Red Hat OpenShift Serverless 1.31

Installing Serverless

Installing the Serverless Operator, Knative CLI, Knative Serving, and Knative Eventing

Last Updated: 2023-12-14

Red Hat OpenShift Serverless 1.31 Installing Serverless

Installing the Serverless Operator, Knative CLI, Knative Serving, and Knative Eventing

Legal Notice

Abstract

This document contains information about installing the Serverless Operator, Knative CLI, Knative Serving, and Knative Eventing. It also provides details about configuring Knative for Apache Kafka and about configuring Serverless Functions.

Table of Contents

CHAPTER 1. PREPARING TO INSTALL OPENSHIFT SERVERLESS

Read the following information about supported configurations and prerequisites before you install OpenShift Serverless.

For OpenShift Container Platform:

- OpenShift Serverless is supported for installation in a restricted network environment.
- OpenShift Serverless currently cannot be used in a multi-tenant configuration on a single cluster.

1.1. SUPPORTED CONFIGURATIONS

The set of supported features, configurations, and integrations for OpenShift Serverless, current and past versions, are available at the Supported Configurations page.

1.2. SCALABILITY AND PERFORMANCE ON OPENSHIFT CONTAINER PLATFORM

OpenShift Serverless has been tested with a configuration of 3 main nodes and 3 worker nodes, each of which has 64 CPUs, 457 GB of memory, and 394 GB of storage each.

The maximum number of Knative services that can be created using this configuration is 3,000. This corresponds to the OpenShift Container Platform Kubernetes services limit of 10,000 , since 1 Knative service creates 3 Kubernetes services.

The average scale from zero response time was approximately 3.4 seconds, with a maximum response time of 8 seconds, and a 99.9th percentile of 4.5 seconds for a simple Quarkus application. These times might vary depending on the application and the runtime of the application.

NOTE

The following section on defining cluster size requirements applies to these distributions:

- OpenShift Container Platform
- OpenShift Dedicated
- Red Hat OpenShift Service on AWS

1.3. DEFINING CLUSTER SIZE REQUIREMENTS

To install and use OpenShift Serverless, the cluster must be sized correctly.

NOTE

The following requirements relate only to the pool of worker machines of the cluster. Control plane nodes are not used for general scheduling and are omitted from the requirements.

The minimum requirement to use OpenShift Serverless is a cluster with 10 CPUs and 40GB memory. By default, each pod requests ~400m of CPU, so the minimum requirements are based on this value.

The total size requirements to run OpenShift Serverless are dependent on the components that are installed and the applications that are deployed, and might vary depending on your deployment.

1.4. SCALING YOUR CLUSTER USING COMPUTE MACHINE SETS ON OPENSHIFT CONTAINER PLATFORM

You can use the OpenShift Container Platform **MachineSet** API to manually scale your cluster up to the desired size. The minimum requirements usually mean that you must scale up one of the default compute machine sets by two additional machines. See Manually scaling a compute machine set .

1.4.1. Additional requirements for advanced use-cases

For more advanced use-cases such as logging or metering on OpenShift Container Platform, you must deploy more resources. Recommended requirements for such use-cases are 24 CPUs and 96GB of memory.

If you have high availability (HA) enabled on your cluster, this requires between 0.5 - 1.5 cores and between 200MB - 2GB of memory for each replica of the Knative Serving control plane. HA is enabled for some Knative Serving components by default. You can disable HA by following the documentation on "Configuring high availability replicas".

1.5. ADDITIONAL RESOURCES IN OPENSHIFT CONTAINER PLATFORM DOCUMENTATION

- Using Operator Lifecycle Manager on restricted networks
- Understanding OperatorHub
- Cluster capabilities

CHAPTER 2. INSTALLING THE OPENSHIFT SERVERLESS OPERATOR

Installing the OpenShift Serverless Operator enables you to install and use Knative Serving, Knative Eventing, and the Knative broker for Apache Kafka on a OpenShift Container Platform cluster. The OpenShift Serverless Operator manages Knative custom resource definitions (CRDs) for your cluster and enables you to configure them without directly modifying individual config maps for each component.

2.1. INSTALLING THE OPENSHIFT SERVERLESS OPERATOR FROM THE WEB CONSOLE

You can install the OpenShift Serverless Operator from the OperatorHub by using the OpenShift Container Platform web console. Installing this Operator enables you to install and use Knative components.

Prerequisites

- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- For OpenShift Container Platform, your cluster has the Marketplace capability enabled or the Red Hat Operator catalog source configured manually.
- You have logged in to the web console.

Procedure

1. In the web console, navigate to the **Operators → OperatorHub** page.
2. Scroll, or type the keyword **Serverless** into the **Filter by keyword** box to find the OpenShift Serverless Operator.
3. Review the information about the Operator and click **Install**.
4. On the **Install Operator** page:
    a. The **Installation Mode** is **All namespaces on the cluster (default)** This mode installs the Operator in the default **openshift-serverless** namespace to watch and be made available to all namespaces in the cluster.
    b. The **Installed Namespace** is **openshift-serverless**.
    c. Select an **Update Channel**.
        - The **stable** channel enables installation of the latest stable release of the OpenShift Serverless Operator. The **stable** channel is the default.
        - To install another version, specify the corresponding **stable-x.y** channel, for example **stable-1.29**.
    d. Select the **stable** channel as the **Update Channel**. The **stable** channel will enable installation of the latest stable release of the OpenShift Serverless Operator.

e. Select **Automatic** or **Manual** approval strategy.

5. Click **Install** to make the Operator available to the selected namespaces on this OpenShift Container Platform cluster.

6. From the **Catalog → Operator Management** page, you can monitor the OpenShift Serverless Operator subscription's installation and upgrade progress.

   a. If you selected a **Manual** approval strategy, the subscription's upgrade status will remain **Upgrading** until you review and approve its install plan. After approving on the **Install Plan** page, the subscription upgrade status moves to **Up to date**.

   b. If you selected an **Automatic** approval strategy, the upgrade status should resolve to **Up to date** without intervention.

Verification

After the Subscription's upgrade status is **Up to date**, select **Catalog → Installed Operators** to verify that the OpenShift Serverless Operator eventually shows up and its **Status** ultimately resolves to **InstallSucceeded** in the relevant namespace.

If it does not:

1. Switch to the **Catalog → Operator Management** page and inspect the **Operator Subscriptions** and **Install Plans** tabs for any failure or errors under **Status**.

2. Check the logs in any pods in the **openshift-serverless** project on the **Workloads → Pods** page that are reporting issues to troubleshoot further.

IMPORTANT

If you want to use Red Hat OpenShift distributed tracing with OpenShift Serverless , you must install and configure Red Hat OpenShift distributed tracing before you install Knative Serving or Knative Eventing.

2.2. INSTALLING THE OPENSHIFT SERVERLESS OPERATOR FROM THE CLI

You can install the OpenShift Serverless Operator from the OperatorHub by using the CLI. Installing this Operator enables you to install and use Knative components.

Prerequisites

- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- For OpenShift Container Platform, your cluster has the Marketplace capability enabled or the Red Hat Operator catalog source configured manually.
- You have logged in to the cluster.

Procedure

1. Create a YAML file containing **Namespace**, **OperatorGroup**, and **Subscription** objects to subscribe a namespace to the OpenShift Serverless Operator. For example, create the file **serverless-subscription.yaml** with the following content:

   ### Example subscription

   ```
   ---
   apiVersion: v1
   kind: Namespace
   metadata:
     name: openshift-serverless
   ---
   apiVersion: operators.coreos.com/v1
   kind: OperatorGroup
   metadata:
     name: serverless-operators
     namespace: openshift-serverless
   spec: {}
   ---
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     name: serverless-operator
     namespace: openshift-serverless
   spec:
     channel: stable (1)
     name: serverless-operator (2)
     source: redhat-operators (3)
     sourceNamespace: openshift-marketplace (4)
   ```

   (1) The channel name of the Operator. The **stable** channel enables installation of the most recent stable version of the OpenShift Serverless Operator. To install another version, specify the corresponding **stable-x.y** channel, for example **stable-1.29**.

   (2) The name of the Operator to subscribe to. For the OpenShift Serverless Operator, this is always **serverless-operator**.

   (3) The name of the CatalogSource that provides the Operator. Use **redhat-operators** for the default OperatorHub catalog sources.

   (4) The namespace of the CatalogSource. Use **openshift-marketplace** for the default OperatorHub catalog sources.

2. Create the **Subscription** object:

   ```
   $ oc apply -f serverless-subscription.yaml
   ```

Verification

Check that the cluster service version (CSV) has reached the **Succeeded** phase:

Example command

```
$ oc get csv
```

Example output

```
NAME                              DISPLAY                       VERSION   REPLACES                  PHASE
serverless-operator.v1.25.0   Red Hat OpenShift Serverless   1.25.0    serverless-operator.v1.24.0
Succeeded
```

IMPORTANT

If you want to use Red Hat OpenShift distributed tracing with OpenShift Serverless , you must install and configure Red Hat OpenShift distributed tracing before you install Knative Serving or Knative Eventing.

2.3. GLOBAL CONFIGURATION

The OpenShift Serverless Operator manages the global configuration of a Knative installation, including propagating values from the **KnativeServing** and **KnativeEventing** custom resources to system config maps. Any updates to config maps which are applied manually are overwritten by the Operator. However, modifying the Knative custom resources allows you to set values for these config maps.

Knative has multiple config maps that are named with the prefix **config-**. All Knative config maps are created in the same namespace as the custom resource that they apply to. For example, if the **KnativeServing** custom resource is created in the **knative-serving** namespace, all Knative Serving config maps are also created in this namespace.

The **spec.config** in the Knative custom resources have one **<name>** entry for each config map, named **config-<name>**, with a value which is be used for the config map **data**.

2.4. ADDITIONAL RESOURCES FOR OPENSHIFT CONTAINER PLATFORM

- Managing resources from custom resource definitions
- Understanding persistent storage
- Configuring a custom PKI

2.5. NEXT STEPS

- After the OpenShift Serverless Operator is installed, you can install Knative Serving or install Knative Eventing.

CHAPTER 3. INSTALLING THE KNATIVE CLI

The Knative (**kn**) CLI does not have its own login mechanism. To log in to the cluster, you must install the OpenShift CLI (**oc**) and use the **oc login** command. Installation options for the CLIs may vary depending on your operating system.

For more information on installing the OpenShift CLI (**oc**) for your operating system and logging in with **oc**, see the OpenShift CLI getting started documentation.

OpenShift Serverless cannot be installed using the Knative (**kn**) CLI. A cluster administrator must install the OpenShift Serverless Operator and set up the Knative components, as described in the Installing the OpenShift Serverless Operator documentation.

IMPORTANT

If you try to use an older version of the Knative (**kn**) CLI with a newer OpenShift Serverless release, the API is not found and an error occurs.

For example, if you use the 1.23.0 release of the Knative (**kn**) CLI, which uses version 1.2, with the 1.24.0 OpenShift Serverless release, which uses the 1.3 versions of the Knative Serving and Knative Eventing APIs, the CLI does not work because it continues to look for the outdated 1.2 API versions.

Ensure that you are using the latest Knative (**kn**) CLI version for your OpenShift Serverless release to avoid issues.

3.1. INSTALLING THE KNATIVE CLI USING THE OPENSHIFT CONTAINER PLATFORM WEB CONSOLE

Using the OpenShift Container Platform web console provides a streamlined and intuitive user interface to install the Knative (**kn**) CLI. After the OpenShift Serverless Operator is installed, you will see a link to download the Knative (**kn**) CLI for Linux (amd64, s390x, ppc64le), macOS, or Windows from the **Command Line Tools** page in the OpenShift Container Platform web console.

Prerequisites

- You have logged in to the OpenShift Container Platform web console.
- The OpenShift Serverless Operator and Knative Serving are installed on your OpenShift Container Platform cluster.

IMPORTANT

If **libc** is not available, you might see the following error when you run CLI commands:

```
$ kn: No such file or directory
```

- If you want to use the verification steps for this procedure, you must install the OpenShift (**oc**) CLI.

Procedure

1. Download the Knative (**kn**) CLI from the **Command Line Tools** page. You can access the **Command Line Tools** page by clicking the  icon in the top right corner of the web console and selecting **Command Line Tools** in the list.

2. Unpack the archive:

```
$ tar -xf <file>
```

3. Move the **kn** binary to a directory on your **PATH**.

4. To check your **PATH**, run:

```
$ echo $PATH
```

Verification

- Run the following commands to check that the correct Knative CLI resources and route have been created:

```
$ oc get ConsoleCLIDownload
```

Example output

```
NAME                DISPLAY NAME                                        AGE
kn                  kn - OpenShift Serverless Command Line Interface (CLI)   2022-09-
20T08:41:18Z
oc-cli-downloads    oc - OpenShift Command Line Interface (CLI)              2022-09-
20T08:00:20Z
```

```
$ oc get route -n openshift-serverless
```

Example output

```
NAME   HOST/PORT                              PATH   SERVICES                  PORT
TERMINATION     WILDCARD
kn     kn-openshift-serverless.apps.example.com          knative-openshift-metrics-3   http-cli
edge/Redirect   None
```

3.2. INSTALLING THE KNATIVE CLI FOR LINUX BY USING AN RPM PACKAGE MANAGER

For Red Hat Enterprise Linux (RHEL), you can install the Knative (**kn**) CLI as an RPM by using a package manager, such as **yum** or **dnf**. This allows the Knative CLI version to be automatically managed by the system. For example, using a command like **dnf upgrade** upgrades all packages, including **kn**, if a new version is available.

Prerequisites

- You have an active OpenShift Container Platform subscription on your Red Hat account.

Procedure

1. Register with Red Hat Subscription Manager:

```
# subscription-manager register
```

2. Pull the latest subscription data:

```
# subscription-manager refresh
```

3. Attach the subscription to the registered system:

```
# subscription-manager attach --pool=<pool_id> 1
```

1 Pool ID for an active OpenShift Container Platform subscription

4. Enable the repositories required by the Knative (**kn**) CLI:
   - Linux (x86_64, amd64)

```
# subscription-manager repos --enable="openshift-serverless-1-for-rhel-8-x86_64-rpms"
```

   - Linux on IBM zSystems and IBM® LinuxONE (s390x)

```
# subscription-manager repos --enable="openshift-serverless-1-for-rhel-8-s390x-rpms"
```

   - Linux on IBM Power (ppc64le)

```
# subscription-manager repos --enable="openshift-serverless-1-for-rhel-8-ppc64le-rpms"
```

5. Install the Knative (**kn**) CLI as an RPM by using a package manager:

Example yum command

```
# yum install openshift-serverless-clients
```

3.3. LOCKING VERSION FOR THE KNATIVE CLI INSTALLED WITH RPM PACKAGE MANAGER

You might require to use Knative (**kn**) CLI version that is not the most recent, but is in Maintenance Phase. You can achieve that by locking the version of **kn**, which prevents it from being upgraded.

Procedure

1. Install the **versionlock** plugin for the DNF package manager:

```
# dnf install 'dnf-command(versionlock)'
```

2. Lock the version of **kn** by running:

```
# dnf versionlock add --raw 'openshift-serverless-clients-1.7.*'
```

This command locks **kn** to be the version based on Knative 1.7, which is in Maintenance Phase at the time of release of OpenShift Serverless 1.29.

3.4. UPGRADING THE KNATIVE CLI WITH LOCKED VERSION

You can manually upgrade the Knative (**kn**) CLI version that has previously been version-locked.

Procedure

1. Check whether upgraded packages are available:

```
# dnf search --showduplicates openshift-serverless-clients
```

2. Remove the version lock of **kn**:

```
# dnf versionlock delete openshift-serverless-clients
```

1. Lock **kn** to the new version:

```
# dnf versionlock add --raw 'openshift-serverless-clients-1.8.*'
```

+ This example uses the **kn** version that is based on Knative 1.8.

1. Upgrade to the new available version:

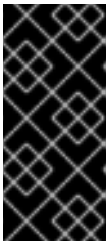
```
# dnf install --upgrade openshift-serverless-clients
```

3.5. INSTALLING THE KNATIVE CLI FOR LINUX

If you are using a Linux distribution that does not have RPM or another package manager installed, you can install the Knative (**kn**) CLI as a binary file. To do this, you must download and unpack a **tar.gz** archive and add the binary to a directory on your **PATH**.

Prerequisites

- If you are not using RHEL or Fedora, ensure that **libc** is installed in a directory on your library path.



IMPORTANT

If **libc** is not available, you might see the following error when you run CLI commands:

```
$ kn: No such file or directory
```

Procedure

1. Download the relevant Knative (**kn**) CLI **tar.gz** archive:

- Linux (x86_64, amd64)
- Linux on IBM zSystems and IBM® LinuxONE (s390x)

- Linux on IBM Power (ppc64le)

You can also download any version of **kn** by navigating to that version's corresponding directory in the Serverless client download mirror .

2. Unpack the archive:

```
$ tar -xf <filename>
```

3. Move the **kn** binary to a directory on your **PATH**.

4. To check your **PATH**, run:

```
$ echo $PATH
```

3.6. INSTALLING THE KNATIVE CLI FOR MACOS

If you are using macOS, you can install the Knative (**kn**) CLI as a binary file. To do this, you must download and unpack a **tar.gz** archive and add the binary to a directory on your **PATH**.

Procedure

1. Download the Knative (**kn**) CLI **tar.gz** archive.
You can also download any version of **kn** by navigating to that version's corresponding directory in the Serverless client download mirror .

2. Unpack and extract the archive.

3. Move the **kn** binary to a directory on your **PATH**.

4. To check your **PATH**, open a terminal window and run:

```
$ echo $PATH
```

3.7. INSTALLING THE KNATIVE CLI FOR WINDOWS

If you are using Windows, you can install the Knative (**kn**) CLI as a binary file. To do this, you must download and unpack a ZIP archive and add the binary to a directory on your **PATH**.

Procedure

1. Download the Knative (**kn**) CLI ZIP archive.
You can also download any version of **kn** by navigating to that version's corresponding directory in the Serverless client download mirror .

2. Extract the archive with a ZIP program.

3. Move the **kn** binary to a directory on your **PATH**.

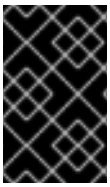
4. To check your **PATH**, open the command prompt and run the command:

```
C:\> path
```

CHAPTER 4. INSTALLING KNATIVE SERVING

Installing Knative Serving allows you to create Knative services and functions on your cluster. It also allows you to use additional functionality such as autoscaling and networking options for your applications.

After you install the OpenShift Serverless Operator, you can install Knative Serving by using the default settings, or configure more advanced settings in the **KnativeServing** custom resource (CR). For more information about configuration options for the **KnativeServing** CR, see Global configuration.



IMPORTANT

If you want to use Red Hat OpenShift distributed tracing with OpenShift Serverless , you must install and configure Red Hat OpenShift distributed tracing before you install Knative Serving.

4.1. INSTALLING KNATIVE SERVING BY USING THE WEB CONSOLE

After you install the OpenShift Serverless Operator, install Knative Serving by using the OpenShift Container Platform web console. You can install Knative Serving by using the default settings or configure more advanced settings in the **KnativeServing** custom resource (CR).

Prerequisites

- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- You have logged in to the OpenShift Container Platform web console.
- You have installed the OpenShift Serverless Operator.

Procedure

1. In the **Administrator** perspective of the OpenShift Container Platform web console, navigate to **Operators → Installed Operators**.
2. Check that the **Project** dropdown at the top of the page is set to **Project: knative-serving**.
3. Click **Knative Serving** in the list of **Provided APIs** for the OpenShift Serverless Operator to go to the **Knative Serving** tab.
4. Click **Create Knative Serving**
5. In the **Create Knative Serving** page, you can install Knative Serving using the default settings by clicking **Create**.
   You can also modify settings for the Knative Serving installation by editing the **KnativeServing** object using either the form provided, or by editing the YAML.
   - Using the form is recommended for simpler configurations that do not require full control of **KnativeServing** object creation.
   - Editing the YAML is recommended for more complex configurations that require full control of **KnativeServing** object creation. You can access the YAML by clicking the **edit YAML** link in the top right of the **Create Knative Serving** page.

After you complete the form, or have finished modifying the YAML, click **Create**.

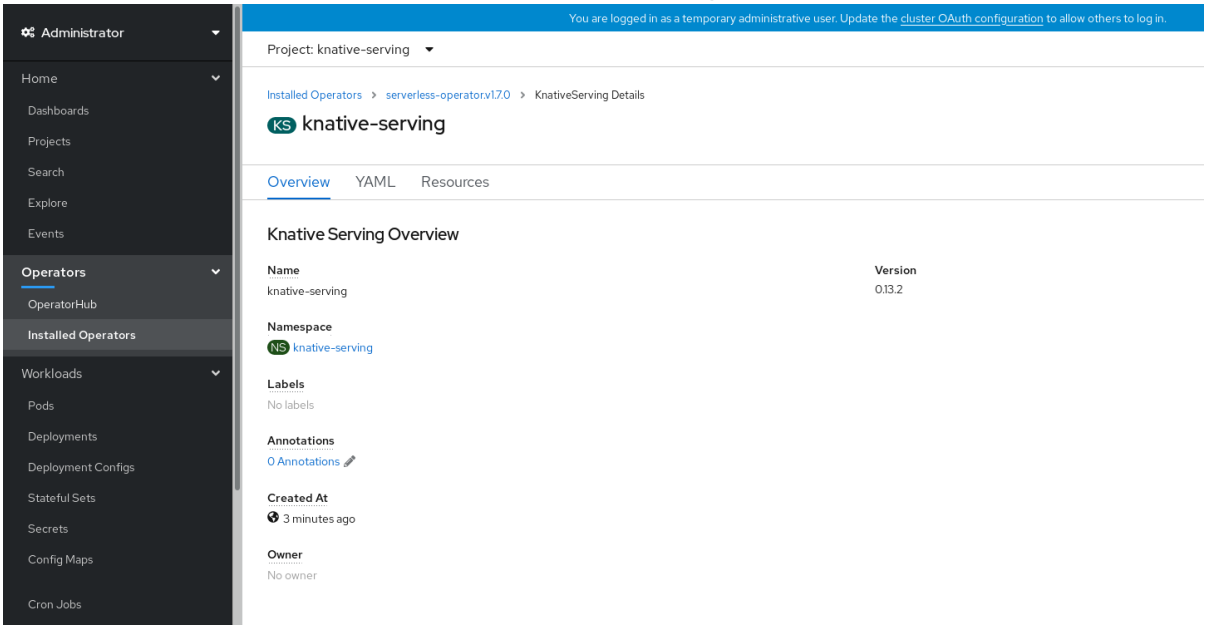
NOTE

For more information about configuration options for the KnativeServing custom resource definition, see the documentation on *Advanced installation configuration options*.

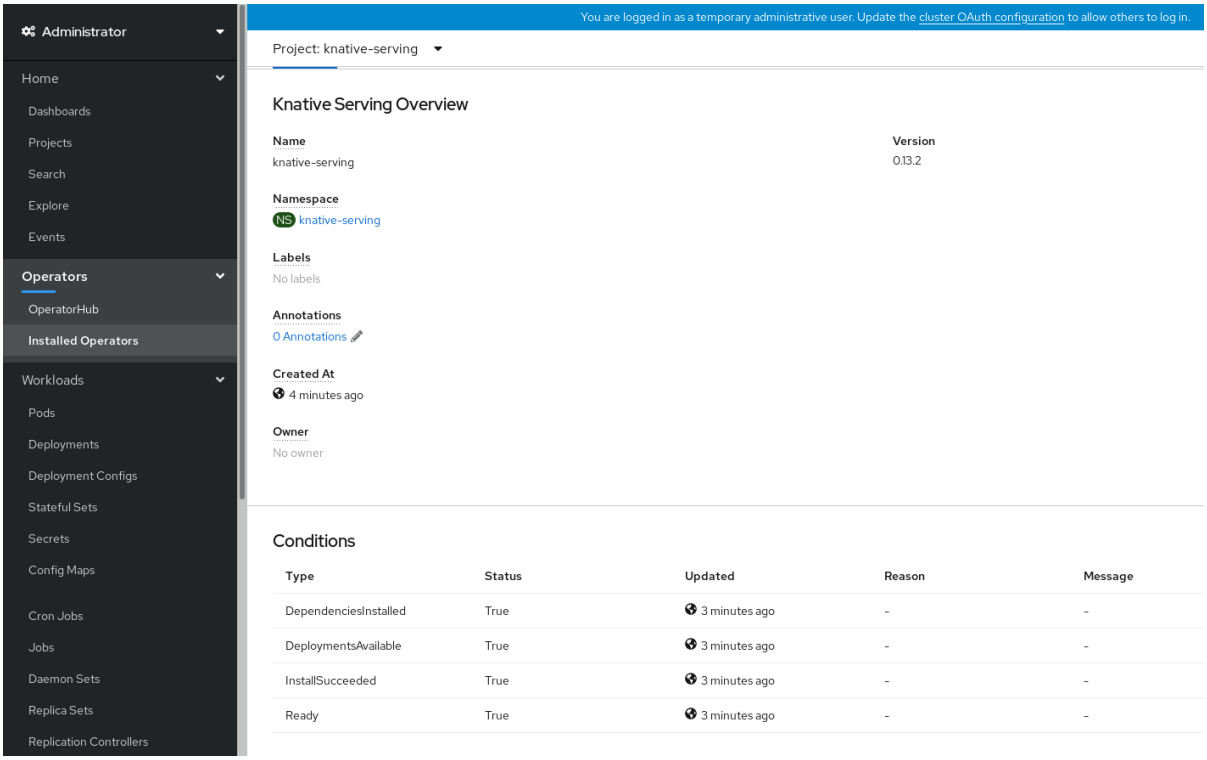
6. After you have installed Knative Serving, the **KnativeServing** object is created, and you are automatically directed to the **Knative Serving** tab. You will see the **knative-serving** custom resource in the list of resources.

Verification

1. Click on **knative-serving** custom resource in the **Knative Serving** tab.
2. You will be automatically directed to the **Knative Serving Overview** page.



3. Scroll down to look at the list of **Conditions**.
4. You should see a list of conditions with a status of **True**, as shown in the example image.



NOTE

It may take a few seconds for the Knative Serving resources to be created. You can check their status in the **Resources** tab.

5. If the conditions have a status of **Unknown** or **False**, wait a few moments and then check again after you have confirmed that the resources have been created.

4.2. INSTALLING KNATIVE SERVING BY USING YAML

After you install the OpenShift Serverless Operator, you can install Knative Serving by using the default settings, or configure more advanced settings in the **KnativeServing** custom resource (CR). You can use the following procedure to install Knative Serving by using YAML files and the **oc** CLI.

Prerequisites

- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- You have installed the OpenShift Serverless Operator.
- Install the OpenShift CLI (**oc**).

Procedure

1. Create a file named **serving.yaml** and copy the following example YAML into it:

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeServing
metadata:
```

```
    name: knative-serving
    namespace: knative-serving
```

2. Apply the **serving.yaml** file:

```
$ oc apply -f serving.yaml
```

Verification

1. To verify the installation is complete, enter the following command:

```
$ oc get knativeserving.operator.knative.dev/knative-serving -n knative-serving --
template='{{range .status.conditions}}{{printf "%s=%s\n" .type .status}}{{end}}'
```

Example output

```
DependenciesInstalled=True
DeploymentsAvailable=True
InstallSucceeded=True
Ready=True
```

NOTE

It may take a few seconds for the Knative Serving resources to be created.

If the conditions have a status of **Unknown** or **False**, wait a few moments and then check again after you have confirmed that the resources have been created.

2. Check that the Knative Serving resources have been created:

```
$ oc get pods -n knative-serving
```

Example output

```
NAME                                                READY   STATUS      RESTARTS   AGE
activator-67ddf8c9d7-p7rm5                          2/2     Running     0          4m
activator-67ddf8c9d7-q84fz                          2/2     Running     0          4m
autoscaler-5d87bc6dbf-6nqc6                         2/2     Running     0          3m59s
autoscaler-5d87bc6dbf-h64rl                         2/2     Running     0          3m59s
autoscaler-hpa-77f85f5cc4-lrts7                     2/2     Running     0          3m57s
autoscaler-hpa-77f85f5cc4-zx7hl                     2/2     Running     0          3m56s
controller-5cfc7cb8db-nlccl                         2/2     Running     0          3m50s
controller-5cfc7cb8db-rmv7r                         2/2     Running     0          3m18s
domain-mapping-86d84bb6b4-r746m                     2/2     Running     0          3m58s
domain-mapping-86d84bb6b4-v7nh8                     2/2     Running     0          3m58s
domainmapping-webhook-769d679d45-bkcnj              2/2     Running     0          3m58s
domainmapping-webhook-769d679d45-fff68              2/2     Running     0          3m58s
storage-version-migration-serving-serving-0.26.0--1-6qlkb   0/1     Completed   0          3m56s
webhook-5fb774f8d8-6bqrt                            2/2     Running     0          3m57s
webhook-5fb774f8d8-b8lt5                            2/2     Running     0          3m57s
```

3. Check that the necessary networking components have been installed to the automatically created **knative-serving-ingress** namespace:

```
$ oc get pods -n knative-serving-ingress
```

Example output

```
NAME                                    READY   STATUS    RESTARTS   AGE
net-kourier-controller-7d4b6c5d95-62mkf   1/1     Running   0          76s
net-kourier-controller-7d4b6c5d95-qmgm2   1/1     Running   0          76s
3scale-kourier-gateway-6688b49568-987qz   1/1     Running   0          75s
3scale-kourier-gateway-6688b49568-b5tnp   1/1     Running   0          75s
```

4.3. NEXT STEPS

- If you want to use Knative event-driven architecture you can install Knative Eventing.

CHAPTER 5. INSTALLING KNATIVE EVENTING

To use event-driven architecture on your cluster, install Knative Eventing. You can create Knative components such as event sources, brokers, and channels and then use them to send events to applications or external systems.

After you install the OpenShift Serverless Operator, you can install Knative Eventing by using the default settings, or configure more advanced settings in the **KnativeEventing** custom resource (CR). For more information about configuration options for the **KnativeEventing** CR, see Global configuration.

IMPORTANT

If you want to use Red Hat OpenShift distributed tracing with OpenShift Serverless , you must install and configure Red Hat OpenShift distributed tracing before you install Knative Eventing.

5.1. INSTALLING KNATIVE EVENTING BY USING THE WEB CONSOLE

After you install the OpenShift Serverless Operator, install Knative Eventing by using the OpenShift Container Platform web console. You can install Knative Eventing by using the default settings or configure more advanced settings in the **KnativeEventing** custom resource (CR).

Prerequisites

- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- You have logged in to the OpenShift Container Platform web console.
- You have installed the OpenShift Serverless Operator.

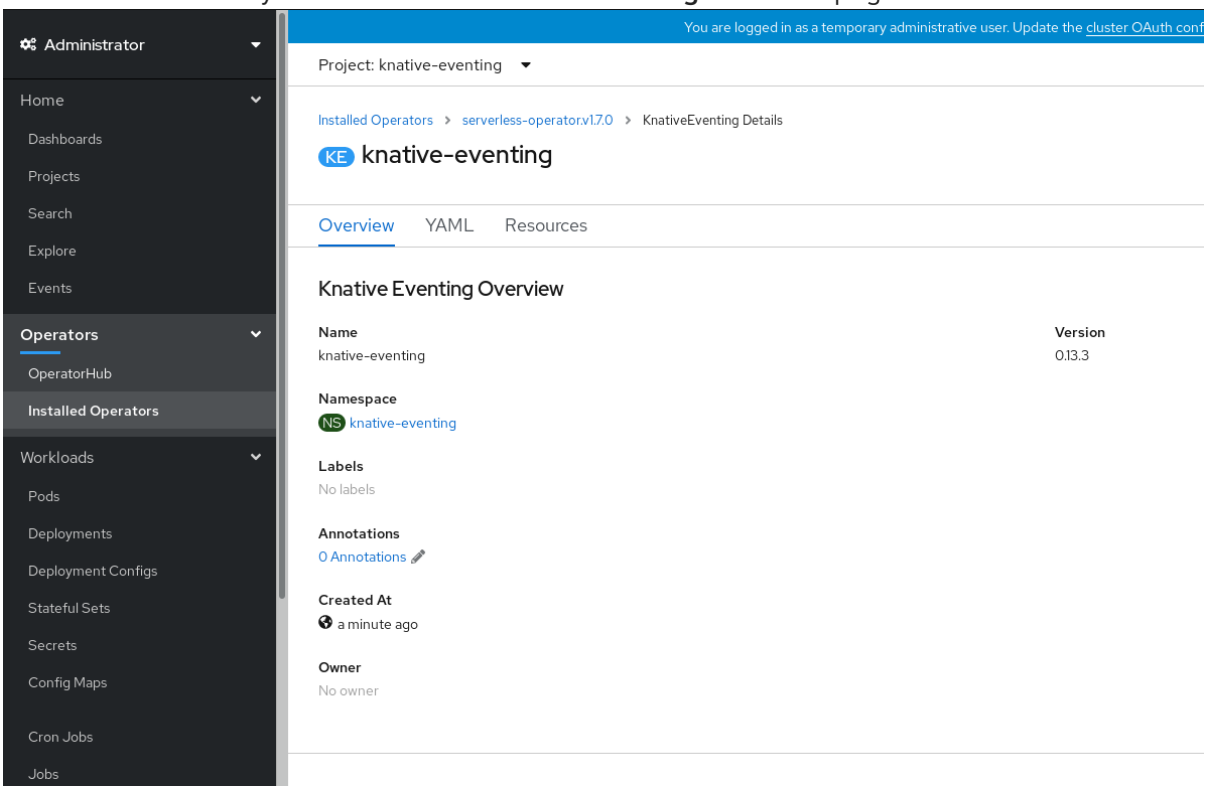
Procedure

1. In the **Administrator** perspective of the OpenShift Container Platform web console, navigate to **Operators → Installed Operators**.
2. Check that the **Project** dropdown at the top of the page is set to **Project: knative-eventing**.
3. Click **Knative Eventing** in the list of **Provided APIs** for the OpenShift Serverless Operator to go to the **Knative Eventing** tab.
4. Click **Create Knative Eventing**.
5. In the **Create Knative Eventing** page, you can configure the **KnativeEventing** object by using either the form provided, or by editing the YAML file.
    - Use the form for simpler configurations that do not require full control of **KnativeEventing** object creation.
6. Click **Create**.
    - Edit the YAML file for more complex configurations that require full control of **KnativeEventing** object creation. To access the YAML editor, click **edit YAML** on the **Create Knative Eventing** page.

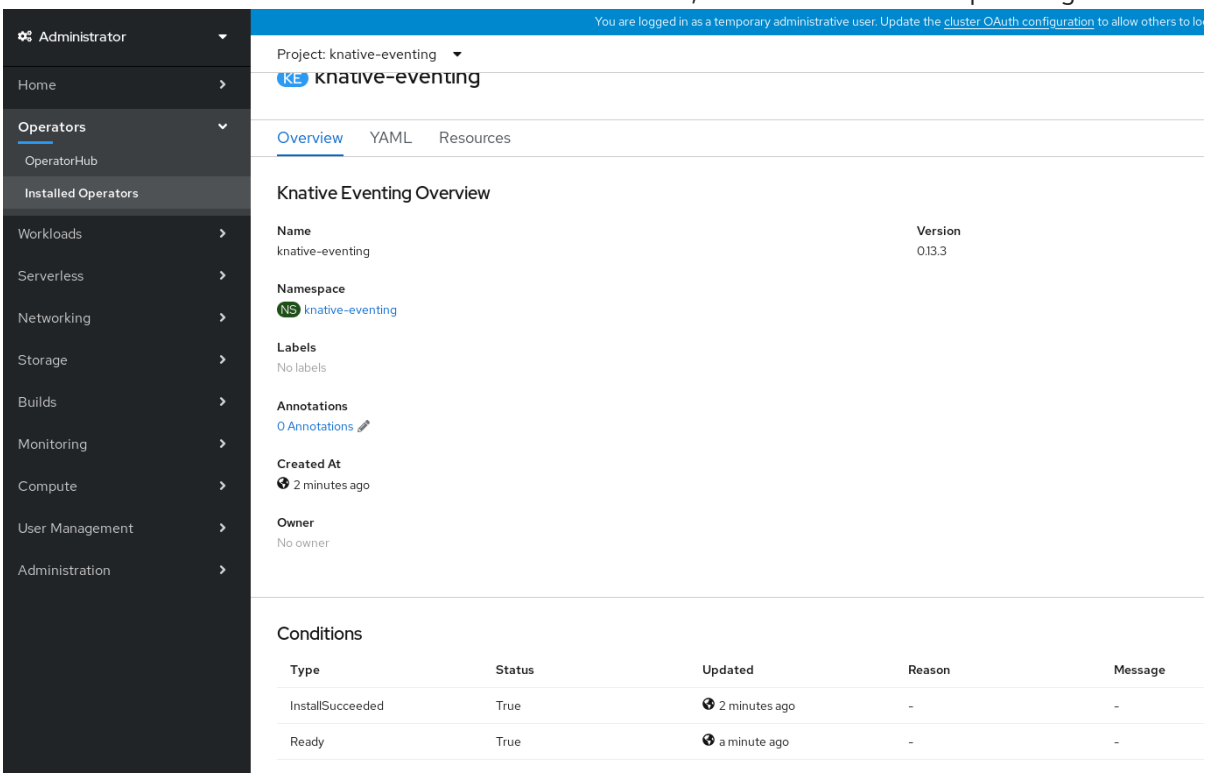
7. After you have installed Knative Eventing, the **KnativeEventing** object is created, and you are automatically directed to the **Knative Eventing** tab. You will see the **knative-eventing** custom resource in the list of resources.

Verification

1. Click on the **knative-eventing** custom resource in the **Knative Eventing** tab.
2. You are automatically directed to the **Knative Eventing Overview** page.



3. Scroll down to look at the list of **Conditions**.
4. You should see a list of conditions with a status of **True**, as shown in the example image.



NOTE

It may take a few seconds for the Knative Eventing resources to be created. You can check their status in the **Resources** tab.

5. If the conditions have a status of **Unknown** or **False**, wait a few moments and then check again after you have confirmed that the resources have been created.

5.2. INSTALLING KNATIVE EVENTING BY USING YAML

After you install the OpenShift Serverless Operator, you can install Knative Eventing by using the default settings, or configure more advanced settings in the **KnativeEventing** custom resource (CR). You can use the following procedure to install Knative Eventing by using YAML files and the **oc** CLI.

Prerequisites

- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- You have installed the OpenShift Serverless Operator.
- Install the OpenShift CLI (**oc**).

Procedure

1. Create a file named **eventing.yaml**.
2. Copy the following sample YAML into **eventing.yaml**:

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeEventing
metadata:
    name: knative-eventing
    namespace: knative-eventing
```

3. Optional. Make any changes to the YAML that you want to implement for your Knative Eventing deployment.
4. Apply the **eventing.yaml** file by entering:

```
$ oc apply -f eventing.yaml
```

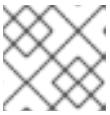
Verification

1. Verify the installation is complete by entering the following command and observing the output:

```
$ oc get knativeeventing.operator.knative.dev/knative-eventing \
  -n knative-eventing \
  --template='{{range .status.conditions}}{{printf "%s=%s\n" .type .status}}{{end}}'
```

Example output

```
InstallSucceeded=True
Ready=True
```



NOTE

It may take a few seconds for the Knative Eventing resources to be created.

2. If the conditions have a status of **Unknown** or **False**, wait a few moments and then check again after you have confirmed that the resources have been created.

3. Check that the Knative Eventing resources have been created by entering:

```
$ oc get pods -n knative-eventing
```

Example output

```
NAME                                  READY   STATUS    RESTARTS   AGE
broker-controller-58765d9d49-g9zp6    1/1     Running   0          7m21s
eventing-controller-65fdd66b54-jw7bh  1/1     Running   0          7m31s
eventing-webhook-57fd74b5bd-kvhlz     1/1     Running   0          7m31s
imc-controller-5b75d458fc-ptvm2       1/1     Running   0          7m19s
imc-dispatcher-64f6d5fccb-kkc4c       1/1     Running   0          7m18s
```

5.3. INSTALLING KNATIVE BROKER FOR APACHE KAFKA

The Knative broker implementation for Apache Kafka provides integration options for you to use supported versions of the Apache Kafka message streaming platform with OpenShift Serverless. Knative broker for Apache Kafka functionality is available in an OpenShift Serverless installation if you have installed the **KnativeKafka** custom resource.

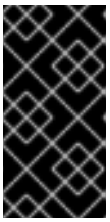
Prerequisites

- You have installed the OpenShift Serverless Operator and Knative Eventing on your cluster.
- You have access to a Red Hat AMQ Streams cluster.
- Install the OpenShift CLI (**oc**) if you want to use the verification steps.
- You have cluster administrator permissions on OpenShift Container Platform, or you have cluster or dedicated administrator permissions on Red Hat OpenShift Service on AWS or OpenShift Dedicated.
- You are logged in to the OpenShift Container Platform web console.

Procedure

1. In the **Administrator** perspective, navigate to **Operators → Installed Operators**.
2. Check that the **Project** dropdown at the top of the page is set to **Project: knative-eventing**.
3. In the list of **Provided APIs** for the OpenShift Serverless Operator, find the **Knative Kafka** box and click **Create Instance**.

4. Configure the **KnativeKafka** object in the **Create Knative Kafka** page.



IMPORTANT

To use the Kafka channel, source, broker, or sink on your cluster, you must toggle the **enabled** switch for the options you want to use to **true**. These switches are set to **false** by default. Additionally, to use the Kafka channel, broker, or sink you must specify the bootstrap servers.

- Use the form for simpler configurations that do not require full control of **KnativeKafka** object creation.
- Edit the YAML for more complex configurations that require full control of **KnativeKafka** object creation. You can access the YAML by clicking the **Edit YAML** link on the **Create Knative Kafka** page.

Example KnativeKafka custom resource

```
apiVersion: operator.serverless.openshift.io/v1alpha1
kind: KnativeKafka
metadata:
  name: knative-kafka
  namespace: knative-eventing
spec:
  channel:
    enabled: true (1)
    bootstrapServers: <bootstrap_servers> (2)
  source:
    enabled: true (3)
  broker:
    enabled: true (4)
    defaultConfig:
      bootstrapServers: <bootstrap_servers> (5)
      numPartitions: <num_partitions> (6)
      replicationFactor: <replication_factor> (7)
  sink:
    enabled: true (8)
  logging:
    level: INFO (9)
```

1. Enables developers to use the **KafkaChannel** channel type in the cluster.
2. A comma-separated list of bootstrap servers from your AMQ Streams cluster.
3. Enables developers to use the **KafkaSource** event source type in the cluster.
4. Enables developers to use the Knative broker implementation for Apache Kafka in the cluster.
5. A comma-separated list of bootstrap servers from your Red Hat AMQ Streams cluster.
6. Defines the number of partitions of the Kafka topics, backed by the **Broker** objects. The default is **10**.

7. Defines the replication factor of the Kafka topics, backed by the **Broker** objects. The default is **3**. The **replicationFactor** value must be less than or equal to the number of nodes of your Red Hat AMQ Streams cluster.

8. Enables developers to use a Kafka sink in the cluster.

9. Defines the log level of the Kafka data plane. Allowed values are **TRACE**, **DEBUG**, **INFO**, **WARN** and **ERROR**. The default value is **INFO**.

WARNING

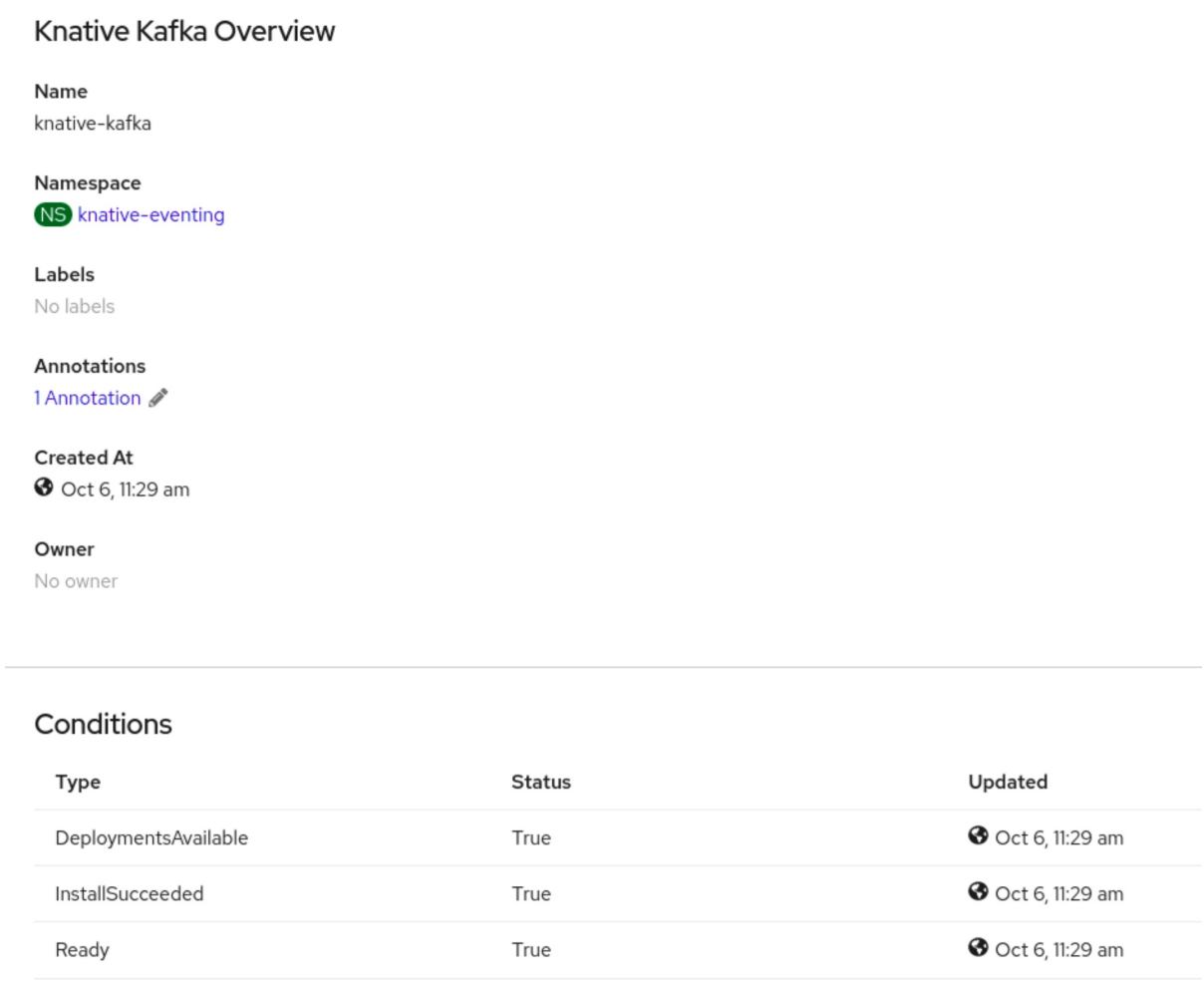
Do not use **DEBUG** or **TRACE** as the logging level in production environments. The outputs from these logging levels are verbose and can degrade performance.

5. Click **Create** after you have completed any of the optional configurations for Kafka. You are automatically directed to the **Knative Kafka** tab where **knative-kafka** is in the list of resources.

Verification

1. Click on the **knative-kafka** resource in the **Knative Kafka** tab. You are automatically directed to the **Knative Kafka Overview** page.

2. View the list of **Conditions** for the resource and confirm that they have a status of **True**.



If the conditions have a status of **Unknown** or **False**, wait a few moments to refresh the page.

3. Check that the Knative broker for Apache Kafka resources have been created:

```
$ oc get pods -n knative-eventing
```

Example output

```
NAME                                  READY   STATUS    RESTARTS   AGE
kafka-broker-dispatcher-7769fbbcbb-xgffn    2/2     Running   0          44s
kafka-broker-receiver-5fb56f7656-fhq8d      2/2     Running   0          44s
kafka-channel-dispatcher-84fd6cb7f9-k2tjv   2/2     Running   0          44s
kafka-channel-receiver-9b7f795d5-c76xr      2/2     Running   0          44s
kafka-controller-6f95659bf6-trd6r           2/2     Running   0          44s
kafka-source-dispatcher-6bf98bdfff-8bcsn    2/2     Running   0          44s
kafka-webhook-eventing-68dc95d54b-825xs     2/2     Running   0          44s
```

5.4. NEXT STEPS

- If you want to use Knative services you can install Knative Serving.

CHAPTER 6. CONFIGURING KNATIVE BROKER FOR APACHE KAFKA

The Knative broker implementation for Apache Kafka provides integration options for you to use supported versions of the Apache Kafka message streaming platform with OpenShift Serverless. Kafka provides options for event source, channel, broker, and event sink capabilities.

In addition to the Knative Eventing components that are provided as part of a core OpenShift Serverless installation, the **KnativeKafka** custom resource (CR) can be installed by:

- Cluster administrators, for OpenShift Container Platform
- Cluster or dedicated administrators, for Red Hat OpenShift Service on AWS or for OpenShift Dedicated.

The **KnativeKafka** CR provides users with additional options, such as:

- Kafka source
- Kafka channel
- Kafka broker
- Kafka sink

CHAPTER 7. CONFIGURING KUBE-RBAC-PROXY FOR KNATIVE FOR APACHE KAFKA

The **kube-rbac-proxy** component provides internal authentication and authorization capabilities for Knative for Apache Kafka.

7.1. CONFIGURING KUBE-RBAC-PROXY RESOURCES FOR KNATIVE FOR APACHE KAFKA

You can globally override resource allocation for the **kube-rbac-proxy** container by using the OpenShift Serverless Operator CR.

> You can also override resource allocation for a specific deployment.

The following configuration sets Knative Kafka **kube-rbac-proxy** minimum and maximum CPU and memory allocation:

KnativeKafka CR example

```
apiVersion: operator.serverless.openshift.io/v1alpha1
kind: KnativeKafka
metadata:
  name: knative-kafka
  namespace: knative-kafka
spec:
  config:
    workload:
      "kube-rbac-proxy-cpu-request": "10m" 1
      "kube-rbac-proxy-memory-request": "20Mi" 2
      "kube-rbac-proxy-cpu-limit": "100m" 3
      "kube-rbac-proxy-memory-limit": "100Mi" 4
```

1. Sets minimum CPU allocation.
2. Sets minimum RAM allocation.
3. Sets maximum CPU allocation.
4. Sets maximum RAM allocation.

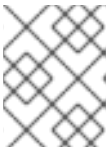
CHAPTER 8. CONFIGURING OPENSHIFT SERVERLESS FUNCTIONS

To improve the process of deployment of your application code, you can use OpenShift Serverless to deploy stateless, event-driven functions as a Knative service on OpenShift Container Platform. If you want to develop functions, you must complete the set up steps.

8.1. PREREQUISITES

To enable the use of OpenShift Serverless Functions on your cluster, you must complete the following steps:

- The OpenShift Serverless Operator and Knative Serving are installed on your cluster.



NOTE

Functions are deployed as a Knative service. If you want to use event-driven architecture with your functions, you must also install Knative Eventing.

- You have the **oc** CLI installed.
- You have the Knative (**kn**) CLI installed. Installing the Knative CLI enables the use of **kn func** commands which you can use to create and manage functions.
- You have installed Docker Container Engine or Podman version 3.4.7 or higher.
- You have access to an available image registry, such as the OpenShift Container Registry.
- If you are using Quay.io as the image registry, you must ensure that either the repository is not private, or that you have followed the OpenShift Container Platform documentation on Allowing pods to reference images from other secured registries.
- If you are using the OpenShift Container Registry, a cluster administrator must expose the registry.

8.2. SETTING UP PODMAN

To use advanced container management features, you might want to use Podman with OpenShift Serverless Functions. To do so, you need to start the Podman service and configure the Knative (**kn**) CLI to connect to it.

Procedure

1. Start the Podman service that serves the Docker API on a UNIX socket at **${XDG_RUNTIME_DIR}/podman/podman.sock**:

```
$ systemctl start --user podman.socket
```

NOTE

On most systems, this socket is located at **/run/user/$(id -u)/podman/podman.sock**.

2. Establish the environment variable that is used to build a function:

```
$ export DOCKER_HOST="unix://${XDG_RUNTIME_DIR}/podman/podman.sock"
```

3. Run the build command inside your function project directory with the **-v** flag to see verbose output. You should see a connection to your local UNIX socket:

```
$ kn func build -v
```

8.3. SETTING UP PODMAN ON MACOS

To use advanced container management features, you might want to use Podman with OpenShift Serverless Functions. To do so on macOS, you need to start the Podman machine and configure the Knative (**kn**) CLI to connect to it.

Procedure

1. Create the Podman machine:

```
$ podman machine init --memory=8192 --cpus=2 --disk-size=20
```

2. Start the Podman machine, which serves the Docker API on a UNIX socket:

```
$ podman machine start
Starting machine "podman-machine-default"
Waiting for VM ...
Mounting volume... /Users/myuser:/Users/user

[...truncated output...]

You can still connect Docker API clients by setting DOCKER_HOST using the
following command in your terminal session:

 export
DOCKER_HOST='unix:///Users/myuser/.local/share/containers/podman/machine/podman-
machine-default/podman.sock'

Machine "podman-machine-default" started successfully
```

NOTE

On most macOS systems, this socket is located at **/Users/myuser/.local/share/containers/podman/machine/podman-machine-default/podman.sock**.

3. Establish the environment variable that is used to build a function:

```
$ export
DOCKER_HOST='unix:///Users/myuser/.local/share/containers/podman/machine/podman-
machine-default/podman.sock'
```

4. Run the build command inside your function project directory with the **-v** flag to see verbose output. You should see a connection to your local UNIX socket:

```
$ kn func build -v
```

8.4. NEXT STEPS

- For more information about Docker Container Engine or Podman, see Container build tool options.
- See Getting started with functions.

CHAPTER 9. SERVERLESS UPGRADES

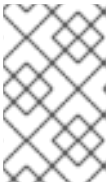
OpenShift Serverless should be upgraded without skipping release versions. This section shows how to resolve problems with upgrading.

9.1. SERVERLESS OPERATOR MAINTENANCE RELEASE UPGRADES

9.1.1. Latest and maintenance releases

Beginning with OpenShift Serverless 1.29, the different product versions are available as follows:

- The latest release is available through the **stable** channel.
- The maintenance release is available through its version-based channel, for example, **stable-1.29**.



NOTE

The maintenance release is the release prior to the latest release. For example, if the **stable** channel includes version 1.30, then the maintenance release will be available in the **stable-1.29** channel.

Using a version-based channel allows you to stay within a specific **x.y** stream. Additionally, it prevents an upgrade to the latest version of the product, which might contain breaking changes.

To switch to the maintenance release, update the channel parameter in the subscription object YAML file from **stable** to the corresponding version-based channel, such as **stable-1.29**.

9.1.2. Patches and hotfixes for maintenance releases

As with the stable release, the maintenance release is subject to patches and hotfixes, which help to keep your deployment up to date with critical bug and security fixes.

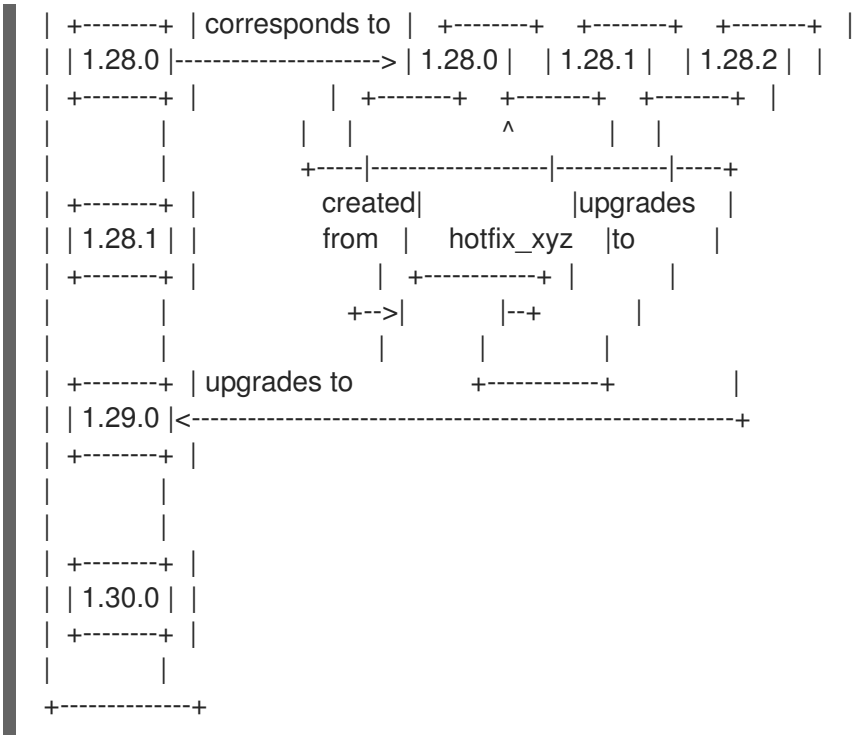
- Patches are updates that are distributed as z-releases, for example, OpenShift Serverless 1.29.1 is the patch that provides updates made since version 1.29.0.
- Hotfixes are fixes that require zero downtime and are used directly in production. They are different from the usual updates in that they upgrade the customer's deployed version, and not the latest released version.
  Hotfixes might not be available to all customers immediately. However, changes introduced by hotfixes often become available to all customers as part of a future release.

  When a hotfix pertaining to your deployment is available, you will be given the hotfix CatalogSource to update your subscription and obtain the hotfix.

  After a new operator release is available, deployed operators with hotfixes can also be upgraded. To use the latest GA version, modify the subscription to use the public CatalogSource instead of the hotfix one.

The following diagram illustrates how patches and hotfixes work:

```
    stable                          stable-1.28
+--------------+            +-----------------------------------------+
|              |            |                                         |
```



9.1.3. Upgrade path for maintenance releases

If you use a version-based channel, you can always upgrade to the latest version in the channel, or head. For example, you can upgrade from 1.29.0 to 1.29.2 in the **stable-1.29** channel.

Additionally, from the head of the channel, you can upgrade to the next **x.y** release. For example, if 1.29.2 is the head in the **stable-1.29** channel, you can upgrade from 1.29.2 to 1.30. Such cross-channel updates are not done automatically, and the administrator needs to switch the channel manually by updating the subscription.

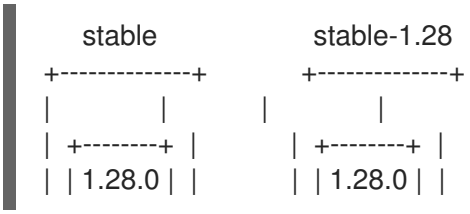
9.1.4. Upgrade examples

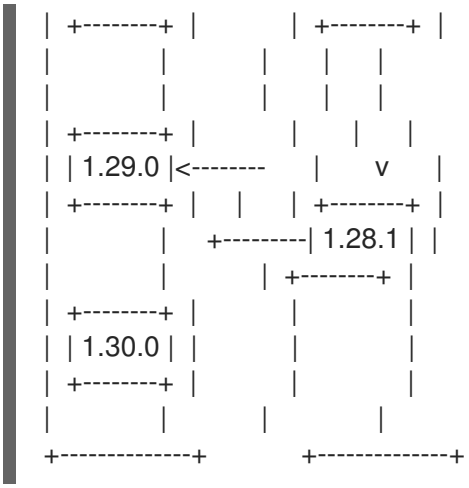
9.1.4.1. Scenario 1

In this scenario, the following circumstances are true:

- The channel is **stable-1.28**
- You are switching to the **stable** channel
- The currently installed version is 1.28.0
- 1.29.0 was released before 1.28.1
- 1.30.0 is the head of the **stable** channel

The upgrade path from 1.28.0 on **stable-1.28** to 1.29.0 on **stable** in this scenario is 1.28.0 to 1.28.1 to 1.29.0.



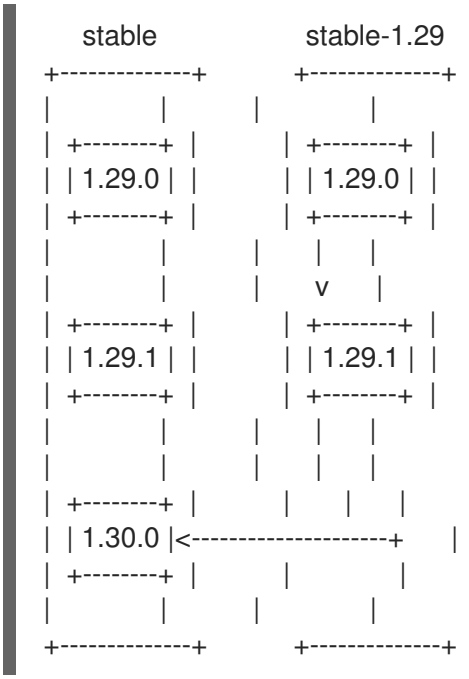


9.1.4.2. Scenario 2

In this scenario, the following circumstances are true:

- The channel is **stable-1.29**
- The currently installed version is 1.29.0
- 1.29.1 was released to both the **stable-1.29** and **stable** channels before **1.30.0** was released to the **stable** channel

The upgrade path from 1.29.0 on **stable-1.29** to 1.30.0 on **stable** in this scenario is 1.29.0 to 1.29.1 to 1.30.0.



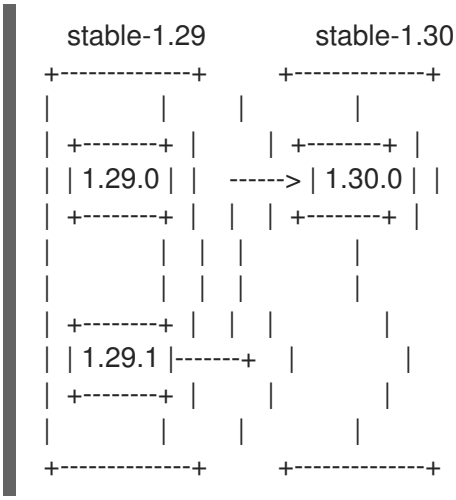
9.1.4.3. Scenario 3

In this scenario, the following circumstances are true:

- The channel is **stable-1.29**
- You are switching to the **stable-1.30** channel

- The currently installed version is 1.29.1
- 1.29.1 is the head of the **stable-1.29** channel

The upgrade path from 1.29.1 on **stable-1.29** to 1.30.0 on **stable-1.30** in this scenario is 1.29.1 to 1.30.0.



9.2. RESOLVING AN OPENSHIFT SERVERLESS OPERATOR UPGRADE FAILURE

You might encounter an error when upgrading OpenShift Serverless Operator, for example, when performing manual uninstalls and reinstalls. If you encounter an error, you must manually reinstall OpenShift Serverless Operator.

Procedure

1. Identify the version of OpenShift Serverless Operator that was installed originally by searching in the OpenShift Serverless Release Notes.
   For example, the error message during attempted upgrade might contain the following string:

   ```
   The installed KnativeServing version is v1.5.0.
   ```

   In this example, the KnativeServing **MAJOR.MINOR** version is **1.5**, which is covered in the release notes for OpenShift Serverless 1.26: *OpenShift Serverless now uses Knative Serving 1.5* .

2. Uninstall OpenShift Serverless Operator and all of its install plans.

3. Manually install the version of OpenShift Serverless Operator that you discovered in the first step. To install, first create a **serverless-subscription.yaml** file as shown in the following example:

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     name: serverless-operator
     namespace: openshift-serverless
   spec:
     channel: stable
     name: serverless-operator
     source: redhat-operators
   ```

```
sourceNamespace: openshift-marketplace
installPlanApproval: Manual
startingCSV: serverless-operator.v1.26.0
```

4. Then, install the subscription by running the following command:

```
$ oc apply -f serverless-subscription.yaml
```

5. Upgrade by manually approving the upgrade install plans as they appear.

Additional resources

- OpenShift Serverless Release Notes
- Deleting Operators from a cluster using the web console
- Installing the OpenShift Serverless Operator from the web console