEASE_TAG=11.0.0.8-r1
export NAMESPACE=namespace_name
helm install --name releaseName ibm-ace-dashboard-prod \
  --tiller-namespace=tiller \
  --set license=accept \
  --set image.contentServer=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-ace-content-server-prod:${RELEASE_TAG} \
  --set image.controlUI=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/
ibm-ace-dashboard-prod:${RELEASE_TAG} \
  --set image.configurator=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-acecc-configurator-prod:${RELEASE_TAG} \
  --set image.infra=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/ibm-
acecc-infra-prod:${RELEASE_TAG} \
  --set ssoEnabled=false \
  --set tillerNamespace=tiller --set helmTlsSecret=helmcerts --tls
```

- If TLS is not configured in your cluster, run the following commands collectively to set environment variables and install the dashboard Helm chart:

```
export RELEASE_TAG=11.0.0.8-r1
export NAMESPACE=namespace_name
helm install --name releaseName ibm-ace-dashboard-prod \
  --tiller-namespace=tiller \
  --set license=accept \
  --set image.contentServer=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/
ibm-ace-content-server-prod:${RELEASE_TAG} \
  --set image.controlUI=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/ibm-
ace-dashboard-prod:${RELEASE_TAG} \
  --set image.configurator=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/
ibm-acecc-configurator-prod:${RELEASE_TAG} \
  --set image.infra=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/ibm-
acecc-infra-prod:${RELEASE_TAG} \
  --set ssoEnabled=false \
```

```
--set tillerNamespace=tiller
```

For a list of the fields in these commands and their default values, see the `ibm-ace-dashboard-prod/values.yaml` file that was extracted from the IBM Passport Advantage package. For an explanation of what the fields represent, see the `ibm-ace-dashboard-prod/values-metadata.yaml` file.

4. Run the following command to check the status of the Helm release:

```
helm status releaseName --tiller-namespace=tiller
```

This command outputs some information in a `NOTES:` section, which includes a command that you can run to discover the URL of your dashboard. Look for the line that starts with this command:

```
echo && export ACE_DASHBOARD_URL=...
```

5. Copy and run this command to obtain the dashboard URL.

What to do next

You can distribute the dashboard URL to all users with the required access. To obtain the URL on subsequent occasions, repeat steps “4” on page 12 and “5” on page 12 in the preceding task.

IBM Cloud Pak for Integration only: Authority to use the App Connect Dashboard

A user's authority to use the App Connect Dashboard in specific roles is configured by use of the Identity and Access Management (IAM) service.

Authority to use an App Connect Dashboard is determined by the user's authority for the namespace that contains the dashboard. The relation between user authority and namespace is explained here:

1. The App Connect Dashboard is created in a namespace.
2. The namespace is a resource to which an IAM team can be granted access.
3. The IAM team consists of users and each user has an assigned role.
4. Each user role has a specified authority.

Users of the App Connect Dashboard have authorities that correspond to the roles described in the following table:

IAM service role	App Connect Enterprise role	App Connect Enterprise security permissions
ClusterAdministrator/ Administrator	admin	read+:write+:execute+
Operator	operator	read+:write-:execute+
Editor	editor	read+:write+:execute-
Auditor	audit	read+:write-:execute-
Viewer	viewer	read+:write-:execute-

Note: App Connect Enterprise integration servers are preconfigured with security permissions that map directly to IAM roles. The permissions are `server.conf.yaml` configuration file overrides that do not require modification.

If a user needs App Connect Enterprise to share REST APIs with IBM API Connect, they must be authorized in the `admin` or `operator` App Connect Enterprise role.

For information about mapping user actions and roles to permissions, see [Table 2: File-based permissions required for acting on an integration node](#) and [Controlling access to data and resources in the web user interface](#).

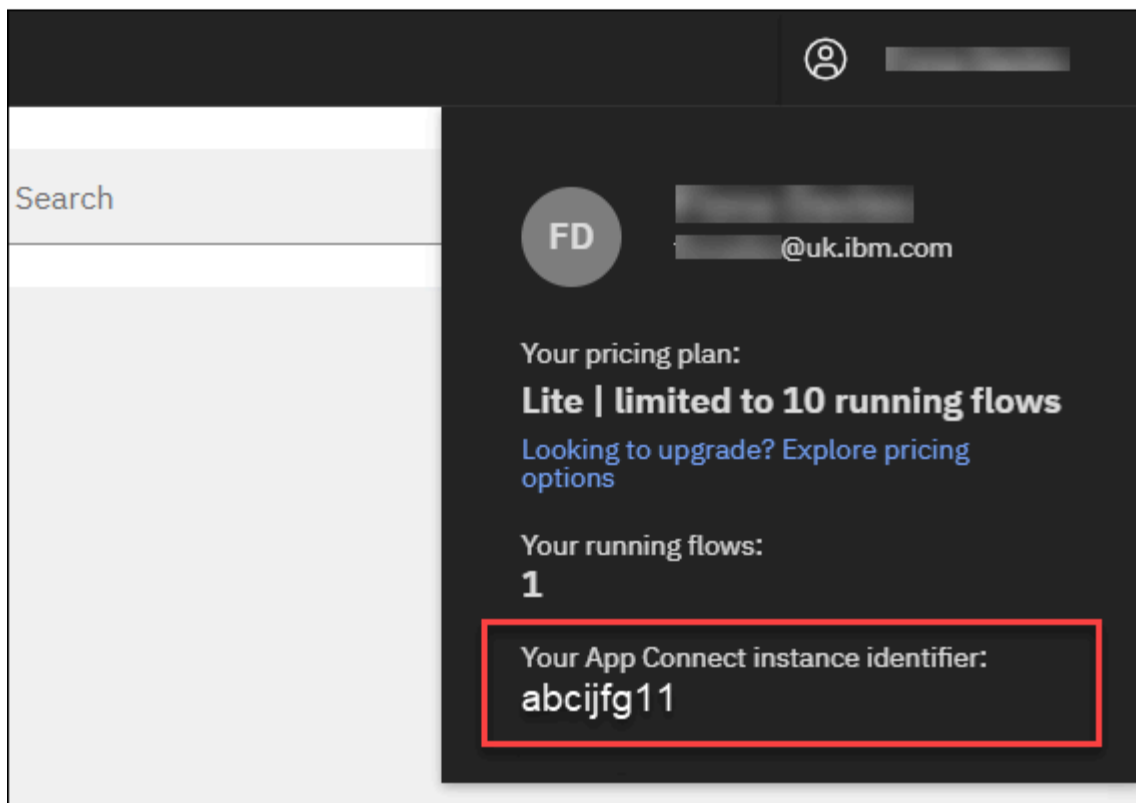
Creating an instance of App Connect Designer

You can create one or more instances of App Connect Designer within a namespace in your cluster by using a Helm chart that is supplied as part of your IBM App Connect Enterprise certified container installation. Users in this namespace can then use their allocated instance to develop and manage flows for an API.

Before you begin

In the Helm chart for an App Connect Designer instance, you can enable the use of local connectors only, or enable both local and cloud-managed connectors to run the API operations in a flow. To use cloud-managed connectors, the following prerequisites must be met:

1. [An instance of App Connect on IBM Cloud](#) is required, which hosts the cloud-managed connectors.
2. Launch the instance and make a note of the instance identifier.



3. Make a note of the region in which the instance is provisioned.
4. Create an IBM Cloud API key while logged in to the same IBM Cloud account as the App Connect on IBM Cloud instance. You can create this key from the IBM Cloud API Keys page at: <https://cloud.ibm.com/iam/apikeys>. Download the API key to your browser's default location for later use. The file is named `apiKey.json`.

Access considerations for cloud-managed connectors:

The number of App Connect Designer instances that you create should depend on your team requirements, but take note of the following consideration for access.

If you create one App Connect Designer instance for a team of users under a namespace, each user will require access to the (single) App Connect on IBM Cloud instance that hosts the connectors. (An App Connect Designer instance can be coupled with only one App Connect on IBM Cloud instance.)

Each user will require Developer access to the **Cloud Foundry** organization and space that the App Connect on IBM Cloud instance belongs to. For more information about allocating Developer access to the App Connect on IBM Cloud instance, see [Inviting users to an account](#) in the IBM Cloud *Managing identity and access* documentation.

Procedure

1. Ensure that you are logged in to your cluster.
2. From the command line, navigate to the directory that contains the extracted files from the IBM Passport Advantage package.
3. **Applicable only if both local and cloud-managed connectors are enabled:** Run the following command to create a secret to store the IBM Cloud API key that you created earlier. This secret will be used for authenticating to the App Connect on IBM Cloud instance.

```
oc create secret generic IBMCloud_secret --from-literal=apikey=IBMCloud_APIkey --namespace=namespace_name
```

Where:

- *IBMCloud_secret* is the name of the secret for storing the IBM Cloud API key.
 - *IBMCloud_APIkey* is the key within the downloaded `apiKey.json` file.
 - *namespace_name* is the namespace where you want to deploy the chart.
4. Run the appropriate commands to install the App Connect Designer Helm chart named `ibm-app-connect-designer`:

Note: Append `--tls` to the `helm` command if TLS is enabled in your cluster.

- To enable local connectors only, run the following commands collectively to set environment variables and install the chart, where:
 - *namespace_name* is the namespace where you want to deploy the chart.
 - *releaseName* is a unique Helm release name for the App Connect Designer instance.

```
export RELEASE_TAG=11.0.0.8-r1
export NAMESPACE=namespace_name
export designerReleaseName=releaseName
helm install --name {designerReleaseName} ibm-app-connect-designer --tiller-namespace=tiller \
  --set ibm-ace-server-dev.image.aceonly=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-ace-server-dev:{RELEASE_TAG} \
  --set ibm-ace-server-dev.image.connectors=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-ace-lcp-dev:{RELEASE_TAG} \
  --set ibm-ace-server-dev.image.proxy=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-proxy-prod:{RELEASE_TAG} \
  --set ibm-ace-server-dev.image.designerflows=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-ace-designer-flows-dev:{RELEASE_TAG} \
  --set ibm-ace-server-dev.image.configurator=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-acecc-configurator-dev:{RELEASE_TAG} \
  --set image.fireflyFlowdocAuthoring=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-flowdoc-authoring-prod:{RELEASE_TAG} \
  --set image.fireflyRuntime=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-runtime-prod:{RELEASE_TAG} \
  --set image.fireflyFlowTestManager=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-flow-test-manager-prod:{RELEASE_TAG} \
  --set image.connectorAuthService=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-connector-auth-service-prod:{RELEASE_TAG} \
  --set image.fireflyUi=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-ui-prod:{RELEASE_TAG} \
  --set image.proxy=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-app-connect-proxy-prod:{RELEASE_TAG} \
  --set image.configurator=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-acecc-configurator-prod:{RELEASE_TAG} \
  --set couchdb.image.repository=image-registry.openshift-image-registry.svc:5000/{NAMESPACE}/ibm-couchdb2 \
  --set global.designerFlowsOperationMode=local \
  --set license=accept \
  --set tillerNamespace=tiller \
```



```
--set ssoEnabled=false
```

- To enable both local and cloud-managed connectors, run the following commands collectively to set environment variables and install the chart, where:
 - *namespace_name* is the namespace where you want to deploy the chart.
 - *instanceID* is the ID of your App Connect on IBM Cloud instance.
 - *serviceURL* is a base URL that's associated with the region where the App Connect on IBM Cloud instance is provisioned. Specify the value that matches the region:

Region	<i>serviceURL</i> value
us-south or sydney	https://firefly-api-prod.appconnect.ibmcloud.com
eu-gb	https://firefly-api-produk.eu-gb.appconnect.ibm.com

- *IBMCloud_secret* is the secret that you created to store the IBM Cloud API key.
- *releaseName* is a unique Helm release name for the App Connect Designer instance.

```
export RELEASE_TAG=11.0.0.8-r1
export NAMESPACE=namespace_name
export appConnectInstanceID=instanceID
export appConnectURL="serviceURL"
export appConnectSecret=IBMCloud_secret
export designerReleaseName=releaseName
helm install --name ${designerReleaseName} ibm-app-connect-designer --tiller-
namespace=tiller \
  --set ibm-ace-server-dev.image.aceonly=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-ace-server-dev:${RELEASE_TAG} \
  --set ibm-ace-server-dev.image.connectors=image-registry.openshift-image-
registry.svc:5000/${NAMESPACE}/ibm-ace-lcp-dev:${RELEASE_TAG} \
  --set ibm-ace-server-dev.image.proxy=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-app-connect-proxy-prod:${RELEASE_TAG} \
  --set ibm-ace-server-dev.image.designerflows=image-registry.openshift-image-
registry.svc:5000/${NAMESPACE}/ibm-ace-designer-flows-dev:${RELEASE_TAG} \
  --set ibm-ace-server-dev.image.configurator=image-registry.openshift-image-
registry.svc:5000/${NAMESPACE}/ibm-acecc-configurator-dev:${RELEASE_TAG} \
  --set image.fireflyFlowdocAuthoring=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-app-connect-flowdoc-authoring-prod:${RELEASE_TAG} \
  --set image.fireflyRuntime=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-app-connect-runtime-prod:${RELEASE_TAG} \
  --set image.fireflyFlowTestManager=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-app-connect-flow-test-manager-prod:${RELEASE_TAG} \
  --set image.connectorAuthService=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-app-connect-connector-auth-service-prod:${RELEASE_TAG} \
  --set image.fireflyUi=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/ibm-
app-connect-ui-prod:${RELEASE_TAG} \
  --set image.proxy=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/ibm-app-
connect-proxy-prod:${RELEASE_TAG} \
  --set image.configurator=image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/
ibm-acecc-configurator-prod:${RELEASE_TAG} \
  --set couchdb.image.repository=image-registry.openshift-image-registry.svc:5000/
${NAMESPACE}/ibm-couchdb2 \
  --set global.designerFlowsOperationMode=all \
  --set global.appConnectSecret=${appConnectSecret} \
  --set global.appConnectInstanceID=${appConnectInstanceID} \
  --set global.appConnectURL=${appConnectURL} \
  --set license=accept \
  --set ssoEnabled=false
```

For a list of the fields in these commands and their default values, see the `ibm-app-connect-designer/values.yaml` file that was extracted from the IBM Passport Advantage package. For an explanation of what the fields represent, see the `ibm-app-connect-designer/values-metadata.yaml` file.

5. Run the following command to check the status of the Helm release:

```
helm status releaseName --tiller-namespace=tiller
```

This command outputs some information in a NOTES section, including a command that you can run to discover the URL of the App Connect Designer instance. Look for the line that starts with this command:

```
echo && echo ...
```

6. Copy and run this command to obtain the App Connect Designer instance URL.

What to do next

You can distribute the instance URL to one or more users with the required access. To obtain the URL on subsequent occasions, repeat steps [“5” on page 16](#) and [“6” on page 16](#) in the preceding task.

Upgrading the Helm release for an App Connect Dashboard or App Connect Designer instance

You can upgrade a Helm release if you want to modify the configuration for your instance.

Procedure

To modify the configuration for the Helm release:

1. Ensure that you are logged in to your cluster.
2. Run the **helm upgrade** command with the `--set` flag for the values that you want to override or add.

For example:

- App Connect Dashboard:

```
helm upgrade MyHelmReleaseName ibm-ace-dashboard-prod \  
  --set contentServer.resources.requests.cpu=200m \  
  --tiller-namespace=tiller \  
  --reuse-values
```

You can view the configurable parameters and their default values in the `ibm-ace-dashboard-prod/values.yaml` file that was extracted from the installation package. For a description of what the parameters represent, see the `ibm-ace-dashboard-prod/values-metadata.yaml` file.

- App Connect Designer:

```
helm upgrade MyHelmReleaseName ibm-app-connect-designer \  
  --set configurator.resources.requests.cpu=200m \  
  --set log.format=basic \  
  --tiller-namespace=tiller \  
  --reuse-values
```

You can view the configurable parameters and their default values in the `ibm-app-connect-designer/values.yaml` file that was extracted from the installation package. For a description of what the parameters represent, see the `ibm-app-connect-designer/values-metadata.yaml` file.

Deleting App Connect Designer or App Connect Dashboard instances

If no longer required, you can delete an instance of App Connect Designer or App Connect Dashboard by deleting the Helm release that was created for that instance. You can delete the Helm release from the command line.

Procedure

To delete the Helm release from a command window:

1. Ensure that you are logged in to your cluster.

2. Run the following command, where *releaseName* is the name of the Helm release to be deleted:

```
helm delete releaseName --purge --tiller-namespace=tiller
```

Creating and managing flows for an API in App Connect Designer

App Connect Designer provides an authoring environment in which you can create, test, and share flows for an API. You can share your flows by using the export and import functions, or by adding them to an Asset Repository for reuse.

About this task

When you create flows in your App Connect Designer instance, connectors are used to establish connections and interact with the target applications that are referenced in the flows.

Depending on the configuration enabled for flows in your App Connect Designer instance, you can use *local connectors* only, or both local and *cloud-managed connectors* to run the API operations that are defined in a flow.

- A local connector is deployed locally within the cluster. For such connectors, you can specify connection credentials for a target application directly within the UI in your App Connect Designer instance.
- A cloud-managed connector is hosted within an instance of App Connect on IBM Cloud. To use these connectors, you must have access to an App Connect on IBM Cloud instance that is associated with your App Connect Designer instance.

Accessing your App Connect Designer instance

To create and manage flows for an API in your cluster, you require an instance of App Connect Designer. An App Connect Enterprise team administrator typically creates this instance after IBM Cloud Pak for Integration or IBM App Connect Enterprise certified container is installed.

Before you begin

- If both local and cloud-managed connectors were enabled while creating the App Connect Designer instance, you'll require access to the App Connect on IBM Cloud instance that was assigned as the host for the cloud-managed connectors. If you are not the owner of this App Connect on IBM Cloud instance, you'll require Developer access to the **Cloud Foundry** org and space that the instance belongs to.

For more information about creating an App Connect Designer instance and allocating access to an App Connect on IBM Cloud instance, see:

- [Application integration Designer deployment](#) in the IBM Cloud Pak for Integration documentation in IBM Knowledge Center
 - [Creating an instance of App Connect Designer in IBM App Connect Enterprise certified container](#)
 - [Inviting users to an account](#) in the IBM Cloud "*Managing identity and access*" documentation
- If you are using IBM Cloud Pak for Integration, obtain the URL of the Platform Navigator, the name of the App Connect Designer instance, and the namespace from your administrator.

- If you are using App Connect Enterprise certified container:
 - Obtain the URL of your App Connect Designer instance and corresponding Helm release name from your administrator.
 - If you'd like to run commands against resources in your cluster, ensure that these command-line interface (CLI) tools are installed on your local workstation:
 - **OpenShift Container Platform CLI (oc):** For installation instructions, see [Getting started with the CLI](#) in the Red Hat OpenShift documentation.
 - **Helm CLI (helm):** For installation instructions, see [Introduction to Helm](#) in the Helm documentation.
 - **Kubernetes CLI (kubectl):** For installation instructions, see [Install and Set Up kubectl](#) in the Kubernetes documentation.

About this task

- [“IBM Cloud Pak for Integration only: Accessing your App Connect Designer instance” on page 18](#)
- [“App Connect Enterprise certified container only: Accessing your App Connect Designer instance” on page 20](#)

IBM Cloud Pak for Integration only: Accessing your App Connect Designer instance

Procedure

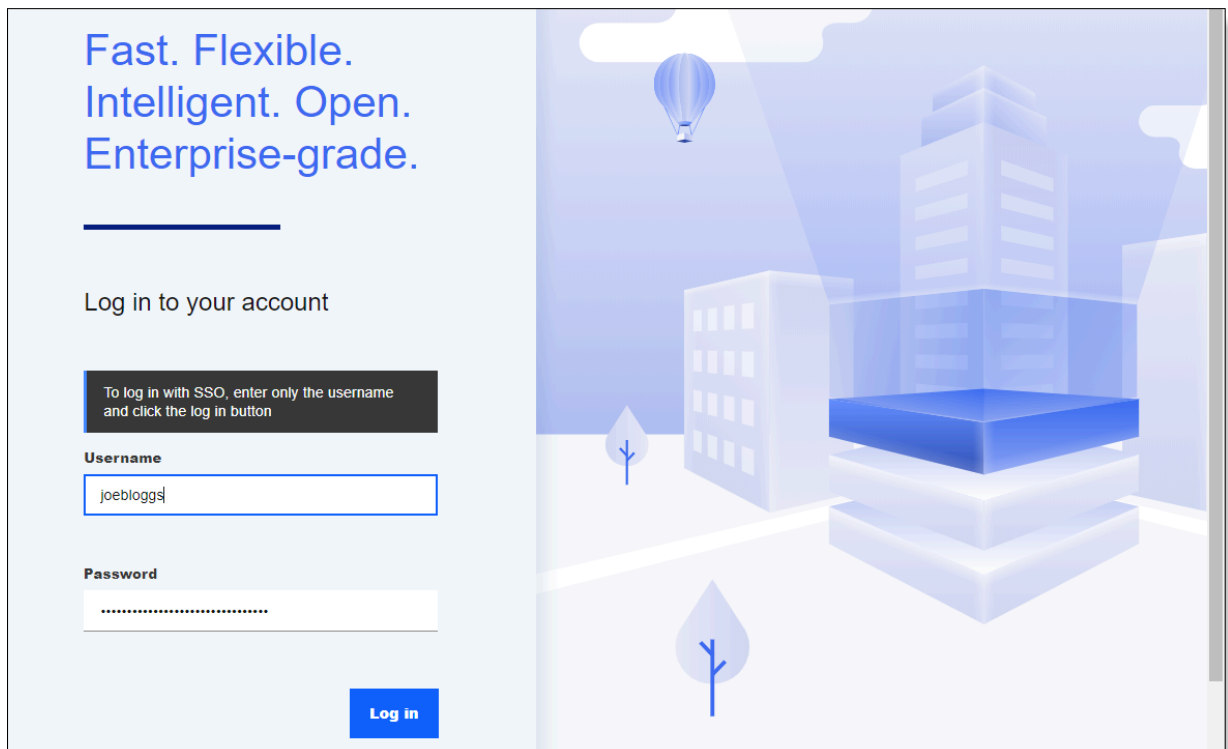
To access your App Connect Designer instance.

1. From a browser window, specify the URL of the Platform Navigator:

For example:

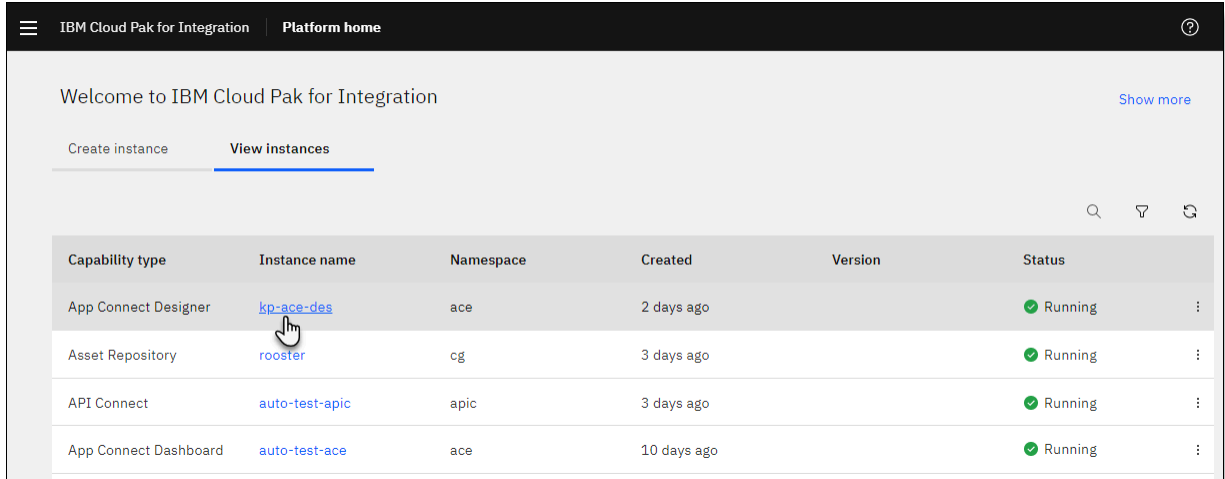
`https://ibm-icp4i-prod-integration.apps-value.domain-value`

2. Specify a valid user name and password, and then click **Log in**.



The IBM Cloud Pak for Integration Platform Navigator opens.




3. Click **View instances** and then click the name of the App Connect Designer instance that you want to open.

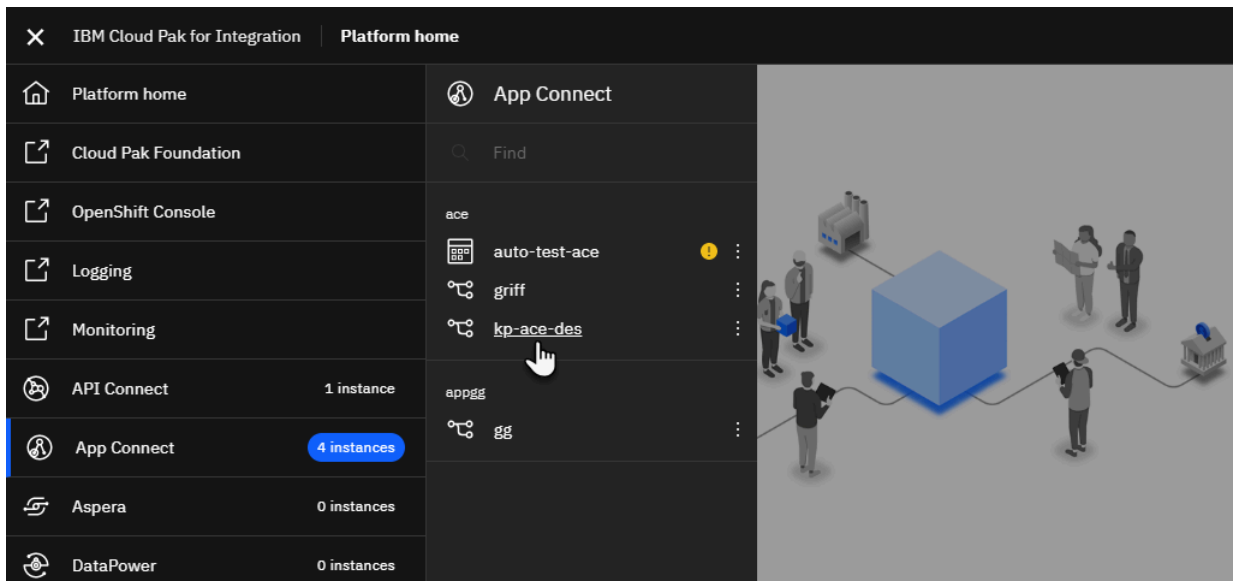


Welcome to IBM Cloud Pak for Integration Show more

Create instance View instances

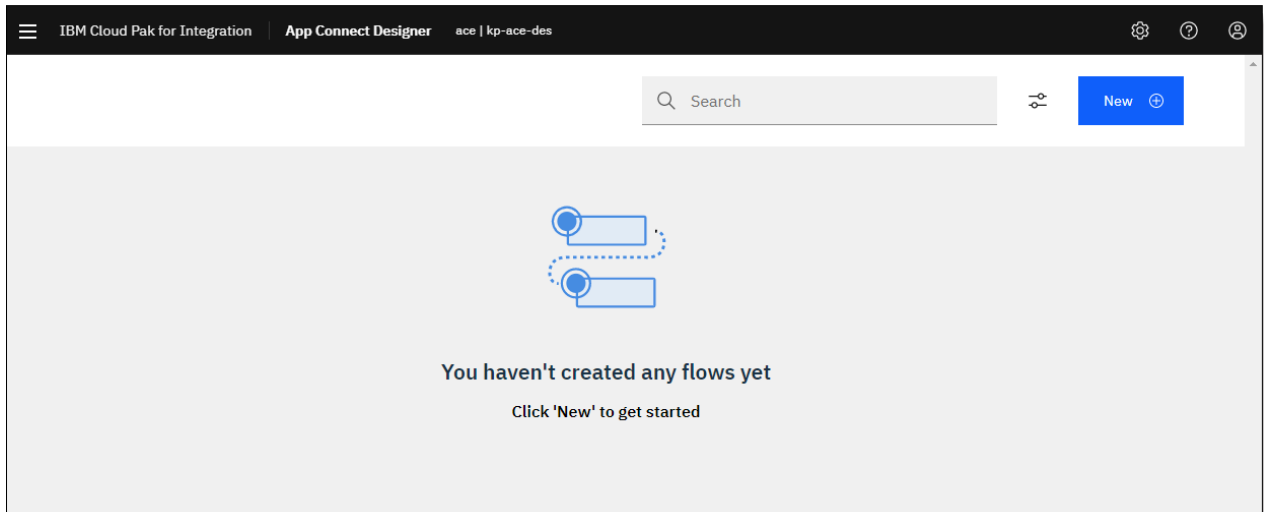
Capability type	Instance name	Namespace	Created	Version	Status
App Connect Designer	kp-ace-des	ace	2 days ago		Running
Asset Repository	rooster	cg	3 days ago		Running
API Connect	auto-test-apic	apic	3 days ago		Running
App Connect Dashboard	auto-test-ace	ace	10 days ago		Running

Tip: You can also open the App Connect Designer instance from the IBM Cloud Pak menu  in the Platform Navigator. Open the menu, click **App Connect** and then click the name of the instance under your namespace. The **App Connect Dashboard** icon  and **App Connect Designer** icon  are used to depict the instance type.



Results

The App Connect Designer window opens. If any flows have been created, they are displayed as tiles in the dashboard.



App Connect Enterprise certified container only: Accessing your App Connect Designer instance

Procedure

To access your App Connect Designer instance, complete either of the following steps:

- From a browser window, enter the URL that your administrator provided for the App Connect Designer instance, and log in when prompted.
- Run the following commands to obtain the URL and then access your App Connect Designer instance:

- a) Log in to your cluster by running the **oc login** command and provide the OpenShift Container Platform server URL (and optionally a token). Examples:

```
oc login https://Mycluster_hostName:8443
```

```
oc login --token=AbcdE1fgHijKLM2moP3q4rs5TU6vw7xYz --server=https://  
Mycluster_hostName:6443
```

- b) Indicate whether to use insecure connections, and then specify a user name and password if required.
- c) Run the following command, where *releaseName* is the Helm release name of the instance:

```
helm status releaseName --tiller-namespace=tiller
```

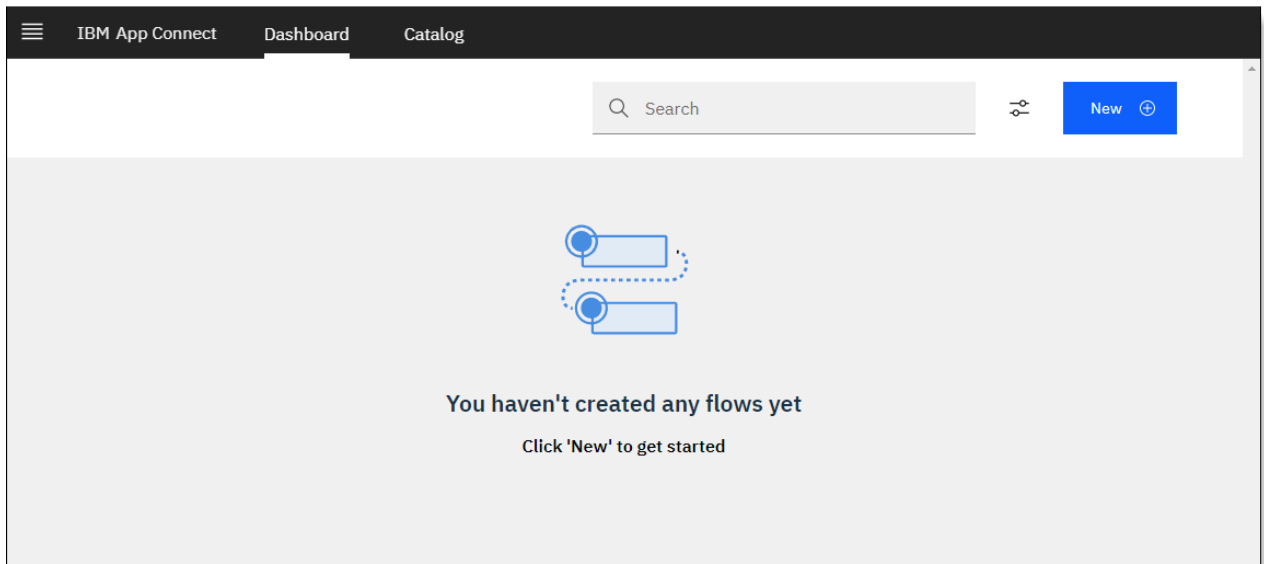
This command outputs some information in a NOTES section, including a command that you can run to discover the App Connect Designer instance URL. Look for the line that starts with this command:

```
echo && echo ...
```

- d) Copy and run this command to obtain the App Connect Designer instance URL.
- e) Copy and paste the URL into a browser window, and log in when prompted.

Results

The App Connect Designer window opens. If any flows have been created, they are displayed as tiles in the dashboard.



Creating flows for an API

You can create *flows for an API* (alternatively referred to as an *API flow*) in your App Connect Designer instance. The defined configuration provides an API that exposes one or more operations that enable you to call out to an endpoint and pass data between that endpoint and applications in the flow.

Before you begin

- To create flows for an API, you must have access to an App Connect Designer instance in your cluster.
- If your administrator has enabled the use of both local and cloud-managed connectors, you must also have access to the App Connect on IBM Cloud instance that hosts the cloud-managed connectors.
- If you know which applications you want to interact with, open the App Connect Designer catalog and create accounts for the local or cloud-managed connectors that will run API operations against the applications. (It is also possible to create accounts while creating a flow.) For more information, see [“Connecting to accounts”](#) on page 33.

About this task

When you create flows for an API, each individual flow is the implementation for an API operation (such as 'GET order' or 'POST order') that is typically invoked from mobile and web applications. The flow for each operation contains a request, actions for one or more applications, optional toolbox nodes for specialized data processing, and a response for the API operation. The request uses a model that you define to request the creation, retrieval, or replacement of data objects in your applications. When the request is submitted, each target application performs its action, and then the flow returns a response that either confirms that the actions were successful, or returns the data that was requested.


Defining the API involves the following steps:

1. Create one or more models that define the structure of the objects that you want to create or retrieve. Up to 10 models are allowed.
2. Choose the built-in create, retrieve, or replace or update operations to perform against each model, or define your own custom operations.
3. Configure a flow that implements each operation, adding actions for one or more target applications. You must connect App Connect to each of these application by providing account details for the applications. Depending on how your App Connect Designer instance was configured, you can use either local or cloud-managed connectors to establish the connection. For more information, see [“Connecting to accounts”](#) on page 33.

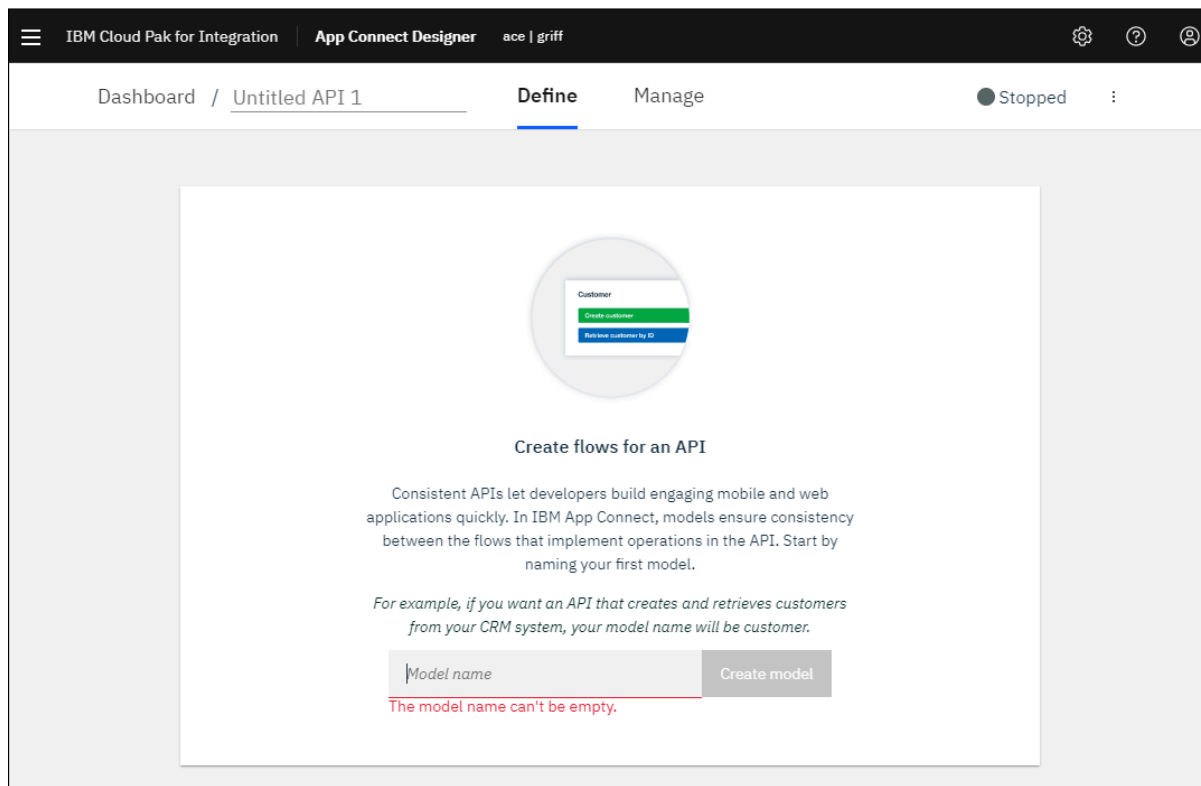
You can also add toolbox utilities to the flow for an operation. For the list of supported utilities, see [“Supported components for flows”](#) on page 41.

Procedure

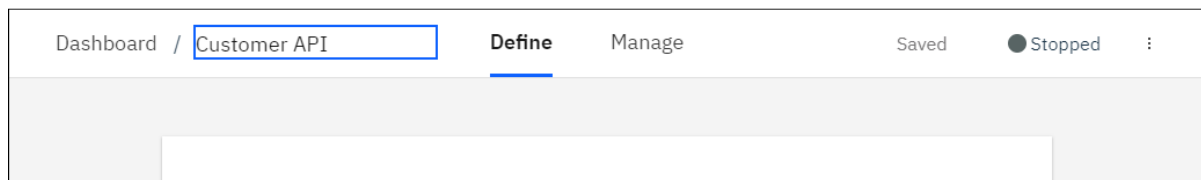
To create an API flow:

1. From the App Connect Designer UI, open the App Connect Designer dashboard if not currently on display:
 - **Applicable to IBM Cloud Pak for Integration only:** Click the settings icon  in the banner and then click **Dashboard**.
 - **Applicable to App Connect Enterprise certified container only:** Click the **Dashboard** tab.
2. From the App Connect Designer dashboard, click **New > Flows for an API**.

The API editor opens with a **Define** tab and a **Manage** tab in view. You create models and operations for the API in the **Define** tab. After you start the API, you can then use the **Manage** tab to obtain an auto-generated base URL for the API, and a user name and password that can be specified when invoking the API operations.



3. Enter a name that identifies the purpose of your flow.



4. Create a model that defines the structure of an object that you want to perform operations on; for example, a customer object for which records can be created, retrieved, or updated.
 - a) In the **Model name** field, enter a name for your model.
 - b) Click **Create model**.

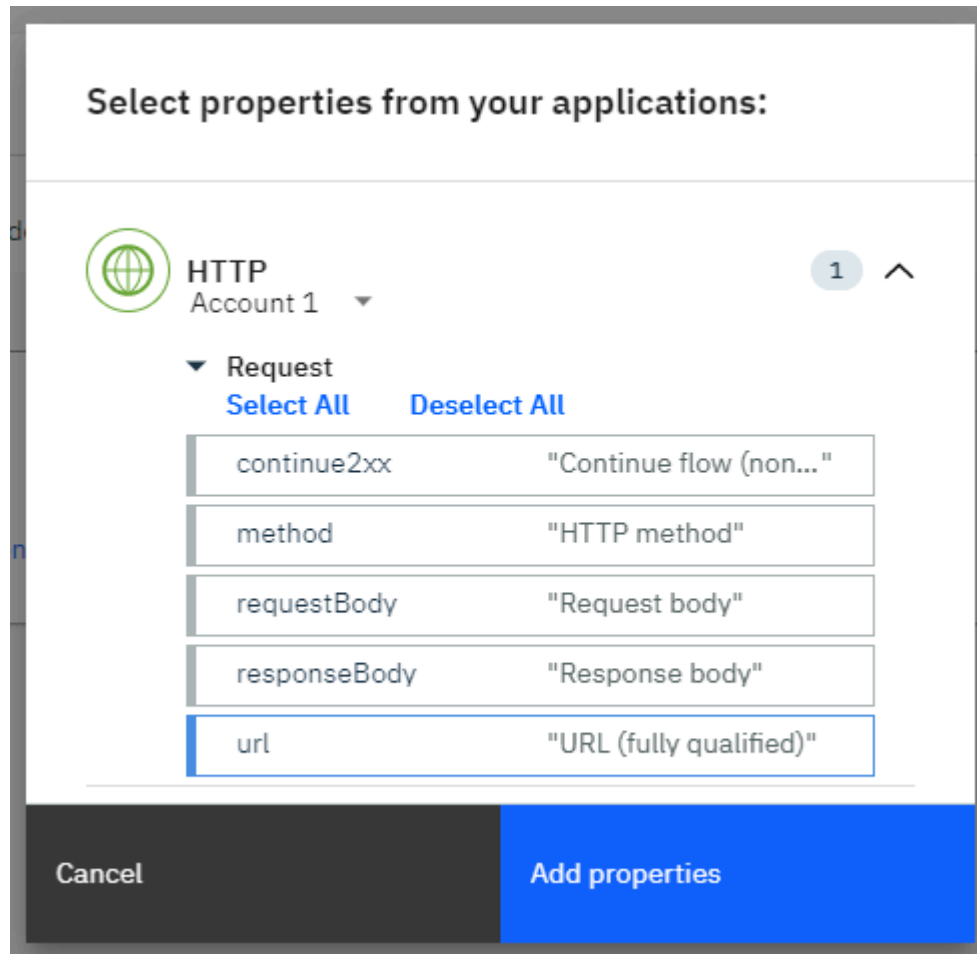
The model panel is displayed with two tabs: **Properties** and **Operations**.

5. From the **Properties** tab, define the structure of the model.
 - a) To add the first property, type the name in the field provided and then select a data type for that property.

By default, the first property that you add has the **ID** option selected. A property that is set as the ID for the model indicates that your flow must return this property when creating an object or that

the property must be sent in a request to update or retrieve an object by using its ID. You must set an ID against one property in order to define operations for the model.

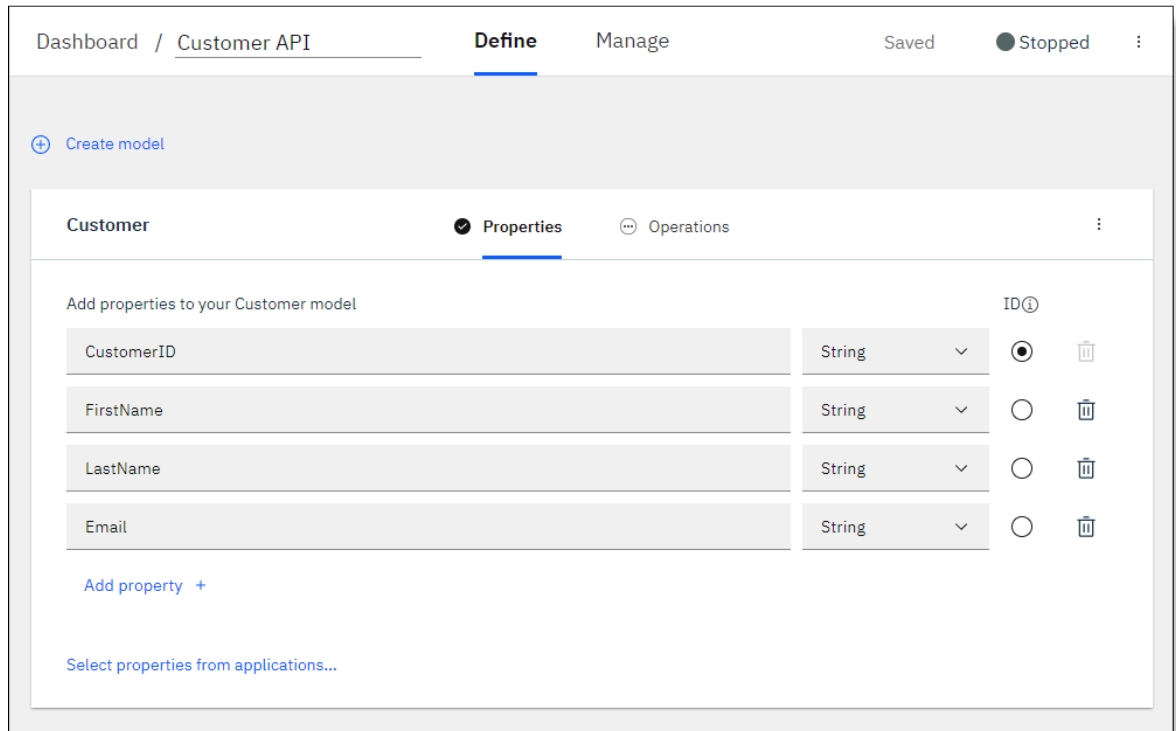
Tip: To add a property, you can also click **Select properties from applications** to choose properties from one or more of the applications that you're connected to. Ensure that the required account is selected. Then select one or more properties and click **Add properties**.



b) Add other properties for the model by clicking **Add a property +** .

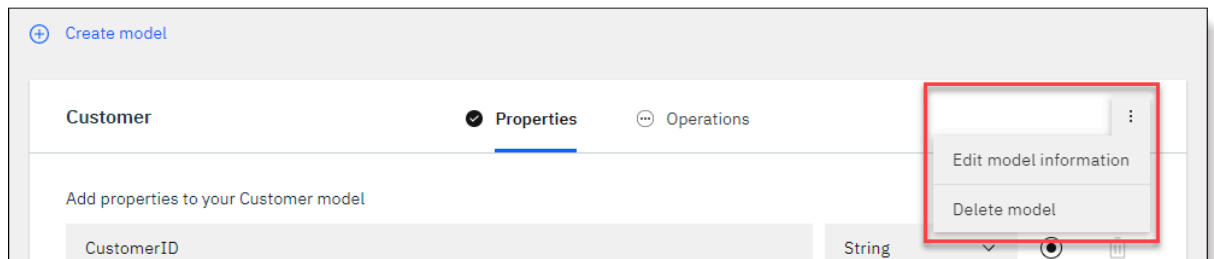
Note:

- Each property name must be unique.
- Spaces are not allowed in the name, but you can use an underscore character (_) to separate words.
- The name must contain only letters, numbers, or the underscore character.
- The name must begin with a letter or an underscore.
- The name must contain at least two characters.



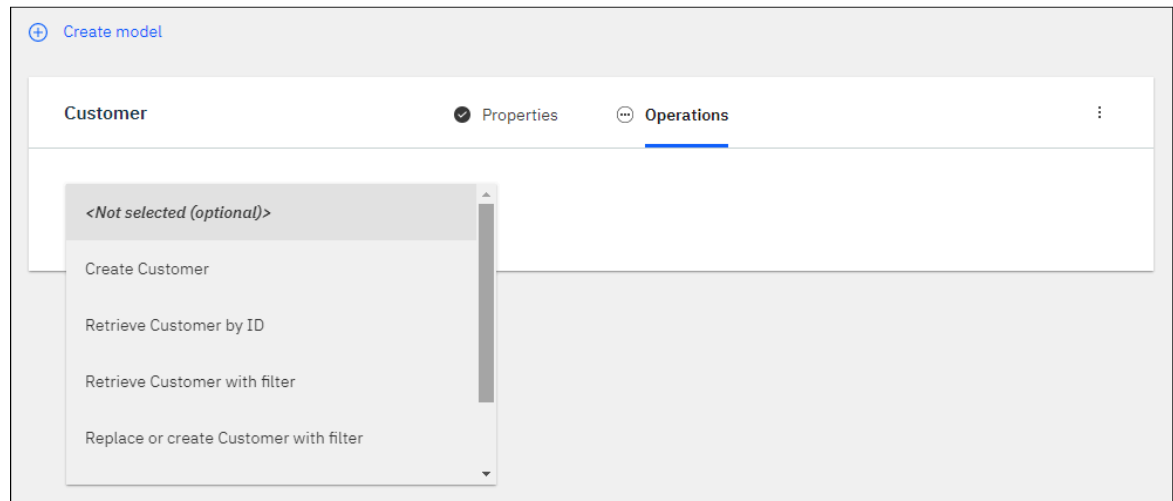
6. Create more models by clicking **Create model** and add properties for each new model as described in step “5” on page 23.

If you want to edit the name of a model, you can select **Edit model information** from the menu. To delete a model, you can select **Delete model**.



7. To define how the API will interact with the model, click **Operations**.

- To add one of the built-in operations, click within the **Select an operation to add** drop-down list, and then choose a **Create**, **Retrieve**, or **Replace or create** operation to perform against the model.



- If you choose an operation that enables you to apply filters (for example, **Retrieve *model_name* with filter** or **Replace or create *model_name* with filter**), select the required filter properties. For the **Retrieve *model_name* with filter** operation, optionally enable and configure pagination

settings for the results. (To apply filters, at least two properties, including the "ID" property for the model, must be defined because the "ID" property cannot be selected as a filter.)

Retrieve Customer with filter GET /Customer?Email=SampleEmail Implement flow

Select filter properties for Customer to be retrieved: 1

FirstName

LastName

Email

Enable pagination for retrieved Customer ⓘ

Paginate results using:

'skip' and 'limit' parameters

'token' and 'limit' parameters

Select an operation to add

For more information about using these operations, see [Introducing filter parameters for API flows in IBM App Connect](#) and [Configuring pagination for the 'Retrieve with filter' operation in API flows](#).

- To define your own operations, click within the **Select an operation to add** drop-down list, and then choose **Add a custom operation**. The following restrictions apply:
 - The operation name cannot be any of these keywords: create, updateOrCreate, all, updateAttributes, update, updateAll, upsertWithWhere, replaceOrCreate, replaceById, destroy, destroyAll, or executeAssembly.
 - The query parameter cannot be the same as the model ID.

For more information about adding custom operations see, [Create your own custom HTTP operations on API flows](#).

<DisplayName> HTTP Verb /Users/<Operation name> Implement flow

*Display name Display Name

Description Description

*HTTP verb Select

Use Users ID ⓘ

*Operation name ⓘ Operation name

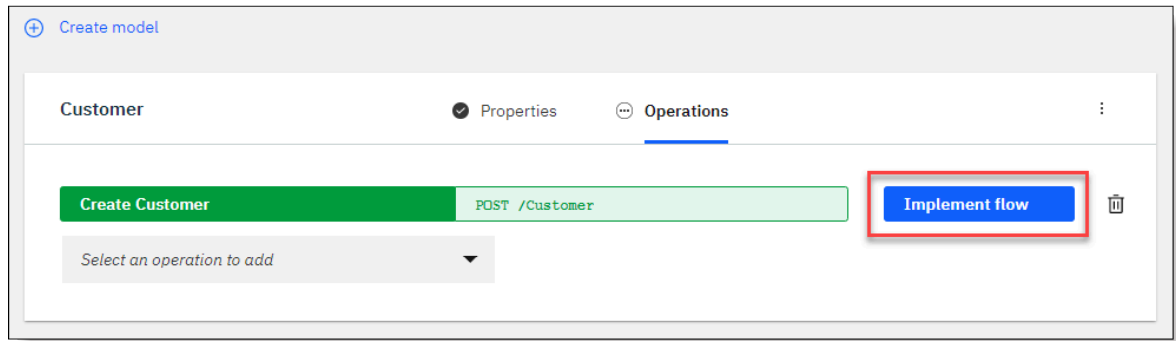
Query parameters (+) Add query parameter

*Request body Select

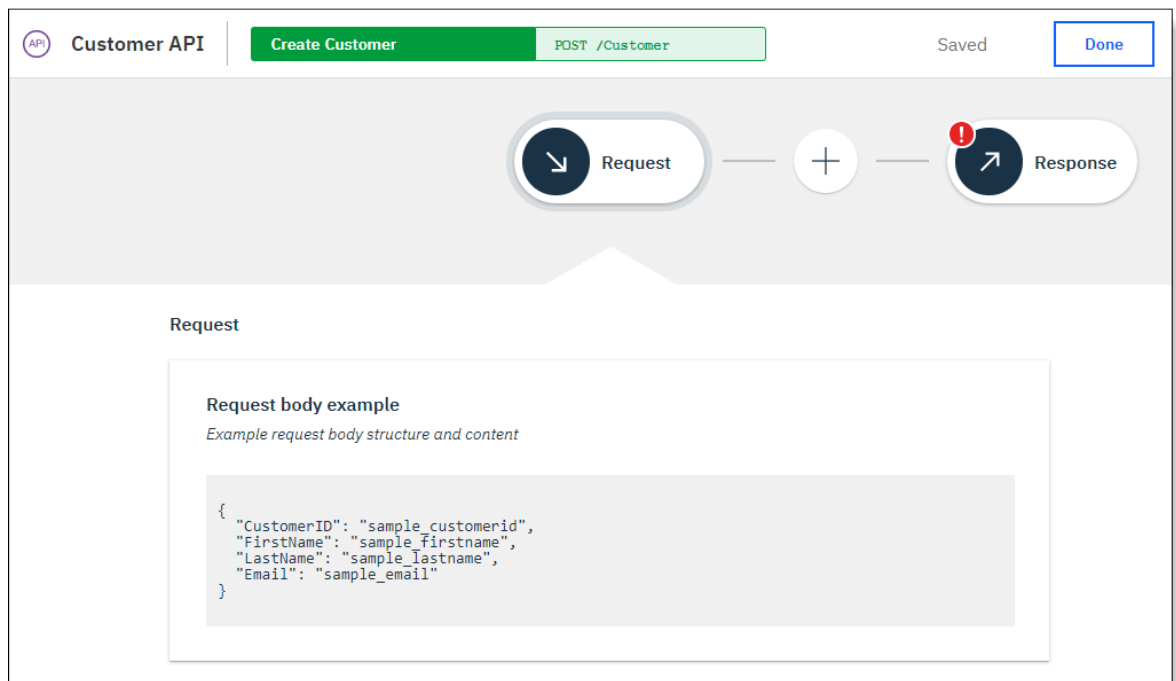
*Response body Select

8. Configure a flow that implements the operation:

a) Click **Implement flow**. For example:



You'll see a basic flow structure in the flow editor, with a **Request** node, a **Response** node, and a space to add one or more target applications. Notice the structure of the Request body example - it is constructed from the properties in your model, with some sample data.

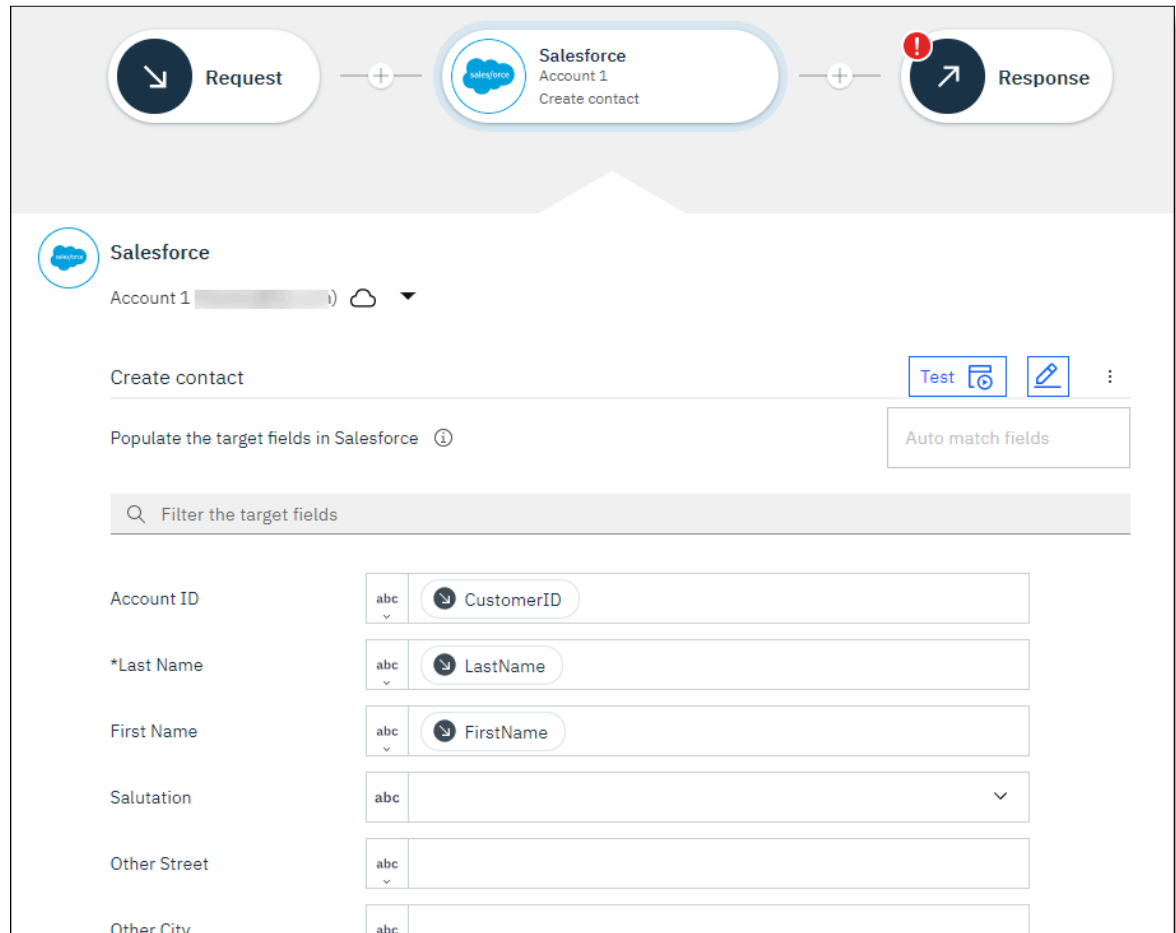


b) To add an application to the flow, click **(+)** and then select the application and action. Also ensure that the correct account, which App Connect will use to connect to the application, is selected. If

there are no connected accounts, you can create one for local or cloud-managed connectors, as described in [“Connecting to accounts”](#) on page 33.

- c) Populate the fields for the action with text, mapped fields from previous nodes in the flow, or JSONata expressions.

For more information, see [Completing the fields for an action](#).



- d) Optional: Add further applications if required.
- e) Optional: Use one or more supported toolbox utilities to provide specialized processing. For example, you can add an If node to provide conditional processing, or a For each node to process retrieved items. For more information, see [Toolbox utilities for specialized processing](#).
- f) Click the **Response** node in the flow to define the response that should be returned when the operation completes successfully.
- g) In the **Response header** section, specify your preferred response status code.
The following response codes are returned for the different operations:
- Create operations return a response code of 201 (record created).
 - Retrieve operations return a response code of 200 (record retrieved).
 - Replace or create operations return a response code of 200 (record replaced) or 201 (record created).
- h) In the **Response body** section, define which fields should be returned in the response body by using text, mapped fields from previous nodes in the flow, or JSONata expressions.

+

Salesforce

Account 1

Create contact

+

↗

Response

Response

Populate the successful response message

Response header ⓘ

*Status Code 123 201 ⋮ fx

Response body

*CustomerID abc Contact ID

FirstName abc

LastName abc

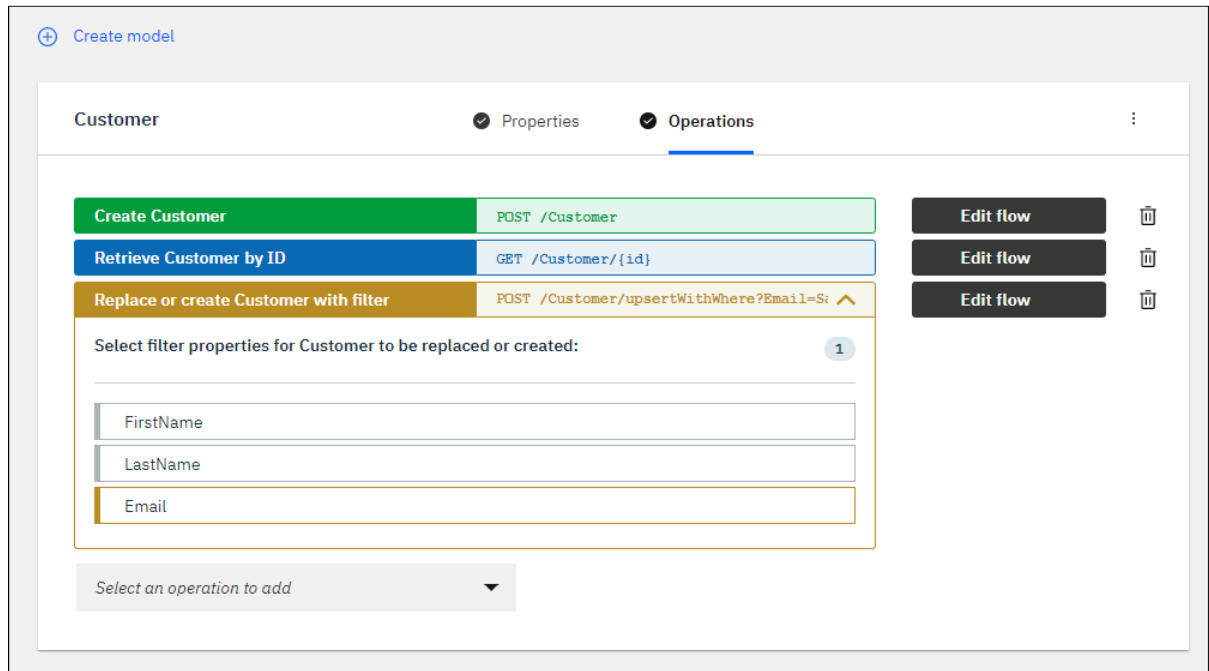
Email abc

Tip: When you're creating an object, typically only the ID from the target application is returned in the response message. If you're retrieving an object, the response message will show you all the fields that you've requested from the target application.

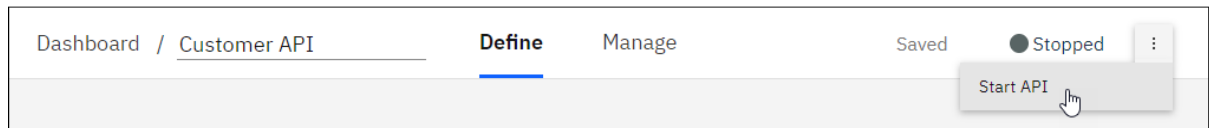
- i) Click **Done** to return to your model.

9. Define further operations for the model and configure a flow that implements each operation, as described in steps “7” on page 26 and “8” on page 28.

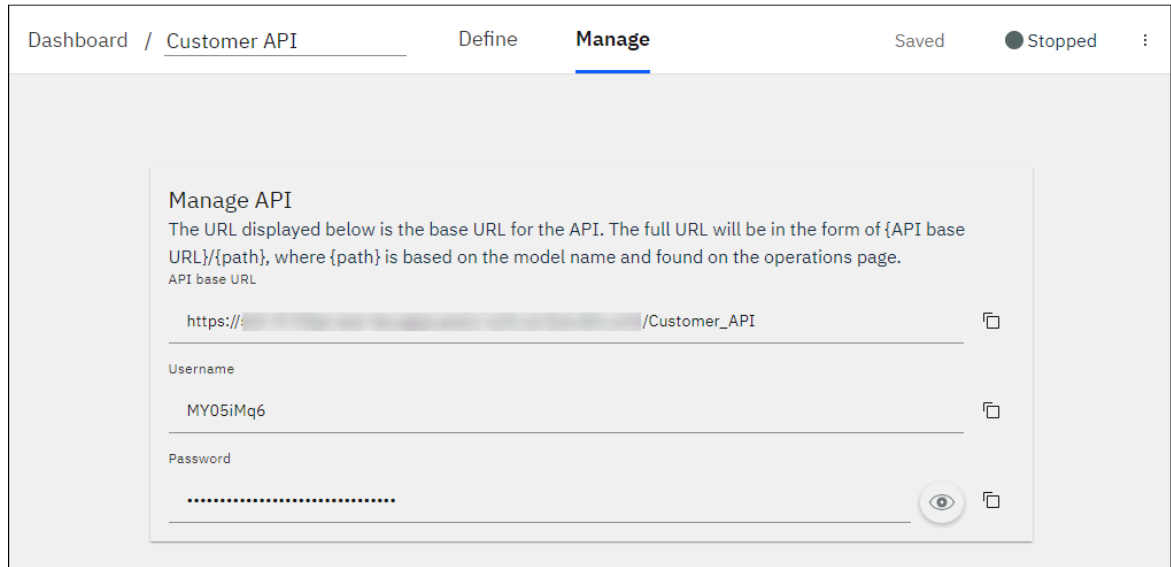
For example:



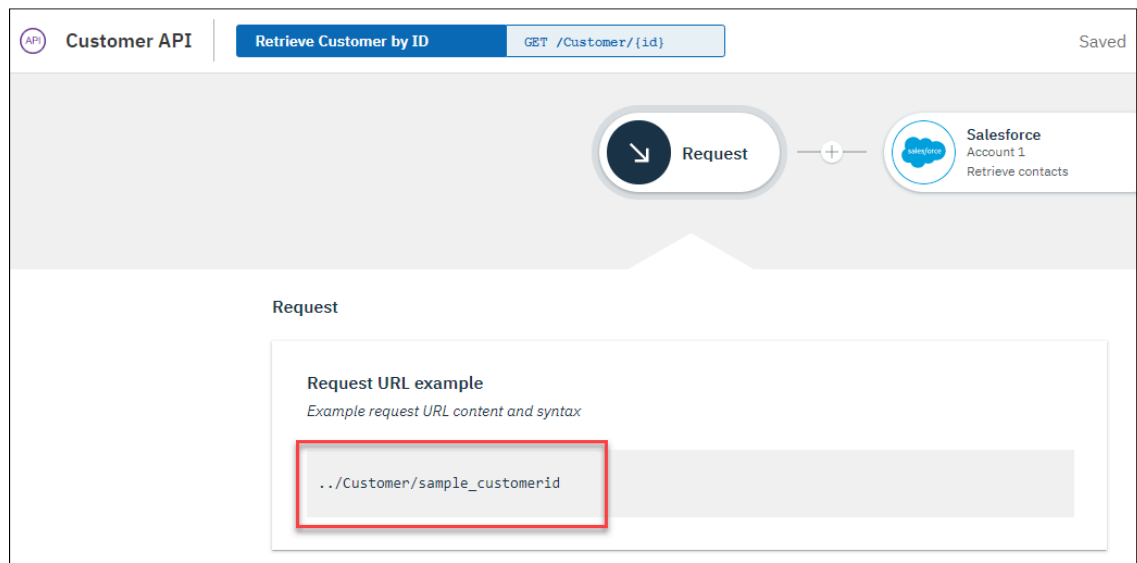
10. Define operations for any additional models and implement their flows.
11. Ensure that there are no validation errors in any of the implemented flows for the operations.
For more information about validating the nodes in a flow and resolving errors, see [Validating your flow is ready to run](#).
12. From the options menu on the **Define** tab, click **Start API** to save and start your flow.



13. Test your API by using your preferred method for invoking APIs, such as Postman, the [HttpRequester](#) add-on for Firefox, or curl.
- a) Click the **Manage** tab and make a note of the base URL for the API, and the user name and password that are generated.



- b) Open your preferred testing utility.
- c) To test each operation:
- 1) Specify the relevant HTTP method.
 - 2) Construct the request URL by appending the path segment for the operation to the base URL that is shared by all operations in your API. The full URL will be in the form *API_base_URL/operation_path_segment*. You can obtain the path segment for the operation from the **Request** node.



- 3) Specify the user name and password from the **Manage** tab.

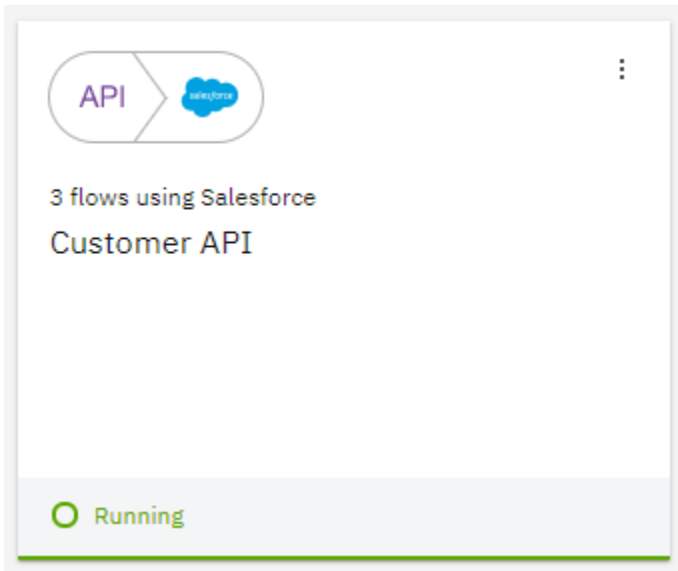
For example, if the base URL is `https://someslocation.abc.com/Customer_API` and the path segment for the operation is `/Customer/sample_customerid`, where `sample_customerid` represents the ID of a contact record to be retrieved, the request URL can be specified as:

`https://someslocation.abc.com/Customer_API/Customer/sample_customerid`

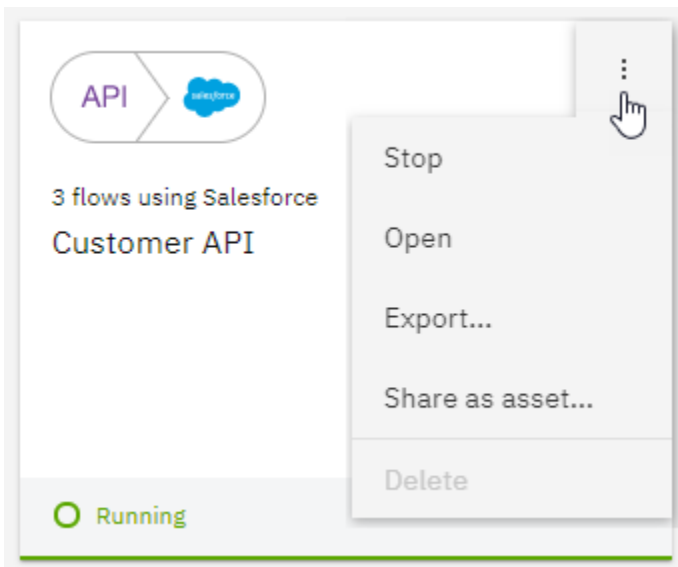
- d) Invoke the API operation and then check for the expected results in the target application.

What to do next

Click **Dashboard** to return to the dashboard. You should be able to see the tile for your API flow.



From the flow menu, you can start, stop, export, or delete the API flow, and can choose to share it with other users as an asset if an Asset Repository instance is available in your cluster. While the API flow is running, you can also open it to view the configuration, but you'll have to stop the flow if you want to edit it.



Connecting to accounts

App Connect Designer provides a catalog of IBM, partner, and third-party applications for you to use in your flows. To create flows that transfer data across these applications, a connection needs to be established to each application in a flow, and you can do so by setting up accounts with the required credentials.

About this task

The account credentials that you need to provide vary by application, and you might need to work with an administrator in some cases to obtain the values. You can view the type of credentials that each application requires from the App Connect Designer Catalog page, and can review the [How-to guides for apps](#) for guidance on obtaining the values.

When you create your flows, it is possible to use local connectors only, or a combination of local and cloud-managed connectors to connect to the target applications and run the defined API operations. Your ability to use local or cloud-managed connectors depends on configuration settings that were enabled when your App Connect Designer instance was created.

- If only local connectors were enabled, you'll see only the supported subset of connectors on the Catalog page. When you try to connect, the connection fields are presented for your input.
- If both local and cloud-managed connectors were enabled, you'll see the full set of connectors. When you try to connect, you can choose whether you want to create a local or cloud account if both account types are supported for that connector, and can either then specify the local credentials or link out to the App Connect on IBM Cloud instance that hosts that connector in the cloud. If only a cloud account is supported, you are immediately presented with a panel to link out to the App Connect on IBM Cloud instance that hosts that connector in the cloud.

Note: If accounts are already configured in your App Connect on IBM Cloud instance, when you open the App Connect Designer Catalog page, those accounts are immediately shown and you can select them for use.


You can add account details from the App Connect Designer Catalog page before you create a flow, or you can do so from within the API editor of a flow that you are creating. While the connection is being established, you might be informed that App Connect wants to access your account for the application. If you see this prompt, be sure to grant App Connect access to the application.

The following instructions document how to add accounts from the Catalog page. The instructions for adding accounts while you are creating a flow are similar, except that you are prompted to connect *after* you select an application and action that you want to add to your flow.


Procedure





To add account details from the App Connect Designer Catalog page:

1. Open the Catalog page if not currently displayed:

- **Applicable to IBM Cloud Pak for Integration only:** Click the settings icon  in the banner and then click **Catalog**.
- **Applicable to App Connect Enterprise certified container only:** Click the **Catalog** tab.

For each application, the account status is displayed as follows:

- An application with no accounts is shown in a *Not connected* state.
- An application with a single local or cloud-managed account shows the account name. A cloud icon  is appended to the name if the account is detected to be cloud-managed from your App Connect on IBM Cloud instance.
- An application that contains multiple cloud-managed or local accounts will display the total number of connected accounts.

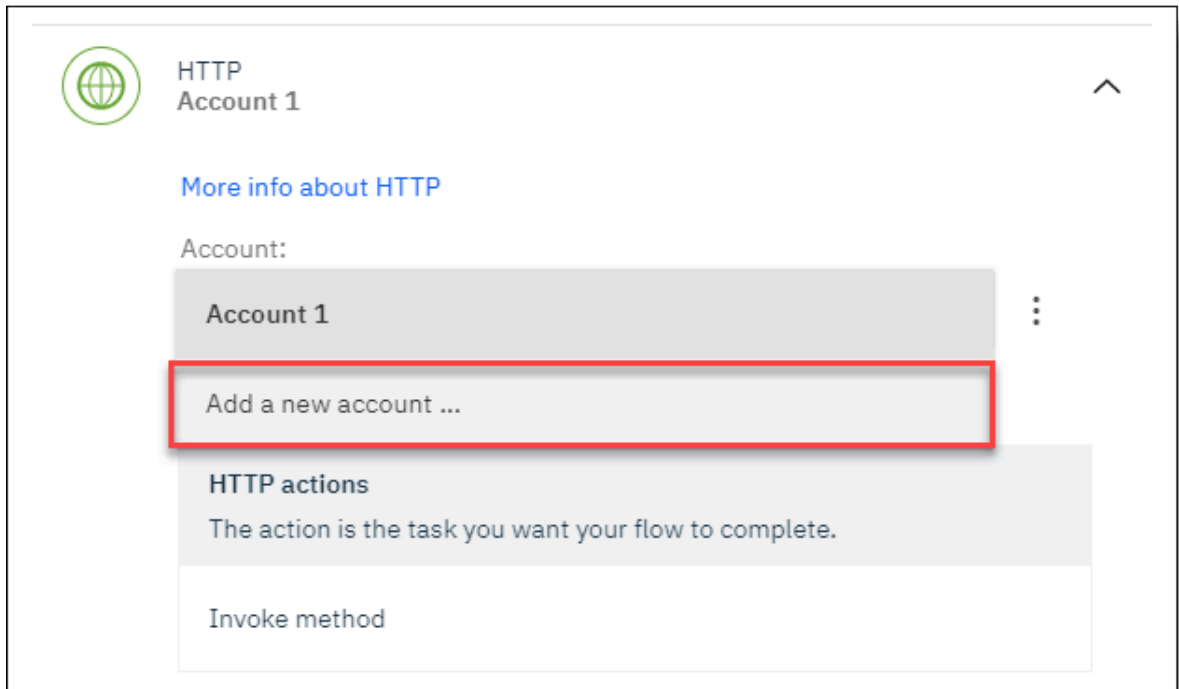
	Coupa Not connected	▼
	IBM Watson Language Translator Account 1 	▼
	IBM Watson Tone Analyzer 2 accounts	▼

2. Click to expand the application for which you want to create an account.

3. If no accounts have yet been created for this application, click the **Connect** button. Then go to step [“5” on page 37](#).

4. If an account already exists for this application and you want to add another account, complete the following steps:

a) Click the **Account** drop-down list and then click **Add a new account**.

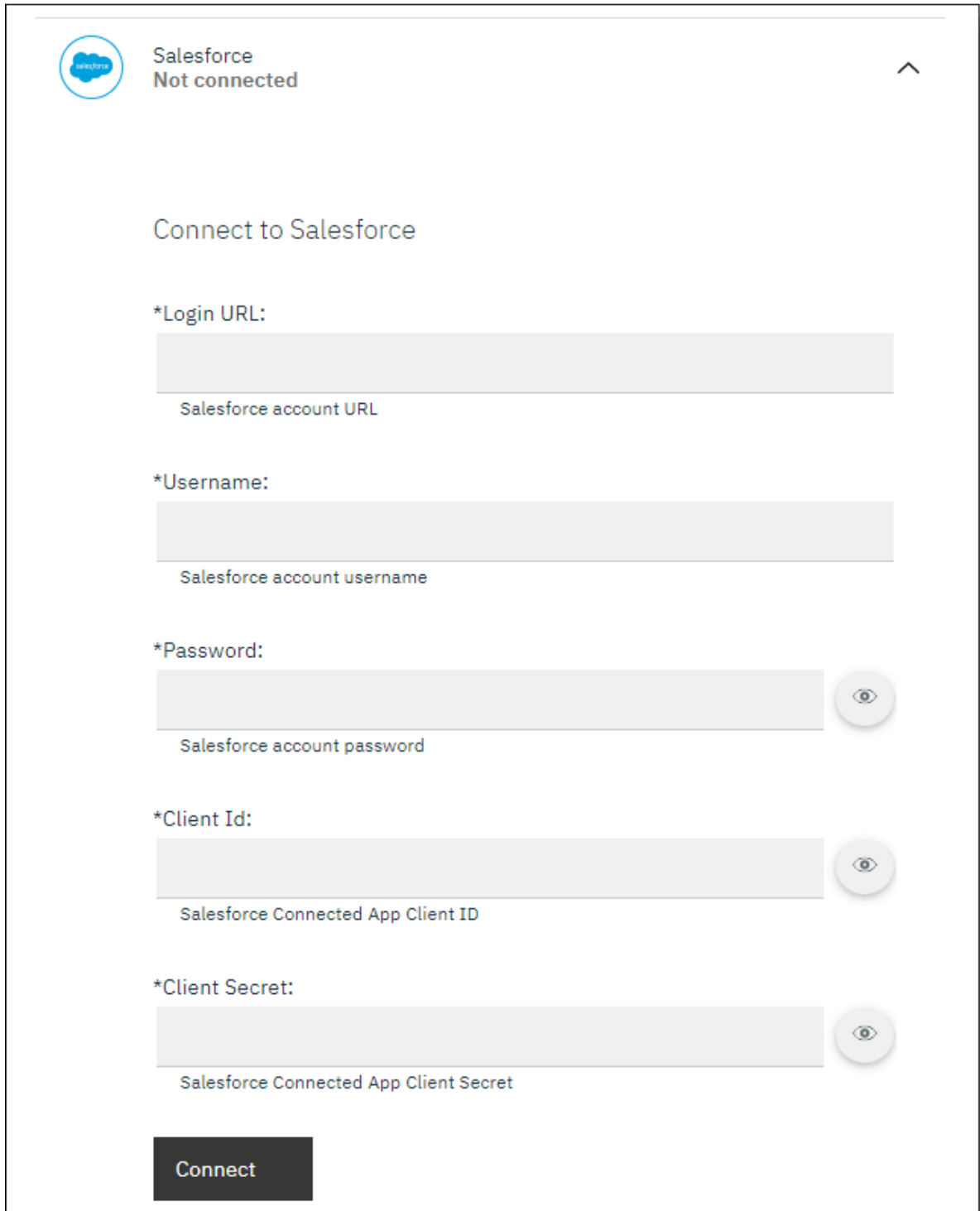


b) Go to step [“5”](#) on page 37.

5. Add connection details by completing one of the following steps:

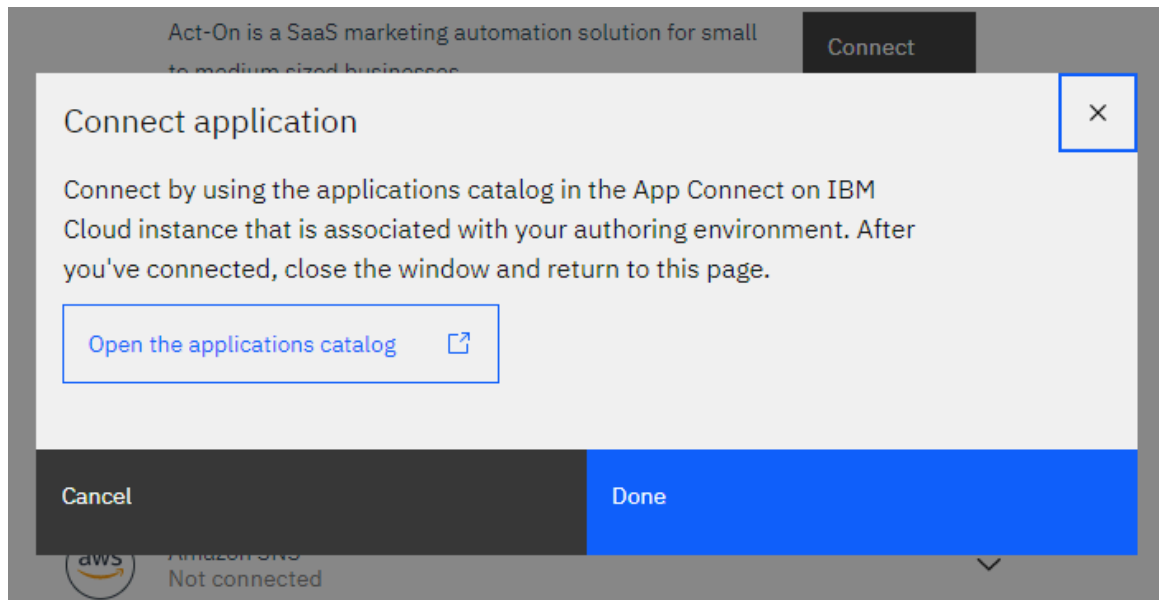
- If only local connectors are enabled for flows, the connection fields are immediately displayed. Enter the account information and then click **Connect**.

Note: Local connectors do not require the Secure Gateway connection that is typically required for cloud-managed connectors in App Connect on IBM Cloud.



The screenshot shows a web interface for connecting to Salesforce. At the top left is the Salesforce logo and the text "Salesforce Not connected" with an upward arrow icon. Below this is the heading "Connect to Salesforce". There are five input fields, each with a label and a placeholder text: "*Login URL:" with "Salesforce account URL", "*Username:" with "Salesforce account username", "*Password:" with "Salesforce account password", "*Client Id:" with "Salesforce Connected App Client ID", and "*Client Secret:" with "Salesforce Connected App Client Secret". Each of the last three fields has a toggle icon (an eye) to the right. At the bottom left is a dark "Connect" button.

- If both local and cloud-managed connectors are enabled for flows, but only cloud accounts can be created for the application, you'll need to create the account in your App Connect on IBM Cloud instance as follows:
 - a. In the "Connect application" panel that is displayed, click **Open the applications catalog**.



- b. Log in to the App Connect on IBM Cloud instance if prompted.
- c. From the App Connect on IBM Cloud Catalog page, locate and click the application for which you want to create an account, and then click the **Connect** button.
- d. Enter the account information and click **Connect** again.

aws Amazon S3 Not connected

Connect to Amazon S3

*Secret access key:

.....

The secret access key for your Amazon S3 account, as generated in the Security Credentials page in the AWS Management Console

*Access key ID:

.....

The access key ID for your Amazon S3 account, as generated in the Security Credentials page in the AWS Management Console

Region: (optional)

us-east-1

The region of your Amazon S3 instance; for example, us-east-1

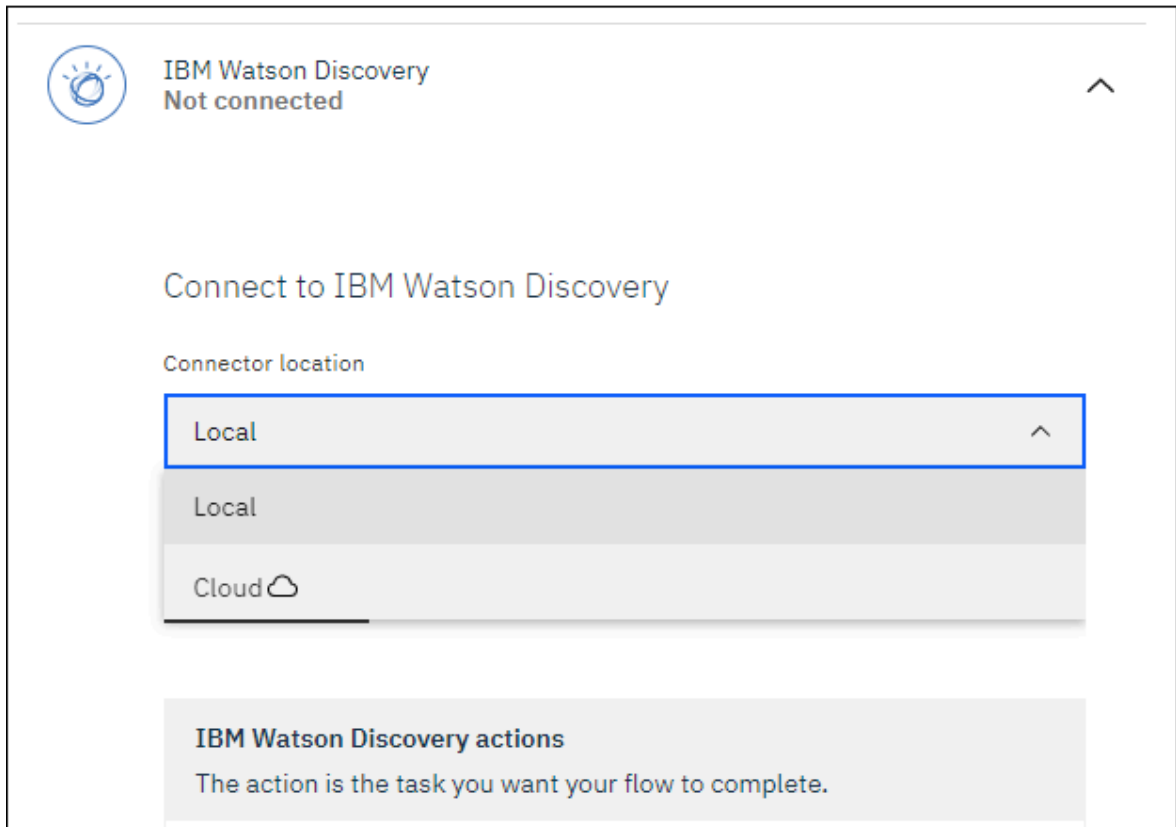
Connect

e. When it's confirmed that an account has been created, close the App Connect on IBM Cloud browser tab and return to your App Connect Designer instance. Then click **Done** in the "Connect application" panel.

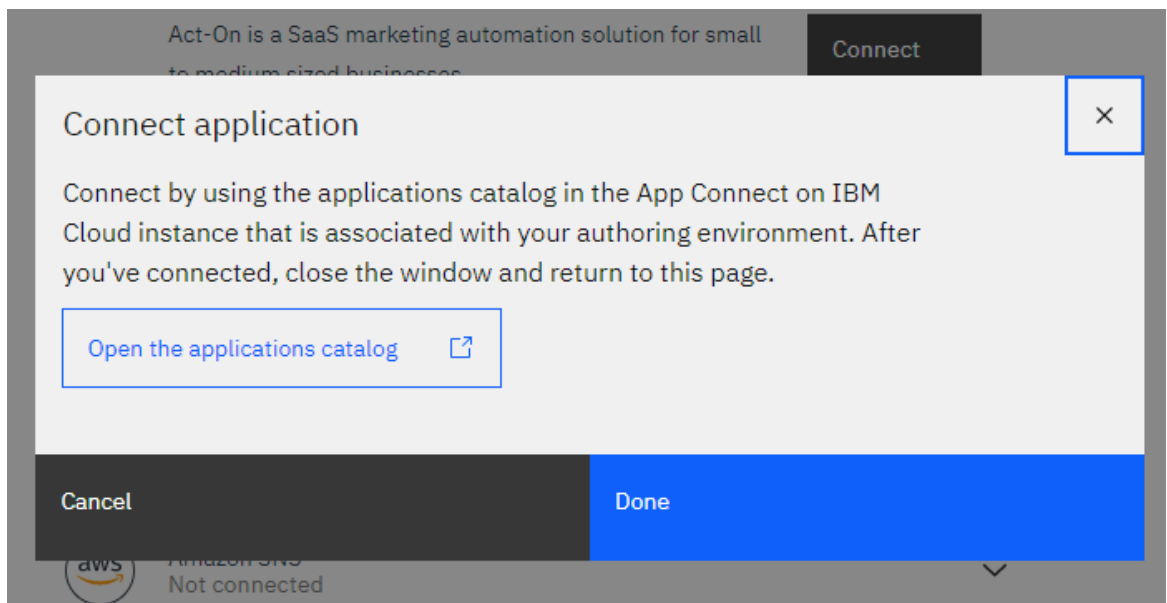
You should now see the account in the Catalog page.



- If both local and cloud-managed connectors are enabled, and both local and cloud accounts can be created for the application, you can choose the type of account that you want to create. Open the **Connector location** drop-down list and choose the required option.



- To create a local account, click **Local**. Then enter the account information in the connection fields that are displayed, and click **Connect**.
- To create a cloud account, click **Cloud** ☁. In the "Connect application" panel that appears, click **Open the applications catalog** and then create the account in your App Connect on IBM Cloud instance.



When it's confirmed that an account has been created, close the App Connect on IBM Cloud browser tab and return to your App Connect Designer instance. Then click **Done** in the "Connect application" panel. You should now see the account in the Catalog page.

Results

After you connect the first account from the Catalog page or from the API editor, a default name for the account is shown in a drop-down list below the application name.

Accounts are typically added using the following naming convention:

Account *n* (*user_identifier*)

Where:

- Account *n* represents an account name. When you add an account, its default name is given as Account *n*, where *n* is a number that starts from 1 and increments for each account added for that application. You can change the default name to make it more meaningful; for example, by assigning a name such as Test instance, US Lab, or Inventory API (1000 calls per min), which helps you identify the instance to which you are connected or the credentials used to connect.
- (*user_identifier*) provides additional contextual information such as the email address or user name that is associated with the account, or a generic label; for example: (my_email@example.com) or (username). This value varies based on the application, and might not be applicable for some applications.

Note: If you have a cloud account with the same name as a locally created account, the local account takes precedence and is the account you'll see. If you subsequently remove the local account, the cloud account becomes again.

What to do next

After you create a local account, it's good practice to rename the account for ease of identification. You cannot rename the account for an application if that account is currently being used in a flow; so it's best to rename your account immediately after you connect. To rename an account, click the Accounts options menu for that application on the Catalog page, and then select **Rename Account**.

Cloud accounts can only be managed from the App Connect on IBM Cloud host instance. For example, if you want to rename or remove an account, you must do so from the cloud instance.

After you add an account for an application, you can use that account in multiple flows. You can also add multiple accounts for an app if required, and can update your account credentials in App Connect or remove redundant accounts. For more information, see [Managing accounts in App Connect](#).

Supported components for flows

Use this information as a reference to help you identify which connectors and toolbox utilities are supported when authoring an API flow.

- [“Cloud-managed connectors” on page 41](#)
- [“Local connectors” on page 41](#)
- [“Toolbox utilities” on page 42](#)

Cloud-managed connectors

When local and cloud-managed connectors are enabled for use, the complete set of connectors for which you can set up cloud accounts is shown within the Catalog page or the API editor.

Local connectors

When local connectors are enabled for use, the following subset of connectors are available as local connectors:

- Confluence
- Domino
- HTTP
- IBM Cloudant
- IBM Food Trust

- IBM Maximo
- IBM Watson Campaign Automation
- IBM Watson Discovery
- IBM Watson Language Translator
- IBM Watson Natural Language Classifier
- IBM Watson Personality Insights
- IBM Watson Tone Analyzer
- IBM Watson Visual Recognition
- IBM Weather Company Data Limited Edition
- Jira
- Microsoft SharePoint
- Redis
- Salesforce
- ServiceNow
- Workday

Toolbox utilities

Any of these toolbox utilities can be included in the flow:

- If
- For each
- Log
- Set variable
- JSON parser
- CSV parser
- XML parser

Exporting and importing API flows

You can use the export and import functions in App Connect Designer to share API flows with other users who want to configure identical or similar integrations.

About this task

To share an API flow, you must first export it from the App Connect Designer dashboard as a YAML or BAR file. Users with access to the exported flow can then import it into their own environments.

Note: If using IBM Cloud Pak for Integration, you can also share your flows by adding them as assets to the Asset Repository, which provides a centralized location for storing, managing, and sharing integration assets across capabilities. For more information, see [“IBM Cloud Pak for Integration only: Creating and reusing assets in the Asset Repository”](#) on page 46.


- [“Exporting an API flow”](#) on page 42
- [“Importing an API flow”](#) on page 44

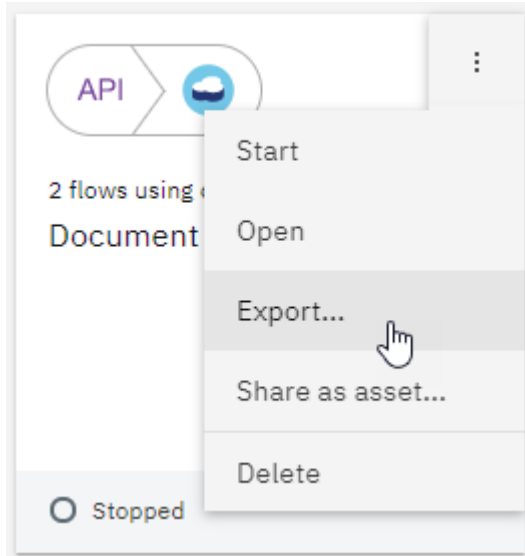
Exporting an API flow

When you export an API flow, all its configuration settings, other than the connection credentials for your accounts, are preserved in the exported definition.

Procedure

To export a flow:

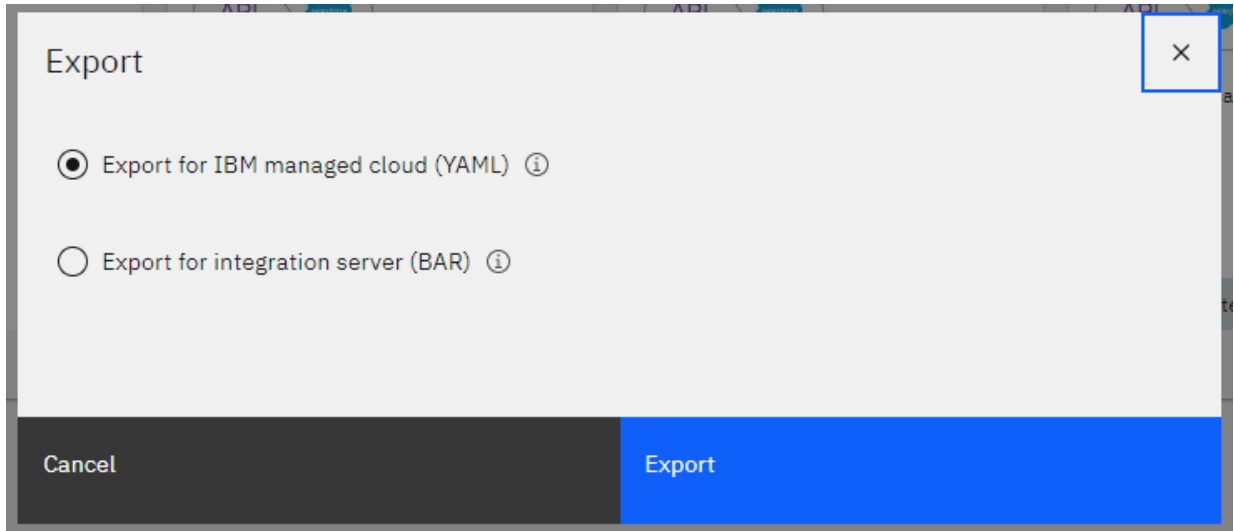
1. From the App Connect Designer UI, open the App Connect Designer dashboard if not currently on display:
 - **Applicable to IBM Cloud Pak for Integration only:** Click the settings icon  in the banner and then click **Dashboard**.
 - **Applicable to App Connect Enterprise certified container only:** Click the **Dashboard** tab.
2. From the App Connect Designer dashboard, locate the API flow, open its menu, and then click **Export**. The flow can be in a Stopped or Running state.



Tip: Avoid exporting flows with an Incomplete status or configuration errors that will need to be investigated after an import. Errors that are related to your account credentials for example, do not have an effect on the exported flow.

3. Choose the required option:
 - Click **Export for IBM managed cloud (YAML)** to export the flow as a YAML file that can be subsequently imported into the same or a different App Connect Designer instance. For more information about importing a flow into an App Connect Designer instance in a cluster, see [“Importing an API flow”](#) on page 44.
 - Click **Export for integration server (BAR)** to export the flow as a BAR file that can be subsequently deployed into an App Connect Enterprise dashboard as an integration server with an API integration. To export a Designer-authored API flow as a BAR file, the API flow must meet a set of export requirements. For more information about the considerations for exporting an API flow, and the supported and unsupported components for export, see [Export guidelines](#). For more

information about deploying an exported flow to an integration server in an App Connect Enterprise dashboard, see [Deploying IBM App Connect Designer APIs to integration servers](#).



4. Click **Export**.
5. If prompted, choose to save the file, which is named after your flow by default, as `flow_name.yaml` or `flow_name.bar`. (If saved as a BAR file, spaces in the flow name are replaced with underscores.) Depending on your browser, the file might also be automatically downloaded to a configured download location.
6. Make the exported YAML file available to other users. For example, you can copy the file to a file system directory, or a content management and file sharing system. You can also upload the file to a server with a public HTTP or HTTPS URL.

Restriction: The URL must point to a public network that anyone can freely connect to without login credentials; for example, a public GitHub repository or GitHub Pages site.

If the file is hosted on a public GitHub repository, ensure that you provide the "raw" URL for sharing. To do so, navigate to the YAML file location; for example:

```
https://github.com/username/myrepo/blob/master/Document%20API.yaml
```

Then click **Raw** to obtain the "raw" URL; for example:

```
https://raw.githubusercontent.com/username/myrepo/master/Document%20API.yaml
```

Importing an API flow

You can import API flows from other users or another App Connect environment into your App Connect Designer instance. You import a flow by using a YAML file that contains an exported flow definition.


About this task

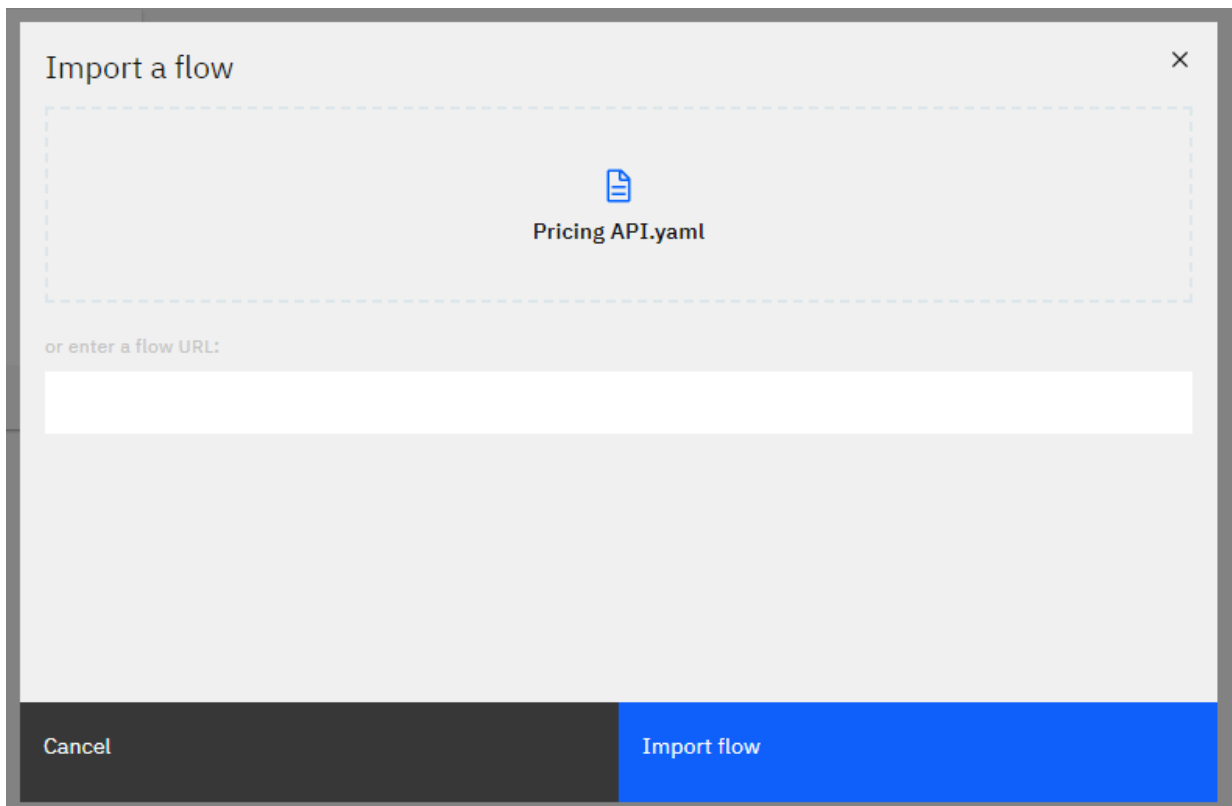
Because connection details are not saved with an exported flow, you'll need to set up your own accounts for each application in the flow if you do not already have one. You can add these accounts from the Catalog page before you import the flow, or from the API editor of the imported flow. For information about the connection details required, see the [How to guides](#) for apps.

After you import the flow, you must ensure that the configuration settings for the nodes are validated. You can also modify the flow in the API editor if required. Do not attempt to modify the YAML file before you import it.

Procedure

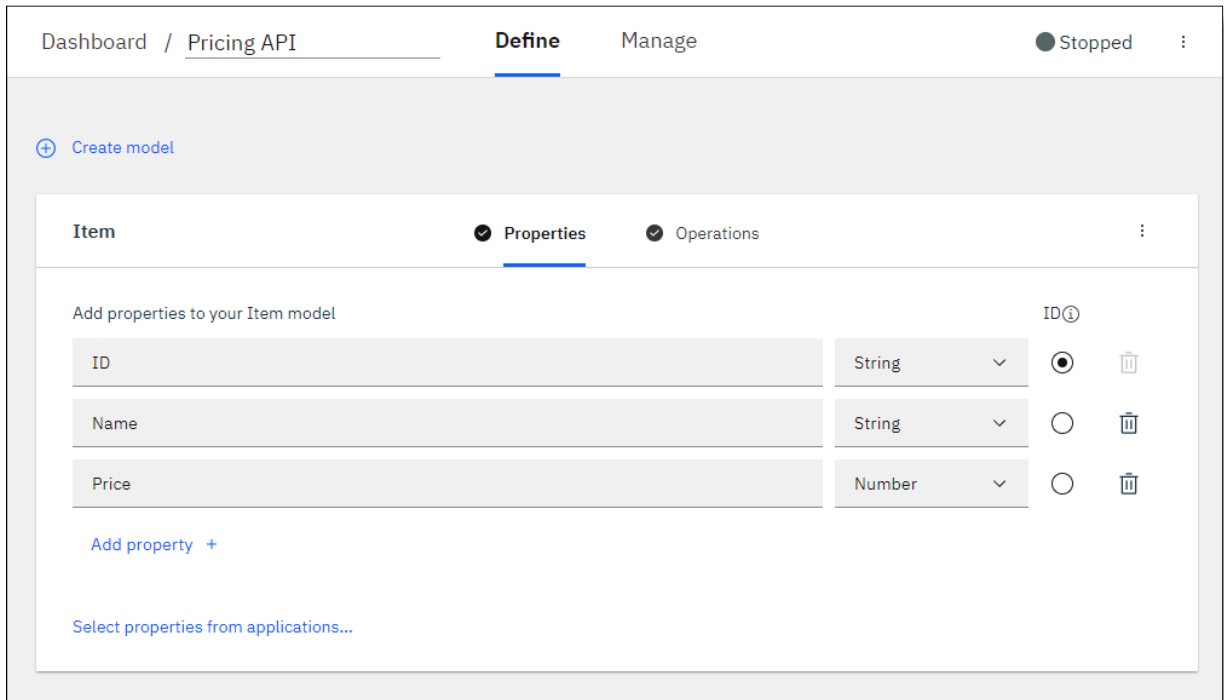
To import an exported flow:

1. From the App Connect Designer UI, open the App Connect Designer dashboard if not currently on display:
 - **Applicable to IBM Cloud Pak for Integration only:** Click the settings icon  in the banner and then click **Dashboard**.
 - **Applicable to App Connect Enterprise certified container only:** Click the **Dashboard** tab.
2. From the App Connect Designer dashboard, click **New > Import flow** to open the "Import a flow" panel.
3. Specify a file or URL location for the YAML file that you want to import:
 - If stored in a file system directory, select the file by using either of these methods:
 - Click within the **Add a YAML file** boxed area to open a file browser and locate the YAML file.
 - Drag and drop the file from its location in an open file browser into the **Add a YAML file** area.You'll see the file name in the boxed area.
 - If stored on a server that is accessible from a URL, enter a valid fully qualified URL for the YAML file in the **or enter a flow URL** field. This must be a public URL that does not require login credentials. Only the HTTP and HTTPS transports are supported. Examples:
`http://www.example.com/path/Pricing%20API.yaml`
`https://raw.githubusercontent.com/username/myrepo/master/Pricing%20API.yaml`




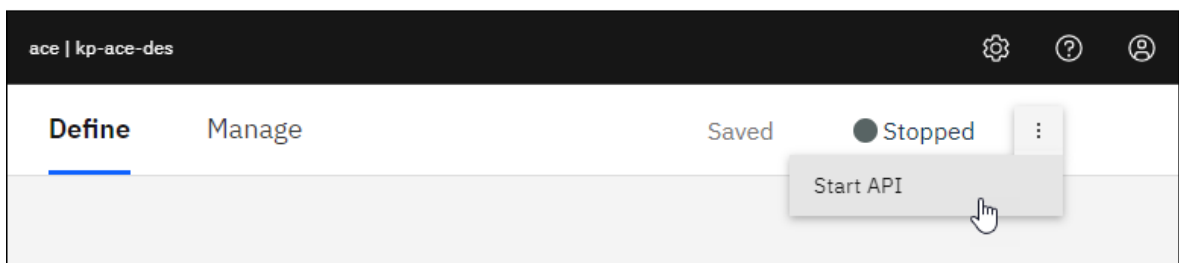
4. Click **Import flow**.

The imported flow opens within the API editor, with the **Define** tab on display, and the inherited flow name from the exported flow definition.



5. Validate the settings for each model and its operations:

- a) From the **Define** tab, click the **Operations** tab for a model.
- b) For *each operation*, edit the flow as follows to review its configuration and resolve any validation errors:
 - 1) Click **Edit flow**. A warning icon  is displayed on any nodes that require your attention.
 - 2) Ensure that the preferred accounts are selected for each action.
 - 3) Resolve any validation errors as described in [Validating your flow is ready to run](#).
 - 4) After updating the operation, click **Done**.
- c) When you've validated all your models and operations, start the API by selecting **Start API** from the flow menu.



IBM Cloud Pak for Integration only: Creating and reusing assets in the Asset Repository

IBM Cloud Pak for Integration provides an Asset Repository as an add-on component for storing, managing, and sharing integration assets across capabilities. You can add API flows that you've created to this repository for sharing, and can also search for and reuse flows that other users have shared. Flows stored in the repository can be used as templates for building the same flow or a customized version within your development environment.

Tip: You can also share your flows by using the export and import functions in App Connect Designer. For more information, see [“Exporting and importing API flows” on page 42](#).

Before you begin

To store and reuse assets, you must have access to an Asset Repository instance. A user with the [Integration Specialist](#) role typically creates the instance from the Platform Navigator and grants access to members of one or more teams. For more information about creating an Asset Repository instance, see [Asset Repository](#) in the IBM Cloud Pak for Integration documentation in IBM Knowledge Center.

About this task

- [“Adding an API flow to the Asset Repository” on page 47](#)
- [“Creating an API flow from an asset” on page 51](#)

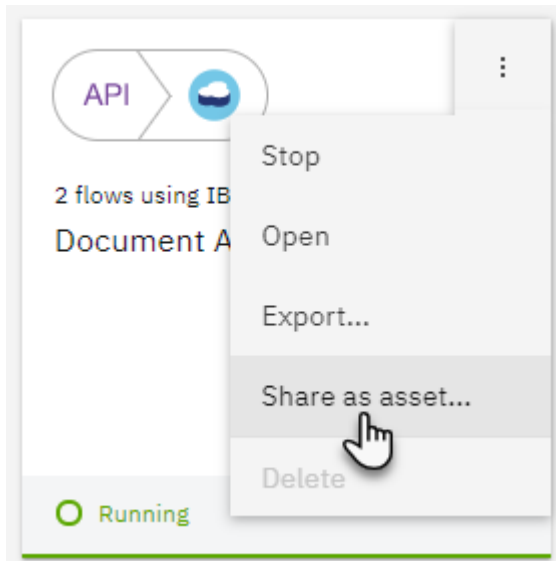
Adding an API flow to the Asset Repository

From the App Connect Designer dashboard, you can add any of your API flows to the Asset Repository for reuse. If you'd like to share YAML files for API flows that were exported from other App Connect Designer or App Connect on IBM Cloud instances, you can also directly add these files to the Asset Repository. When you add a flow as an asset, all its configuration settings, other than your connection details for the integrating apps, are preserved in the asset definition.

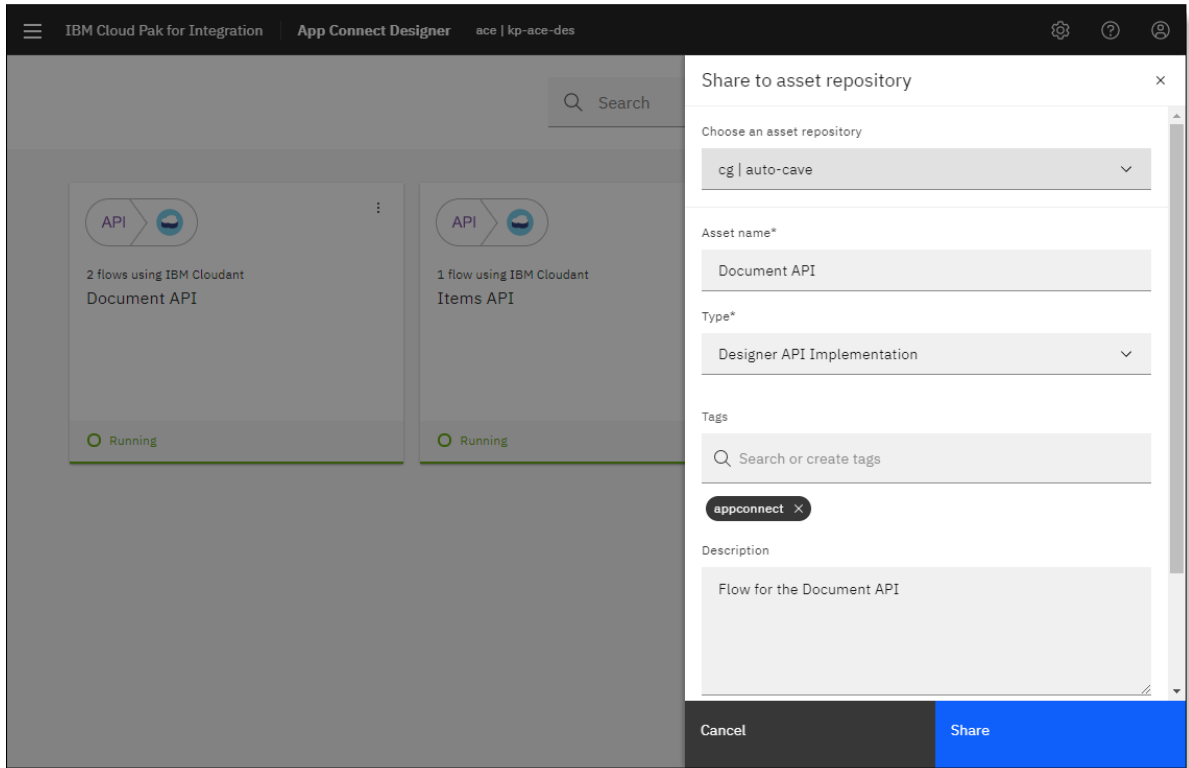
Procedure

To add an API flow as an asset, complete either of the following steps:

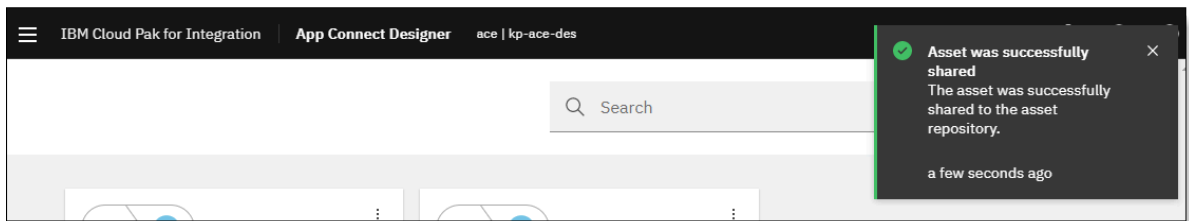
- From the App Connect Designer dashboard, add an asset as follows:
 - a) Open the options menu for the flow that you want to add to the Asset Repository and then click the **Share as asset** option.
This option is visible only if you have access to a repository instance.




- b) Complete the "Share to asset repository" panel as follows:
 - a. In the **Choose an asset repository** field, select the repository where you want to store the asset.
 - b. In the **Asset name** field, specify a name for the asset. By default, the flow name is used.
 - c. In the **Type** field, specify **Designer API Implementation** as the format in which the asset should be saved.
 - d. In the **Tags** field, select or create one or more tags that can be used to categorize the asset within the repository.
 - e. In the **Description** field, specify the purpose of the flow. Asset names in the repository are not required to be unique, so a meaningful description might help other users determine whether an asset could be used to build their solution.
 - f. Click **Share**.

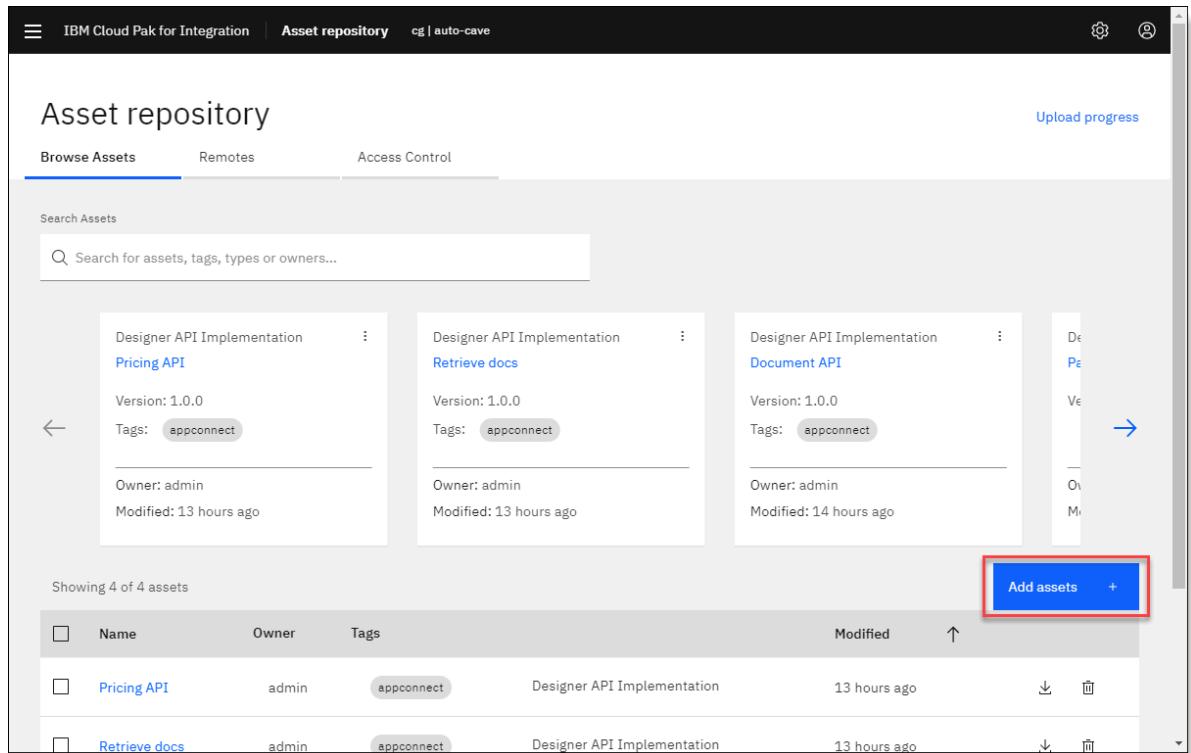


A confirmation message is displayed after the asset is saved. All users with access to the selected Asset Repository will be able to view the asset's definition and metadata in the repository when accessed from the Platform Navigator, and can reuse this asset if required.



Tip: If you'd subsequently like to delete this asset from the Asset Repository, you must open the repository from the Platform Navigator and then use the **Delete** option to permanently remove the asset.

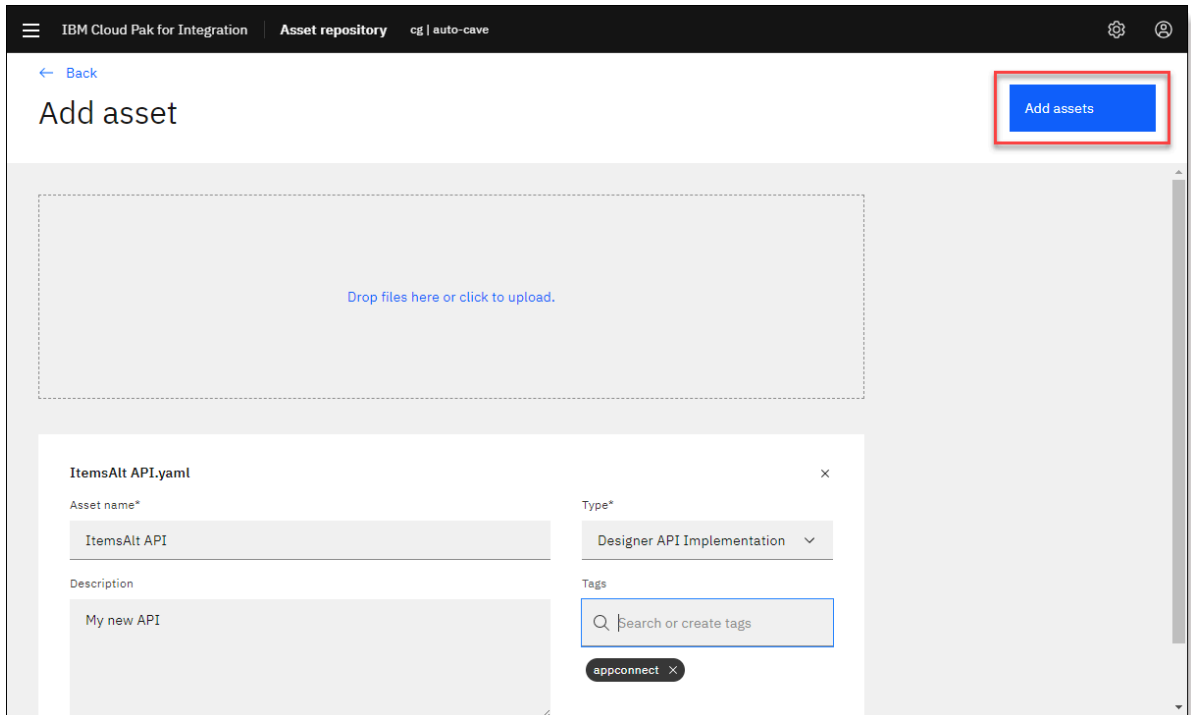
- From the Asset Repository, add an asset as follows:
 - If not already open, use the **View instances** tab in the Platform Navigator, or the IBM Cloud Pak menu  to select the Asset Repository instance that you want to open.
 - From the Asset Repository window, click **Add assets**.



- From the "Add asset" window, drag and drop the YAML file from its location in an open file browser into the boxed area, or click within the area to open a file browser and locate the file. The file details

are displayed with default values for the asset name (that is, the flow name) and type (that is, **Designer API Implementation**).

- d) Add a description that summarizes the purpose of the flow, and specify tags, if required. Then click **Add assets**.



The asset is displayed in the **Browse Assets** tab. All users with access to this Asset Repository will be able to view the asset's definition and metadata, and can reuse the asset if required.

Creating an API flow from an asset

You can create an API flow in App Connect Designer by importing an asset of type **Designer API Implementation**, which contains a flow definition, from the Asset Repository.

About this task

Because connection details are not saved with an asset, you'll need to set up your own accounts for each app in the flow if you do not already have one. You'll also need to connect to each of these accounts before or after you create the flow, and ensure the configuration settings for the nodes are validated, in order to start and run the flow. You can connect from the Catalog page or while within the flow or API editor. For information about the connection details required, see the [How to guides](#) for apps.

If required, you can modify the flow in the flow or API editor after you've created it.

Procedure

To create a flow from an asset:

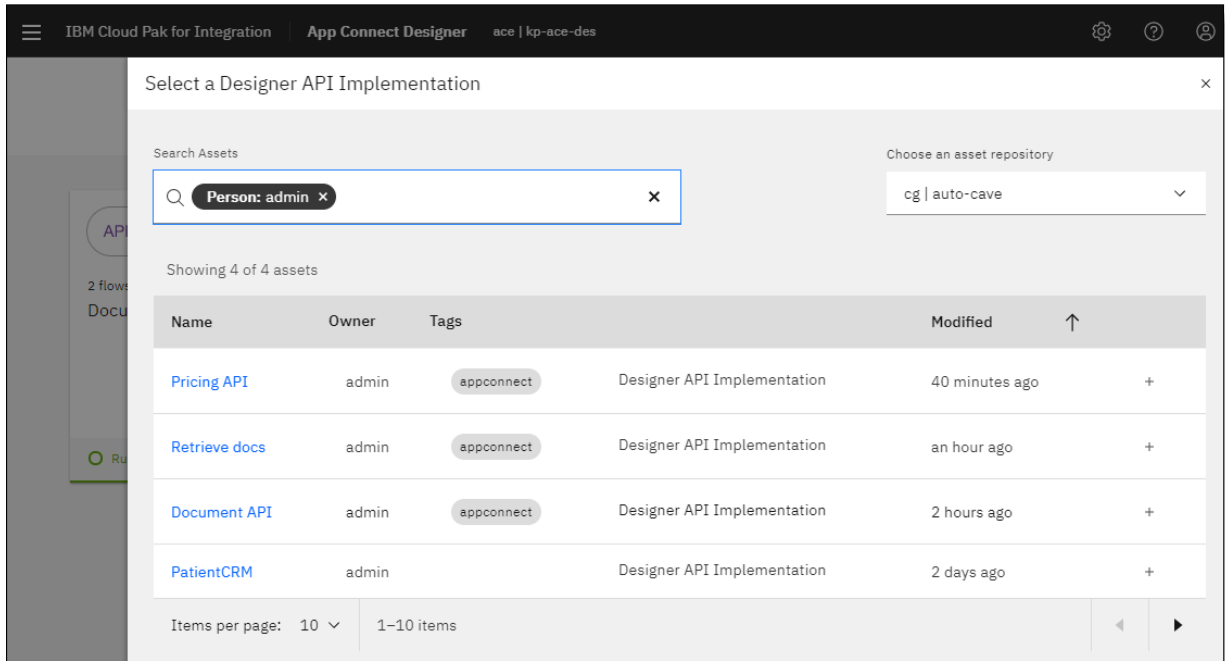
1. From the App Connect Designer dashboard, click **New > Create from asset** to open the "Select a Designer API Implementation" panel.

(The **Create from asset** option is visible only if you have access to a repository instance.) In the "Select a Designer API Implementation" panel, only those assets of a type that can be added to the App Connect Designer dashboard are shown.

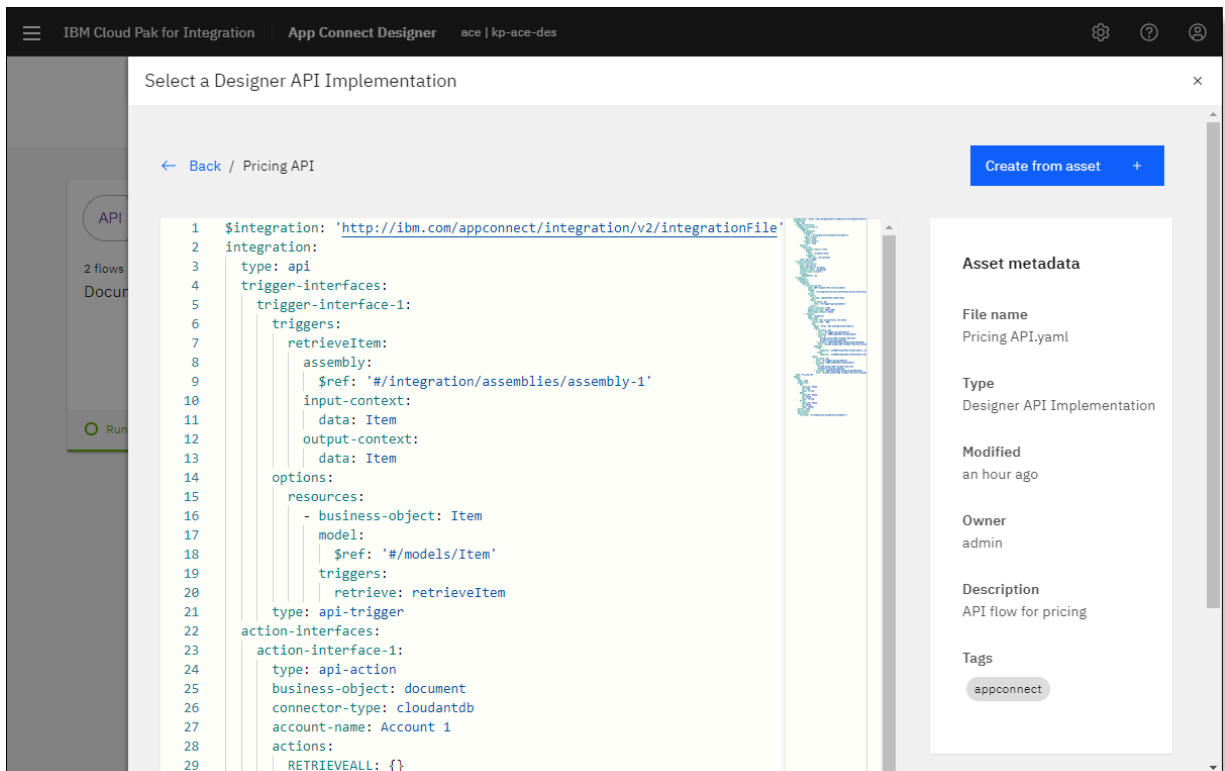
2. Locate the asset as follows:

- a) In the **Choose an asset repository** field, select the repository where the asset is stored.
- b) Optional: To narrow down the search for the asset that you want to locate, specify search criteria in the **Search Assets** field.

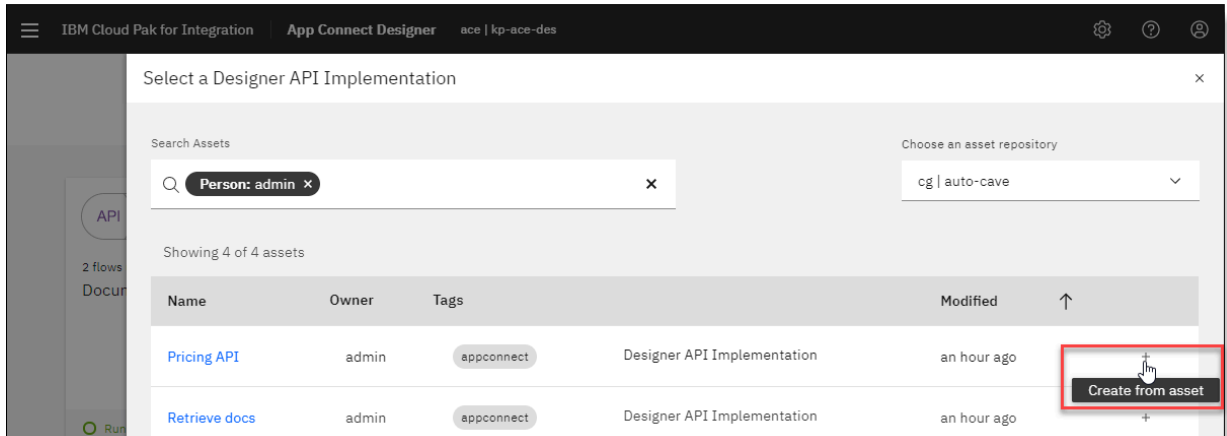
You can search for the name of the asset, as well as by tags and owners. Within the table, you can also click values in the Owner, Tags, or "asset type" columns to filter by those values.



If required, you can also click the name of any asset to review its definition and metadata, and then click **Back** to return to the previous view.




3. When you've located the required asset, click **+ (Create from asset)** to add it as a flow.



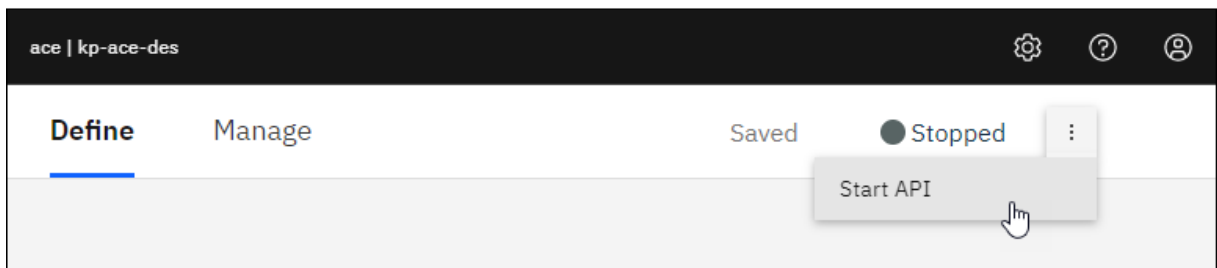
The flow opens within the API editor, with the **Define** tab on display, and the inherited flow name from the asset definition.

4. Review the settings for each model and its operations, and customize the flow as required:
- From the **Define** tab, add or remove models, or update the properties of existing models.
 - From the **Operations** tab for a model, add new operations, or remove or update existing operations.
 - For each defined operation, click **Edit flow** or **Implement flow**:

- Ensure that the preferred accounts are selected for each action.
- If required, customize the flow by adding or removing nodes, and populate the fields with your preferred values.
- Resolve any validation errors as described in [Validating your flow is ready to run](#). (A warning icon  is displayed on any nodes that require your attention.)

After creating or updating the flow, click **Done**.

5. To start the API, select **Start API** from the flow menu.



Deploying IBM App Connect Designer APIs to integration servers

From an App Connect Designer authoring environment, you can export flows for an API (alternatively referred to as an API flow) as a BAR file, and then use an App Connect Dashboard instance to deploy the BAR file as an integration server. The integration server runs the API integration that is packaged in the BAR file.

Within your App Connect Dashboard instance, you can upload, store, and manage BAR files, create integration servers to run the BAR files, and view information about the integrations that are packaged in the deployed BAR files.

High-level sequence for running Designer-authored APIs in integration servers

The sequence for running an App Connect Designer API in an integration server is outlined in the following steps.

- [“IBM Cloud Pak for Integration only: High-level sequence” on page 54](#)
- [“IBM App Connect Enterprise certified container only: High-level sequence” on page 54](#)

IBM Cloud Pak for Integration only: High-level sequence

Prerequisite:

- A Red Hat OpenShift cluster must have been set up with an installation of Cloud Pak for Integration, which includes IBM App Connect Enterprise by default.
- An instance of App Connect Dashboard must also have been created to provide a runtime environment for integration servers.

The high-level steps are as follows:

1. From an App Connect Designer instance in your cluster, or from IBM App Connect on IBM Cloud, create an API flow that meets the requirements for export, and then export the flow as a BAR file.
2. Optional. From your local computer, run **mqsapplybaroverride** commands to override the values for the App Connect instance ID and region-specific URL in the BAR file. These values will be used to identify the App Connect on IBM Cloud instance that hosts cloud-managed connectors, when enabled.
3. Access the Cloud Pak for Integration Platform Navigator in your cluster, and download a set of command-line tools (if not yet installed on your local computer). Then install the CLIs locally. Also ensure that the OpenShift Container Platform CLI is installed, as described in the Red Hat OpenShift documentation.
4. In your IBM Cloud account, create an IBM Cloud API key that can be used to generate a Kubernetes secret when you deploy an integration server in the cluster. This step is required only if you want to use cloud-managed connectors to run the API operations.
5. Upload the BAR file to your App Connect Dashboard instance in the cluster, and generate a Kubernetes secret to store details of the IBM Cloud API key or account credentials that will enable the deployed API to use cloud-managed connectors or local connectors in your cluster. Then create an integration server that will run all the artifacts that are packaged for the API within the BAR file.
6. Invoke the API.

IBM App Connect Enterprise certified container only: High-level sequence

Prerequisite:

- A Red Hat OpenShift cluster must have been set up with an installation of App Connect Enterprise certified container.
- An instance of App Connect Dashboard must also have been created to provide a runtime environment for integration servers.

The high-level steps are as follows:

1. From an App Connect Designer instance in your cluster, or from App Connect on IBM Cloud, create an API flow that meets the requirements for export, and then export the flow as a BAR file.
2. Optional. From your local computer, run **mqsapplybaroverride** commands to override the values for the App Connect instance ID and region-specific URL in the BAR file. These values will be used to identify the App Connect on IBM Cloud instance that hosts cloud-managed connectors, when enabled.
3. Download a set of command-line tools (if not yet installed on your local computer). Then install the CLIs locally. Also ensure that the OpenShift Container Platform CLI is installed, as described in the Red Hat OpenShift documentation.
4. In your IBM Cloud account, create an IBM Cloud API key that can be used to generate a Kubernetes secret when you deploy an integration server in the cluster. This step is required only if you want to use cloud-managed connectors to run the API operations.

5. Upload the BAR file to your App Connect Dashboard instance in your cluster, and generate a Kubernetes secret to store details of the IBM Cloud API key or account credentials that will enable the deployed API to use cloud-managed connectors or local connectors in your cluster. Then create an integration server that will run all the artifacts that are packaged for the API within the BAR file.
6. Invoke the API.

Prerequisites for deploying an App Connect Designer API to an integration server

Before attempting to deploy an App Connect Designer API to an integration server, first find or create everything that you'll need.

- You must have access to an App Connect Designer authoring environment, which contains an [API flow that meets the requirements for exporting to a BAR file](#). The authoring environment can be in a private or managed cloud setting:
 - An App Connect Designer instance that was created in IBM Cloud Pak for Integration or an IBM App Connect Enterprise certified container deployment in a cluster
 - A provisioned instance of [IBM App Connect on IBM Cloud](#)
- If you want to use cloud-managed connectors to interact with applications that are referenced in the flow, you must have a provisioned instance of App Connect on IBM Cloud.
- You must have an installation of IBM App Connect Enterprise on your computer so that you can run the **mqsipbaroverride** command to update the configuration in an exported BAR file (if required). If you don't have an App Connect Enterprise installation, you can install App Connect Enterprise for Developers, as described in [Get started with IBM App Connect Enterprise](#).

Also ensure that you've initialized the command environment by running the **mqsiprofile** command. For more information, see [Setting up a command environment](#).

- You must have access to an OpenShift cluster that contains an installation of Cloud Pak for Integration or App Connect Enterprise certified container, and must have been assigned to a team and namespace with the required pod security policies by the cluster or App Connect Enterprise administrator. You must also have access to an App Connect Dashboard instance that you can use to deploy BAR files as integration servers. For more information, see the following topics in IBM Knowledge Center:
 - **Cloud Pak for Integration:** [Installation and Application integration Dashboard deployment](#) in the Cloud Pak for Integration documentation
 - **App Connect Enterprise certified container:** [Installing IBM App Connect Enterprise certified container](#) and [Creating an instance of App Connect Dashboard](#)
- Obtain the following information about your cluster and App Connect Dashboard instance from your cluster or App Connect Enterprise administrator:
 - The URL that you can use to access the Platform Navigator (if using Cloud Pak for Integration), or the URL for accessing the App Connect Dashboard instance (if using App Connect Enterprise certified container)
 - The user credentials for accessing the cluster; for example, LDAP login credentials
 - The namespace to which you are assigned
 - The instance name of your App Connect Dashboard (if using Cloud Pak for Integration), or the Helm release name of your App Connect Dashboard instance (if using App Connect Enterprise certified container)
- Ensure that curl is installed on your computer so that you can run commands to download command-line interface (CLI) installers from your cluster. Also ensure that you have a utility for extracting files from a .tar.gz archive. (Your operating system might contain the **curl** and **tar** executables by default.)

Tip: Windows 10 (Build 17063 or later) provides native support for these executables, which are typically stored in the C:\Windows\System32 directory. This directory is, by default, included in your System PATH environment variable, which should enable you to run **curl.exe** and **tar.exe** from any location.

- Ensure that you have a Bash utility installed on your computer to enable you to run `.sh` scripts.

Tip: On Windows, you can use utilities such as Git or Cygwin, and should also ensure that the path to the `bash.exe` file (for example, `C:\Program Files\Git\bin`) is included in your System PATH environment variable.

Exporting an API flow to a BAR file

From an App Connect Designer authoring environment, you can export a Designer-authored API flow as a BAR file, and subsequently use that file to deploy an integration server and its API integration to an App Connect Dashboard instance in your cluster. You can export the API flow from an App Connect Designer instance in IBM Cloud Pak for Integration or IBM App Connect Enterprise certified container, or from IBM App Connect on IBM Cloud.

- [“Before you begin” on page 56](#)
- [“Export guidelines” on page 56](#)
- [“Exporting an API flow” on page 58](#)

Before you begin

From your App Connect Designer authoring environment, create an API flow that meets the export requirements outlined in the [“Export guidelines” on page 56](#) section. (For information about creating API flows from an App Connect Designer instance in your cluster, see [“Creating flows for an API” on page 21](#). For information about creating API flows from App Connect on IBM Cloud, see [Creating flows for an API from scratch or from a template.](#))

Export guidelines

These guidelines identify the requirements for successfully exporting a BAR file that can be deployed to an integration server.

Connection considerations for running the deployed flow

When you deploy a BAR file in your cluster, you’ll need to indicate which connectors (and consequently, endpoints) the API flow can use to run its operations. You can configure the API flow to either access your App Connect on IBM Cloud instance to use *cloud-managed connectors* that are referenced in the flow, or you can choose to deploy and use local versions of these connectors in your cluster (that is, *local connectors*). The subset of connectors that can be locally added to your cluster are listed under the [“Supported components for export” on page 57](#) section.

An IBM Cloud API key will be needed for connection to your cloud-managed connectors, and account credentials will need to be defined for connection to your local connectors. Further details are provided in subsequent topics.

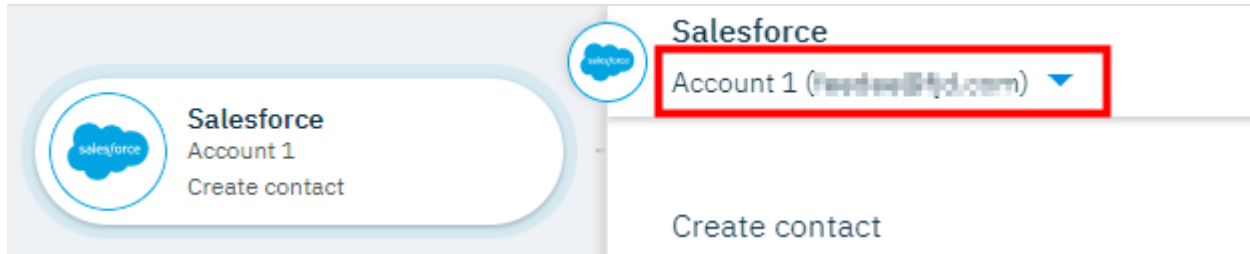
Minimum conditions for export

At a minimum:

- Do not use [unsupported components](#) in the API flow.
- Test the flow to verify that all its defined API operations can be successfully invoked.

Important: The accounts that you select for the defined actions in each operation must remain active because the deployed API in your private cloud will need to access these accounts. You must not delete

or rename any of these accounts in your App Connect on IBM Cloud instance. (This is not applicable if you are using local connectors only.)



Supported components for export

The following components are allowed in the API flow:

- Up to 10 models are allowed, and any of the supported operation types can be configured.
- If using App Connect on IBM Cloud, the actions for most of the software-as-a-service connectors (under **Catalog > Applications**) can be included in the flow. For exceptions, see the [“Unsupported components for export”](#) on page 58 section. If using an App Connect Designer instance in your cluster, only supported actions are shown in the Catalog page.
- The available actions for connectors that require a Secure Gateway connection are supported. (A Secure Gateway connection is relevant for cloud-managed connectors only.)
- If using App Connect on IBM Cloud, operations (exposed as actions) for any API under **Catalog > APIs** can be included in the flow.
- Any of these toolbox utilities can be included in the flow:
 - If
 - For each
 - Log
 - Set variable
 - JSON parser
 - CSV parser
 - XML parser

- If any of these connectors are included in your flow, you can choose to run them as local connectors in your cluster:
 - Confluence
 - Domino
 - HTTP
 - IBM Cloudant
 - IBM Food Trust
 - IBM Maximo
 - IBM Watson Campaign Automation
 - IBM Watson Discovery
 - IBM Watson Language Translator
 - IBM Watson Natural Language Classifier
 - IBM Watson Personality Insights
 - IBM Watson Tone Analyzer
 - IBM Watson Visual Recognition
 - IBM Weather Company Data Limited Edition
 - Jira
 - Microsoft SharePoint
 - Redis
 - Salesforce
 - ServiceNow
 - Workday

Unsupported components for export

The following components are not supported in an API flow that you want to export as a BAR file:

- Actions for the following connector:
 - Callable flow
- The following toolbox utilities:
 - Batch process
 - Notification
 - Situation detector


Exporting an API flow

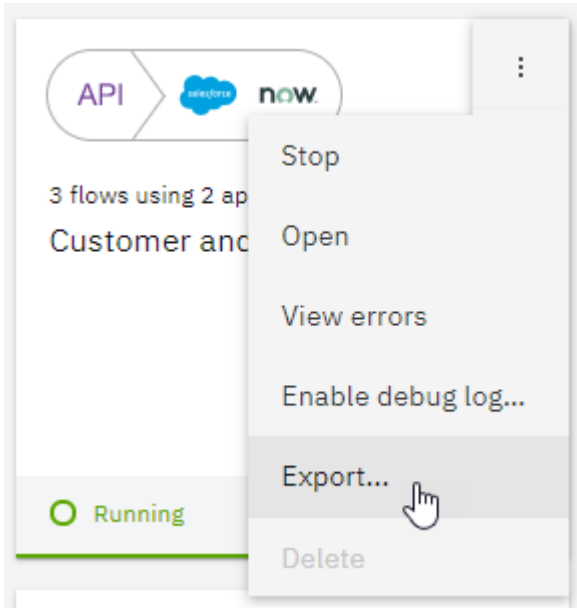
When you export an API flow, you can choose to export its configuration as a YAML or BAR file. To deploy the flow to an integration server, you must export it as a BAR file. All the configuration settings, other than the connection credentials for your accounts, are preserved in the exported archive.

Note: Event-driven flows cannot be exported as BAR files. For information about exporting event-driven or API flows as YAML files, see [Exporting and importing flows](#).

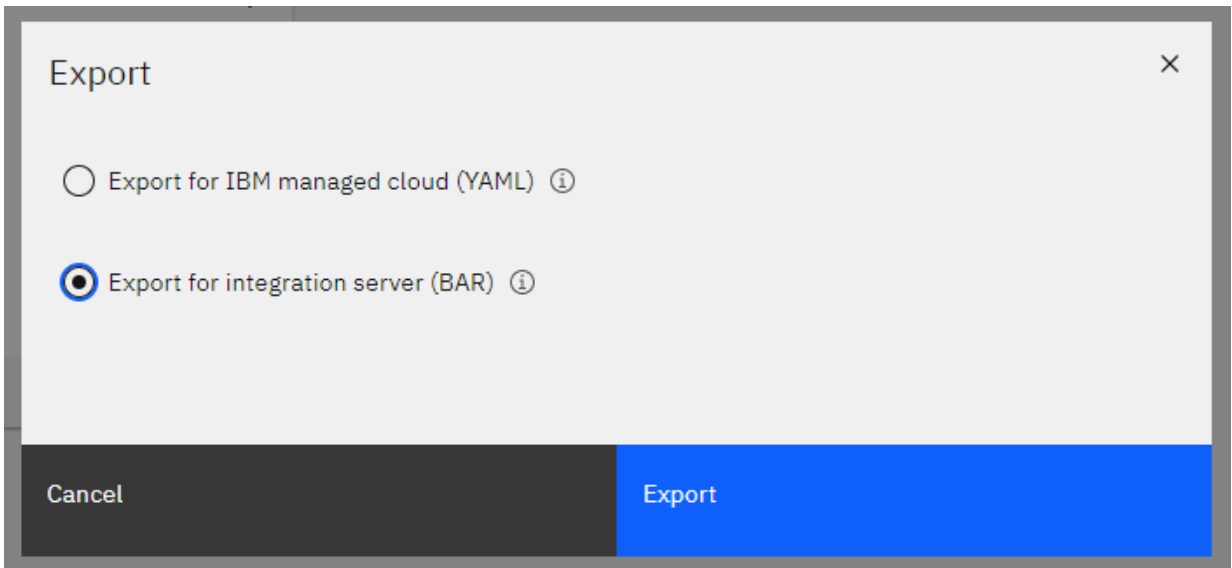
Procedure

To export an API flow:

1. Complete the relevant step to access the dashboard in your App Connect Designer authoring environment:
 - From an App Connect Designer instance in your cluster, either click the settings icon  in the banner and then click **Dashboard**, or click the **Dashboard** tab.
 - From App Connect on IBM Cloud, click **Dashboard**.
2. From the dashboard, locate the API flow, open its menu, and then click **Export**. (The options that you see depend on your authoring environment.) The flow can be in a Stopped or Running state.



3. Click **Export for integration server (BAR)** and then click **Export**.



4. If prompted, choose to save the file, which is named after your flow by default, as *flow_name.bar*. (Spaces in the flow name are replaced with underscores.) Depending on your browser, the file might also be automatically downloaded to a configured download location.

What to do next

- Optional. Override the values for the App Connect instance ID and region-specific URL in the BAR file.
- Access the management console for your cluster.

Overriding the values for the App Connect instance ID and region-specific URL in the BAR file

When you deploy the exported BAR file for an API flow in your cluster, you can indicate whether you want to use cloud-managed connectors to run API operations on the applications that are referenced in the flow. To use cloud-managed connectors, access to an IBM App Connect on IBM Cloud instance is required.

About this task

If the BAR file was exported from an App Connect Designer instance in your cluster, the deployed API will use the cloud-managed connectors in the App Connect on IBM Cloud instance that is linked to your App Connect Designer instance. (Details of this App Connect on IBM Cloud instance are specified when the App Connect Designer instance is created.) If the BAR file was directly exported from an App Connect on IBM Cloud instance, the deployed API will use the cloud-managed connectors in that cloud instance.

The exported BAR file includes configuration parameters that define the following values:

- The ID of the App Connect on IBM Cloud instance that is linked to the App Connect Designer instance, or the ID of the App Connect on IBM Cloud instance from which the file was exported
- A base URL that identifies the region where the App Connect on IBM Cloud instance was provisioned

These configuration values will be used to establish a connection to the accounts of any cloud-managed connectors that are used in the App Connect on IBM Cloud instance.

Note: If you exported the BAR file from your own App Connect Designer or App Connect on IBM Cloud instance (for your own use), you can skip this task because the configuration values will be correctly set by default.

If you'd like to use a BAR file that was exported from a different App Connect Designer or App Connect on IBM Cloud instance (or shared by another user), you *must* complete the steps in this task, to update the BAR file configuration with values for your own App Connect on IBM Cloud instance and region-specific URL. You'll also need to ensure that your App Connect on IBM Cloud instance contains connected accounts with identical names to those defined in the exported flow.

- Possible methods for checking account names
 - If possible, check in the original App Connect on IBM Cloud instance and flow to see which account names are used.
 - Otherwise, extract the contents of the compressed *flow_name*.appzip file that's contained in the BAR file, and then locate and open the *flow_name*.yaml file to view the account-name values for

each action. This example shows account names in a YAML file that was extracted from the APPZIP file in a myApiFlowName.bar archive.

```
View: myApiFlowName.yaml
action-interfaces:
  action-interface-1:
    type: api-action
    business-object: Contact
    connector-type: salesforce
    account-name: Account 1
    actions:
      CREATE: {}
  action-interface-2:
    type: api-action
    business-object: Contact
    connector-type: salesforce
    account-name: Account 1
    actions:
      RETRIEVEALL: {}
  action-interface-3:
    type: api-action
    business-object: Contact
    connector-type: salesforce
    account-name: Account 1
    actions:
      UPSERTWITHWHERE: {}
  action-interface-4:
    type: api-action
    business-object: sys_user
    connector-type: servicenow
    account-name: Account 1
    actions:
      CREATE: {}
assemblies:
  assembly-1:
    assembly:
      execute:
        - create-action:
            name: Salesforce Create contact
```

Procedure

To update the BAR file configuration, run the `mqsipplybaroverride` command to override the values in the broker archive deployment descriptor:

1. Open the App Connect Enterprise Console that was supplied as part of your local App Connect Enterprise installation. For example, on Windows, select **Start > IBM App Connect Enterprise > IBM App Connect Enterprise Console**.

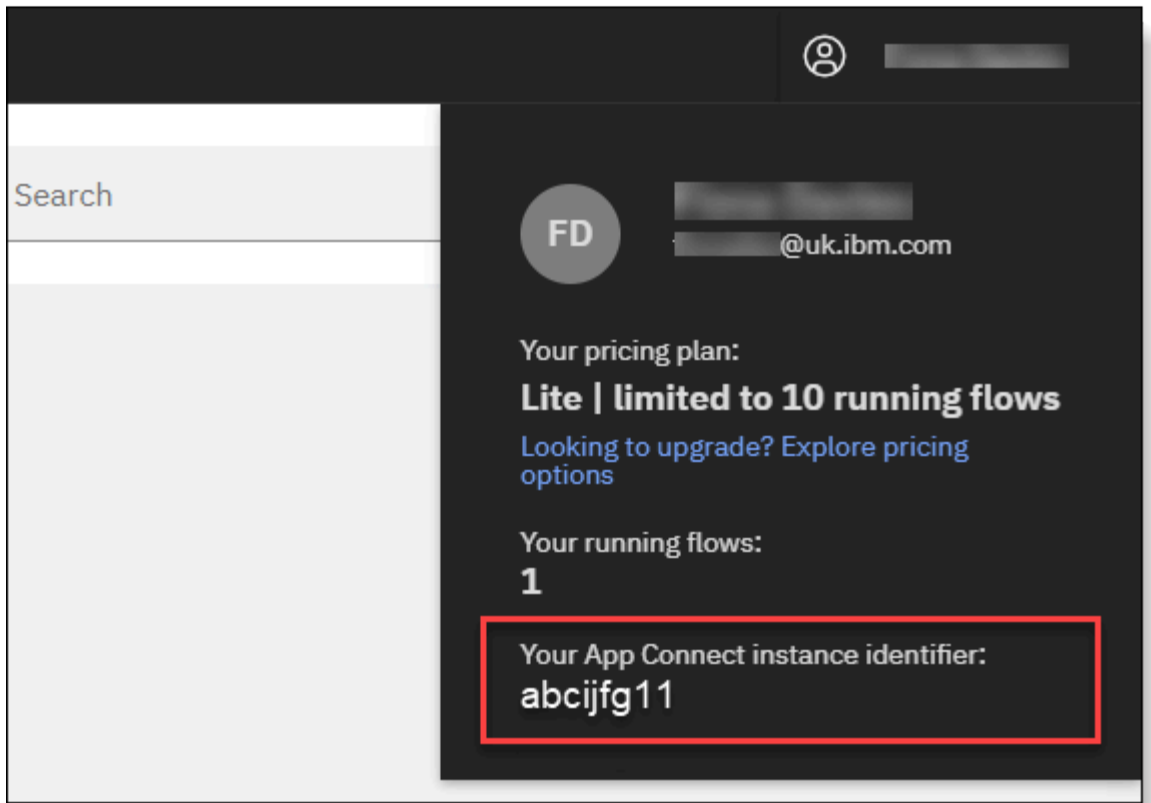
2. Run each of the following commands in turn:

```
mqsiaapplybaroverride -b barfilePath -k applicationName -m  
gen.applicationName#connectorServiceInstanceId=instanceID
```

```
mqsiaapplybaroverride -b barfilePath -k applicationName -m  
gen.applicationName#connectorServiceUrl=serviceURL
```

Where:

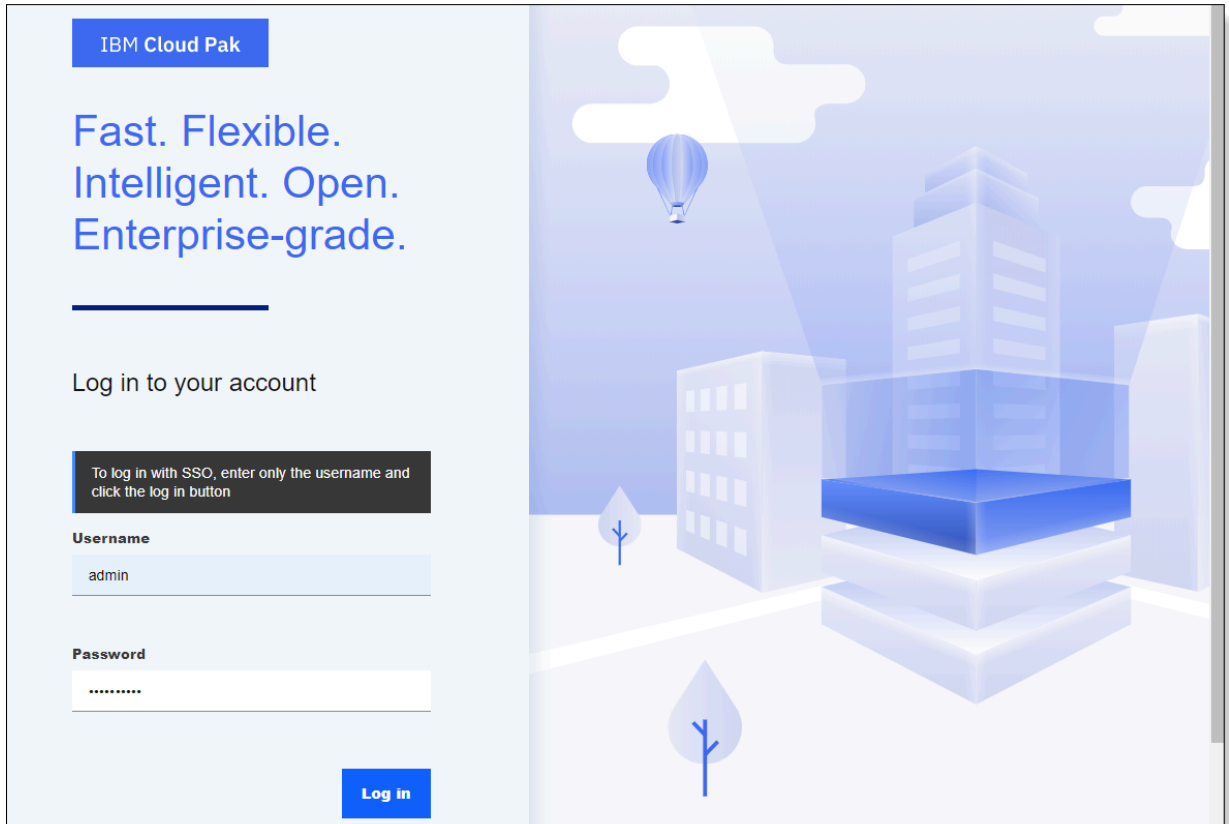
- *barfilePath* is the file path and name of the exported BAR file.
- *applicationName* is the file name stem of the *flow_name*.bar file; for example, *flow_name*.
- *instanceID* is the ID of your App Connect on IBM Cloud instance. You can find this value by clicking the profile icon in your instance.



- *serviceURL* is a region-specific base URL. Specify the value that matches the region where your App Connect production instance is provisioned:

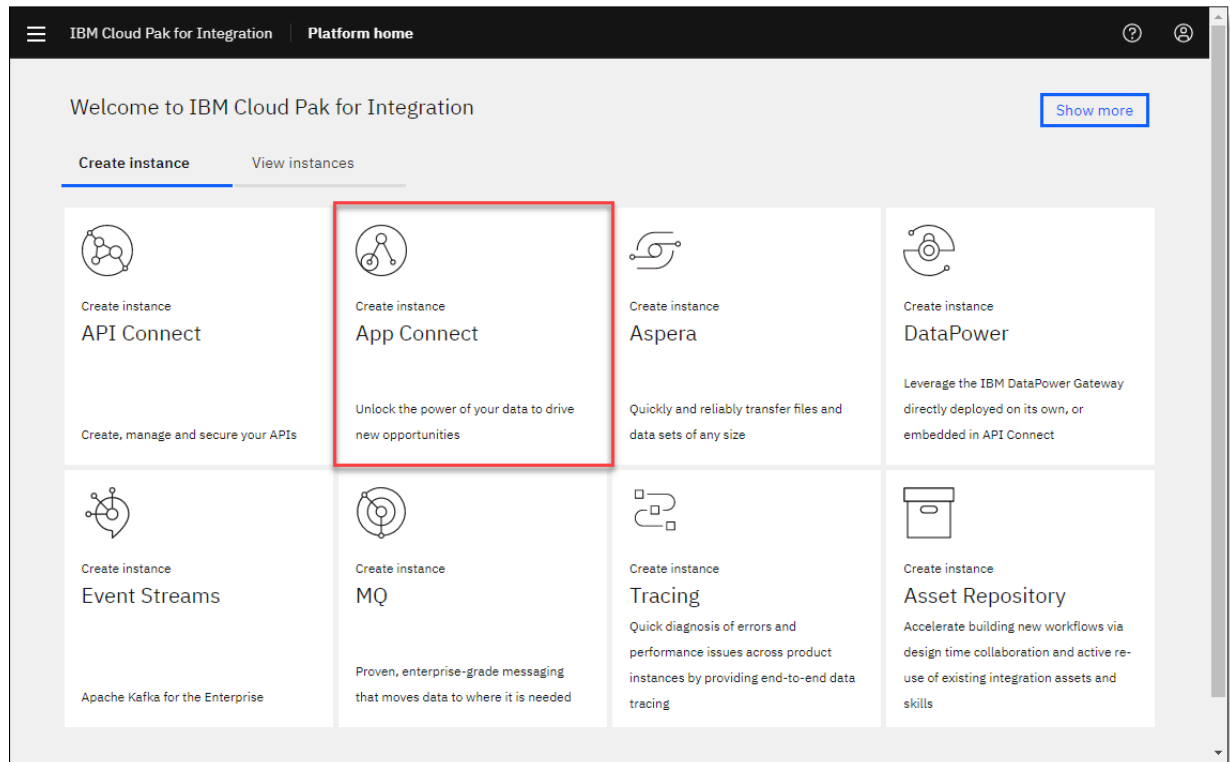
Region	<i>serviceURL</i> value
us-south or sydney	https://firefly-api-prod.appconnect.ibmcloud.com


1. From a browser window, access the Platform Navigator by entering the URL in the following format:
`https://Helm_releaseName-namespace.apps.domain`
For example:
`https://ibm-icp4i-prod-integration.apps-value.domain-value`
2. Specify a valid user name and password, and then click **Log in**.




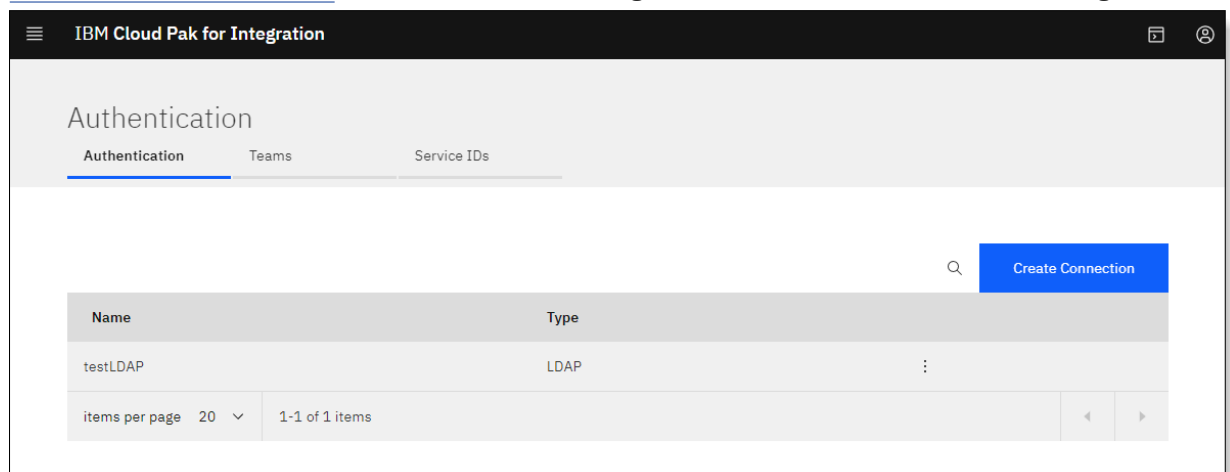
The Cloud Pak for Integration Platform Navigator opens. From this navigator, a user with appropriate access can click the App Connect tile to create instances of App Connect Designer and App Connect Dashboard. For more information about creating an App Connect Designer instance, see [Application integration Designer deployment](#) in the Cloud Pak for Integration documentation in IBM Knowledge Center. For more information about creating an App Connect Dashboard instance, see

Application integration Dashboard deployment in the Cloud Pak for Integration documentation.



3. To access the Cloud Pak management console, open the IBM Cloud Pak menu  in the Platform Navigator, and then click **Cloud Pak Foundation**.
The management console opens.

From this console, you can use the IBM Cloud Pak menu  to administer common services provided by the Cloud Pak; for example, logging, monitoring, and metering. For more information, see [Cloud Platform Common Services](#) in the Cloud Pak for Integration documentation in IBM Knowledge Center.



What to do next

- [Install the command-line tools for your cluster](#) (if not yet installed).
- [Create an IBM Cloud API key in your IBM Cloud account](#) (if using cloud-managed connectors).

Installing the command-line tools for your cluster

You can use a set of command-line interface (CLI) tools to manage the cluster, containers, infrastructures, services, and other resources.

About this task

As a one-time task, install these CLI tools locally if you don't already have them installed.

You'll require these CLI tools:

- **OpenShift Container Platform CLI (oc):** You can use this CLI to log in to your cluster, and to run commands to manage the resources (such as projects, pods, and deployments) in the cluster.
- **IBM Cloud Pak CLI (cloudctl):** If using IBM Cloud Pak for Integration, you can use this CLI to view information about the cluster, install Helm charts and repositories, manage identity and access, and perform other operations. This CLI enables you to use the command line to administer common services that are provided by the Cloud Pak. (You can alternatively use the [Cloud Pak management console](#).)
- **Kubernetes CLI (kubectl):** You can use this CLI to run commands against your cluster. To learn about the **kubectl** commands, see [Overview of kubectl](#) in the Kubernetes documentation.


Note: The **oc** CLI provides similar capabilities as **kubectl**, in addition to native OpenShift Container Platform support. So if you're familiar with **oc**, you might prefer to run **oc** commands in place of the **kubectl** commands that are provided in these topics.

- **Helm CLI (helm):** You can use this CLI to manage releases in your cluster.
- [“IBM Cloud Pak for Integration only: Installing the CLI tools” on page 66](#)
- [“IBM App Connect Enterprise certified container only: Installing the CLI tools” on page 66](#)

IBM Cloud Pak for Integration only: Installing the CLI tools

Procedure

To install the CLI tools:

1. Download and install the OpenShift Container Platform CLI, as described in [Getting started with the CLI](#) in the Red Hat OpenShift documentation. Ensure that you are using the version of **oc** that's provided for the Red Hat OpenShift version on which Cloud Pak for Integration is running.
2. From Cloud Pak for Integration, download and install the supplied CLI tools as follows:
 - a) Open the IBM Cloud Pak menu  and then click **Cloud Pak Foundation**.
 - b) If prompted, enter credentials to log in to the management console that provides common services. (These should be the same credentials that you used to access the Platform Navigator.)
 - c) From the Cloud Pak management console, download and install the CLI tools. For more information, see [CLI tools guide](#) in the Cloud Pak for Integration documentation in IBM Knowledge Center.

Now that the CLIs are installed, you can [log in to your cluster](#) at any time to run **oc**, **cloudctl**, **kubectl**, and **helm** commands.

IBM App Connect Enterprise certified container only: Installing the CLI tools

Procedure

To install the CLI tools:

1. Download and install the OpenShift Container Platform CLI, as described in [Getting started with the CLI](#) in the Red Hat OpenShift documentation. Ensure that you are using the version of **oc** that's provided for the Red Hat OpenShift version on which App Connect Enterprise certified container is running.

2. Download and install the Kubernetes CLI, as described in [Install and Set Up kubectl](#) in the Kubernetes documentation.
3. Download and install the Helm CLI, as described in [Introduction to Helm](#) in the Helm documentation.

Now that the CLIs are installed, you can [log in to your cluster](#) at any time to run **oc**, **kubectl**, and **helm** commands.

Logging in to your cluster from the command-line interface

After you install the CLIs, you can log in to your cluster from the command-line interface to view information about the cluster and manage it.

Procedure

To log in to your cluster:

1. From a command window, run the **oc login** command and provide the OpenShift Container Platform server URL (and optionally a token).

Examples:

```
oc login https://Mycluster_hostName:8443
```

```
oc login --token=AbcdE1fgHijKLM2moP3q4rs5TU6vw7xYz --server=https://
Mycluster_hostName:6443
```

2. Indicate whether to use insecure connections, and then specify a user name and password if required.

```
Administrator: Command Prompt

Since OpenShift runs on top of Kubernetes, your favorite kubectl commands are also present in oc,
allowing you to quickly switch between development and debugging. You can also run kubectl directly
against any OpenShift cluster using the kubeconfig file created by 'oc login'.

For more on OpenShift, see the documentation at https://docs.openshift.com.

To see the full list of commands supported, run 'oc --help'.

C:\WINDOWS\system32>oc login --token=B...k --server=https://...:6443
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server could be intercepted by others.
Use insecure connections? (y/n): y

Logged into "https://...:6443" as "kube:admin" using the token provided.

You have access to 64 projects, the list has been suppressed. You can list all projects with 'oc projects'

Using project "default".

C:\WINDOWS\system32>
```

3. If you intend to run commands against resources in a specific namespace (or project), which is not currently the default, switch to that project:

```
oc project projectName
```

Creating an IBM Cloud API key

In this task, you'll create an IBM Cloud API key that your deployed API can use to authenticate to your IBM App Connect on IBM Cloud instance, in order to connect to the connectors and accounts that were referenced in the exported flow.

About this task

Note: You need to complete this task only if you want your deployed API to use cloud-managed connectors to run one or more operations that are defined in your BAR file configuration. You can skip this task if your deployed API will be configured to use only local connectors in your cluster.

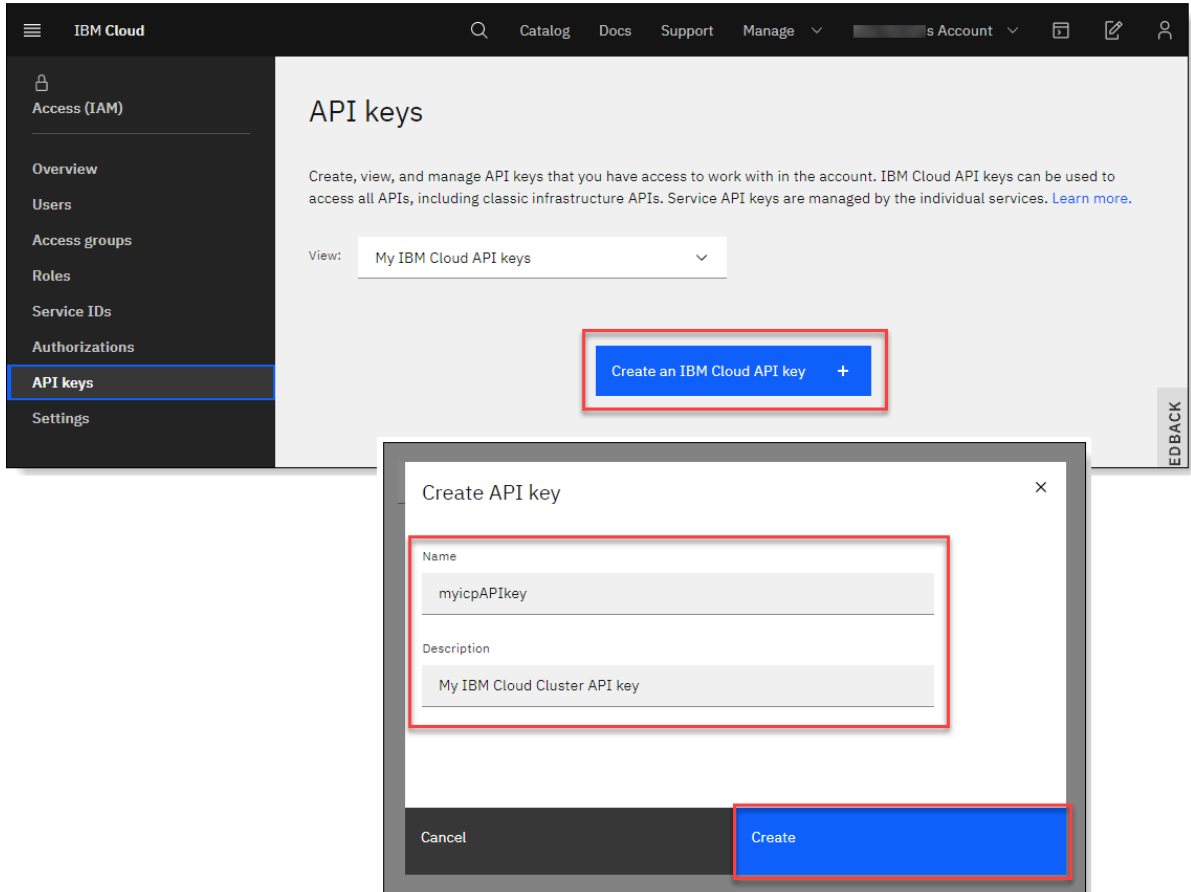
Procedure

You can create an IBM Cloud API key by using IBM Cloud Identity and Access Management (IAM), as in the following steps:

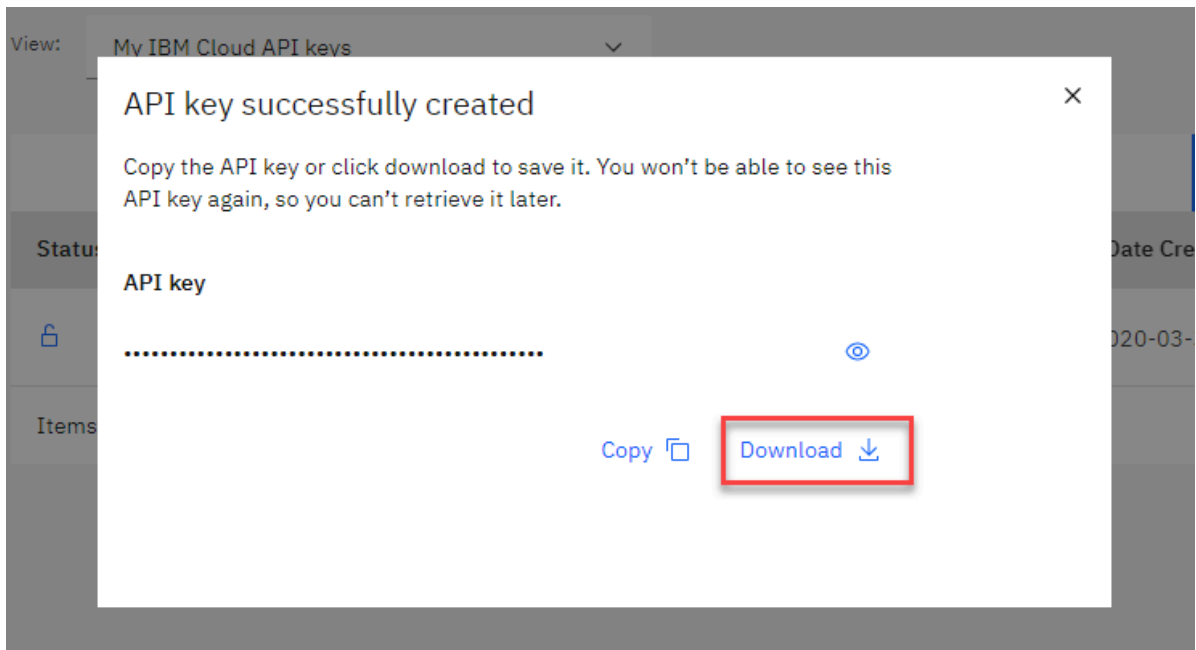
1. Ensure that you are logged in to the same IBM Cloud account as the App Connect on IBM Cloud instance that your deployed API will need to communicate with.

2. Create an API key as follows:

- a) Click **Manage > Access (IAM) > API keys** to open the "API keys" page (URL: <https://cloud.ibm.com/iam/apikeys>).
- b) Ensure that **My IBM Cloud API keys** is selected in the **View** list.
- c) Click **Create an IBM Cloud API key**, and then specify a name and description.
- d) Click **Create**.



- e) Click **Download** to download the API key to a file named `apiKey.json` in your browser's default location for later use.



Note:

- In a subsequent task, you'll create a Kubernetes secret (that is, a security identity) to store this API key, and then configure the deployed API to use that secret to authenticate to the App Connect on IBM Cloud instance.
- You can create a single IBM Cloud API key per App Connect instance, and then use that API key to create a different secret for each BAR file that you deploy as an integration server.
Alternatively, you can create multiple IBM Cloud API keys per App Connect instance, and then use any of those keys to create secrets for your exported BAR files. If you decide to revoke an IBM Cloud API key, be aware that API calls from all deployed integration servers that use that key, will fail.

What to do next

- [Cloud Pak for Integration only: Create an integration server to run your BAR file resources.](#)
- [App Connect Enterprise certified container only: Create an integration server to run your BAR file resources.](#)

Cloud Pak for Integration only: Creating an integration server to run your BAR file resources

The BAR file for your exported API flow contains all the resources that are needed to deploy an integration that exposes the API and its operations. To run this API, you'll need to deploy the BAR file to an integration server by completing a multi-step process.

Before you begin

1. [From an App Connect Designer instance, create an API flow that meets the conditions for export, and export the API flow to a BAR file. Or from App Connect on IBM Cloud, create an API flow that meets the conditions for export, and then export the API flow to a BAR file.](#)
2. [Optional. Override the values for the App Connect instance ID and region-specific URL in the BAR file.](#)
3. [If not yet installed, install the command-line tools for your cluster.](#)
4. [If you do not have one available, create an IBM Cloud API key in your IBM Cloud account to enable you to use cloud-managed connectors.](#)

About this task

You can deploy the BAR file to an integration server by completing a multi-step process:

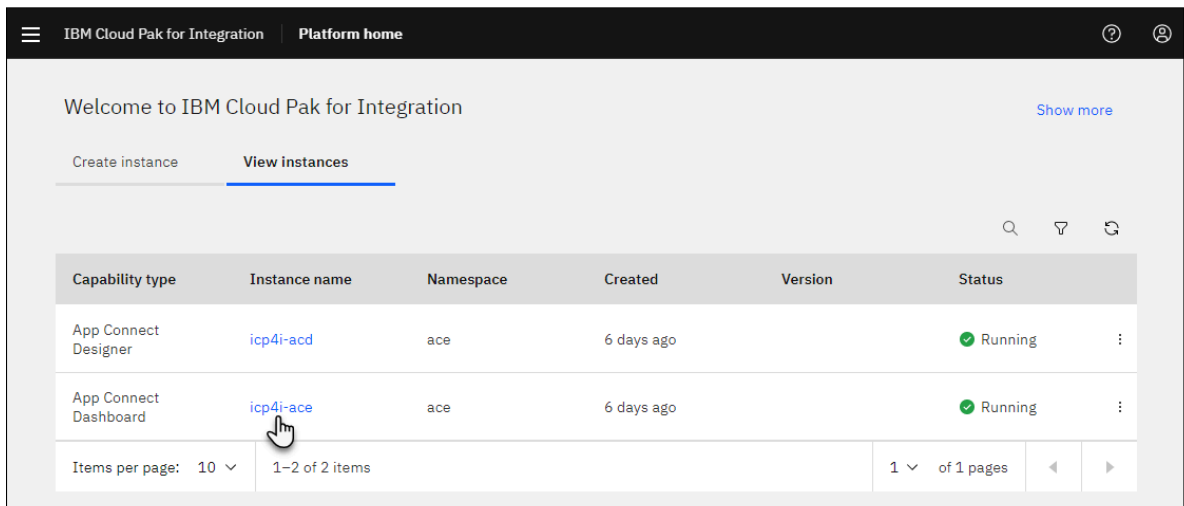
1. Use the App Connect Dashboard to upload the BAR file to a content server in your cluster.
2. Use a supplied configuration package to generate a Kubernetes secret that the deployed API can use to establish a connection to the connectors and accounts that it requires.
 - To use cloud-managed (App Connect on IBM Cloud) connectors to run one or more API operations, you must use your IBM Cloud API key and the configuration package to generate the Kubernetes secret.
 - If you want to deploy and use local connectors in your cluster, you must use the configuration package to configure and store the account credentials for these connectors in the Kubernetes secret. (To check which connectors can be deployed locally, see [Exporting an API flow to a BAR file.](#))
3. Configure and install a Helm release for the integration server.


When the deployment completes, an integration server is created and started, and it reads the BAR file to run the integration.

Procedure

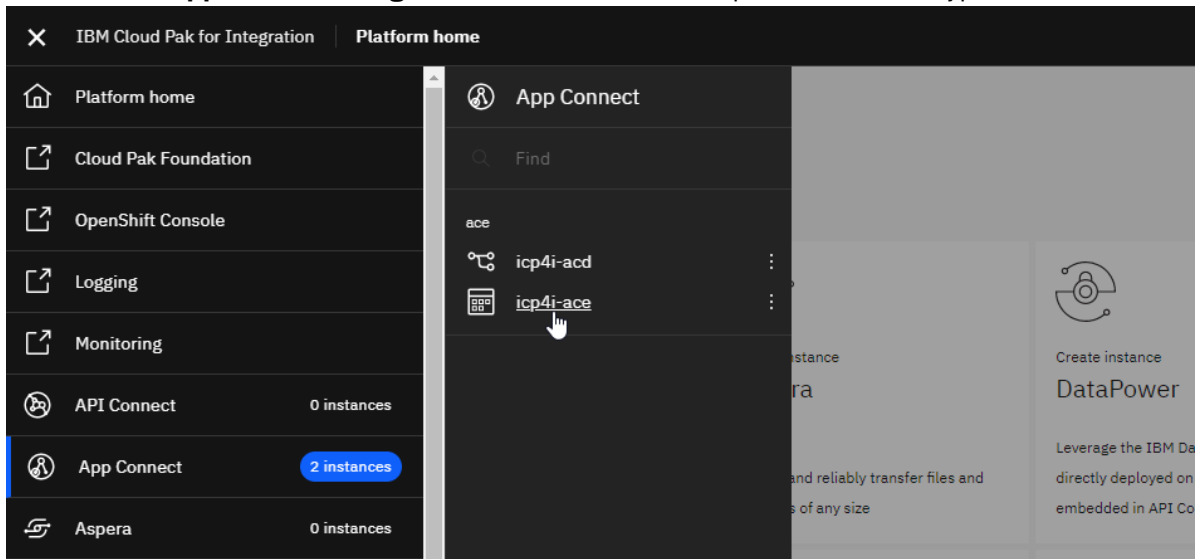
To create an integration server, complete the following steps:

1. From your cluster, launch the App Connect Dashboard:
 - a) Obtain the App Connect instance name for the dashboard from your administrator.
 - b) If not already logged in, [log in to the Platform Navigator for your cluster.](#)
 - c) Click **View instances** and then click the name of the App Connect Dashboard instance for your dashboard.

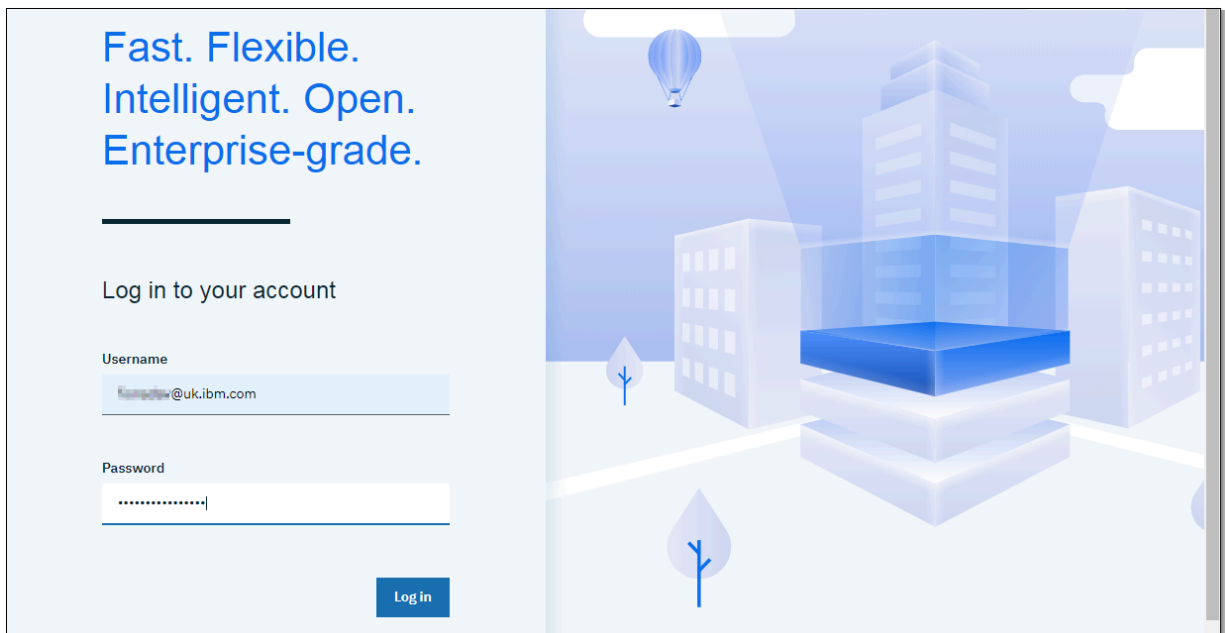


Tip: You can also open the App Connect Dashboard instance from the IBM Cloud Pak menu  in the Platform Navigator. Open the menu, click **App Connect** and then click the name of the instance under your namespace. The **App Connect Dashboard**

icon  and **App Connect Designer** icon  are used to depict the instance type.

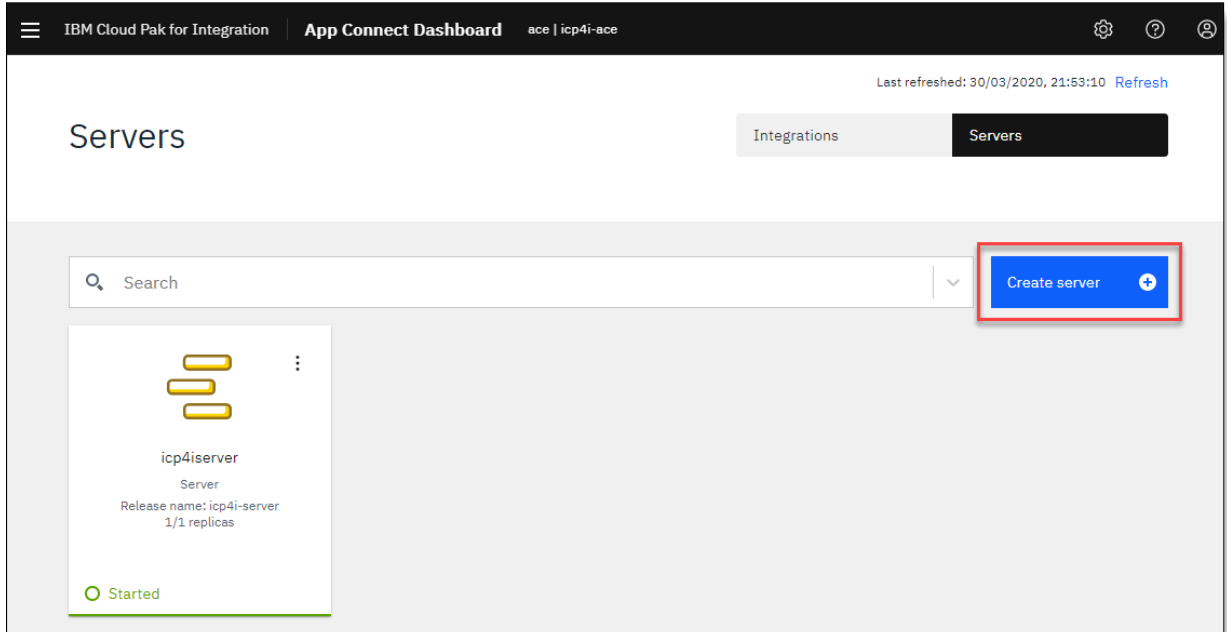


2. If prompted, enter your login credentials to log in to your App Connect Dashboard.



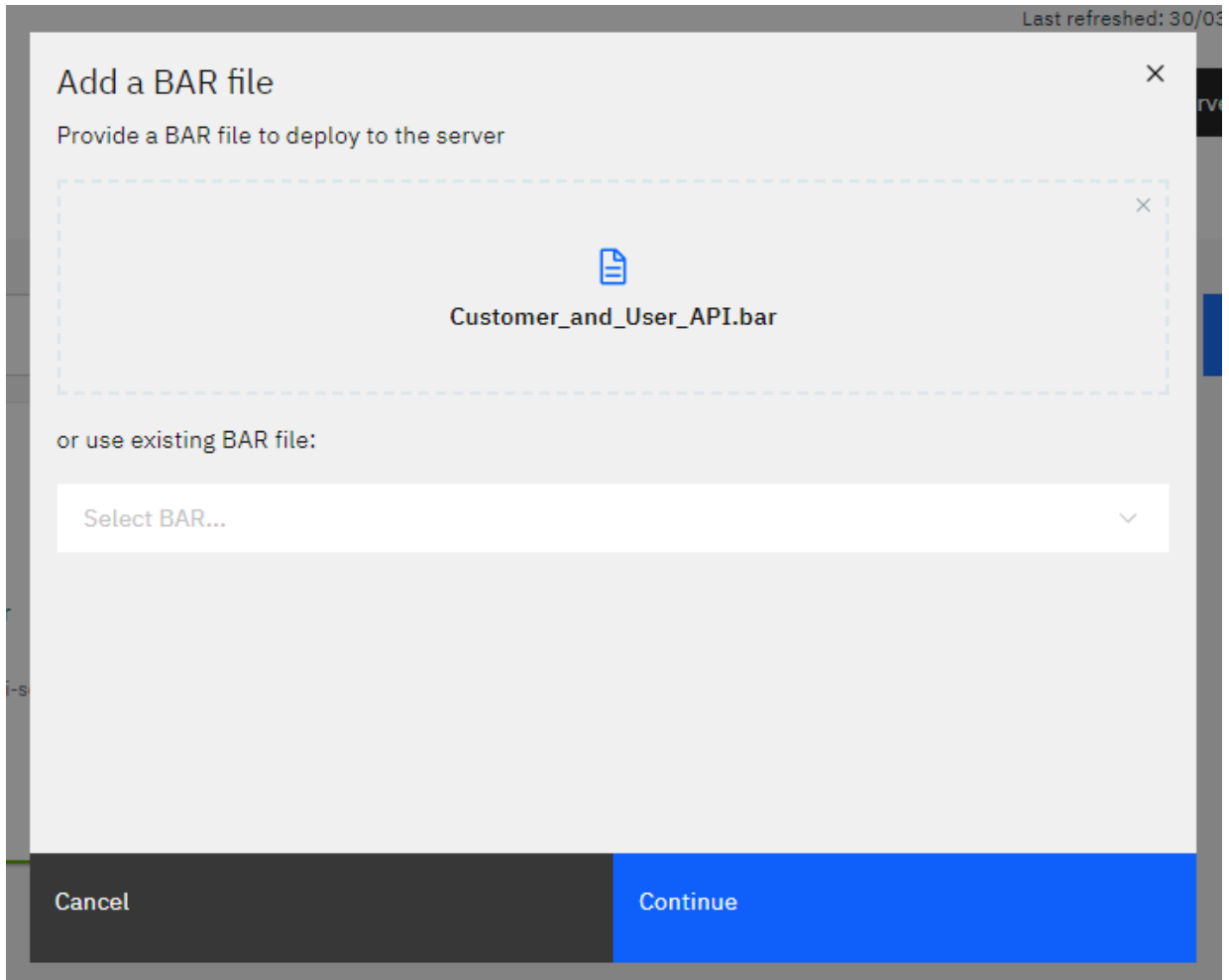
The App Connect Dashboard window opens. If any integration servers have been created, they are displayed as tiles in the dashboard.

3. From the Servers page on the dashboard, click **Create server** to deploy the BAR file to the content server.

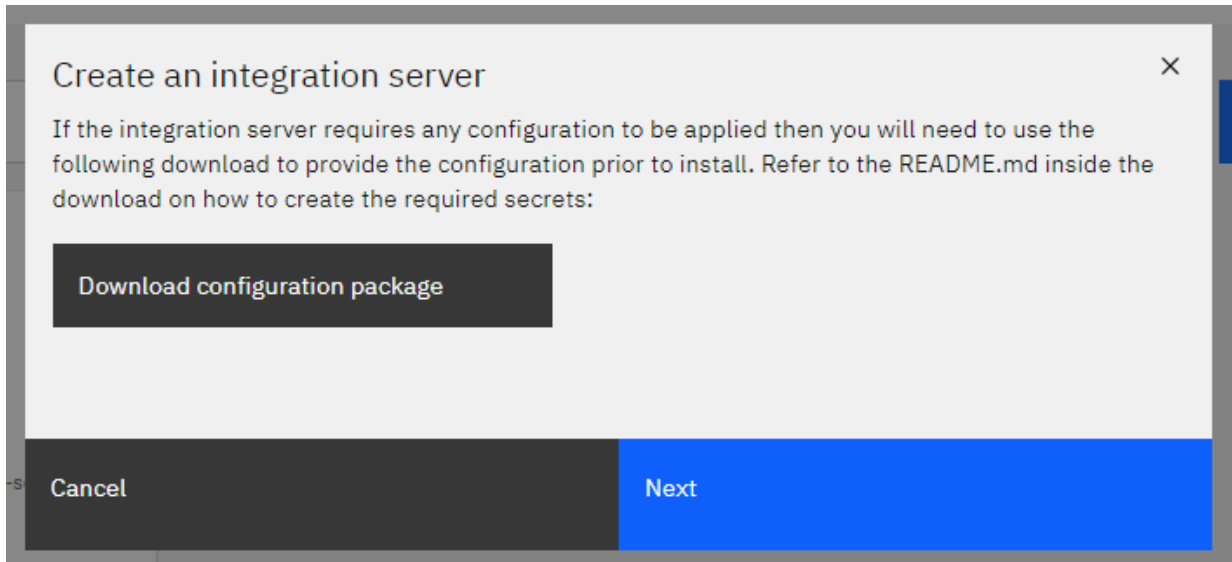


4. From the "Add a BAR file" panel, complete any of these steps:

- Drag and drop the file from its location in an open file browser into the boxed area.
- Click within the boxed area to open a file browser and locate the BAR file.
- From the drop-down list, select an existing BAR file that was previously uploaded to the content server. (This option is useful if you want to create more than one integration server from the same BAR file to manage your workloads.)



5. Click **Continue**.



6. Download a configuration package for connectors.

The configuration package contains a set of files that you can use to generate a Kubernetes secret that stores the account credentials for your local connectors, as well as the IBM Cloud API key that permits access to your cloud-managed connectors. The configuration package is downloaded to your browser's default location as a `config.tar.gz` archive.

a) Click **Download configuration package**

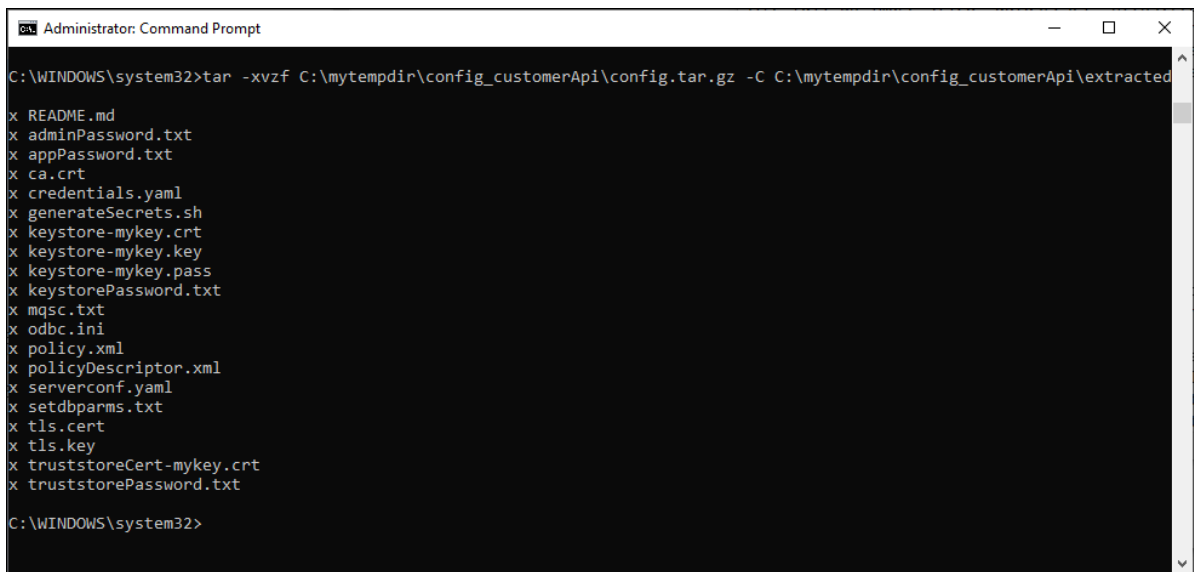
b) Extract the contents of the `config.tar.gz` archive to a temporary directory:

- **Linux or macOS:** Open a terminal window and run the following command from the directory that contains the downloaded file:

```
tar -xvf config.tar.gz
```

- **Windows:** Run **cmd.exe** as an administrator, and then run the **tar** command in the following format:

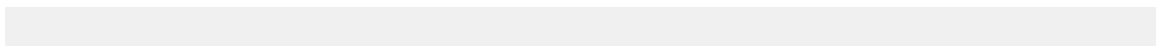
```
tar -xvzf sourcePath\config.tar.gz -C extractedPath
```



```
Administrator: Command Prompt
C:\WINDOWS\system32>tar -xvzf C:\mytempdir\config_customerApi\config.tar.gz -C C:\mytempdir\config_customerApi\extracted
x README.md
x adminPassword.txt
x appPassword.txt
x ca.crt
x credentials.yaml
x generateSecrets.sh
x keystore-mykey.crt
x keystore-mykey.key
x keystore-mykey.pass
x keystorePassword.txt
x mqsc.txt
x odbc.ini
x policy.xml
x policyDescriptor.xml
x serverconf.yaml
x setdbparms.txt
x tls.cert
x tls.key
x truststoreCert-mykey.crt
x truststorePassword.txt
C:\WINDOWS\system32>
```

c) If you want to use cloud-managed connectors, edit the extracted `setdbparms.txt` file to provide your IBM Cloud API key details.

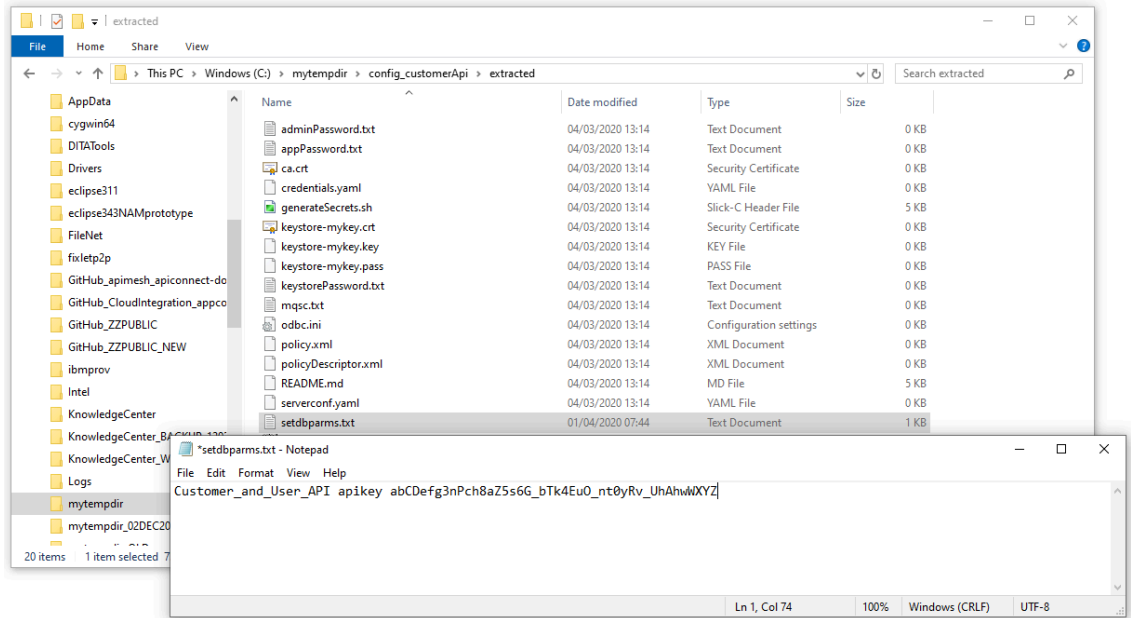
1) Add a line in this format to the empty file:



`flow_name apikey IBMCloud_APIkey`

Where:

- `flow_name` is the file name stem of the BAR file.
- `IBMCloud_APIkey` is the "apiKey" value in the `apiKey.json` file that contains details of the IBM Cloud API key.



2) Save and close the file.

d) If you want to locally deploy connectors to run API operations, configure account credentials for these connectors.

(For details of the supported connectors, see [Exporting an API flow to a BAR file](#).)

- 1) Go to the directory where you extracted the contents of the `config.tar.gz` archive and locate the `credentials.yaml` file.
- 2) Edit this file to add connection credentials for the connectors that you want to deploy locally.

The first line of the file must start with the text `accounts:`, followed by account metadata details for each unique "connector and account" combination that's used in the flow. The following example shows an entry that defines the account credentials for HTTP and Salesforce connectors.

```
accounts:
  http:
    - credentials:
        authType: "none"
        username: ""
        password: ""
      endpoint:
        endpointUrl: ""
        name: "Account 1"
  salesforce:
    - credentials:
        authType: "oauth2Password"
        username: "janedoe"
        password: "mysecret"
        clientIdentity: "XXXXXX"
        clientSecret: "XXXXXX"
      endpoint:
        loginUrl: "https://eu8.salesforce.com"
```

```
name: "Account 1"
```

The attached [sample-credential.yaml.zip](#) file contains a YAML file with sample entries for each connector that can be deployed locally. You can copy the contents of the YAML file into the `credentials.yaml` file and then edit it to define the credentials that you require.

The following guidelines apply:

- An entry for a connector starts with the name, followed by account parameters that define the credentials, endpoint, and account name.
- The account parameters are shown as key:value pairs; for example, `username: "janedoe"`.
- The key names must not be edited.
- The value for each key must be enclosed in double quotation marks ("").
- You must provide a value for each key/value pair that's identified as "required".
- The name value, which identifies the account name, must be identical to the account name that is used in the API flow that was exported. You can use your preferred values for the `credentials` parameters.
- For an optional parameter, you can either specify a blank value of "" or omit the key/value line from the file if you don't want to specify a value.
- Ensure that your updated `credentials.yaml` file contains valid YAML. Tab characters are not permitted, and must be replaced with spaces if used. You might find it helpful to use a YAML validation tool to check the content.

Note: Additional references:

- See also the attached [schema.yaml.zip](#) file, which contains a YAML file that describes the schema that the `credentials.yaml` file needs to adhere to.

3) Save and close the `credentials.yaml` file.

Note: If the account credentials for a local connector change after you've deployed the integration server, you'll need to configure the Helm release to use the updated credentials. To do this, you must update the `credentials.yaml` file and generate a new secret. Therefore, it's advisable for you to retain the extracted contents of the `config.tar.gz` archive so that you can quickly update the relevant values in the `credentials.yaml` file. For more information, see [Updating account credentials for a local connector](#).

e) Use the `generateSecrets.sh` script to run **kubectl** commands to create a secret:

- 1) Ensure that you are logged in to your namespace in the cluster, as described in [Logging in to your cluster from the command-line interface](#).
- 2) From the same command window, change to the directory that contains the extracted contents of the `config.tar.gz` archive. For example on Windows: `cd C:\mytempdir\config_customerApi\extracted`
- 3) Run the following command to create the secret, where *secretName* is a short string value. (You'll need to generate a secret for each BAR file that you deploy, so you might find it useful to adopt a naming convention for your secrets.)

• **Linux or macOS:**

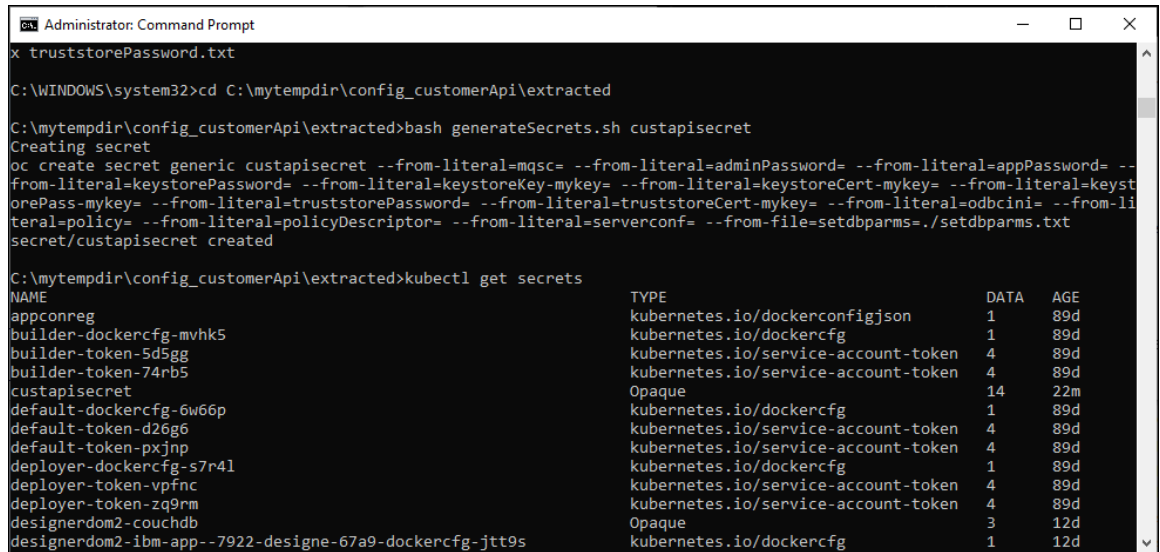
```
./generateSecrets.sh secretName
```

• **Windows:**

```
bash generateSecrets.sh secretName
```


You can check for this new secret in your cluster by running the following command:

```
kubectl get secrets
```



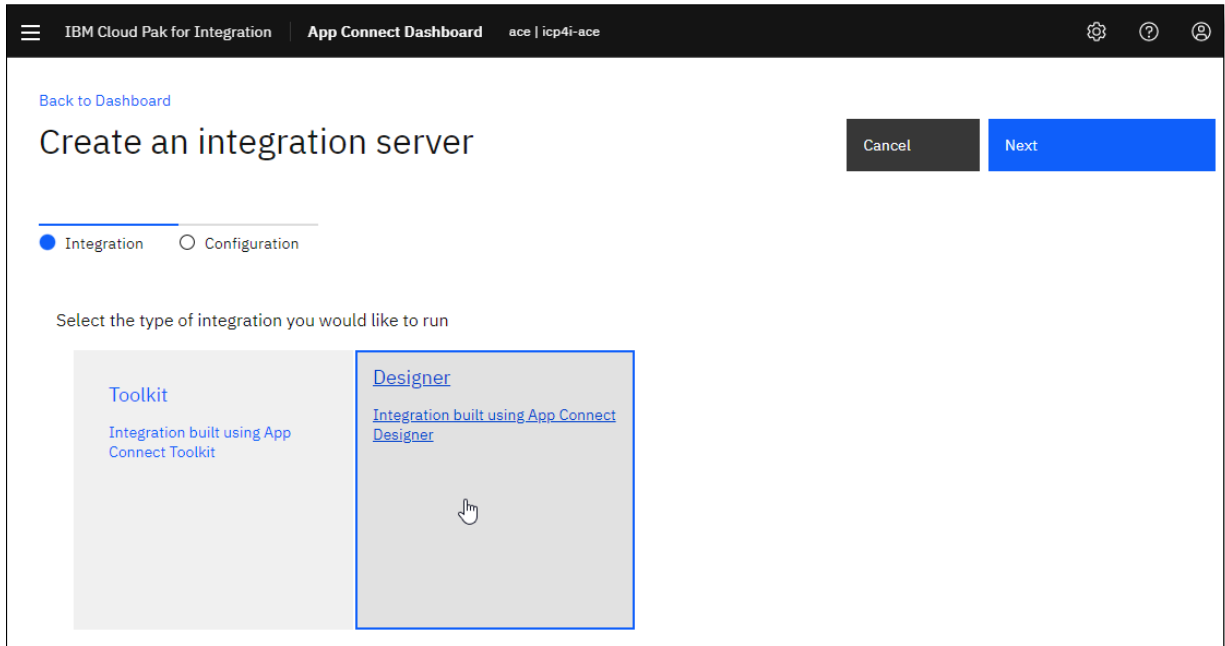
```
Administrator: Command Prompt
x truststorePassword.txt

C:\WINDOWS\system32>cd C:\mytempdir\config_customerApi\extracted

C:\mytempdir\config_customerApi\extracted>bash generateSecrets.sh custapisecret
Creating secret
oc create secret generic custapisecret --from-literal=mqsc= --from-literal=adminPassword= --from-literal=appPassword= --
from-literal=keystorePassword= --from-literal=keystoreKey-mykey= --from-literal=keystoreCert-mykey= --from-literal=keyst
orePass-mykey= --from-literal=truststorePassword= --from-literal=truststoreCert-mykey= --from-literal=odbcini= --from-li
teral=policy= --from-literal=policyDescriptor= --from-literal=serverconf= --from-file=setdbparms=./setdbparms.txt
secret/custapisecret created

C:\mytempdir\config_customerApi\extracted>kubectl get secrets
NAME                                TYPE                                DATA  AGE
appconreg                           kubernetes.io/dockerconfigjson    1      89d
builder-dockercfg-mvhk5              kubernetes.io/dockercfg           1      89d
builder-token-5d5gg                  kubernetes.io/service-account-token 4      89d
builder-token-74rb5                  kubernetes.io/service-account-token 4      89d
custapisecret                        Opaque                             14     22m
default-dockercfg-6w66p              kubernetes.io/dockercfg           1      89d
default-token-d26g6                  kubernetes.io/service-account-token 4      89d
default-token-pxjnp                  kubernetes.io/service-account-token 4      89d
deployer-dockercfg-s7r41              kubernetes.io/dockercfg           1      89d
deployer-token-vpfnc                 kubernetes.io/service-account-token 4      89d
deployer-token-zq9rm                 kubernetes.io/service-account-token 4      89d
designerdom2-couchdb                  Opaque                             3      12d
designerdom2-ibm-app--7922-designe-67a9-dockercfg-jtt9s kubernetes.io/dockercfg           1      12d
```

- 4) Make a note of this secret. When you specify configuration values for the integration server, you'll need to specify this secret in one of the fields.
7. Go back to the "Create an integration server" panel and click **Next**.
8. From the **Integration** tab on the "Create an integration server" page, choose **Designer** as the type of integration that you want to run. Then click **Next**.



The **Configuration** tab on the "Create an integration server" page opens. The initial view shows a subset of the available fields, which are considered sufficient for a standard configuration for the

integration server. For the remaining "hidden" fields, default values will be assumed. For the most part, you can simply accept the default values, which are defined for high availability and large workloads .

9. Complete the **Configuration** tab as follows:

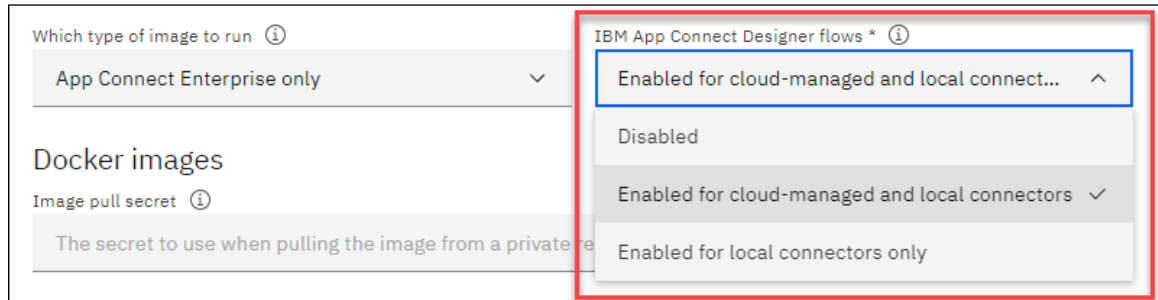
a) Complete the standard configuration fields for installing the BAR file resources:

- **Name:** Enter a short distinctive name that uniquely identifies this Helm release.
- **Which type of image to run:** Accept the default value of **App Connect Enterprise only**. Only change this setting if your integration solution requires an IBM MQ client or IBM MQ server.
- **IBM App Connect Designer flows:** Select an option that enables the use of Designer-authored API flows, and cloud-managed or local connectors.
 - **Enabled for cloud-managed and local connectors:** Select this option to use any combination of cloud-managed or local connectors. This option will extend the functionality of each pod by deploying sidecar containers, which are needed to run APIs that are authored in App Connect Designer, and local connectors. (You'll need to have a secret that contains an IBM Cloud API key and account credentials.)
 If selected, a local connector will be used if a successful connection can be established with the supplied credentials. If the connection fails or if local credentials were not supplied, the cloud-managed connector will be used. The operation will fail if the connection to the cloud-managed connector also fails.
 - **Enabled for local connectors only:** Select this option to use local connectors only. This option will extend the functionality of each pod by deploying sidecar containers, which are needed to

run APIs that are authored in App Connect Designer, and local connectors. (You'll need to have a secret that contains account credentials.)

If selected, a local connector will be used if a successful connection can be established with the supplied credentials. Otherwise, the operation will fail.

- **Disabled:** Select this option (the default) only if you are deploying a BAR file that was created for a Toolkit (non-Designer) integration.



- **Image pull secret:** Specify the secret that can be used to access and pull the Docker images for App Connect Enterprise and the “Designer flows” capability from a private registry. To identify the secret for your namespace, run the following command, where *namespace* is the namespace you specified earlier for this integration server:

```
kubectl get secrets -n namespace | grep default-docker
```

On Windows, you can alternatively run:

```
kubectl get secrets -n namespace | findstr default-docker
```

```
C:\mytempdir\extracted>kubectl get secrets -n ace | findstr default-docker
default-dockercfg-p2gfd          kubernetes.io/dockercfg      1      5d14h
C:\mytempdir\extracted>
```

- **Name of the secret that contains the server configuration:** Specify the secret that you created earlier with the `generateSecrets.sh` script.
- **Enable Operations Dashboard:** Select this check box to enable transaction tracing, which will push trace data to the IBM Cloud Pak for Integration Operations Dashboard to aid with problem investigation and troubleshooting. An Operations Dashboard instance must be available to process the required registration approval for tracing. For more information, see [Capability](#)

[registration](#) and [Operations Dashboard](#) in the IBM Cloud Pak for Integration documentation in IBM Knowledge Center.

- **OD tracing instance namespace:** Specify the namespace where the Operations Dashboard was released. This field is displayed only when **Enable Operations Dashboard** is selected.

The screenshot shows the 'Create an integration server' configuration page. At the top, there is a 'Back to Dashboard' link and a title 'Create an integration server' with 'Back' and 'Create' buttons. Below the title, there are tabs for 'Integration' (selected) and 'Configuration'. Underneath, there are 'UI' and 'Code' tabs, with 'UI' selected. A 'Show everything' toggle is currently 'Off'. A left sidebar contains links for 'Details' (selected), 'Docker images', 'Integration server', 'Operations Dashboard (OD)', and 'configuration'. The main content area is divided into sections: 'Details' with a 'Name *' field containing 'customerapi'; 'Which type of image to run' with a dropdown set to 'App Connect Enterprise only'; 'IBM App Connect Designer flows *' with a dropdown set to 'Enabled for cloud-managed and local connect...'; 'Docker images' with an 'Image pull secret' field containing 'The secret to use when pulling the image from a private registry'; 'Integration server' with a 'Name of the secret that contains the server configuration *' field containing 'custapisecret'; and 'Operations Dashboard (OD) configuration' with an 'Enable Operations Dashboard' checkbox that is currently unchecked.

- b) If you want to view the full set of fields or change some of the default settings, set **Show everything** to **on**.

Back to Dashboard

Create an integration server

Back Create

Integration Configuration

UI Code

Show everything On

Details

Name *
customerapi

License * Content server URL *

Which type of image to run IBM App Connect Designer flows *

Production usage

Docker images

IBM App Connect Enterprise *	IBM App Connect certified container configurator image *
<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>	<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>
IBM App Connect connectors image *	IBM App Connect Designer flows image *
<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>	<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>

Image pull policy Image pull secret

c) To switch to the Code view for the fields, click **Code**.

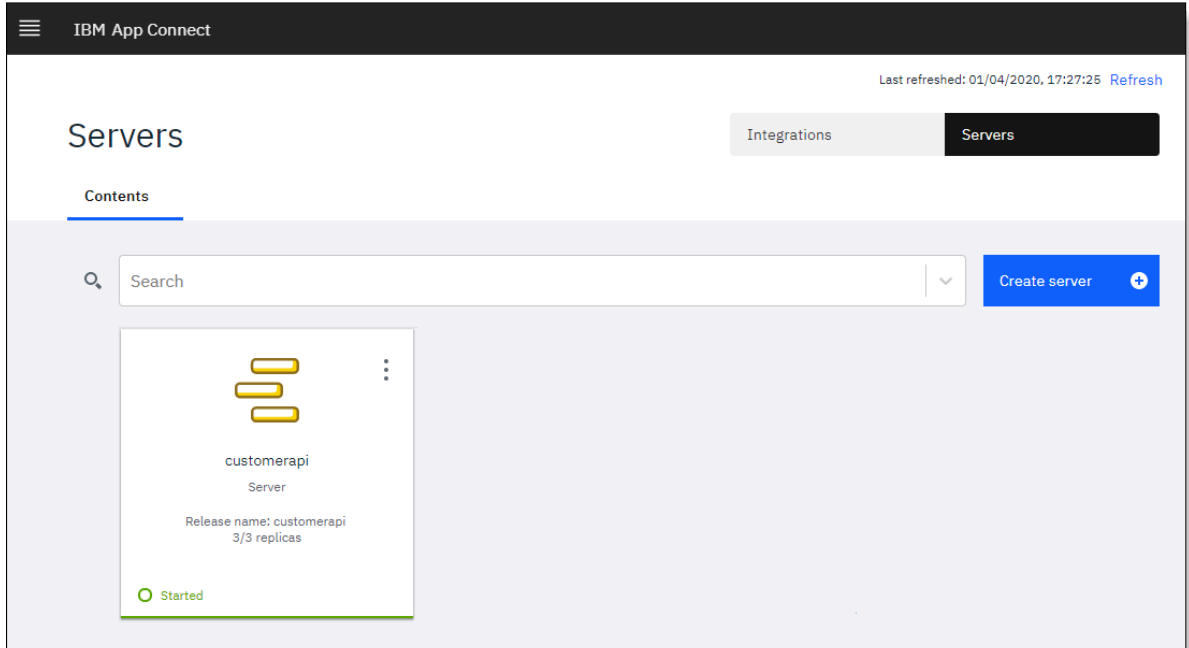
You can manually change the values of the fields in this view. The changes will be reflected in the **UI** view.

```
1 aceonly:
2   replicaCount: 3
3   resources:
4     limits:
5       cpu: '1'
6       memory: 1024Mi
7     requests:
8       cpu: 200m
9       memory: 256Mi
10  adminServerSecure: false
11  arch: amd64
12  connectors:
13    resources:
14      limits:
15        cpu: '1'
16        memory: 768Mi
17      requests:
18        cpu: 150m
19        memory: 200Mi
20  dashboardEnabled: true
21  dataPVC:
22    name: data
23    size: 2Gi
24    storageClassName: ''
25  designerFlowsOperationMode: all
26  designerFlows:
27    resources:
28      limits:
29        cpu: '1'
```

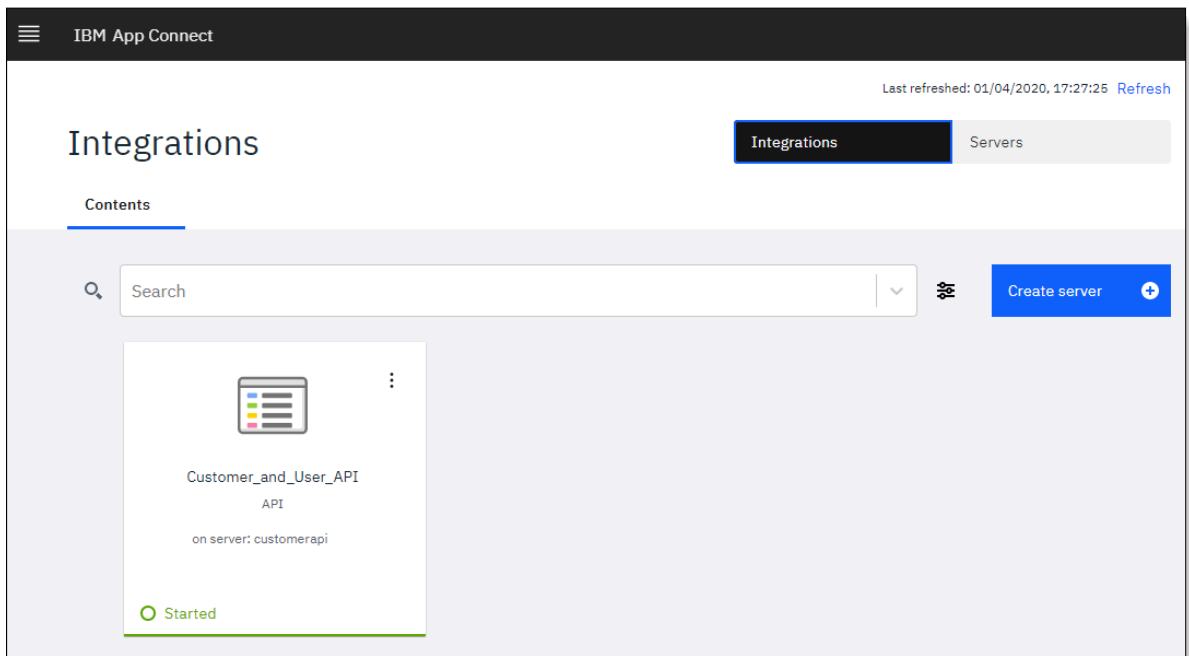
d) Click **Create** to create the Helm release for the integration server.

The integration server is displayed as a tile on the Servers page of the dashboard, with an initial status of *Unavailable* (⚠), which then changes to *Started* when the deployment completes.

The Servers page will also display any other integration servers that are installed in the same namespace.



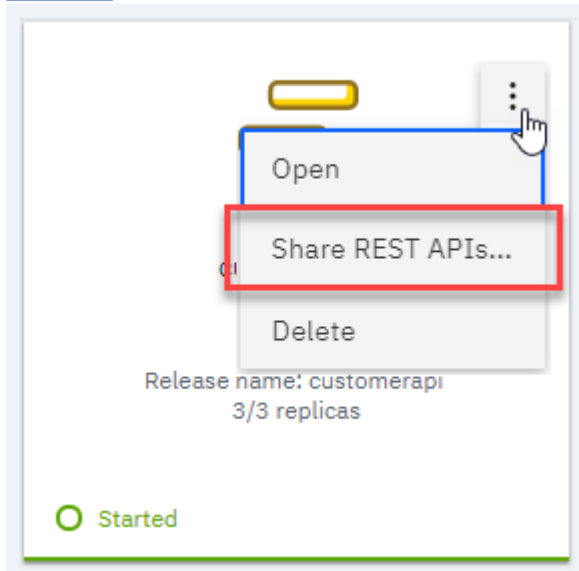
You can click the integration server tile to view the deployed integration, which in this case includes a single API and other resources that are defined in the uploaded BAR file. From the Servers page, you can also view the integrations for all listed integration servers by clicking **Integrations** to open the Integrations page.



Tip:

- If required, you can deploy the same BAR file more than once to accommodate your workloads. A unique Helm release must be created each time, typically with the same Kubernetes secret. Different API endpoints will be generated for each Helm release.
- From the Servers page of the dashboard, you can use the **Share REST APIs** feature to push (or export) the deployed API to [IBM API Connect](#) to take advantage of its advanced API management

capabilities. For more information, see [Pushing REST APIs to IBM API Connect by using the web user interface](#).



What to do next

[Invoke the API.](#)

Updating account credentials for a local connector



If the account credentials change for a local connector that's used in one of your deployed integrations, (for example, because of an expired password), you'll need to update the integration server to use the updated credentials. You can do so by updating the credentials in the `credentials.yaml` file, generating a new secret, and then upgrading the integration server's Helm release to use this new secret.

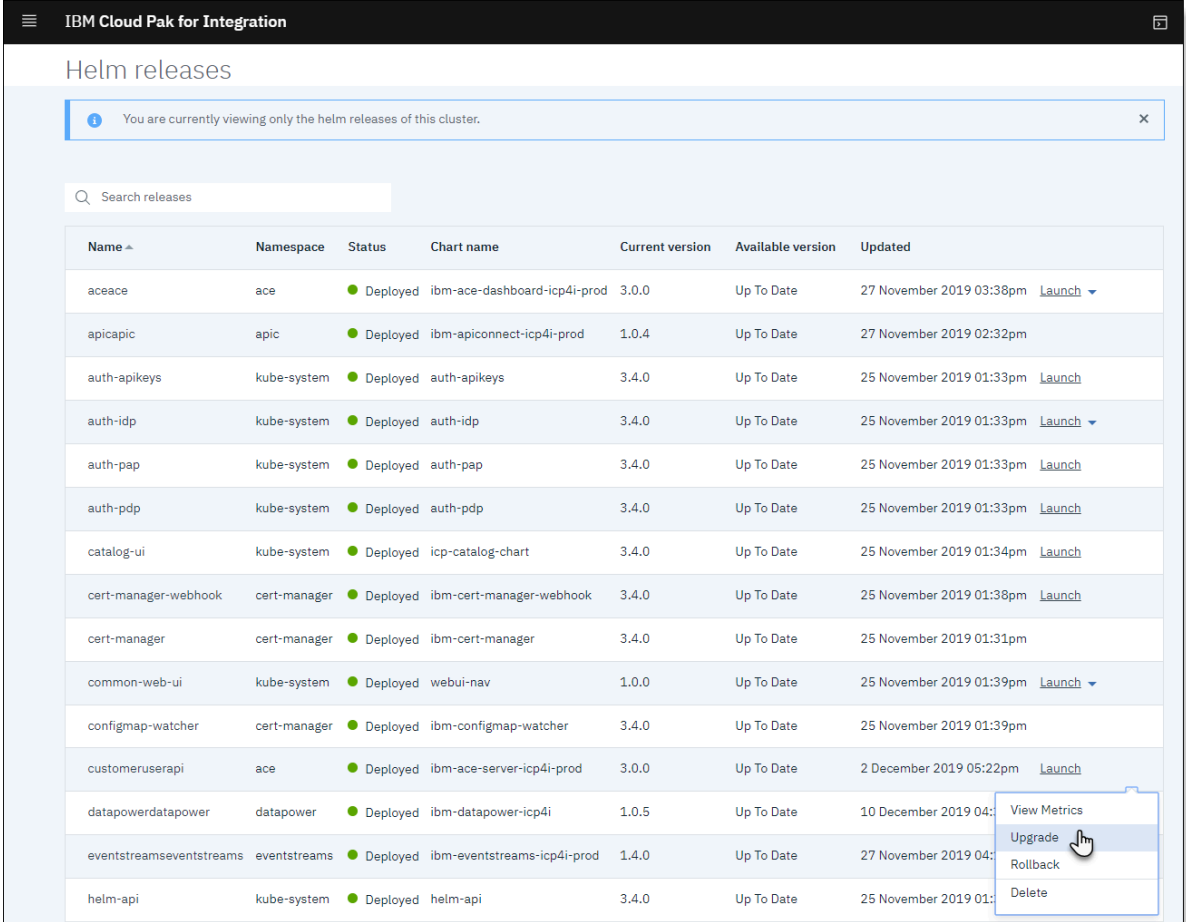
Procedure

To update the account credentials:

1. If not already open, [access your App Connect Dashboard instance](#) from the Platform Navigator for your IBM Cloud Pak for Integration cluster.
2. If you retained the extracted contents of the `config.tar.gz` archive for the deployed integration server, update the credentials as follows:
 - a) From a command window or file browser, go to the directory where the extracted files are stored.
 - b) Locate the `credentials.yaml` file, update the account details as required, and then save and close the file.
For more information about completing the account credentials, see [“Cloud Pak for Integration only: Creating an integration server to run your BAR file resources”](#) on page 70.
3. If you discarded the extracted contents of the `config.tar.gz` archive for the integration server after deployment, update the credentials as follows:
 - a) From the App Connect Dashboard, click **Create server** on the **Servers** page and then use the drop-down list to select the BAR file that was used to deploy the integration server.
 - b) Click **Continue**, and then download and extract the contents of the `config.tar.gz` configuration package.
 - c) Add the account credentials for the local connectors to the `credentials.yaml` file, and then save and close the file.
For more information, see [“Cloud Pak for Integration only: Creating an integration server to run your BAR file resources”](#) on page 70.
4. Generate a new secret by running the `generateSecrets.sh` script, as described in [“Cloud Pak for Integration only: Creating an integration server to run your BAR file resources”](#) on page 70.

5. Upgrade the integration server's Helm release to use the new secret:

- a) From the App Connect Dashboard, open the IBM Cloud Pak menu  and then click **Cloud Pak Foundation**.
- b) From the Cloud Pak management console, open the IBM Cloud Pak menu  and click **Administer > Helm Releases** to open the "**Helm releases**" page.
- c) Locate the Helm release for the integration server that you want to upgrade. Then click the options menu (...) that appears when you hover over the row, and click **Upgrade**.



IBM Cloud Pak for Integration

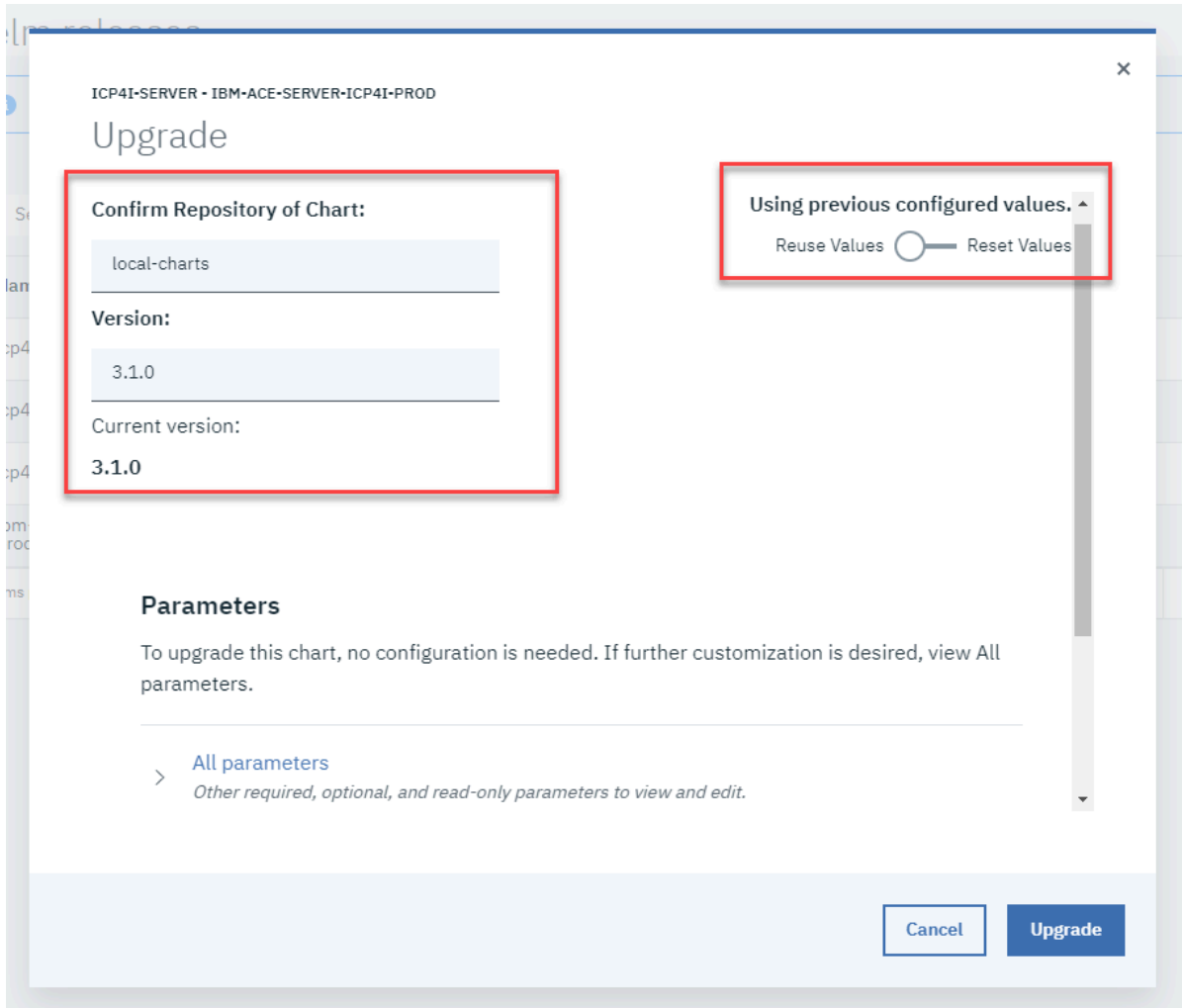
Helm releases

You are currently viewing only the helm releases of this cluster.

Search releases

Name	Namespace	Status	Chart name	Current version	Available version	Updated	
aceace	ace	Deployed	ibm-ace-dashboard-icp4i-prod	3.0.0	Up To Date	27 November 2019 03:38pm	Launch
apicapic	apic	Deployed	ibm-apiconnect-icp4i-prod	1.0.4	Up To Date	27 November 2019 02:32pm	
auth-apikeys	kube-system	Deployed	auth-apikeys	3.4.0	Up To Date	25 November 2019 01:33pm	Launch
auth-idp	kube-system	Deployed	auth-idp	3.4.0	Up To Date	25 November 2019 01:33pm	Launch
auth-pap	kube-system	Deployed	auth-pap	3.4.0	Up To Date	25 November 2019 01:33pm	Launch
auth-pdp	kube-system	Deployed	auth-pdp	3.4.0	Up To Date	25 November 2019 01:33pm	Launch
catalog-ui	kube-system	Deployed	icp-catalog-chart	3.4.0	Up To Date	25 November 2019 01:34pm	Launch
cert-manager-webhook	cert-manager	Deployed	ibm-cert-manager-webhook	3.4.0	Up To Date	25 November 2019 01:38pm	Launch
cert-manager	cert-manager	Deployed	ibm-cert-manager	3.4.0	Up To Date	25 November 2019 01:31pm	
common-web-ui	kube-system	Deployed	webui-nav	1.0.0	Up To Date	25 November 2019 01:39pm	Launch
configmap-watcher	cert-manager	Deployed	ibm-configmap-watcher	3.4.0	Up To Date	25 November 2019 01:39pm	
customeruserapi	ace	Deployed	ibm-ace-server-icp4i-prod	3.0.0	Up To Date	2 December 2019 05:22pm	Launch
datapowerdatapower	datapower	Deployed	ibm-datapower-icp4i	1.0.5	Up To Date	10 December 2019 04:00pm	View Metrics
eventstreameventstreams	eventstreams	Deployed	ibm-eventstreams-icp4i-prod	1.4.0	Up To Date	27 November 2019 04:00pm	Upgrade
helm-api	kube-system	Deployed	helm-api	3.4.0	Up To Date	25 November 2019 01:33pm	Rollback
							Delete

- d) From the **Upgrade** panel, select the repository of the chart (**local-charts**) and select the current version.
- e) Ensure that **Reuse Values** is set to **on** to retain the values that were used during the previous upgrade.



- f) Expand **All parameters**, locate the **The name of the secret to create or to use that contains the server configuration** field, and then overwrite the value with the new secret that you just generated.

CUSTOMERUSERAPI - IBM-ACE-SERVER-ICP4I-PROD

Upgrade

Integration Server
Define configuration for the Integration Server

Integration server name

List of key aliases for the keystore

List of certificate aliases for the truststore

Name of the default application

The name of the secret to create or to use that contains the server configuration

File system group ID

- g) Click **Upgrade** and wait for the processing to complete. An automatic rolling update of the replica pods will occur to pick up the new credentials without any downtime (because the pods become unavailable in turn during the update).
 - h) Go back to the tab that contains the App Connect Dashboard. The integration server that you just upgraded should be in a Started state.
6. Optional: After all the pods are updated, delete the previous secret that was used for this Helm release, as described in [“Deleting a secret”](#) on page 106.

App Connect Enterprise certified container only: Creating an integration server to run your BAR file resources

The BAR file for your exported API flow contains all the resources that are needed to deploy an integration that exposes the API and its operations. To run this API, you’ll need to deploy the BAR file to an integration server by following a multi-step process.

Before you begin

1. [From an App Connect Designer instance, create an API flow that meets the conditions for export, and export the API flow to a BAR file.](#) Or [from App Connect on IBM Cloud, create an API flow that meets the conditions for export, and export the API flow to a BAR file.](#)
2. [Optional. Override the values for the App Connect instance ID and region-specific URL in the BAR file.](#)
3. [If not yet installed, install the command-line tools for your cluster.](#)

4. If you do not have one available, create an IBM Cloud API key in your IBM Cloud account to enable you to use cloud-managed connectors.

About this task

You can deploy the BAR file to an integration server by completing a multi-step process:

1. Use the App Connect Dashboard to upload the BAR file to a content server in your cluster.
2. Use a supplied configuration package to generate a Kubernetes secret that the deployed API can use to establish a connection to the connectors and accounts that it requires.
 - To use cloud-managed (App Connect on IBM Cloud) connectors to run one or more API operations, you must use your IBM Cloud API key and the configuration package to generate the Kubernetes secret.
 - If you want to deploy and use local connectors in your cluster, you must use the configuration package to configure and store the account credentials for these connectors in the Kubernetes secret. (To check which connectors can be deployed locally, see [Exporting an API flow to a BAR file.](#))
3. Configure and install a Helm release for the integration server.

When the deployment completes, an integration server is created and started, and it reads the BAR file to run the integration.

Procedure

To create an integration server, complete the following steps:

1. Access your App Connect Dashboard instance by completing either of the following steps:
 - From a browser window, enter the URL that your administrator provided for the App Connect Dashboard instance, and log in when prompted.
 - From a command window, run the following commands to obtain the URL and then access your App Connect Dashboard instance:
 - a. From a browser window, enter the URL that your administrator provided for the App Connect Dashboard instance, and log in when prompted.
 - b. From a command window, run the following commands to obtain the URL and then access your App Connect Dashboard instance:
 - 1) Log in to your cluster by running the `oc login` command and provide the OpenShift Container Platform server URL (and optionally a token). Examples:

```
oc login https://Mycluster_hostName:8443
```

```
oc login --token=AbcdE1fgHijKLM2moP3q4rs5TU6vw7xYz --server=https://  
Mycluster_hostName:6443
```

- 2) Indicate whether to use insecure connections, and then specify a user name and password if required.
- 3) Run the following command, where *releaseName* is the Helm release name of the instance:

```
helm status releaseName --tiller-namespace=tiller
```

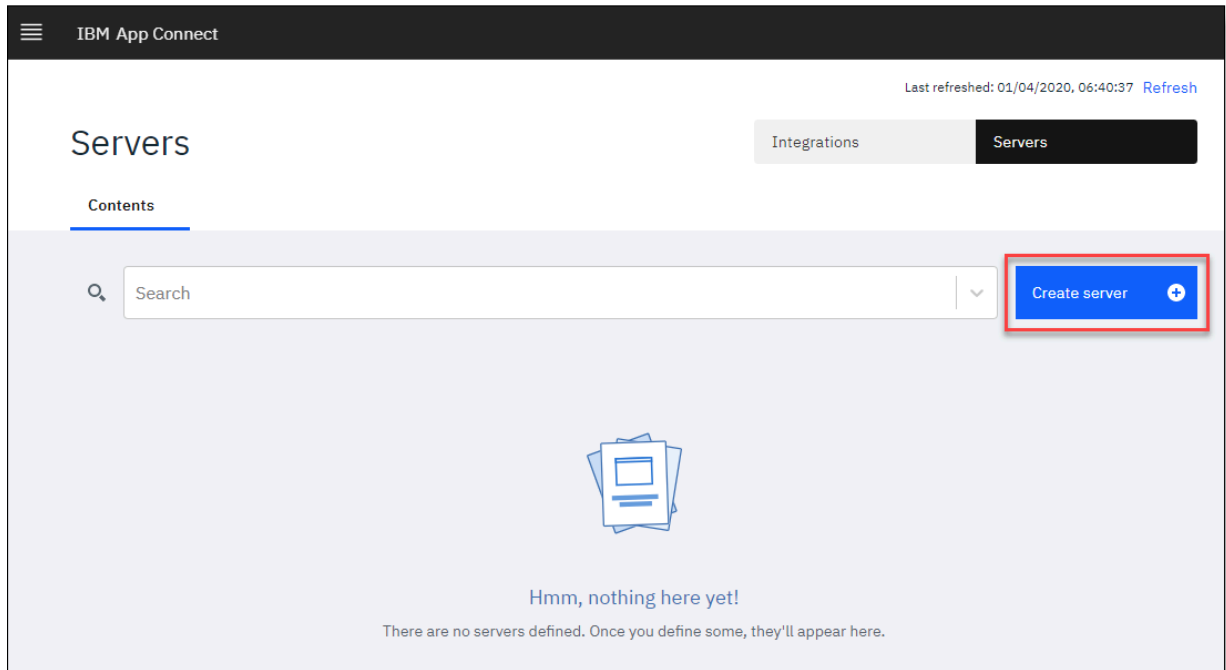
This command outputs some information in a NOTES section, including a command that you can run to discover the App Connect Dashboard instance URL. Look for the line that starts with this command:

```
echo && export ACE_DASHBOARD_URL=...
```

- 4) Copy and run this command to obtain the dashboard URL.
- 5) Copy and paste the URL into a browser window, and log in when prompted.

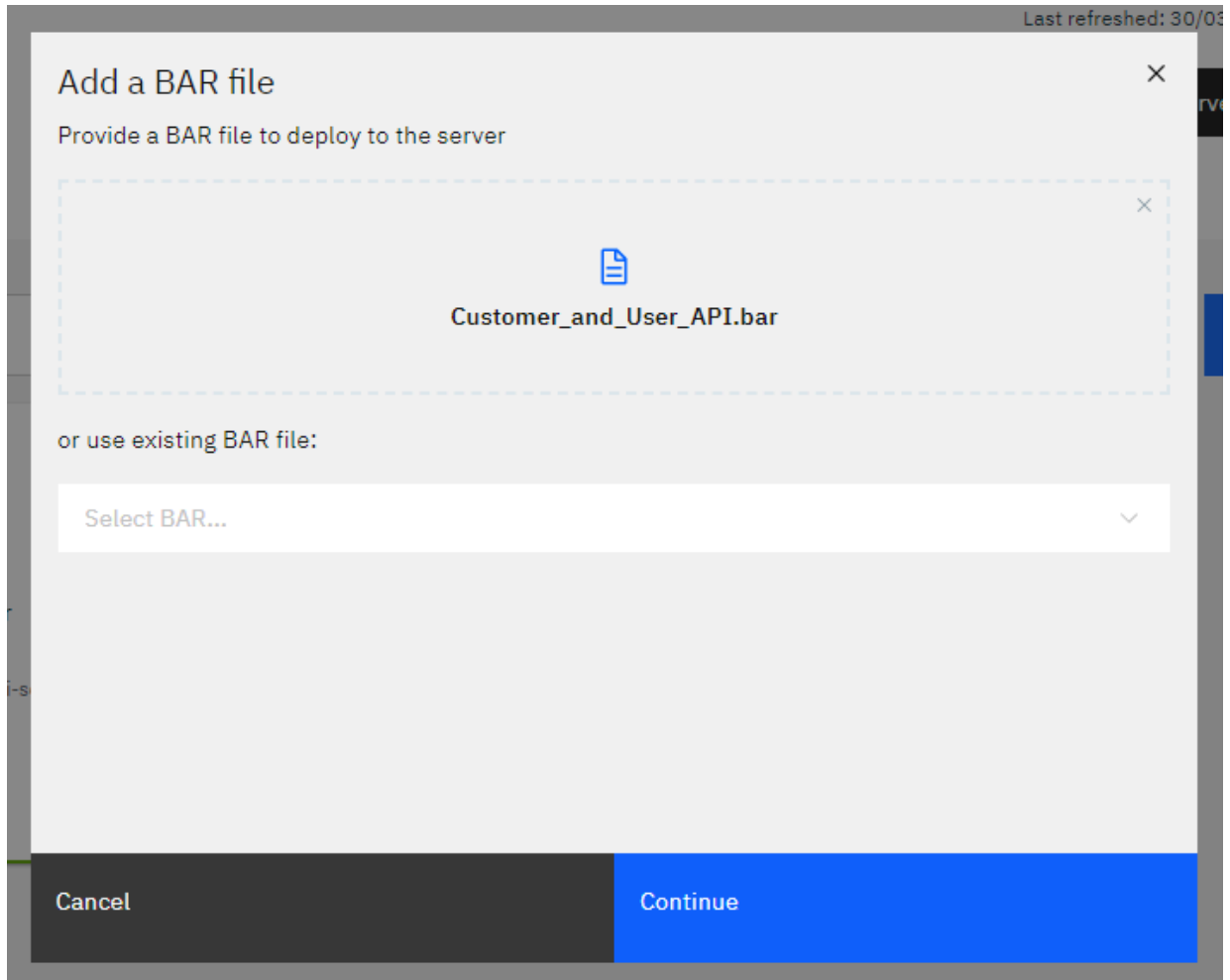
The App Connect Dashboard window opens. If any flows have been created, they are displayed as tiles in the dashboard.

2. From the Servers page on the dashboard, click **Create server** to deploy the BAR file to the content server.

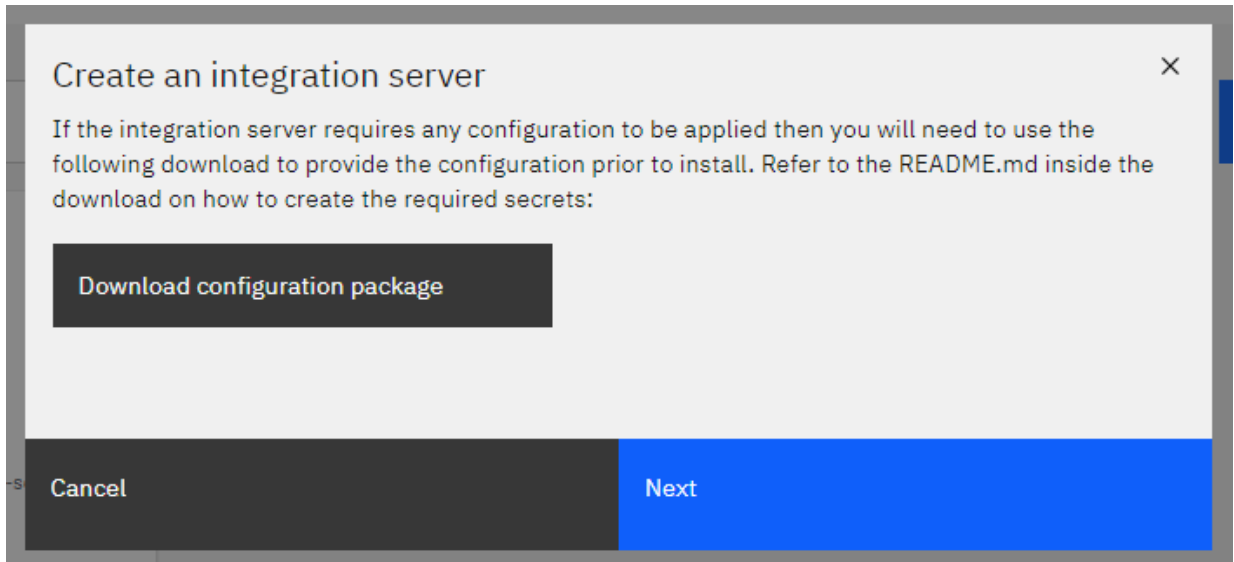


3. From the "Add a BAR file" panel, complete any of these steps:

- Drag and drop the file from its location in an open file browser into the boxed area.
- Click within the boxed area to open a file browser and locate the BAR file.
- From the drop-down list, select an existing BAR file that was previously uploaded to the content server. (This option is useful if you want to create more than one integration server from the same BAR file to manage your workloads.)



4. Click **Continue**.



5. Download a configuration package for connectors.

The configuration package contains a set of files that you can use to generate a Kubernetes secret that stores the account credentials for your local connectors, as well as the IBM Cloud API key that permits access to your cloud-managed connectors. The configuration package is downloaded to your browser's default location as a `config.tar.gz` archive.

a) Click **Download configuration package**

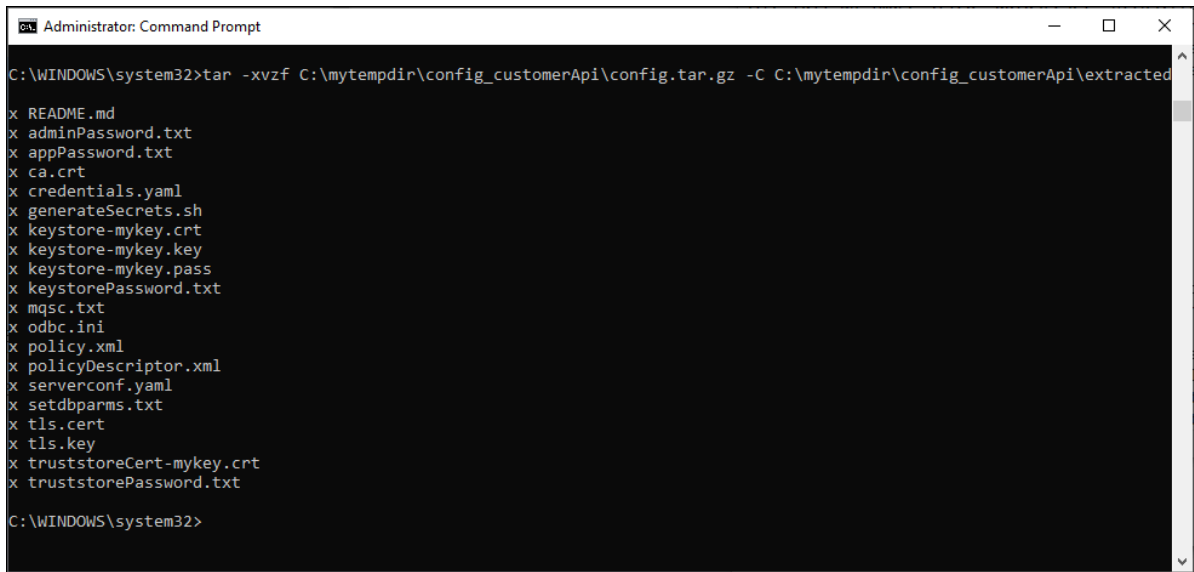
b) Extract the contents of the `config.tar.gz` archive to a temporary directory:

- **Linux or macOS:** Open a terminal window and run the following command from the directory that contains the downloaded file:

```
tar -xvf config.tar.gz
```

- **Windows:** Run **cmd.exe** as an administrator, and then run the **tar** command in the following format:

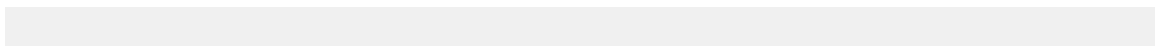
```
tar -xvzf sourcePath\config.tar.gz -C extractedPath
```



```
Administrator: Command Prompt
C:\WINDOWS\system32>tar -xvzf C:\mytempdir\config_customerApi\config.tar.gz -C C:\mytempdir\config_customerApi\extracted
x README.md
x adminPassword.txt
x appPassword.txt
x ca.crt
x credentials.yaml
x generateSecrets.sh
x keystore-mykey.crt
x keystore-mykey.key
x keystore-mykey.pass
x keystorePassword.txt
x mqsc.txt
x odbc.ini
x policy.xml
x policyDescriptor.xml
x serverconf.yaml
x setdbparms.txt
x tls.crt
x tls.key
x truststoreCert-mykey.crt
x truststorePassword.txt
C:\WINDOWS\system32>
```

c) If you want to use cloud-managed connectors, edit the extracted `setdbparms.txt` file to provide your IBM Cloud API key details.

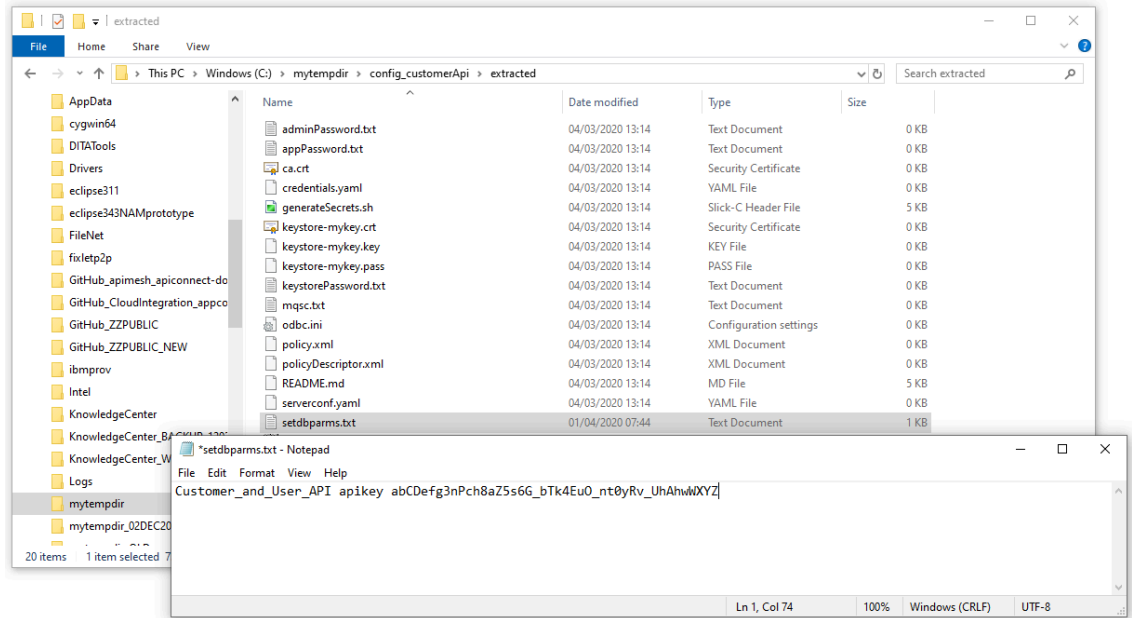
1) Add a line in this format to the empty file:



`flow_name apikey IBMCloud_APIkey`

Where:

- `flow_name` is the file name stem of the BAR file.
- `IBMCloud_APIkey` is the "apiKey" value in the `apiKey.json` file that contains details of the IBM Cloud API key.



2) Save and close the file.

d) If you want to locally deploy connectors to run API operations, configure account credentials for these connectors.

(For details of the supported connectors, see [Exporting an API flow to a BAR file](#).)

- 1) Go to the directory where you extracted the contents of the `config.tar.gz` archive and locate the `credentials.yaml` file.
- 2) Edit this file to add connection credentials for the connectors that you want to deploy locally.

The first line of the file must start with the text `accounts:`, followed by account metadata details for each unique "connector and account" combination that's used in the flow. The following example shows an entry that defines the account credentials for HTTP and Salesforce connectors.

```
accounts:
  http:
    - credentials:
        authType: "none"
        username: ""
        password: ""
      endpoint:
        endpointUrl: ""
        name: "Account 1"
  salesforce:
    - credentials:
        authType: "oauth2Password"
        username: "janedoe"
        password: "mysecret"
        clientIdentity: "XXXXXX"
        clientSecret: "XXXXXX"
      endpoint:
        loginUrl: "https://eu8.salesforce.com"
```

```
name: "Account 1"
```

The attached [sample-credential.yaml.zip](#) file contains a YAML file with sample entries for each connector that can be deployed locally. You can copy the contents of the YAML file into the `credentials.yaml` file and then edit it to define the credentials that you require.

The following guidelines apply:

- An entry for a connector starts with the name, followed by account parameters that define the credentials, endpoint, and account name.
- The account parameters are shown as key:value pairs; for example, `username: "janedoe"`.
- The key names must not be edited.
- The value for each key must be enclosed in double quotation marks ("").
- You must provide a value for each key/value pair that's identified as "required".
- The name value, which identifies the account name, must be identical to the account name that is used in the API flow that was exported. You can use your preferred values for the `credentials` parameters.
- For an optional parameter, you can either specify a blank value of "" or omit the key/value line from the file if you don't want to specify a value.
- Ensure that your updated `credentials.yaml` file contains valid YAML. Tab characters are not permitted, and must be replaced with spaces if used. You might find it helpful to use a YAML validation tool to check the content.

Note: Additional references:

- See also the attached [schema.yaml.zip](#) file, which contains a YAML file that describes the schema that the `credentials.yaml` file needs to adhere to.

3) Save and close the `credentials.yaml` file.

Note: If the account credentials for a local connector change after you've deployed the integration server, you'll need to configure the Helm release to use the updated credentials. To do this, you must update the `credentials.yaml` file and generate a new secret. Therefore, it's advisable for you to retain the extracted contents of the `config.tar.gz` archive so that you can quickly update the relevant values in the `credentials.yaml` file. For more information, see [Updating account credentials for a local connector](#).

e) Use the `generateSecrets.sh` script to run **kubectl** commands to create a secret:

- 1) Ensure that you are logged in to your namespace in the cluster, as described in [Logging in to your cluster from the command-line interface](#).
- 2) From the same command window, change to the directory that contains the extracted contents of the `config.tar.gz` archive. For example on Windows: `cd C:\mytempdir\config_customerApi\extracted`
- 3) Run the following command to create the secret, where *secretName* is a short string value. (You'll need to generate a secret for each BAR file that you deploy, so you might find it useful to adopt a naming convention for your secrets.)

• **Linux or macOS:**

```
./generateSecrets.sh secretName
```

• **Windows:**

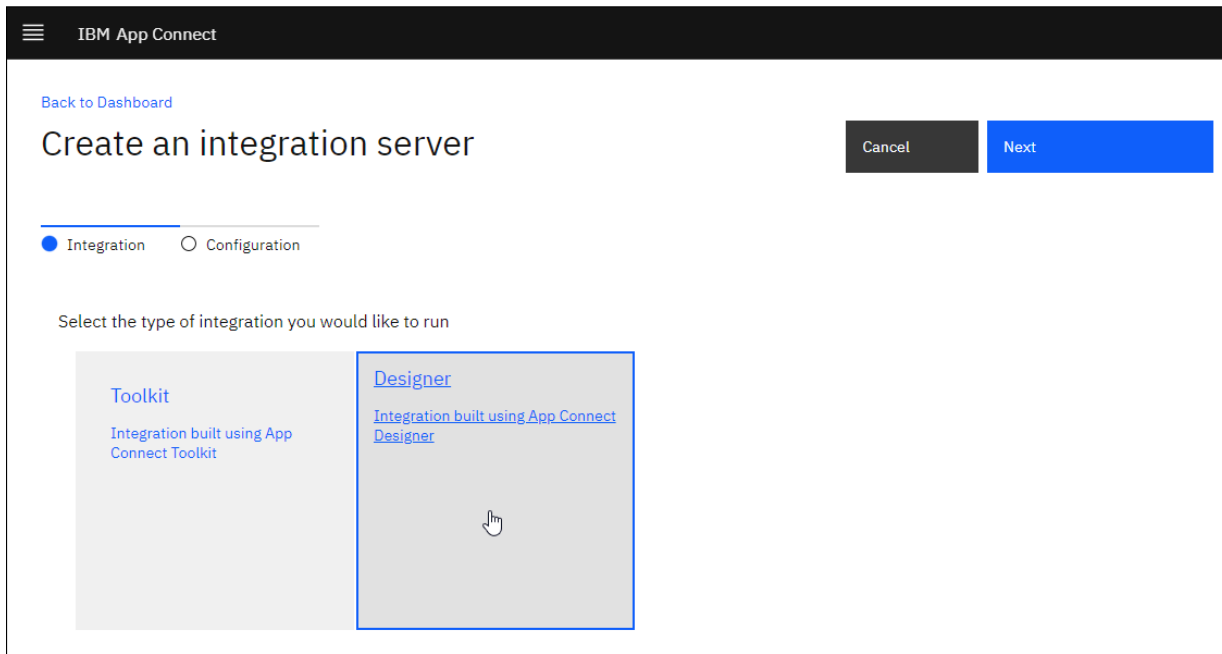
```
bash generateSecrets.sh secretName
```

You can check for this new secret in your cluster by running the following command:

```
kubectl get secrets
```

```
Administrator: Command Prompt
x truststorePassword.txt
C:\WINDOWS\system32>cd C:\mytempdir\config_customerApi\extracted
C:\mytempdir\config_customerApi\extracted>bash generateSecrets.sh custapisecret
Creating secret
oc create secret generic custapisecret --from-literal=mqsc= --from-literal=adminPassword= --from-literal=appPassword= --
from-literal=keystorePassword= --from-literal=keystoreKey-mykey= --from-literal=keystoreCert-mykey= --from-literal=keyst
onePass-mykey= --from-literal=truststorePassword= --from-literal=truststoreCert-mykey= --from-literal=odbcini= --from-li
teral=policy= --from-literal=policyDescriptor= --from-literal=serverconf= --from-file=setdbparms=./setdbparms.txt
secret/custapisecret created
C:\mytempdir\config_customerApi\extracted>kubectl get secrets
NAME                                     TYPE                                     DATA   AGE
appconreg                               kubernetes.io/dockerconfigjson         1       89d
builder-dockercfg-mvhk5                 kubernetes.io/dockercfg                1       89d
builder-token-5d5gg                    kubernetes.io/service-account-token    4       89d
builder-token-74rb5                    kubernetes.io/service-account-token    4       89d
custapisecret                           Opaque                                  14      22m
default-dockercfg-6w66p                 kubernetes.io/dockercfg                1       89d
default-token-d26g6                    kubernetes.io/service-account-token    4       89d
default-token-pxjnp                    kubernetes.io/service-account-token    4       89d
deployer-dockercfg-s7r41                kubernetes.io/dockercfg                1       89d
deployer-token-vpfnc                   kubernetes.io/service-account-token    4       89d
deployer-token-zq9rm                   kubernetes.io/service-account-token    4       89d
designerdom2-couchdb                     Opaque                                  3       12d
designerdom2-ibm-app--7922-designe-67a9-dockercfg-jtt9s kubernetes.io/dockercfg                1       12d
```

- 4) Make a note of this secret. When you specify configuration values for the integration server, you'll need to specify this secret in one of the fields.
6. Go back to the "Create an integration server" panel and click **Next**.
7. From the **Integration** tab on the "Create an integration server" page, choose **Designer** as the type of integration that you want to run.
Then click **Next**.



The **Configuration** tab on the "Create an integration server" page opens. The initial view shows a subset of the available fields, which are considered sufficient for a standard configuration for the

integration server. For the remaining "hidden" fields, default values will be assumed. For the most part, you can simply accept the default values, which are defined for high availability and large workloads .

IBM App Connect

Back to Dashboard

Create an integration server

Back Create

Integration Configuration

UI Code

Show everything Off

Details

Docker images

Integration server

Operations Dashboard (OD) configuration

Details

Name *

name !

Name is required

Which type of image to run ⓘ App Connect Enterprise only

IBM App Connect Designer flows * ⓘ Enabled for cloud-managed and local connect...

Docker images

Image pull secret ⓘ

The secret to use when pulling the image from a private registry

Integration server

Name of the secret that contains the server configuration *

configurationSecret !

Name of the secret that contains the server configuration is required

Operations Dashboard (OD) configuration

Enable Operations Dashboard ⓘ

8. Complete the **Configuration** tab as follows:

a) Complete the standard configuration fields for installing the BAR file resources:

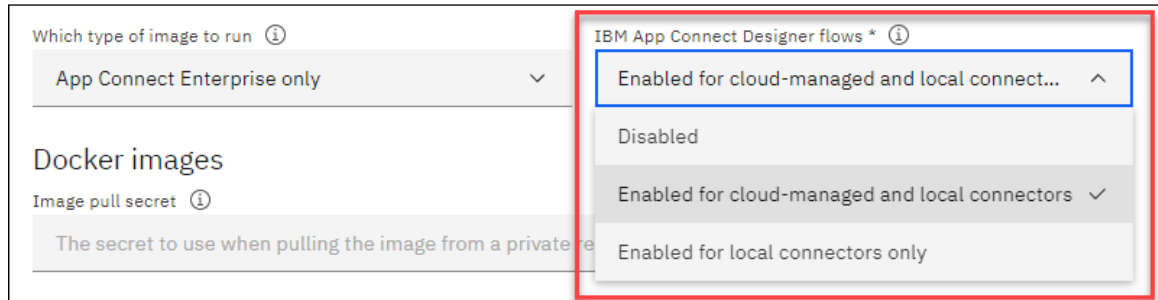
- **Name:** Enter a short distinctive name that uniquely identifies this Helm release.
- **Which type of image to run:** Accept the default value of **App Connect Enterprise only**. Only change this setting if your integration solution requires an IBM MQ client or IBM MQ server.
- **IBM App Connect Designer flows:** Select an option that enables the use of Designer-authored API flows, and cloud-managed or local connectors.
 - **Enabled for cloud-managed and local connectors:** Select this option to use any combination of cloud-managed or local connectors. This option will extend the functionality of each pod by deploying sidecar containers, which are needed to run APIs that are authored in App Connect Designer, and local connectors. (You'll need to have a secret that contains an IBM Cloud API key and account credentials.)

If selected, a local connector will be used if a successful connection can be established with the supplied credentials. If the connection fails or if local credentials were not supplied, the cloud-managed connector will be used. The operation will fail if the connection to the cloud-managed connector also fails.
 - **Enabled for local connectors only:** Select this option to use local connectors only. This option will extend the functionality of each pod by deploying sidecar containers, which are needed to

run APIs that are authored in App Connect Designer, and local connectors. (You'll need to have a secret that contains account credentials.)

If selected, a local connector will be used if a successful connection can be established with the supplied credentials. Otherwise, the operation will fail.

- **Disabled:** Select this option (the default) only if you are deploying a BAR file that was created for a Toolkit (non-Designer) integration.



- **Image pull secret:** Specify the secret that can be used to access and pull the Docker images for App Connect Enterprise and the “Designer flows” capability from a private registry. To identify the secret for your namespace, run the following command, where *namespace* is the namespace you specified earlier for this integration server:

```
kubectl get secrets -n namespace | grep default-docker
```

On Windows, you can alternatively run:

```
kubectl get secrets -n namespace | findstr default-docker
```

```
C:\mytempdir\extracted>kubectl get secrets -n ace | findstr default-docker
default-dockercfg-p2gfd          kubernetes.io/dockercfg           1          5d14h
C:\mytempdir\extracted>
```

- **Name of the secret that contains the server configuration:** Specify the secret that you created earlier with the `generateSecrets.sh` script.
- **Enable Operations Dashboard:** Select this check box to enable transaction tracing, which will push trace data to the IBM Cloud Pak for Integration Operations Dashboard to aid with problem investigation and troubleshooting. An Operations Dashboard instance must be available to process the required registration approval for tracing. For more information, see [Capability](#)

[registration](#) and [Operations Dashboard](#) in the IBM Cloud Pak for Integration documentation in IBM Knowledge Center.

- **OD tracing instance namespace:** Specify the namespace where the Operations Dashboard was released. This field is displayed only when **Enable Operations Dashboard** is selected.

The screenshot shows the 'Create an integration server' configuration page. At the top, there is a 'Back to Dashboard' link and a 'Create' button. Below the title, there are tabs for 'Integration' and 'Configuration', with 'Configuration' selected. There are also tabs for 'UI' and 'Code', with 'UI' selected. A 'Show everything' toggle is set to 'Off'. A sidebar on the left contains links for 'Details', 'Docker images', 'Integration server', 'Operations Dashboard (OD) configuration', and 'configuration'. The main content area is titled 'Details' and contains the following fields:

- Name ***: A text input field containing 'customerapi'.
- Which type of image to run**: A dropdown menu set to 'App Connect Enterprise only'.
- IBM App Connect Designer flows ***: A dropdown menu set to 'Enabled for cloud-managed and local connect...'.
- Docker images**: A section with an 'Image pull secret' field containing the text 'The secret to use when pulling the image from a private registry'.
- Integration server**: A section with a 'Name of the secret that contains the server configuration *' field containing 'custapisecret'.
- Operations Dashboard (OD) configuration**: A section with an 'Enable Operations Dashboard' checkbox, which is currently unchecked.

- b) If you want to view the full set of fields or change some of the default settings, set **Show everything** to on.

Back to Dashboard

Create an integration server

Back Create

Integration Configuration

UI Code

Show everything On

Details

Name *
customerapi

License * Content server URL *

Which type of image to run IBM App Connect Designer flows *

Production usage

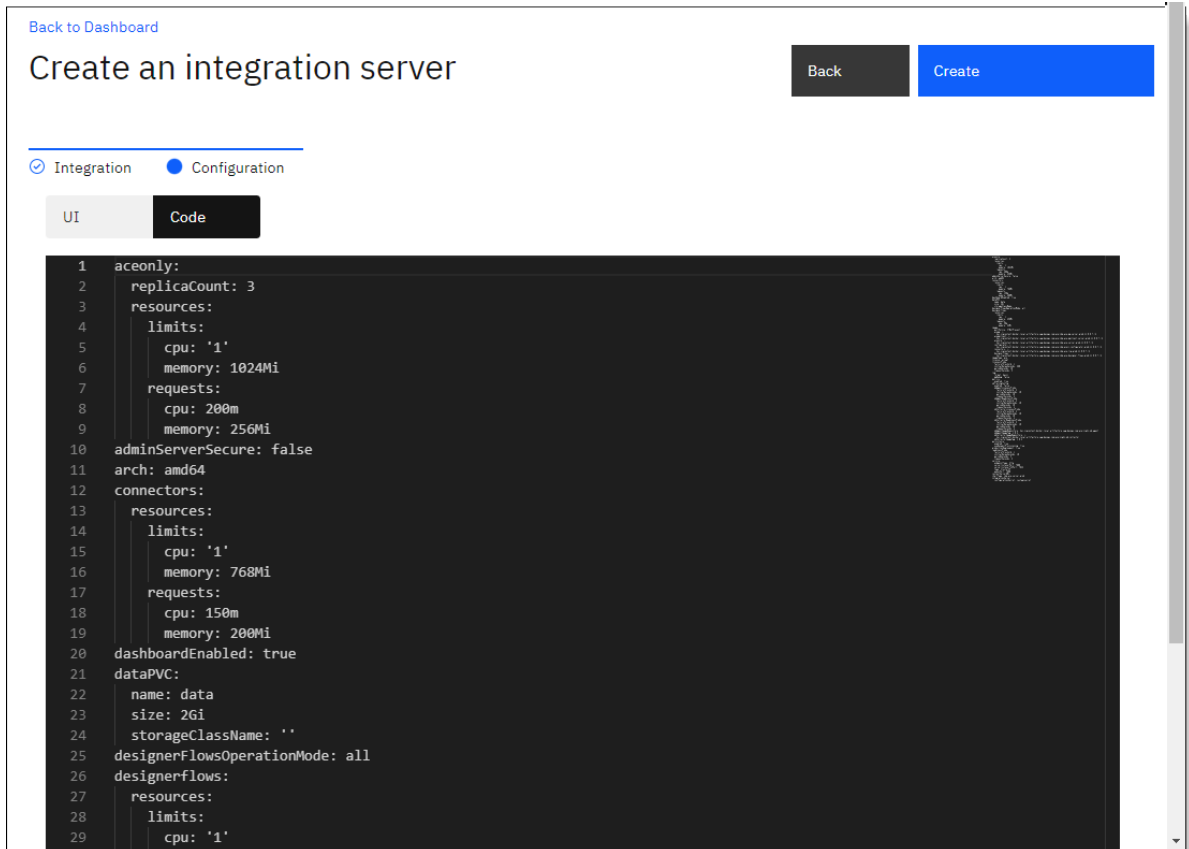
Docker images

IBM App Connect Enterprise *	IBM App Connect certified container configurator image *
<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>	<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>
IBM App Connect connectors image *	IBM App Connect Designer flows image *
<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>	<input type="text" value="hyc-icpcontent-docker-local.artifactory.swg-devops."/>

Image pull policy Image pull secret

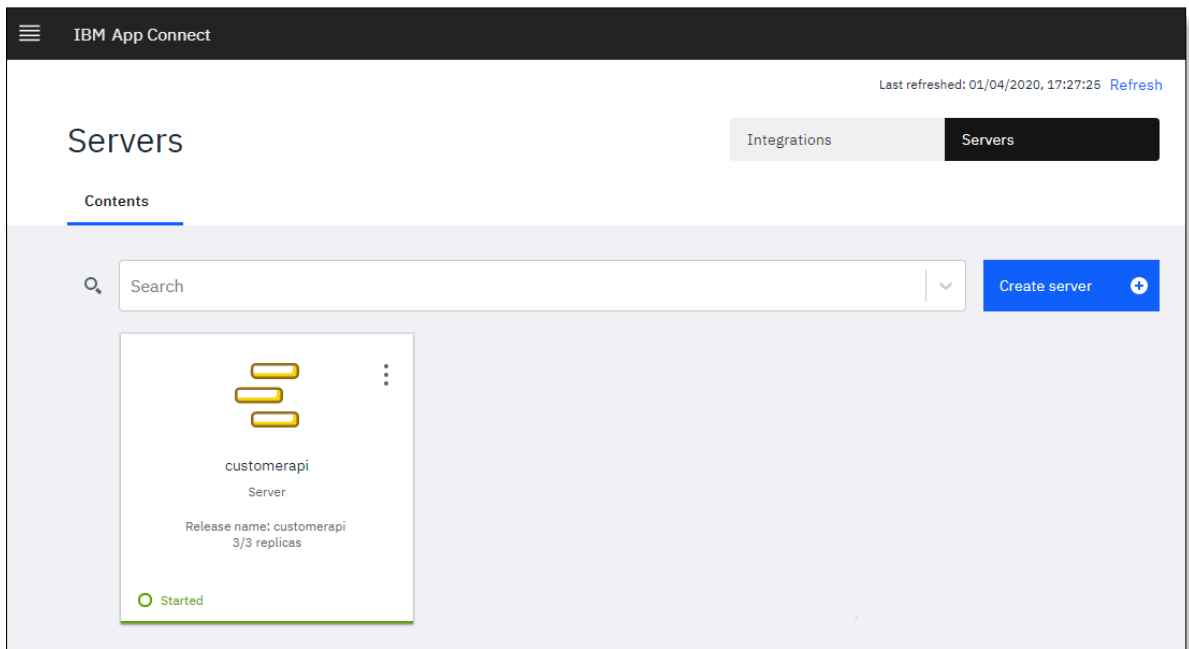
c) To switch to the Code view for the fields, click **Code**.

You can manually change the values of the fields in this view. The changes will be reflected in the **UI** view.



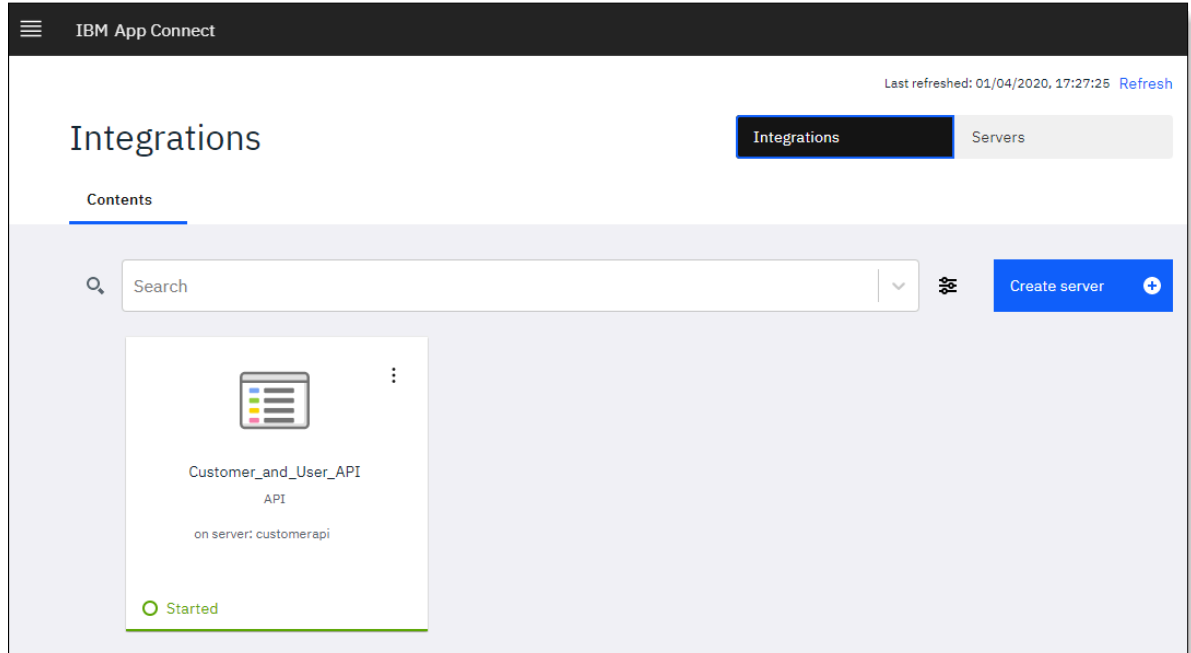
d) Click **Create** to create the Helm release for the integration server.

The integration server is displayed as a tile on the Servers page of the dashboard, with an initial status of *Unavailable* (⚠), which then changes to *Started* when the deployment completes. The Servers page will also display any other integration servers that are installed in the same namespace.



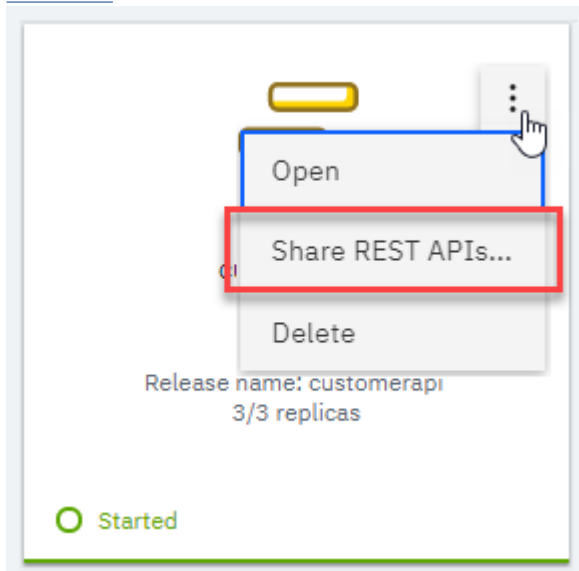
You can click the integration server tile to view the deployed integration, which in this case includes a single API and other resources that are defined in the

uploaded BAR file. From the Servers page, you can also view the integrations for all listed integration servers by clicking **Integrations** to open the Integrations page.



Tip:

- If required, you can deploy the same BAR file more than once to accommodate your workloads. A unique Helm release must be created each time, typically with the same Kubernetes secret. Different API endpoints will be generated for each Helm release.
- From the Servers page of the dashboard, you can use the **Share REST APIs** feature to push (or export) the deployed API to [IBM API Connect](#) to take advantage of its advanced API management capabilities. For more information, see [Pushing REST APIs to IBM API Connect by using the web user interface](#).



What to do next

[Invoke the API.](#)

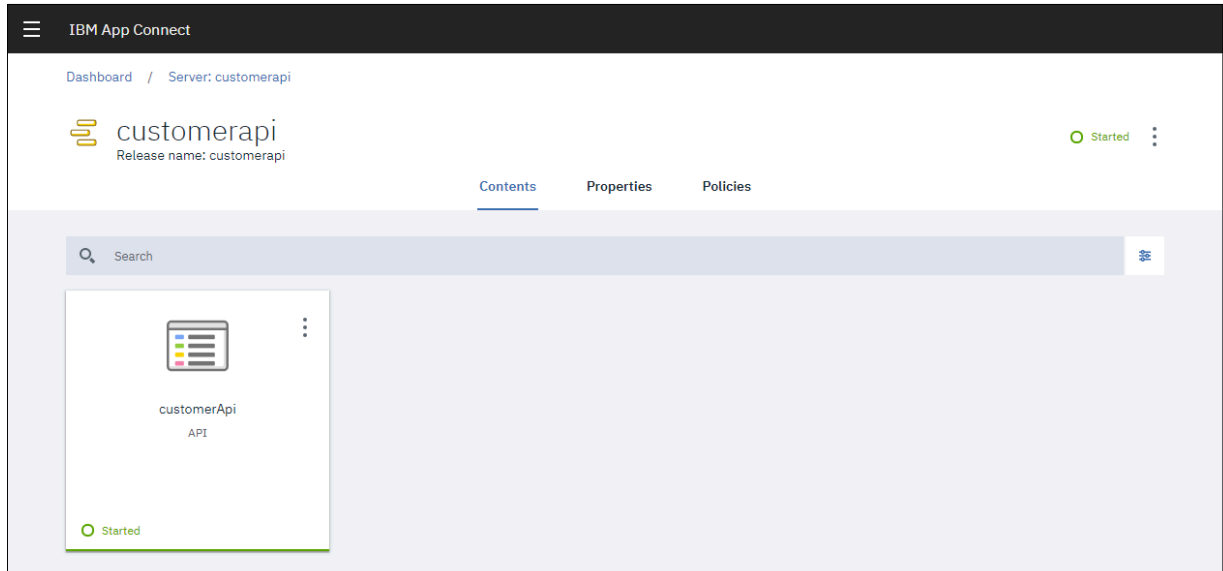
Invoking the deployed API

After you deploy an integration server to the private cloud, you can view the underlying API definition. You can then invoke the API by using the details that are provided for the API endpoint, HTTP methods, and operation paths.

Procedure

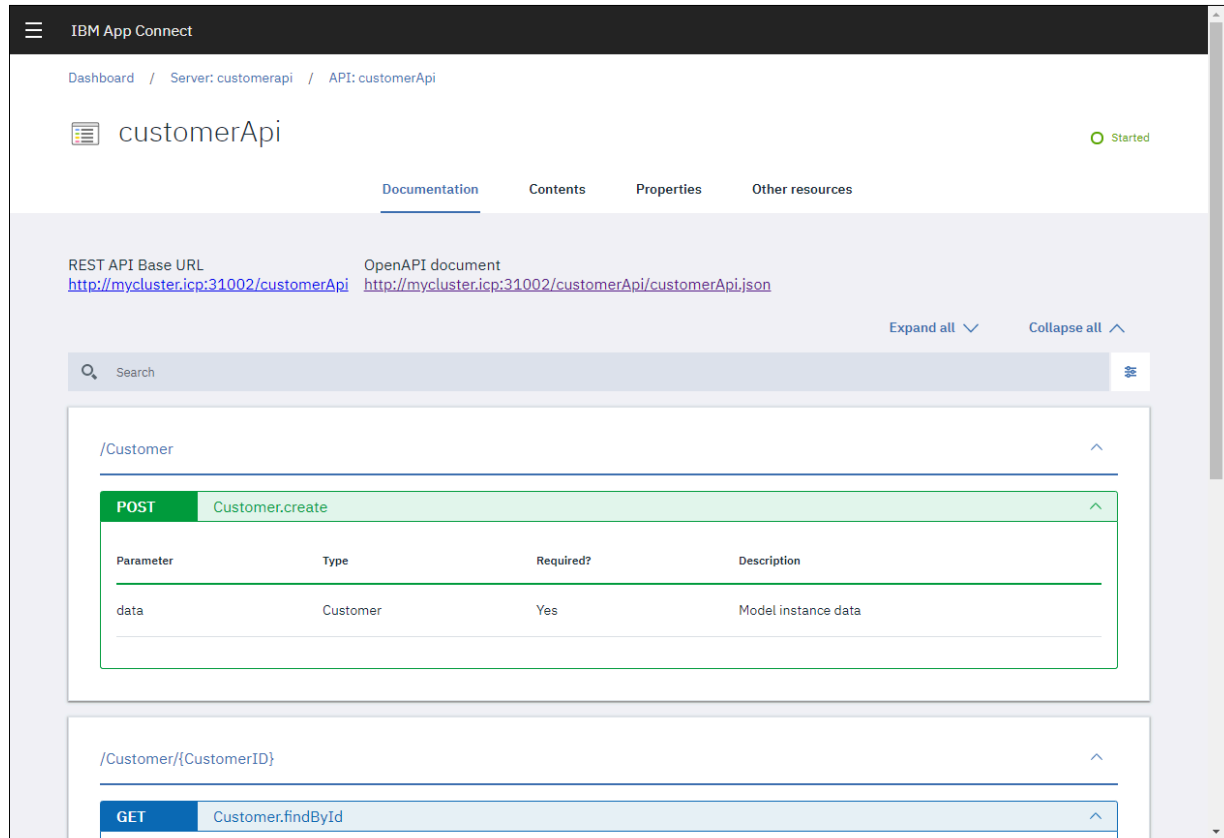
To invoke the API, complete the following steps:

1. From your App Connect Dashboard instance, click the integration server tile on the **Servers** page. The API tile is displayed, with a status of Started.



2. To view the API definition, click the **API tile**.

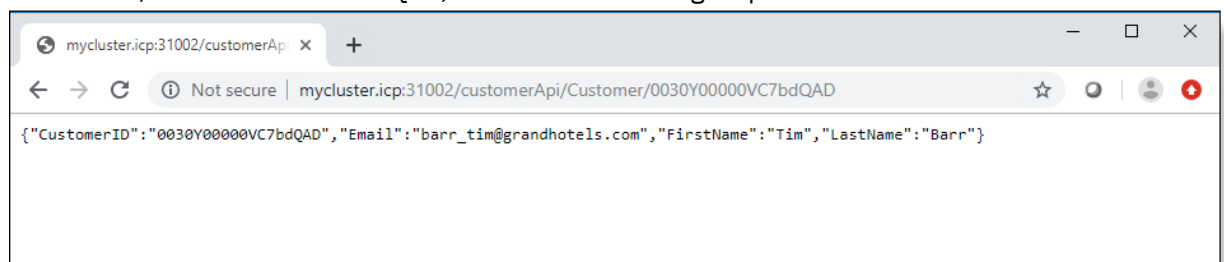
On the **Documentation** tab, the base URL for the API endpoint is shown. A URL is also provided for the OpenAPI document that describes the API. You can view or download the .json file. You can also expand the API operations to view their parameters.



3. To invoke the API, use standard utilities such as curl or Postman.

For GET operations, you can also make calls from a browser window in the usual way.

For example, in the preceding image, the resource path for the GET operation `/Customer/{CustomerID}` retrieves the details of a customer contact in Salesforce by using a unique contact ID. Therefore, an API call to retrieve the details for an existing contact with the ID 0030Y00000VC7bdQAD (specified as `http://mycluster.icp:31002/customerApi/Customer/0030Y00000VC7bdQAD`) returns the following response from the Salesforce instance.



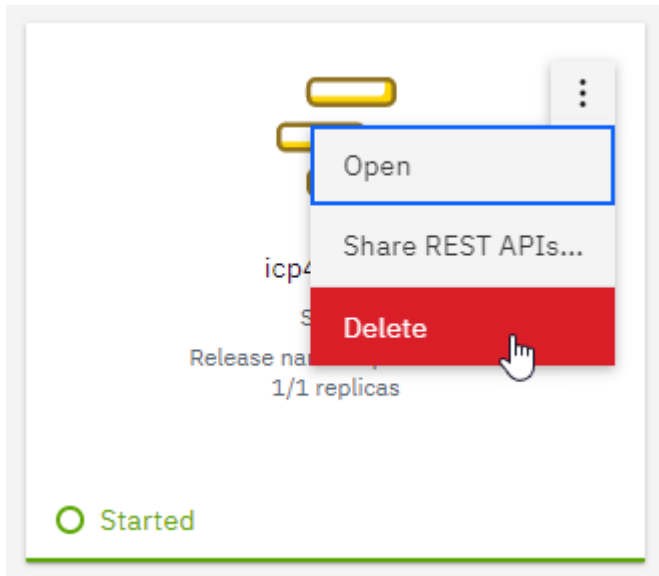
Deleting an integration server

If no longer needed, you can delete an integration server from IBM Cloud Pak for Integration or IBM App Connect Enterprise certified container by deleting the tile in the dashboard.

Procedure

To delete an integration server, complete the following steps.

1. From the **Servers** page of the App Connect Dashboard, locate the tile for the integration server that you want to delete.
2. Open the options menu on the tile and click **Delete**.



3. Optional: If a Kubernetes secret was used only by the deleted Helm release for the integration server, delete the secret, as described in [Deleting a secret](#).

Deleting a secret

After you delete a Helm release, you can use the `generateSecrets.sh` script to delete the Kubernetes secret that you created if that secret was used only by the deleted Helm release. (A secret can be shared by multiple Helm releases if you deployed the same BAR file multiple times to accommodate your workloads. You can also delete the secret that is generated for a Helm release after you update the account credentials for a local connector that is used in the deployed integration.

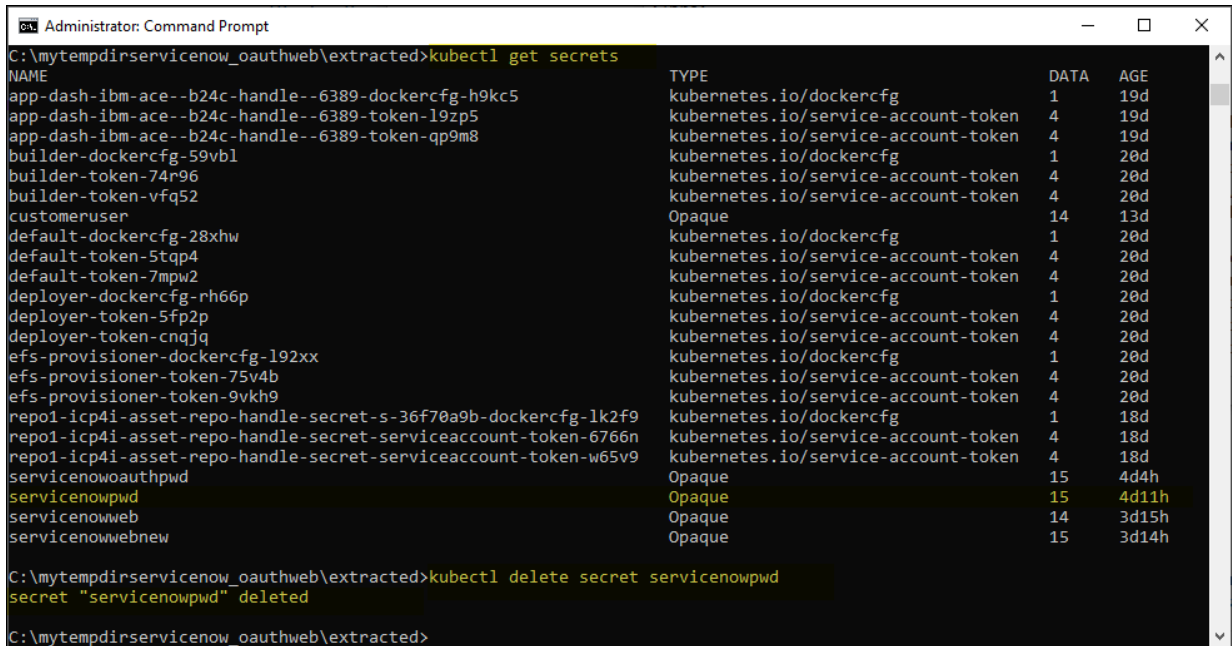
Procedure

To delete a secret, complete the following steps.

1. Ensure that you are logged in to your namespace in the cluster, as described in [Logging in to your cluster from the command-line interface](#).

2. Run the following command to delete the secret, where *secretName* is the secret that you want to delete.

```
kubectl delete secret secretName
```



```
Administrator: Command Prompt
C:\mytempdirservicenow_oauthweb\extracted>kubectl get secrets
NAME                                     TYPE                                     DATA  AGE
app-dash-ibm-ace--b24c-handle--6389-dockercfg-h9kc5  kubernetes.io/dockercfg                1      19d
app-dash-ibm-ace--b24c-handle--6389-token-19zp5     kubernetes.io/service-account-token    4      19d
app-dash-ibm-ace--b24c-handle--6389-token-qp9m8     kubernetes.io/service-account-token    4      19d
builder-dockercfg-59vbl                            kubernetes.io/dockercfg                1      20d
builder-token-74r96                                 kubernetes.io/service-account-token    4      20d
builder-token-vfq52                                 kubernetes.io/service-account-token    4      20d
customeruser                                         Opaque                                  14     13d
default-dockercfg-28xhw                             kubernetes.io/dockercfg                1      20d
default-token-5tqp4                                  kubernetes.io/service-account-token    4      20d
default-token-7mpw2                                  kubernetes.io/service-account-token    4      20d
deployer-dockercfg-rh66p                             kubernetes.io/dockercfg                1      20d
deployer-token-5fp2p                                 kubernetes.io/service-account-token    4      20d
deployer-token-cnjqj                                 kubernetes.io/service-account-token    4      20d
efs-provisioner-dockercfg-192xx                      kubernetes.io/dockercfg                1      20d
efs-provisioner-token-75v4b                          kubernetes.io/service-account-token    4      20d
efs-provisioner-token-9vkh9                          kubernetes.io/service-account-token    4      20d
repo1-icp4i-asset-repo-handle-secret-s-36f70a9b-dockercfg-1k2f9  kubernetes.io/dockercfg                1      18d
repo1-icp4i-asset-repo-handle-secret-serviceaccount-token-6766n  kubernetes.io/service-account-token    4      18d
repo1-icp4i-asset-repo-handle-secret-serviceaccount-token-w65v9  kubernetes.io/service-account-token    4      18d
servicenowauthpwd                                    Opaque                                  15     4d4h
servicenowpwd                                        Opaque                                  15     4d11h
servicenowweb                                        Opaque                                  14     3d15h
servicenowwebnew                                     Opaque                                  15     3d14h

C:\mytempdirservicenow_oauthweb\extracted>kubectl delete secret servicenowpwd
secret "servicenowpwd" deleted

C:\mytempdirservicenow_oauthweb\extracted>
```

Managing BAR files

You can use the BAR file management functions to maintain the set of BAR files within your App Connect Dashboard instance.

About this task

The backup and restore function enables you to collectively export all BAR files to an archive file and subsequently use that archive to restore all the BAR files to your dashboard instance. The backup and restore function is useful when deploying a new dashboard, and provides a full recovery mechanism.



As an alternative to creating an integration server from the Servers page, you can import a BAR file for storage, and then deploy it whenever you choose. You can also update an existing BAR file, or delete it if it's no longer needed.

Note:

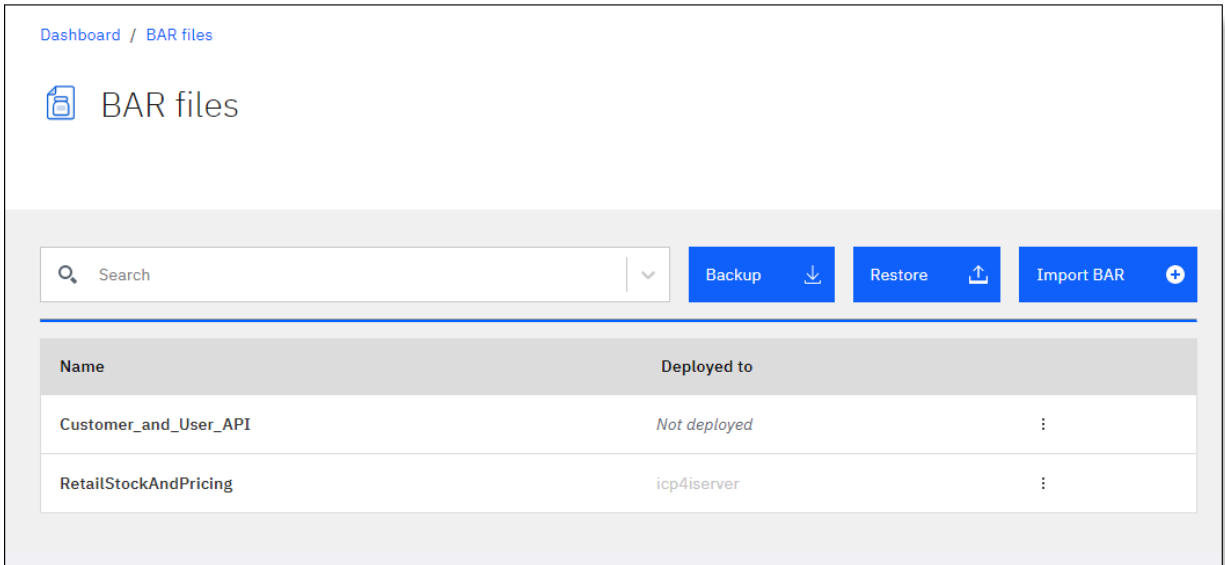
If you are importing or updating a BAR file that was exported from an App Connect Designer authoring environment, you must ensure that the BAR file fulfilled the conditions for export within the authoring environment. For more information, see [Exporting and importing API flows from an App Connect Designer instance in your cluster](#) and [Exporting an API flow to a BAR file from App Connect on IBM Cloud](#).

Procedure


1. From the App Connect Dashboard UI, complete the relevant step:

- **Applicable to IBM Cloud Pak for Integration only:** Click the settings icon  in the banner and then click **BAR files**.
- **Applicable to App Connect Enterprise certified container only:** From the App Connect menu , click **Manage > BAR files**.




The "BAR files" page opens, with a list of BAR files that have been deployed as integration servers.



Dashboard / BAR files

 BAR files

Search

Backup  Restore  Import BAR 

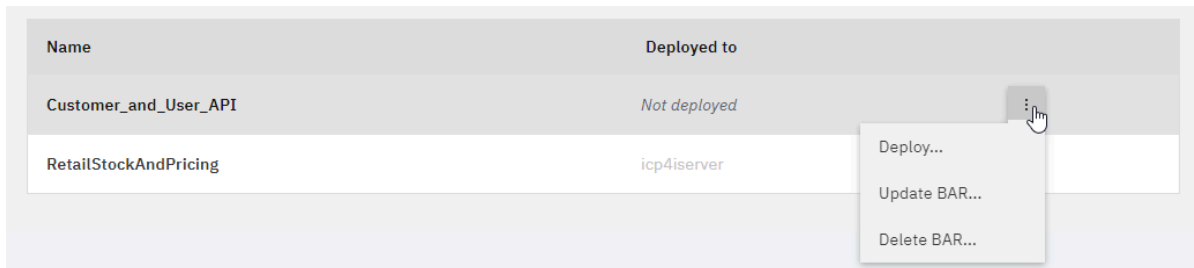
Name	Deployed to	
Customer_and_User_API	Not deployed	:
RetailStockAndPricing	icp4iserver	:

2. Complete any of the following tasks:

- To back up the collection of BAR files, click **Backup**. An archive (named ACE-BAR-Backup.tar.gz) of all the files is downloaded to your browser's configured location. You can subsequently use this archive to restore the files to the same or a different dashboard.
- To restore the BAR files that were backed up to an archive, click **Restore**. Add the ACE-BAR-Backup.tar.gz file to the "Restore" panel, and click **Restore**.

The files are added with a Not deployed status, and can be separately deployed as integration servers by using the **Deploy** option in the options menu for each file.

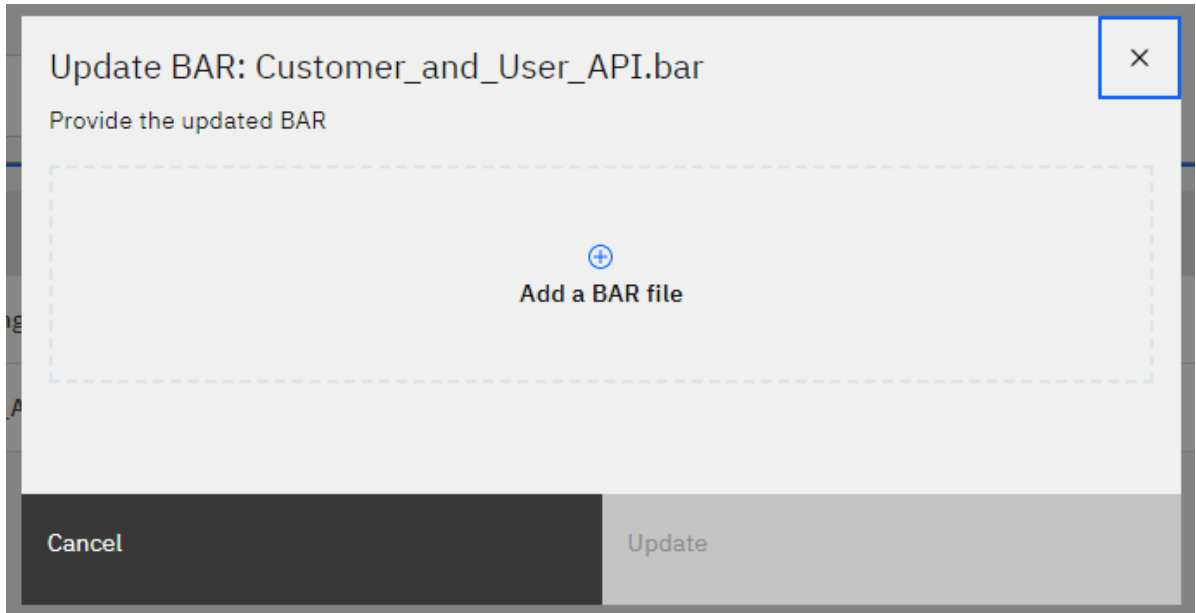
- To import a BAR file, click **Import BAR**. Add the file to the "Add a BAR file" panel, and click **Import**. The file is added with a Not deployed status, and can be deployed as an integration server by using the **Deploy** option in the options menu.
- To replace an existing BAR file with an updated file of the same name, open the options menu for the file and click **Update BAR**.



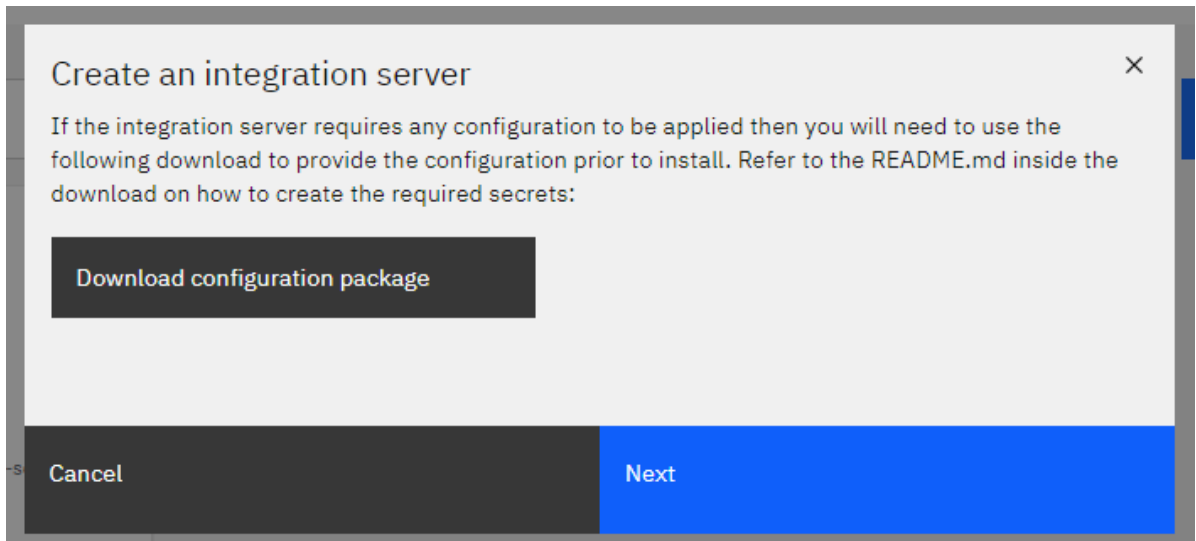
Then add the updated file to the Update BAR panel and click **Update**.

- If the existing version of the BAR file has a Not deployed status, you can subsequently deploy it by using the **Deploy** option in the options menu.
- If the existing version of the BAR file is already deployed, any integration servers that are using that file are restarted. If you are not using App Connect Enterprise with IBM MQ, your deployment is restarted on a rolling basis, because at least one replica is always running, to

avoid downtime. If you are using App Connect Enterprise with IBM MQ, there is only one replica so that is restarted with minimal downtime.



- To deploy (or redeploy) a BAR file, open the options menu for the file and then click **Deploy**. Follow the onscreen guidance to create an integration server. For more information, see [Cloud Pak for Integration only: Creating an integration server to run your BAR file resources](#) and [App Connect Enterprise certified container only: Creating an integration server to run your BAR file resources](#).



- To delete a BAR file, open the options menu for the file and click **Delete BAR**. No confirmation is required.

Troubleshooting and monitoring

Review this information to help resolve issues when you deploy an API that was authored in App Connect Designer to an integration server, or when you invoke the API.

About this task

Tips:

- If you need to run any kubectl commands, you must be logged in to your cluster, as described in [Logging in to your cluster from the command-line interface](#).

- You can view help for the commands from the command line by typing `kubectl help`, and view information about a specific command by typing `kubectl command -help`.

For advice about specific problems that can occur when you deploy or invoke an API, see the following sections.

- [“Checking the deployment status of an integration server” on page 111](#)
- [“Retrieving logs” on page 112](#)
- [“Resolving errors encountered during invocation” on page 113](#)

Checking the deployment status of an integration server

Procedure

To check the deployment status of an integration server, complete the following steps.

1. After you deploy an integration server to the App Connect Dashboard, you can check whether the deployment was successful by running the following command in the CLI.

```
kubectl get pods
```

The command returns a list of pods in the namespace, which are named in the format `helmRelease_name-chartName-generated_characters`.

2. Check the **STATUS** field for your pod for issues. A pod that is working correctly moves from a status of “Pending” to “Container creating” to “Running”.

The screenshot shows a Windows Command Prompt window titled "Administrator: Command Prompt". The user is in the directory `C:\mytempdir\config_customerApi\extracted`. The following commands and outputs are shown:

```
C:\mytempdir\config_customerApi\extracted>kubectl config use-context mycluster-context
Switched to context "mycluster-context".
OK

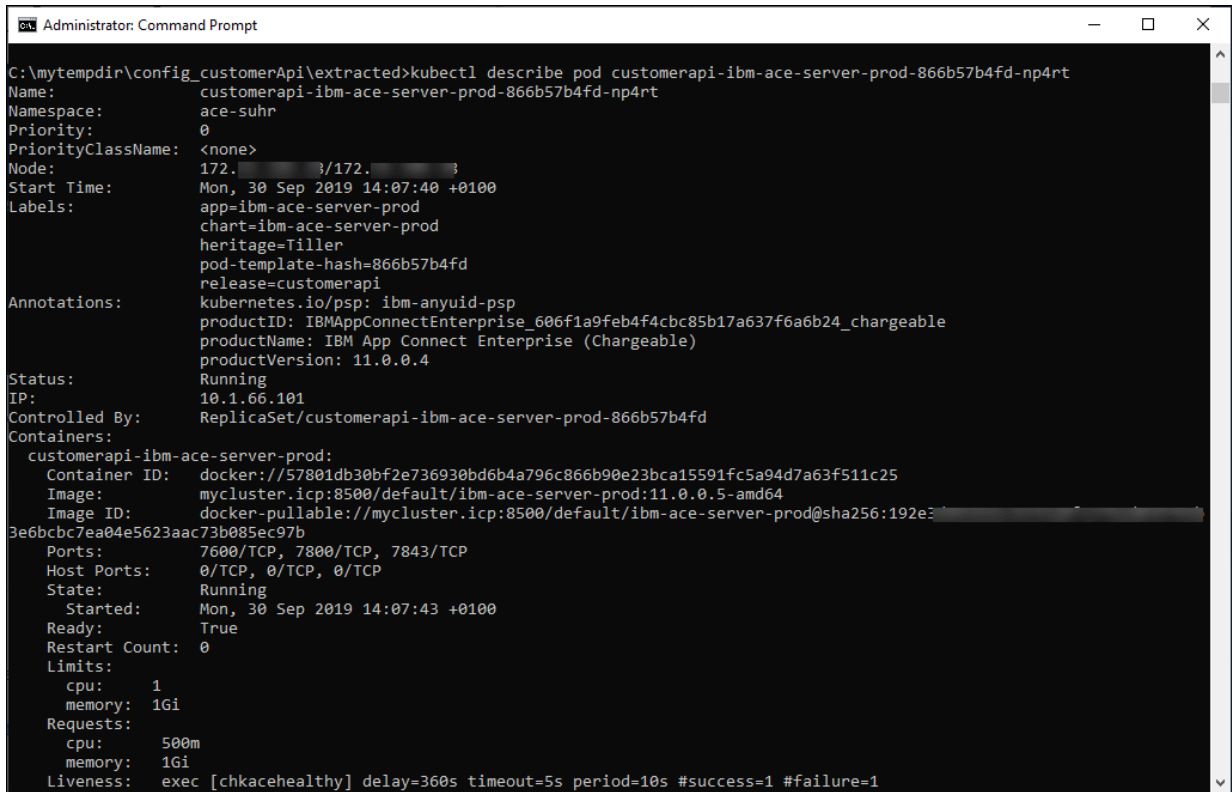
Configuring helm: C:\Users\...\.helm
OK

C:\mytempdir\config_customerApi\extracted>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
customerapi-ibm-ace-server-prod-866b57b4fd-np4rt    2/2     Running   0           10m
dashboard-suhr-ibm-ace-dashboard-prod-86d5866788-wwmn5  2/2     Running   6           52d
httpapi8-ibm-ace-server-prod-cd79d48bd-9mdtn        2/2     Running   0           23d
salesforce-flow-ibm-ace-server-prod-5f7b8757fd-trfnx  2/2     Running   0           24d
sf-phil-ibm-ace-server-prod-794d749d96-1tf8s       2/2     Running   0           6d5h

C:\mytempdir\config_customerApi\extracted>kubectl get pods
```

3. A failed deployment might be due to an issue with the secret that you specified or because you ran out of cluster resources. To get more detailed information about that pod, run the following command:

```
kubectl describe pod pod_name
```



```
Administrator: Command Prompt
C:\mytempdir\config_customerApi\extracted>kubectl describe pod customerapi-ibm-ace-server-prod-866b57b4fd-np4rt
Name: customerapi-ibm-ace-server-prod-866b57b4fd-np4rt
Namespace: ace-suhr
Priority: 0
PriorityClassName: <none>
Node: 172.17.0.1/172.17.0.1
Start Time: Mon, 30 Sep 2019 14:07:40 +0100
Labels: app=ibm-ace-server-prod
chart=ibm-ace-server-prod
heritage=Tiller
pod-template-hash=866b57b4fd
release=customerapi
Annotations: kubernetes.io/psp: ibm-anyuid-ppsp
productID: IBMAppConnectEnterprise_606f1a9feb4f4cbc85b17a637f6a6b24_chargeable
productName: IBM App Connect Enterprise (Chargeable)
productVersion: 11.0.0.4
Status: Running
IP: 10.1.66.101
Controlled By: ReplicaSet/customerapi-ibm-ace-server-prod-866b57b4fd
Containers:
  customerapi-ibm-ace-server-prod:
    Container ID: docker://57801db30bf2e736930bd6b4a796c866b90e23bca15591fc5a94d7a63f511c25
    Image: mycluster.icp:8500/default/ibm-ace-server-prod:11.0.0.5-amd64
    Image ID: docker-pullable://mycluster.icp:8500/default/ibm-ace-server-prod@sha256:192e33e6bcbc7ea04e5623aac73b085ec97b
    Ports: 7600/TCP, 7800/TCP, 7843/TCP
    Host Ports: 0/TCP, 0/TCP, 0/TCP
    State: Running
      Started: Mon, 30 Sep 2019 14:07:43 +0100
    Ready: True
    Restart Count: 0
    Limits:
      cpu: 1
      memory: 1Gi
    Requests:
      cpu: 500m
      memory: 1Gi
    Liveness: exec [chkacehealthy] delay=360s timeout=5s period=10s #success=1 #failure=1
```

Retrieving logs

About this task

You can obtain the logs from your cluster by running kubectl commands from the command line.

Retrieving user logs

1. Retrieve the list of pods in the namespace by running the following command:

```
kubectl get pods
```

The output is in a similar format to the following example.

NAME	READY	STATUS	RESTARTS	AGE
<i>helmRelease_name-2-chartName-generatedCharacter</i>	2/2	Running	0	16h

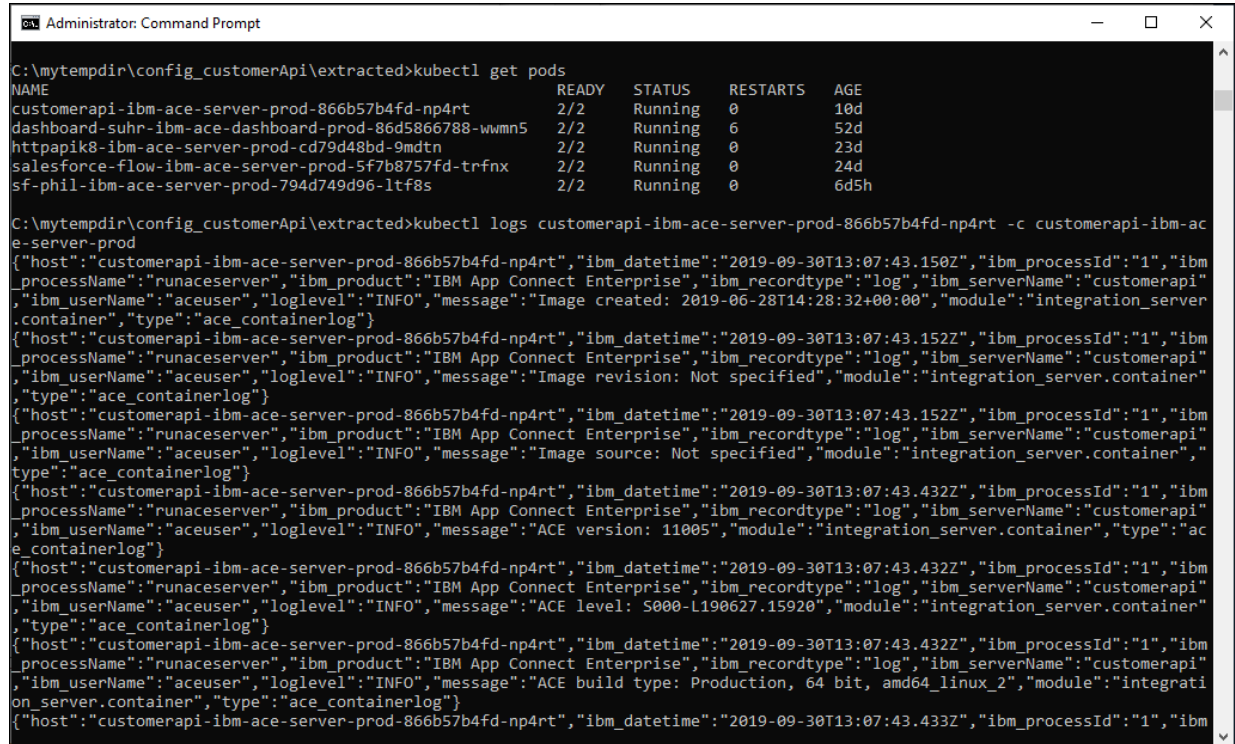
2. Get the App Connect Enterprise user logs for your pod by running the following command.

```
kubectl logs pod_name -c container_name
```

For example:

```
kubectl logs helmRelease_name-chartName-generated_characters -c helmRelease_name-chartName
```

The logs are written to a standard out stream in the terminal window.



```
Administrator: Command Prompt
C:\mytempdir\config_customerApi\extracted>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
customerapi-ibm-ace-server-prod-866b57b4fd-np4rt    2/2     Running   0           10d
dashboard-suhr-ibm-ace-dashboard-prod-86d5866788-wwmn5  2/2     Running   6           52d
httpapi8-ibm-ace-server-prod-cd79d48bd-9mdtn        2/2     Running   0           23d
salesforce-flow-ibm-ace-server-prod-5f7b8757fd-trfnx  2/2     Running   0           24d
sf-phil-ibm-ace-server-prod-794d749d96-1tf8s       2/2     Running   0           6d5h

C:\mytempdir\config_customerApi\extracted>kubectl logs customerapi-ibm-ace-server-prod-866b57b4fd-np4rt -c customerapi-ibm-ace-server-prod
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.150Z","ibm_processId":"1","ibm_processName":"runaceserver","ibm_product":"IBM App Connect Enterprise","ibm_recordtype":"log","ibm_serverName":"customerapi","ibm_userName":"aceuser","loglevel":"INFO","message":"Image created: 2019-06-28T14:28:32+00:00","module":"integration_server.container","type":"ace_containerlog"}
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.152Z","ibm_processId":"1","ibm_processName":"runaceserver","ibm_product":"IBM App Connect Enterprise","ibm_recordtype":"log","ibm_serverName":"customerapi","ibm_userName":"aceuser","loglevel":"INFO","message":"Image revision: Not specified","module":"integration_server.container","type":"ace_containerlog"}
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.152Z","ibm_processId":"1","ibm_processName":"runaceserver","ibm_product":"IBM App Connect Enterprise","ibm_recordtype":"log","ibm_serverName":"customerapi","ibm_userName":"aceuser","loglevel":"INFO","message":"Image source: Not specified","module":"integration_server.container","type":"ace_containerlog"}
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.432Z","ibm_processId":"1","ibm_processName":"runaceserver","ibm_product":"IBM App Connect Enterprise","ibm_recordtype":"log","ibm_serverName":"customerapi","ibm_userName":"aceuser","loglevel":"INFO","message":"ACE version: 11005","module":"integration_server.container","type":"ace_containerlog"}
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.432Z","ibm_processId":"1","ibm_processName":"runaceserver","ibm_product":"IBM App Connect Enterprise","ibm_recordtype":"log","ibm_serverName":"customerapi","ibm_userName":"aceuser","loglevel":"INFO","message":"ACE level: S000-L190627.15920","module":"integration_server.container","type":"ace_containerlog"}
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.432Z","ibm_processId":"1","ibm_processName":"runaceserver","ibm_product":"IBM App Connect Enterprise","ibm_recordtype":"log","ibm_serverName":"customerapi","ibm_userName":"aceuser","loglevel":"INFO","message":"ACE build type: Production, 64 bit, amd64_linux_2","module":"integration_server.container","type":"ace_containerlog"}
{"host":"customerapi-ibm-ace-server-prod-866b57b4fd-np4rt","ibm_datetime":"2019-09-30T13:07:43.433Z","ibm_processId":"1","ibm
```

Retrieving operational logs

If you need to send operational logs to IBM Support to aid with troubleshooting, you can retrieve these logs by running the following command.

```
kubectl exec pod_name -c container_name -ti /bin/bash
```

Where *pod_name* is generally shown in the format *helmRelease_name-chartName-generated_characters*, and *container_name* is shown in the format *helmRelease_name-chartName*.

Logs are written to a file in the `ace-server/log` directory.

Resolving errors encountered during invocation

About this task

An API call returns 200, but contains no response details and an empty log

Check that the secret is set up correctly.

1. If unsure, obtain the name of the secret that was defined for the pod by running the `kubectl get pods` and `kubectl describe pod pod_name` commands.
2. Retrieve details about all secrets by running the following command:

```
kubectl get secrets
```

3. Locate the secret that you generated when you deployed the integration server, and ensure that it has a nonzero value in the Data column.

```
Administrator: Command Prompt
C:\mytempdir\config_customerApi\extracted>kubectl get secrets
NAME                                     TYPE      DATA  AGE
custapisecret                          Opaque    14     10d
customerapi-ibm-ace-server-prod         Opaque    2     10d
customerapi-ibm-ace-server-prod-serviceaccount-token-sgzhv  kubernetes.io/service-account-token  3     10d
dashboard-suhr-ibm-ace-dashboard-prod   Opaque    10     52d
dashboard-suhr-ibm-ace-dashboard-prod-oidc  Opaque    2     52d
dashboard-suhr-ibm-ace-dashboard-prod-serviceaccount-tokenrpfjk  kubernetes.io/service-account-token  3     52d
default-token-d28vc                    kubernetes.io/service-account-token  3     52d
httpapik8-ibm-ace-server-prod           Opaque    2     24d
httpapik8-ibm-ace-server-prod-serviceaccount-token-4hmhb  kubernetes.io/service-account-token  3     24d
httpapik8secret                         Opaque    1     29d
jrsecret                                Opaque    1     43d
salesforce-flow-ibm-ace-server-prod     Opaque    2     30d
salesforce-flow-ibm-ace-server-prod-serviceaccount-token-dr7xf  kubernetes.io/service-account-token  3     30d
salesforce-secret                       Opaque    16     30d
sf-phil                                 Opaque    16     6d6h
sf-phil-enable-trace                   Opaque    16     35d
sf-phil-ibm-ace-server-prod            Opaque    2     6d6h
sf-phil-ibm-ace-server-prod-serviceaccount-token-mtmqr  kubernetes.io/service-account-token  3     6d6h
testflow                                Opaque    16     42d
```

An API call returns an error, but no details

Ensure that the secret, which was specified for the Helm release, is set up correctly.

- Verify that the [IBM Cloud API key](#) (which was used to generate the secret) was created in the same IBM Cloud account as the App Connect instance that your deployed API needs to communicate with.
- Verify that the IBM Cloud API key was not revoked.
- Verify that the `setdbparms.txt` file, which was used to generate the secret, contains the correct values for the flow name and IBM Cloud API key.

Index

I

installing the IBM App Connect Enterprise certified
container
Red Hat OpenShift [10](#)

R

Red Hat OpenShift
installing the IBM App Connect Enterprise certified
container [10](#)

