Red Hat OpenShift Serverless 1.32

安装 Serverless

安装 Serverless Operator、Knative CLI、Knative Serving 和 Knative Eventing

Last Updated: 2024-05-20

Red Hat OpenShift Serverless 1.32 安装 Serverless

安装 Serverless Operator、Knative CLI、Knative Serving 和 Knative Eventing

法律通告

摘要

本文档提供有关安装 Serverless Operator、Knative CLI、Knative Serving 和 Knative Eventing 的信息。它还提供了有关为 Apache Kafka 配置 Knative 以及配置 Serverless 功能的详细信息。

目录

第 1 章 准备安装 OPENSHIFT SERVERLESS

在安装 OpenShift Serverless 前，请阅读以下有关支持的配置和先决条件的信息。

对于 OpenShift Container Platform：

- OpenShift Serverless 支持在受限网络环境中安装。
- OpenShift Serverless 目前无法在单个集群的多租户配置中使用。

1.1. 支持的配置

OpenShift Serverless 支持的功能、配置和集成（当前和过去的版本）包括在 支持的配置页面中。

1.2. OPENSHIFT CONTAINER PLATFORM 的可扩展性和性能

OpenShift Serverless 已使用配置为 3 个主要节点和 3 个 worker 节点进行测试，每个节点都有 64 个 CPU、457 GB 内存和 394 GB 存储。

使用此配置创建的最大 Knative 服务数为 3,000。这与 OpenShift Container Platform Kubernetes 服务限制 10,000 对应，因为 1 个 Knative 服务创建 3 个 Kubernetes 服务。

零响应时间的平均缩放约为 3.4 秒，最大响应时间为 8 秒，而一个简单 Quarkus 应用程序使用 99.9thile 的 4.5 秒。这些时间可能因应用程序和应用程序的运行时的不同而有所不同。

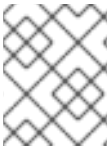
注意

定义集群大小要求的部分适用于这些发行版本：

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

1.3. 定义集群大小要求

要安装和使用 OpenShift Serverless，集群必须正确定义大小。



注意

以下要求仅与集群的 worker 机器池相关。control plane 节点不用于常规调度，它不在要求中。

使用 OpenShift Serverless 的最低要求是集群有 10 个 CPU 和 40GB 内存。默认情况下，每个 Pod 需要大约 400m 的 CPU，最下的要求基于此值。

运行 OpenShift Serverless 的总大小要求取决于安装的组件以及部署的应用程序，并因部署的不同而有所不同。

1.4. 使用 OPENSHIFT CONTAINER PLATFORM 上的计算机器集扩展集群

您可以使用 OpenShift Container Platform **MachineSet** API 手动将集群扩展至所需大小。最低要求通常意味着您必须将一个默认计算机器集扩展两个额外的机器。请参阅 手动扩展计算机器集。

1.4.1. 更复杂用例所需的额外资源

对于更复杂的用例，如日志、OpenShift Container Platform 指标数据等，必须部署更多资源。对这类用例的推荐要求为 24 个 CPU 和 96GB 内存。

如果您在集群中启用了高可用性 (HA)，需要 0.5 - 1.5 个内核，每个 Knative Serving control plane 副本需要 200MB 到 2GB 内存。一些 Knative Serving 组件在默认情况下启用了 HA。您可以按照"配置高可用性副本"的文档禁用 HA。

1.5. OPENSHIFT CONTAINER PLATFORM 文档中的其他资源

- 在受限网络中使用 Operator Lifecycle Manager
- 了解 OperatorHub
- 集群功能

第 2 章 安装 OPENSHIFT SERVERLESS OPERATOR

安装 OpenShift Serverless Operator 可让您在 OpenShift Container Platform 集群中安装和使用 Knative Serving、Knative Eventing 和 Knative broker for Apache Kafka。OpenShift Serverless Operator 管理集群的 Knative 自定义资源定义 (CRD)，并可让您在不直接为每个组件修改单个配置映射的情况下配置它们。

2.1. OPENSHIFT SERVERLESS OPERATOR 资源要求

以下示例设置可帮助您估算 OpenShift Serverless Operator 安装的最低资源要求。您的特定要求可能会有很大不同，随着 OpenShift Serverless 的使用增加，可能会增加。

示例设置中使用的测试套件有以下参数：

- 一个 OpenShift Container Platform 集群，它有以下几项：
  - 10 个 worker (8 个 vCPU，16GiB 内存)
  - 3 个专用于 Kafka 的 worker
  - 2 个专用于 Prometheus 的 worker
  - 5 个 worker 用于 Serverless 和测试部署
- 89 测试并行运行的情况，主要专注于使用 control plane。测试场景通常有一个 Knative Service 通过内存频道、Kafka 频道、内存中代理或 Kafka 代理发送到 Deployment 或 Knative Service。
- 48 个重新创建场景，其中测试场景会被重复删除并重新创建。
- 41 稳定场景，事件在测试过程中发送缓慢但持续运行。
- 测试设置包括：
  - 170 个 Knative Services
  - 20 in-Memory Channels
  - 24 Kafka 频道
  - 52 个订阅
  - 42 代理
  - 68 触发器

下表详细介绍了测试套件发现的高可用(HA)设置的最小资源要求：

组件	RAM 资源	CPU 资源
OpenShift Serverless Operator	1GB	0.2 个内核
Knative Serving	5GB	2.5 个内核
Knative Eventing	2GB	0.5 core

组件	RAM 资源	CPU 资源
Apache Kafka 的 Knative 代理	6GB	1 个内核
sum	14GB	4.2 内核

下表详细介绍了测试套件发现的非 HA 设置的最小资源要求：

组件	RAM 资源	CPU 资源
OpenShift Serverless Operator	1GB	0.2 个内核
Knative Serving	2.5GB	1.2 个内核
Knative Eventing	1GB	0.2 个内核
Apache Kafka 的 Knative 代理	6GB	1 个内核
sum	10.5GB	2.6 内核

2.2. 通过 WEB 控制台安装 OPENSHIFT SERVERLESS OPERATOR

您可以使用 OpenShift Container Platform Web 控制台从 OperatorHub 安装 OpenShift Serverless Operator。安装此 Operator 可让您安装和使用 Knative 组件。

先决条件

- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 对于 OpenShift Container Platform，您的集群启用了 Marketplace 功能，或者手动配置 Red Hat Operator 目录源。
- 已登陆到 web 控制台。

流程

1. 在 Web 控制台中，导航到 **Operators → OperatorHub** 页面。
2. 滚动页面，或在 **Filter by keyword** 框中输入关键字 **Serverless** 来查找 OpenShift Serverless Operator。
3. 查看 Operator 信息并单击 **Install**。
4. 在 **Install Operator** 页面中：
    a. **Installation Mode** 是 **All namespaces on the cluster (default)**。此模式将 Operator 安装至默认 **openshift-serverless** 命名空间，以便供集群中的所有命名空间监视和使用。
    b. **安装的命名空间** 是 **openshift-serverless**。

c. 选择一个 **Update Channel**。

- **stable** 频道启用 OpenShift Serverless Operator 最新稳定版本的安装。 **stable** 频道是默认值。
- 要安装另一个版本，请指定对应的 **stable-x.y** 频道，如 **stable-1.29**。

d. 选择 **stable** 频道作为 **更新频道**。**stable** 频道将启用 OpenShift Serverless Operator 最新稳定版本的安装。

e. 选择 **Automatic** 或 **Manual** 批准策略。

5. 点 **Install** 使 Operator 可供 OpenShift Container Platform 集群上的所选命名空间使用。

6. 在 **Catalog → Operator Management** 页面中，您可以监控 OpenShift Serverless Operator 订阅的安装和升级进度。

a. 如果选择了 **Manual** 批准策略，订阅的升级状态将会一直保持在 **Upgrading**，直到您审阅并批准了它的安装计划。在 **Install Plan** 页面批准后，订阅的升级状态将变为 **Up to date**。

b. 如果选择了 **Automatic** 批准策略，升级状态会在不用人工参与的情况下变为 **Up to date**。

验证

当订阅的升级状态变为**Up to date** 后，选择 **Catalog → Installed Operators** 来验证 OpenShift Serverless Operator 最终出现，它的 **Status** 在相关的命名空间中最终会变为 **InstallSucceeded**。

如果没有：

1. 切换到 **Catalog → Operator Management** 页，检查 **Operator Subscriptions** 和 **Install Plans** 页中的 **Status** 是否有错误。

2. 检查 **Workloads → Pods** 页中提供的关于**openshift-serverless** 项目中的 pod 的日志信息，以便进一步排除故障。

重要

如果要在 OpenShift Serverless 中使用 Red Hat OpenShift distributed tracing，则必须在安装 Knative Serving 或 Knative Eventing 前安装和配置 Red Hat OpenShift distributed tracing。

2.3. 通过 CLI 安装 OPENSHIFT SERVERLESS OPERATOR

您可以使用 CLI 从 OperatorHub 安装 OpenShift Serverless Operator。安装此 Operator 可让您安装和使用 Knative 组件。

先决条件

- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 对于 OpenShift Container Platform，您的集群启用了 Marketplace 功能，或者手动配置 Red Hat Operator 目录源。
- 已登陆到集群。

流程

流程

1. 创建包含 **Namespace**、**OperatorGroup** 和 **Subscription** 对象的 YAML 文件，以便为 OpenShift Serverless Operator 订阅命名空间。例如，使用以下内容创建文件 **serverless-subscription.yaml** ：

   订阅示例

```
---
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) Operator 的频道名称。**stable** 频道启用 OpenShift Serverless Operator 最新稳定版本的安装。要安装另一个版本，请指定对应的 **stable-x.y** 频道，如 **stable-1.29**。

   (2) 要订阅的 Operator 的名称。对于 OpenShift Serverless Operator，这始终为 **serverless-operator**。

   (3) 提供 Operator 的 CatalogSource 的名称。将 **redhat-operators** 用于默认的 OperatorHub 目录源。

   (4) CatalogSource 的命名空间。将 **openshift-marketplace** 用于默认的 OperatorHub 目录源。

2. 创建 **Subscription** 对象：

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

验证

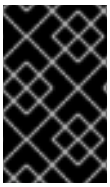
检查集群服务版本 (CSV) 是否已进入 **Succeeded** 阶段：

示例命令

```
$ oc get csv
```

输出示例

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



重要

如果要在 OpenShift Serverless 中使用 Red Hat OpenShift distributed tracing，则必须在安装 Knative Serving 或 Knative Eventing 前安装和配置 Red Hat OpenShift distributed tracing。

2.4. 全局配置

OpenShift Serverless Operator 管理 Knative 安装的全局配置，包括将 **KnativeServing** 和 **KnativeEventing** 自定义资源的值传播到系统配置映射。任何手动应用的配置映射更新都会被 Operator 覆盖。但是，通过修改 Knative 自定义资源，您可以为这些配置映射设置值。

Knative 具有多个配置映射，它们使用前缀 **config-** 命名。所有 Knative 配置映射都与它们应用到的自定义资源在同一命名空间中创建。例如，如果在 **knative-serving** 命名空间中创建 **KnativeServing** 自定义资源，则也会在此命名空间中创建所有 Knative Serving 配置映射。

Knative 自定义资源中的 **spec.config** 为每个配置映射有一个 **<name>** 条目，名为 **config-<name>**，其值为配置映射的 **data**。

2.5. OPENSHIFT CONTAINER PLATFORM 的其他资源

- 管理自定义资源定义中的资源
- 了解持久性存储
- 配置自定义 PKI

2.6. 后续步骤

- 安装 OpenShift Serverless Operator 后，您可以安装 Knative Serving 或 安装 Knative Eventing。

第 3 章 安装 KNATIVE CLI

Knative (**kn**) CLI 本身没有登录机制。要登录到集群，您必须安装 OpenShift CLI (**oc**)，并使用 **oc login** 命令。CLI 的安装选项可能会因您的操作系统而异。

有关为您的操作系统安装 OpenShift CLI (**oc**)并使用 **oc** 登录的更多信息，请参阅 OpenShift CLI 启动 文档。

OpenShift Serverless 不能使用 Knative (**kn**) CLI 安装。集群管理员必须安装 OpenShift Serverless Operator 并设置 Knative 组件，如 安装 OpenShift Serverless Operator 文档所述。

重要

如果您试图将较旧版本的 Knative **kn** CLI 与较新的 OpenShift Serverless 发行版本搭配使用，则不会找到 API，并出现错误。

例如，您使用 1.23.0 版本的 Knative (**kn**) CLI（使用版本 1.2），以及 1.24.0 版本的 OpenShift Serverless（使用版本 1.3 的 Knative Serving 和 Knative Eventing API），则 CLI 将无法正常工作，因为它会一直寻找已过时的 1.2 版本的 API。

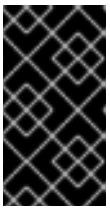
确保为 OpenShift Serverless 版本使用最新的 Knative (**kn**) CLI 版本以避免出现问题。

3.1. 使用 OPENSHIFT CONTAINER PLATFORM WEB 控制台安装 KNATIVE CLI

使用 OpenShift Container Platform Web 控制台提供了一个简洁、直观的用户界面来安装 Knative (**kn**) CLI。安装 OpenShift Serverless Operator 后，您会看到从 OpenShift Container Platform Web 控制台的 **Command Line Tools** 页面中下载适用于 Linux 的 Knative (**kn**) CLI 的链接（amd64、s390x、ppc64le）、macOS 或 Windows。

先决条件

- 已登陆到 OpenShift Container Platform Web 控制台。
- OpenShift Serverless Operator 和 Knative Serving 已安装在 OpenShift Container Platform 集群中。



重要

如果 **libc** 不可用，您在运行 CLI 命令时可能会看到以下错误：

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

- 如果要使用验证步骤，您必须安装 OpenShift (**oc**) CLI。

流程

1. 从 **Command Line Tools** 页面下载 Knative (**kn**) CLI。您可以点 web 控制台右上角的  图标进入 **Command Line Tools** 页，并在列表中选择 **Command Line Tools**。
2. 解包存档：

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

3. 将 **kn** 二进制文件移到 **PATH** 中的目录中。
4. 运行以下命令可以查看 **PATH** 的值：

```
$ echo $PATH
```

验证

- 运行以下命令检查是否已创建了正确的 Knative CLI 资源和路由：

```
$ oc get ConsoleCLIDownload
```

输出示例

```
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
```

输出示例

```
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. 使用 RPM 软件包管理器为 LINUX 安装 KNATIVE CLI

对于 Red Hat Enterprise Linux (RHEL)，您可以使用软件包管理器（如 **yum** 或 **dnf**）将 Knative (**kn**) CLI 作为 RPM 安装。这允许系统自动管理 Knative CLI 版本。例如，如果有新版本可用，使用 **dnf upgrade** 一样的命令升级所有软件包，包括 **kn**。

先决条件

- 您的红帽帐户必须具有有效的 OpenShift Container Platform 订阅。

流程

1. 使用 Red Hat Subscription Manager 注册：

```
# subscription-manager register
```

2. 获取最新的订阅数据：

```
# subscription-manager refresh
```

3. 将订阅附加到注册的系统：

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

1 活跃的 OpenShift Container Platform 订阅的池 ID

4. 启用 Knative (**kn**) CLI 所需的仓库：
   - Linux (x86_64, amd64)

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

   - Linux on IBM zSystems 和 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. 使用软件包管理器将 Knative (**kn**) CLI 作为 RPM 安装：

   yum 命令示例

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

3.3. 使用 RPM 软件包管理器安装的 KNATIVE CLI 锁定版本

您可能需要使用不是最新但处于维护阶段的 Knative (**kn**) CLI 版本。您可以通过锁定 **kn** 的版本来达到此目的，这会阻止它被升级。

流程

1. 为 DNF 软件包管理器安装 **versionlock** 插件：

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

2. 运行以下命令来锁定 **kn** 版本：

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

   此命令会将 **kn** 锁为基于 Knative 1.7 的版本，它在 OpenShift Serverless 1.29 发布时处于维护阶段。

3.4. 使用锁定版本升级 KNATIVE CLI

您可以手动升级之前已经锁定的 Knative (**kn**) CLI 版本。

流程

1. 检查升级的软件包是否可用：

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

2. 删除 **kn** 的版本锁定：

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

1. 将 **kn** 锁定到新版本：

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

+ 本例使用基于 Knative 1.8 的 **kn** 版本。

1. 升级到新的可用版本：

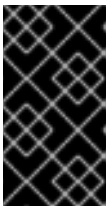
```
# dnf install --upgrade openshift-serverless-clients
```

3.5. 为 LINUX 安装 KNATIVE CLI

如果您使用没有 RPM 或者另一个软件包管理器的 Linux 发行版本，您可以将 Knative (**kn**) CLI 安装为二进制文件。要做到这一点，您必须下载并解包一个 **tar.gz** 存档，并将二进制文件添加到 **PATH** 的目录中。

先决条件

- 如果您不使用 RHEL 或 Fedora，请确保将 **libc** 安装在库路径的目录中。



重要

如果 **libc** 不可用，您在运行 CLI 命令时可能会看到以下错误：

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

流程

1. 下载相关的 Knative (**kn**) CLI **tar.gz** 存档：
   - Linux (x86_64, amd64)
   - Linux on IBM zSystems 和 IBM® LinuxONE (s390x)
   - Linux on IBM Power (ppc64le)

   您还可以通过进入到 Serverless 客户端下载镜像 中的相应目录来下载任何 **kn** 版本。
2. 解包存档：

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

3. 将 **kn** 二进制文件移到 **PATH** 中的目录中。
4. 运行以下命令可以查看 **PATH** 的值：

```
$ echo $PATH
```

3.6. 为 MACOS 安装 KNATIVE CLI

如果使用 macOS，您可以将 Knative (**kn**) CLI 安装为二进制文件。要做到这一点，您必须下载并解包一个 **tar.gz** 存档，并将二进制文件添加到 **PATH** 的目录中。

流程

1. 下载 Knative (**kn**) CLI **tar.gz** 存档。
   您还可以通过进入到 Serverless 客户端下载镜像 中的相应目录来下载任何 **kn** 版本。
2. 解包并提取存档。
3. 将 **kn** 二进制文件移到 **PATH** 中的目录中。
4. 要查看 **PATH**，打开终端窗口并运行：

```
$ echo $PATH
```

3.7. 为 WINDOWS 安装 KNATIVE CLI

如果使用 Windows，您可以将 Knative (**kn**) CLI 安装为二进制文件。要做到这一点，您必须下载并解包 ZIP 存档，并将二进制文件添加到 **PATH** 的目录中。

流程

1. 下载 Knative (**kn**) CLI ZIP 存档。
   您还可以通过进入到 Serverless 客户端下载镜像 中的相应目录来下载任何 **kn** 版本。
2. 使用 ZIP 程序解压存档。
3. 将 **kn** 二进制文件移到 **PATH** 中的目录中。
4. 要查看您的 **PATH**，请打开命令窗口并运行以下命令：

```
C:\> path
```

第 4 章 安装 KNATIVE SERVING

安装 Knative Serving 可让您在集群中创建 Knative 服务和功能。它还允许您为应用程序使用自动扩展和网络选项等其他功能。

安装 OpenShift Serverless Operator 后，您可以使用默认设置安装 Knative Serving，或者在 **KnativeServing** 自定义资源 (CR) 中配置更高级的设置。如需有关 **KnativeServing** CR 的配置选项的更多信息，请参阅全局配置。

重要

如果要在 OpenShift Serverless 中使用 Red Hat OpenShift distributed tracing，则必须在安装 Knative Serving 前安装和配置 Red Hat OpenShift distributed tracing。

4.1. 使用 WEB 控制台安装 KNATIVE SERVING

安装 OpenShift Serverless Operator 后，使用 OpenShift Container Platform Web 控制台安装 Knative Serving。您可以使用默认设置安装 Knative Serving，或者在 **KnativeServing** 自定义资源 (CR) 中配置更高级的设置。

先决条件

- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 已登陆到 OpenShift Container Platform Web 控制台。
- 已安装 OpenShift Serverless Operator。

流程

1. 在 OpenShift Container Platform web 控制台的 **Administrator** 视角中，进入 **Operators → Installed Operators**。
2. 检查页面顶部的 **Project** 下拉菜单是否已设置为 **Project: knative-serving**。
3. 点 OpenShift Serverless Operator 的 **Provided APIs** 列表中的 **Knative Serving** 来进入 **Knative Serving** 选项卡。
4. 点 **Create Knative Serving**。
5. 在 **Create Knative Serving** 页中，您可以使用默认设置安装 Knative Serving。点 **Create**。
   您还可以使用提供的表单或编辑 YAML 来修改 **KnativeServing** 对象来修改 Knative Serving 安装的设置。
   - 建议您在不需要完全控制 **KnativeServing** 对象创建的简单配置中使用该表单。
   - 对于更复杂的配置，建议编辑 YAML，这可以完全控制 **KnativeServing** 对象的创建。您可以通过点 **Create Knative Serving** 页右上角的 **edit YAML** 链接来访问 YAML。
     完成表单后，或者完成对 YAML 的修改后，点 **Create**。

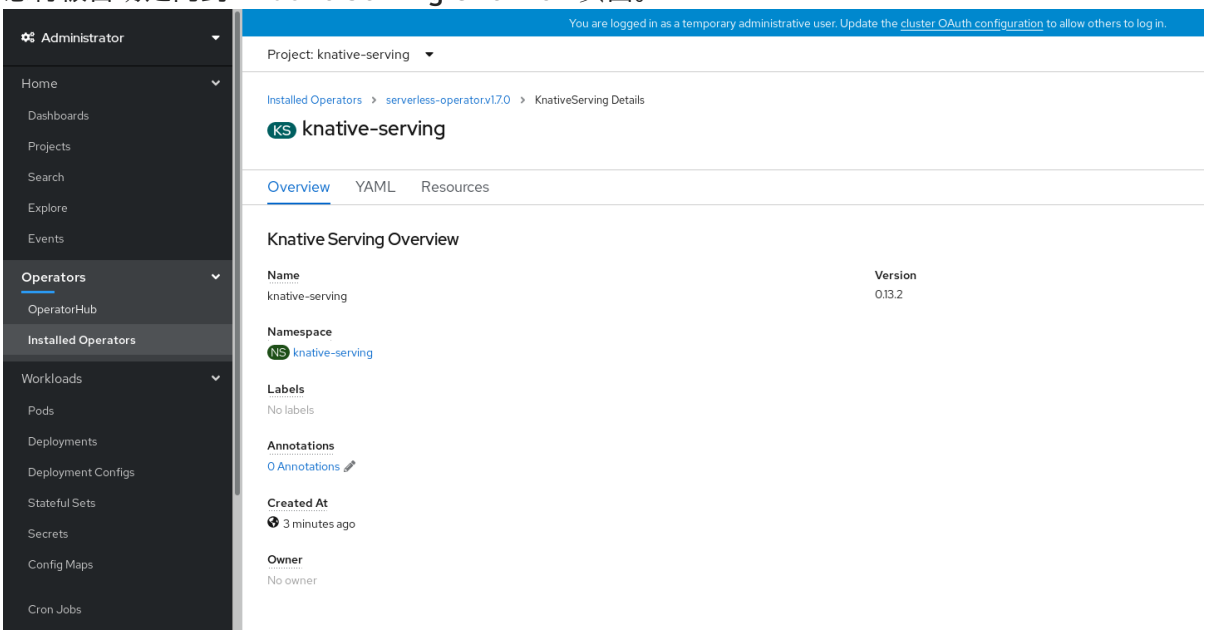
     注意

     如需有关 KnativeServing 自定义资源定义的配置选项的更多信息，请参阅 *高级安装配置选项*。

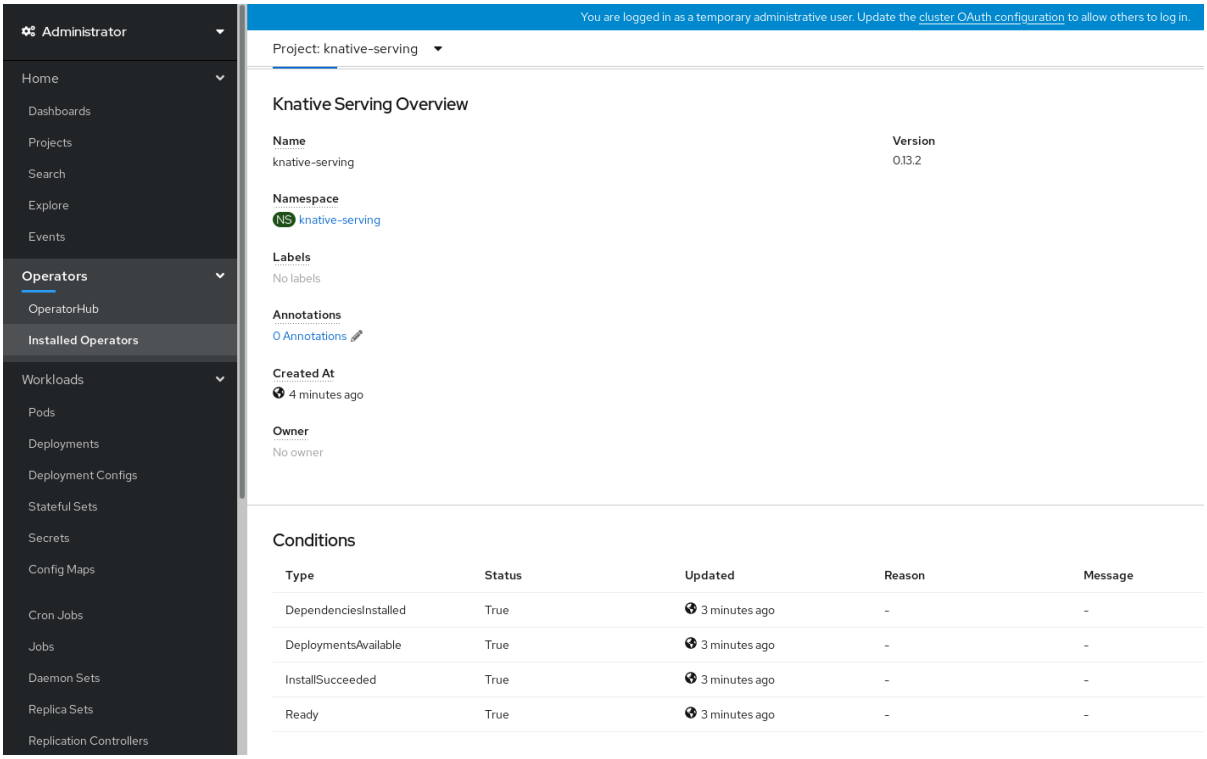
6. 安装 Knative Serving 后，会创建 **KnativeServing** 对象，并自动定向到 **Knative Serving** 选项卡。您可以在资源列表中看到 **knative-serving** 自定义资源。

验证

1. 在 **Knative Serving** 选项卡中点 **knative-serving** 自定义资源。
2. 您将被自动定向到 **Knative Serving Overview** 页面。



3. 向下滚动查看**条件**列表。
4. 您应该看到一个状况为 **True** 的条件列表，如示例镜像所示。



注意

创建 Knative Serving 资源可能需要几秒钟时间。您可以在 **Resources** 选项卡中查看其状态。

5. 如果条件状态为 **Unknown** 或 **False**，请等待几分钟，然后在确认已创建资源后再重新检查。

4.2. 使用 YAML 安装 KNATIVE SERVING

安装 OpenShift Serverless Operator 后，您可以使用默认设置安装 Knative Serving，或者在 **KnativeServing** 自定义资源 (CR) 中配置更高级的设置。您可以使用 YAML 文件和 **oc** CLI 安装 Knative Serving。

先决条件

- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 已安装 OpenShift Serverless Operator。
- 安装 OpenShift CLI (**oc**) 。

流程

1. 创建名为 **serving.yaml** 的文件并将以下示例 YAML 复制到其中：

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeServing
metadata:
  name: knative-serving
  namespace: knative-serving
```

2. 应用 **service.yaml** 文件：

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

验证

1. 使用以下命令校验安装是否完成：

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

输出示例

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

注意

创建 Knative Serving 资源可能需要几秒钟时间。

如果条件状态为 **Unknown** 或 **False**，请等待几分钟，然后在确认已创建资源后再重新检查。

2. 检查是否已创建 Knative Serving 资源：

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

输出示例

```
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. 检查所需的网络组件是否已安装到自动创建的 **knative-serving-ingress** 命名空间：

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

输出示例

```
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. 其他资源

- Kourier 和 Istio ingresses

4.4. 后续步骤

- 如果要使用 Knative 事件驱动的架构，您可以安装 Knative Eventing。

第 5 章 安装 KNATIVE EVENTING

要在集群上使用事件驱动的构架，请安装 Knative Eventing。您可以创建 Knative 组件，如事件源、代理和频道，然后使用它们向应用程序或外部系统发送事件。

安装 OpenShift Serverless Operator 后，您可以使用默认设置安装 Knative Eventing，或者在 **KnativeEventing** 自定义资源 (CR) 中配置更高级的设置。有关 **KnativeEventing** CR 的配置选项的更多信息，请参阅全局配置。

重要

如果要在 OpenShift Serverless 中使用 Red Hat OpenShift distributed tracing，则必须在安装 Knative Eventing 前安装和配置 Red Hat OpenShift distributed tracing。

5.1. 使用 WEB 控制台安装 KNATIVE EVENTING

安装 OpenShift Serverless Operator 后，使用 OpenShift Container Platform Web 控制台安装 Knative Eventing。您可以使用默认设置安装 Knative Eventing，或者在 **KnativeEventing** 自定义资源 (CR) 中配置更高级的设置。

先决条件

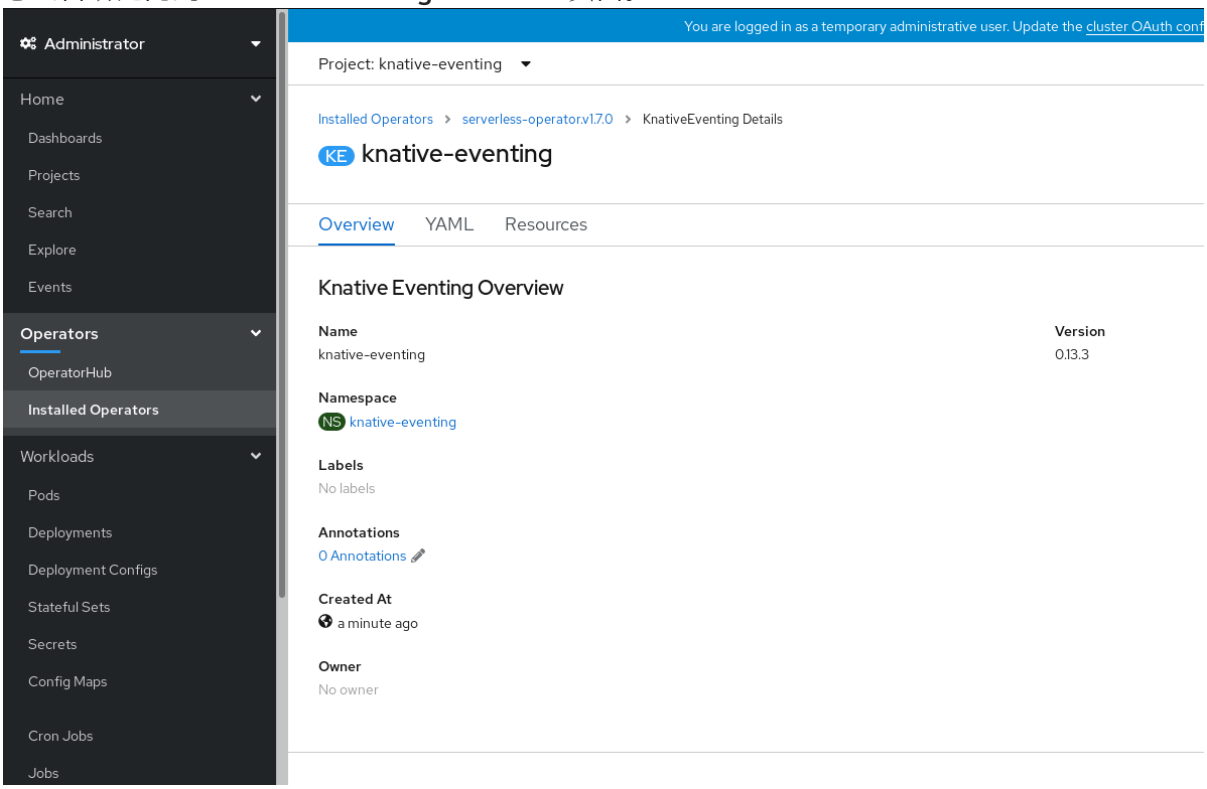
- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 已登陆到 OpenShift Container Platform Web 控制台。
- 已安装 OpenShift Serverless Operator。

流程

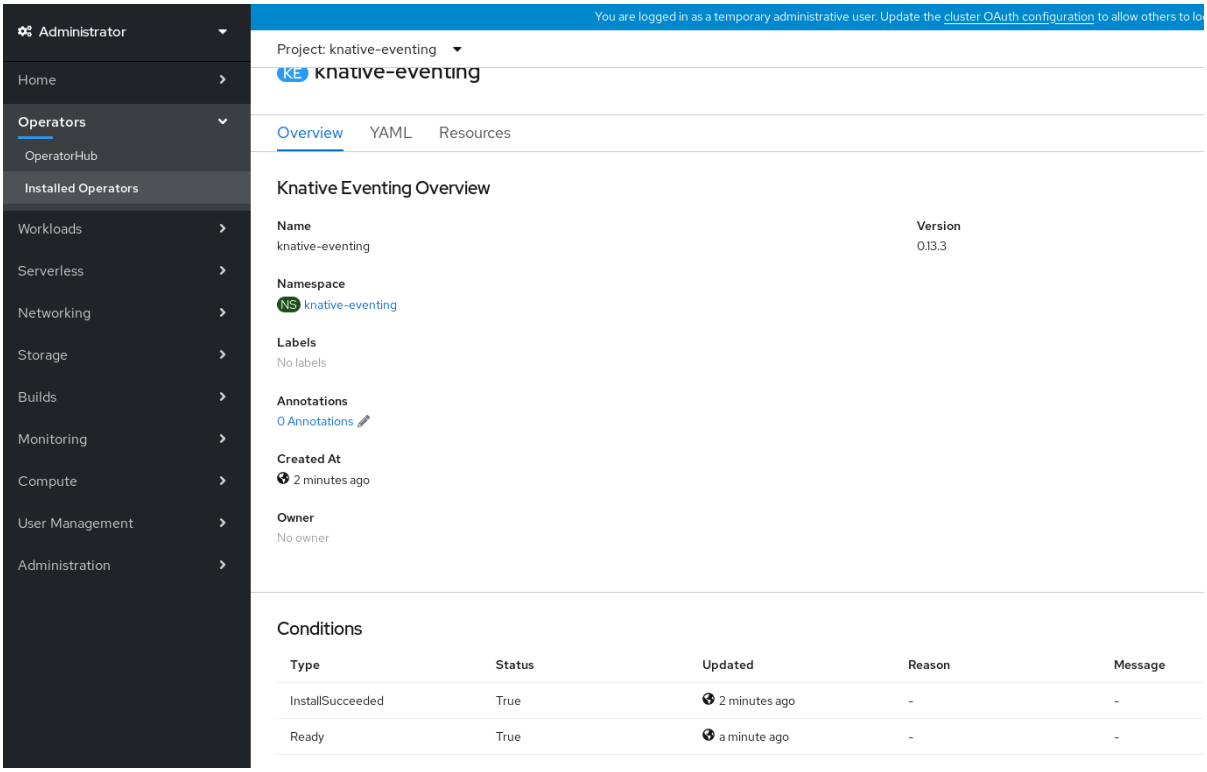
1. 在 OpenShift Container Platform web 控制台的 **Administrator** 视角中，进入 **Operators → Installed Operators**。
2. 检查页面顶部的 **Project** 下拉菜单是否已设置为 **Project: knative-eventing**。
3. 点 OpenShift Serverless Operator 的 **Provided APIs** 列表中的 **Knative Eventing** 来进入 **Knative Eventing** 选项卡。
4. 点 **Create Knative Eventing**。
5. 在 **Create Knative Eventing** 页面中，您可以使用提供的表单或编辑 YAML 文件来配置 **KnativeEventing** 对象。
   - 在不需要完全控制 **KnativeEventing** 对象创建的简单配置中使用表单。
6. 点 **Create**。
   - 编辑 YAML 文件，以获取更复杂的配置，这些配置需要完全控制 **KnativeEventing** 对象创建。要访问 YAML 编辑器，请点击 **Create Knative Eventing** 页面上的 **edit YAML**。
7. 安装 Knative Eventing 后，会创建 **KnativeEventing** 对象，并自动定向到 **Knative Eventing** 选项卡。您可以在资源列表中看到 **knative-eventing** 自定义资源。

验证

1. 点 Knative Eventing 选项卡中的 **knative-eventing** 自定义资源。
2. 您会自动定向到 **Knative Eventing Overview** 页面。



3. 向下滚动查看**条件**列表。
4. 您应该看到一个状况为 **True** 的条件列表，如示例镜像所示。



注意

创建 Knative Eventing 资源可能需要几秒钟时间。您可以在 **Resources** 选项卡中查看其状态。

5. 如果条件状态为 **Unknown** 或 **False**，请等待几分钟，然后在确认已创建资源后再重新检查。

5.2. 使用 YAML 安装 KNATIVE EVENTING

安装 OpenShift Serverless Operator 后，您可以使用默认设置安装 Knative Eventing，或者在 **KnativeEventing** 自定义资源 (CR) 中配置更高级的设置。您可以使用 YAML 文件和 **oc** CLI 安装 Knative Eventing。

先决条件

- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 已安装 OpenShift Serverless Operator。
- 安装 OpenShift CLI (**oc**) 。

流程

1. 创建名为 **eventing.yaml** 的文件。
2. 将以下示例 YAML 复制到 **eventing.yaml**中：

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

3. 可选。根据您的 Knative Eventing 部署，对 YAML 进行相应的更改。
4. 输入以下内容来应用 **eventing.yaml** 文件：

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

验证

1. 输入以下命令验证安装是否完成，并观察输出结果：

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

输出示例

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

注意

创建 Knative Eventing 资源可能需要几秒钟时间。

2. 如果条件状态为 **Unknown** 或 **False**，请等待几分钟，然后在确认已创建资源后再重新检查。

3. 使用以下命令检查是否已创建 Knative Eventing 资源：

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

输出示例

```
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. 为 APACHE KAFKA 安装 KNATIVE 代理

Apache Kafka 的 Knative 代理实现提供了集成选项，供您在 OpenShift Serverless 中使用支持的 Apache Kafka 消息流平台版本。如果安装了 **KnativeKafka** 自定义资源，则 OpenShift Serverless 安装中提供了 Apache Kafka 功能的 Knative 代理。

先决条件

- 在集群中安装了 OpenShift Serverless Operator 和 Knative Eventing。
- 您可以访问 Red Hat AMQ Streams 集群。
- 如果要使用验证步骤，请安装 OpenShift CLI (**oc**) 。
- 在 OpenShift Container Platform 上具有集群管理员权限，或者对 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated 有集群或专用管理员权限。
- 已登陆到 OpenShift Container Platform Web 控制台。

流程

1. 在 **Administrator** 视角中，进入 **Operators → Installed Operators**。
2. 检查页面顶部的 **Project** 下拉菜单是否已设置为 **Project: knative-eventing**。
3. 在 OpenShift Serverless Operator 的 **Provided APIs** 列表中，找到 **Knative Kafka** 复选框并点 **Create Instance**。
4. 在 **Create Knative Kafka** 页面中配置 **KnativeKafka** 对象。

重要

要在集群中使用 Kafka 频道、源、代理或 sink，您需要将要使用的属性的 **enabled** 选项设置为 **true**。这些交换机默认设置为 **false**。另外，要使用 Kafka 频道、代理或接收器，您必须指定 bootstrap 服务器。

- 在不需要完全控制 **KnativeKafka** 对象创建的简单配置中使用表单。

- 编辑 YAML 以获取更复杂的配置，这些配置需要完全控制 **KnativeKafka** 对象创建。您可以通过点 **Create Knative Kafka** 页面中的 **Edit YAML** 链接来访问 YAML。

KnativeKafka 自定义资源示例

```
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. 让开发人员在集群中使用 **KafkaChannel** 频道类型。
2. 以逗号分隔的 AMQ Streams 集群中的 bootstrap 服务器列表。
3. 让开发人员在集群中使用 **KafkaSource** 事件源类型。
4. 让开发人员在集群中使用 Apache Kafka 的 Knative 代理实现。
5. 来自 Red Hat AMQ Streams 集群的 bootstrap 服务器的逗号分隔列表。
6. 定义 Kafka 主题分区的数量，由 **Broker** 对象支持。默认值为 **10**。
7. 定义 Kafka 主题的复制因素，由 **Broker** 对象支持。默认值为 **3**。**replicationFactor** 值必须小于或等于 Red Hat AMQ Streams 集群的节点数量。
8. 让开发人员在集群中使用 Kafka sink。
9. 定义 Kafka 数据平面的日志级别。允许的值有 **TRACE**,**DEBUG**,**INFO**,**WARN** 和 **ERROR**。默认值为 **INFO**。

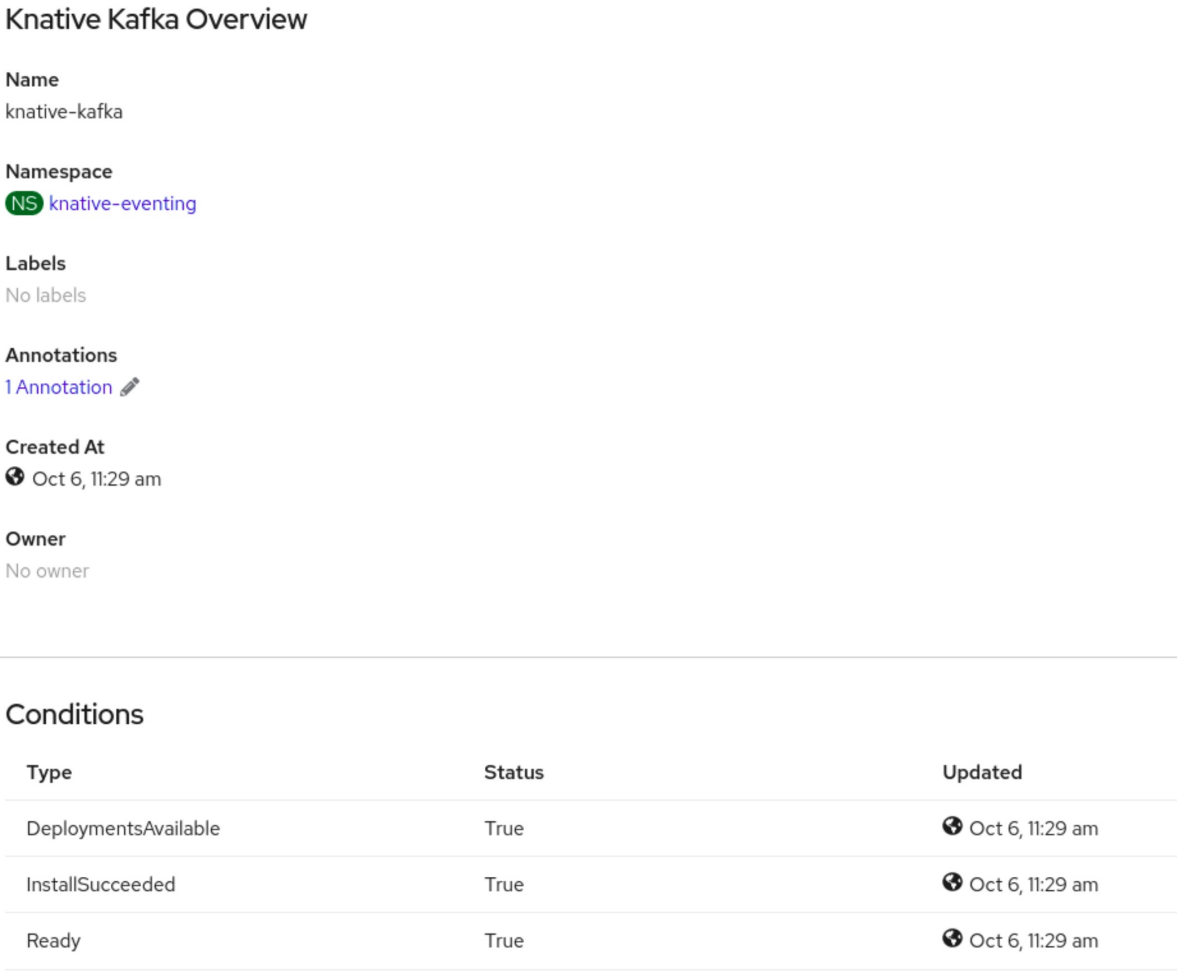
警告

不要在生产环境中使用 **DEBUG** 或 **TRACE** 作为日志级别。这些日志记录级别的输出非常详细，可能会降低性能。

5. 完成 Kafka 的任何可选配置后，点 **Create**。您会自动定向到 **Knative Kafka** 标签页，其中 **knative-kafka** 在资源列表中。

验证

1. 点 **Knative Kafka** 选项卡中的 **knative-kafka** 资源。您会自动定向到 **Knative Kafka Overview** 页面。
2. 查看资源的 **Conditions** 列表，并确认其状态为 **True**。



如果条件的状态为 **Unknown** 或 **False**，请等待几分钟刷新页面。

3. 检查是否已创建 Apache Kafka 资源的 Knative 代理：

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

输出示例

```
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. 后续步骤

- 如果要使用 Knative 服务，可以安装 Knative Serving。

第 6 章 为 APACHE KAFKA 配置 KNATIVE 代理

Apache Kafka 的 Knative 代理实现提供了集成选项，供您在 OpenShift Serverless 中使用支持的 Apache Kafka 消息流平台版本。Kafka 为事件源、频道、代理和事件 sink 功能提供选项。

除了作为 OpenShift Serverless 核心安装一部分的 Knative Eventing 组件外，还可以安装 **KnativeKafka** 自定义资源(CR)：

- 集群管理员，适用于 OpenShift Container Platform
- 集群或专用管理员，用于 Red Hat OpenShift Service on AWS 或 OpenShift Dedicated。

KnativeKafka CR 为用户提供其他选项，例如：

- Kafka 源
- Kafka 频道
- Kafka 代理
- Kafka 接收器

第 7 章 为 APACHE KAFKA 为 KNATIVE 配置 KUBE-RBAC-PROXY

kube-rbac-proxy 组件为 Apache Kafka 提供 Knative 的内部身份验证和授权功能。

7.1. 为 APACHE KAFKA 为 KNATIVE 配置 KUBE-RBAC-PROXY 资源

您可以使用 OpenShift Serverless Operator CR 全局覆盖 **kube-rbac-proxy** 容器的资源分配。

注意

您还可以覆盖特定部署的资源分配。

以下配置设置 Knative Kafka **kube-rbac-proxy** 最小值和最大 CPU 和内存分配：

KnativeKafka CR 示例

```
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. 设置最小 CPU 分配。
2. 设置最小 RAM 分配。
3. 设置最大 CPU 分配。
4. 设置最大 RAM 分配。

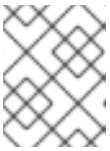
第 8 章 配置 OPENSHIFT SERVERLESS FUNCTIONS

为改进应用程序代码部署的过程，您可以使用 OpenShift Serverless 部署无状态、事件驱动的功能，作为 OpenShift Container Platform 上的 Knative 服务。如果要开发功能，您必须完成设置步骤。

8.1. 先决条件

要在集群中启用 OpenShift Serverless 功能，您必须完成以下步骤：

- 在集群中安装了 OpenShift Serverless Operator 和 Knative Serving。



注意

功能部署为 Knative 服务。如果要将事件驱动的架构与您的功能搭配使用，还必须安装 Knative Eventing。

- 已安装 **oc** CLI。
- 已安装 Knative (**kn**) CLI。安装 Knative CLI 可让您使用 **kn func** 命令来创建和管理功能。
- 已安装 Docker Container Engine 或 Podman 版本 3.4.7 或更高版本。
- 您可以访问可用的镜像 registry，如 OpenShift Container Registry。
- 如果您使用 Quay.io 作为镜像 registry，您必须确保存储库不是私有的，或者按照 OpenShift Container Platform 文档中有关允许 Pod 引用其他安全 registry 中的镜像的内容进行操作。
- 如果使用 OpenShift Container Registry，集群管理员必须公开 registry。

8.2. 设置 PODMAN

要使用高级容器管理功能，您可能需要将 Podman 与 OpenShift Serverless 功能一起使用。要做到这一点，您需要启动 Podman 服务并配置 Knative (**kn**) CLI 来连接它。

流程

1. 在 **${XDG_RUNTIME_DIR}/podman/podman.sock** 的 UNIX 套接字上启动提供 Docker API 的 Podman 服务：

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

注意

在大多数系统中，此套接字位于 **/run/user/$ (id -u) /podman/podman.sock**。

2. 建立用于构建功能的环境变量：

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

3. 在函数项目目录中使用 **-v** 标记运行构建命令，以查看详细的输出。您应该看到到本地 UNIX 套接字的连接：

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

8.3. 在 MACOS 中设置 PODMAN

要使用高级容器管理功能，您可能需要将 Podman 与 OpenShift Serverless 功能一起使用。要在 macOS 中这样做，您需要启动 Podman 机器并配置 Knative (**kn**) CLI 来连接它。

流程

1. 创建 Podman 机器：

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

2. 启动 Podman 机器，该机器在 UNIX 套接字上提供 Docker API：

```
$ 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
```

注意

在大多数 macOS 系统上，此套接字位于 **/Users/myuser/.local/share/containers/podman/machine/podman-machine-default/podman.sock**。

3. 建立用于构建功能的环境变量：

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

4. 在函数项目目录中使用 **-v** 标记运行构建命令，以查看详细的输出。您应该看到到本地 UNIX 套接字的连接：

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

8.4. 后续步骤

- 有关 Docker Container Engine 或 Podman 的更多信息，请参阅 容器构建工具选项。

- 请参阅开始使用功能。

第 9 章 SERVERLESS 升级

在不跳过发行版本的情况下，OpenShift Serverless 应该会被升级。本节演示了如何解决升级的问题。

9.1. SERVERLESS OPERATOR 维护发行版本升级

9.1.1. 最新和维护版本

从 OpenShift Serverless 1.29 开始，不同的产品版本如下：

- 最新版本可通过 **stable** 频道获得。
- 维护版本可以通过其基于版本的频道获得，如 **stable-1.29**。

注意

维护版本是最新版本前的发行版本。例如，如果 **stable** 频道包含 1.30，则维护版本将在 **stable-1.29** 频道中提供。

通过使用基于版本的频道，您可以保留在特定的 **x.y** 流中。另外，它还可防止升级到产品的最新版本，该版本可能包含破坏更改。

要切换到维护版本，请将订阅对象 YAML 文件中的 channel 参数从 **stable** 更新到对应的基于版本的频道，如 **stable-1.29**。

9.1.2. 维护版本的补丁和修补程序

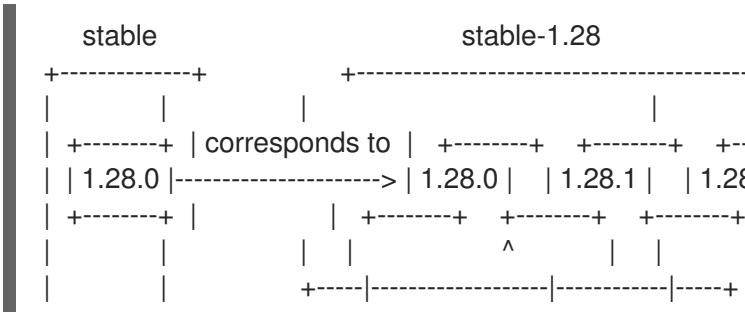
与稳定的版本一样，维护版本会受到补丁和热修补代码，有助于保持部署最新及关键错误和安全修复程序。

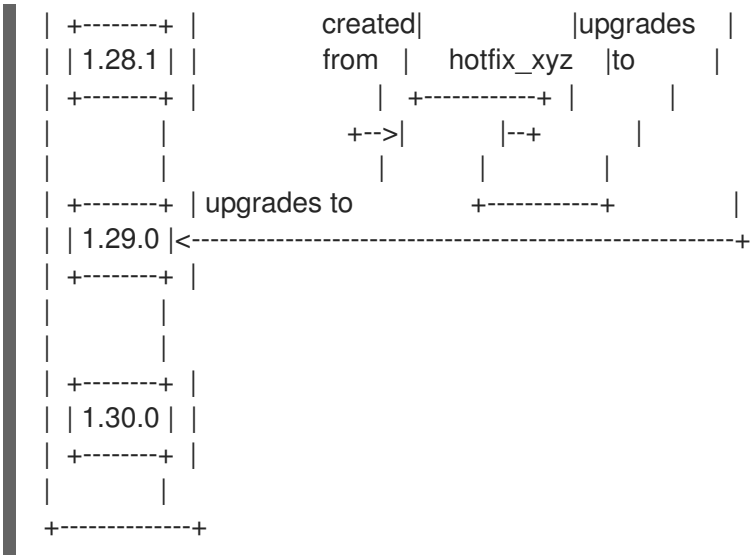
- 补丁是作为 z-releases 分发的更新，例如，OpenShift Serverless 1.29.1 是提供自版本 1.29.0 以来所做的更新。
- 热修补代码是需要零停机时间且直接在生产中使用的修复。它们与常规的更新不同，它们会升级客户部署的版本，而不是最新发布的版本。
  所有客户可能不会立即提供热修补代码。但是，修补程序引入的变化通常作为以后的版本提供给所有客户。

  当有与您的部署相关的热修复程序时，您将获得 hotfix CatalogSource 来更新您的订阅并获取热修补代码。

  在新的 Operator 发行版本可用后，还可以升级使用热修补代码部署的 Operator。要使用最新的 GA 版本，请将订阅修改为使用公共 CatalogSource 而不是热修补代码。

下图说明了补丁和热修补方式：





9.1.3. 维护版本的升级路径

如果您使用基于版本的频道，则始终可以升级到频道中的最新版本，或头。例如，您可以在 **stable-1.29** 频道中从 1.29.0 升级到 1.29.2。

另外，您还可以从频道头升级到下一个 **x.y** 版本。例如，如果 1.29.2 是 **stable-1.29** 频道中的头，您可以从 1.29.2 升级到 1.30。此类跨通道更新不会自动完成，管理员需要通过更新订阅来手动切换频道。

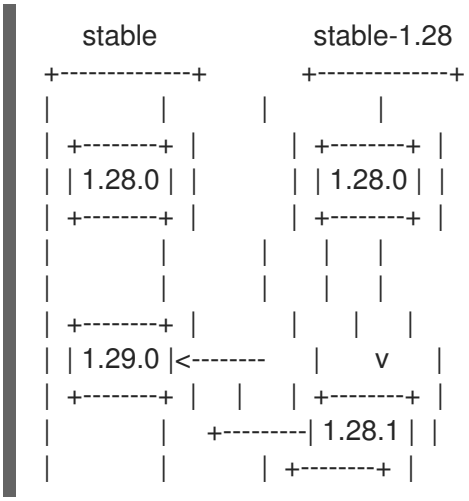
9.1.4. 升级示例

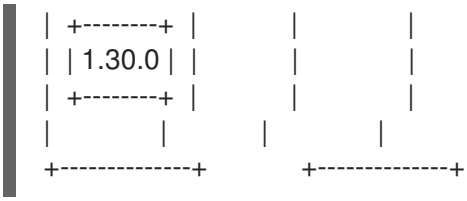
9.1.4.1. 场景 1

在这种情况下，以下情况是 true：

- 频道是 **stable-1.28**
- 您切换到 **stable** 频道
- 当前安装的版本为 1.28.0
- 1.29.0 在 1.28.1 之前发布
- 1.30.0 是 **stable** 频道的头

在这种情况下，在 **stable -1.28** 上的 1.28.0 到 1.29.0 的升级路径是 1.28.0 到 1.28.1 到 1.29.0。



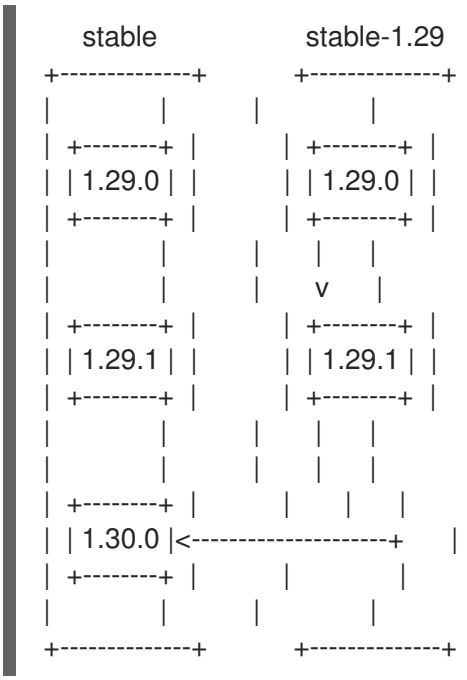


9.1.4.2. 场景 2

在这种情况下，以下情况是 true：

- 频道是 **stable-1.29**
- 当前安装的版本为 1.29.0
- 1.29.1 在 **1.30.0** 之前被发布到 **stable -1.29** 和 **stable** 频道

在这种情况下，在 **stable -1.29** 到 1.30.0 的升级路径是 1.29.0 到 1.29.1 到 1.30.0。

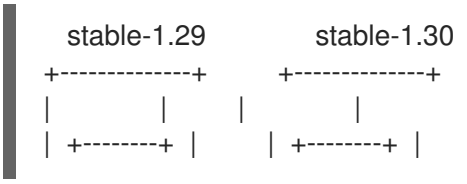


9.1.4.3. 场景 3

在这种情况下，以下情况是 true：

- 频道是 **stable-1.29**
- 您切换到 **stable-1.30** 频道
- 当前安装的版本为 1.29.1
- 1.29.1 是 **stable-1.29** 频道的头

在这种情况下，在 **stable -1.30** 上从 1.29.1 到 1.30.0 的升级路径为 1.29.1 到 1.30.0。



```
| | 1.29.0 | |    ------> | 1.30.0 | |
| +--------+ |    |    | +--------+ |
|            |    |    |            |
|            |    |    |            |
| +--------+ |    |    |            |
| | 1.29.1 |-------+    |            |
| +--------+ |       |    |            |
|            |       |    |            |
+--------------+       +--------------+
```

9.2. 解决 OPENSHIFT SERVERLESS OPERATOR 升级失败

升级 OpenShift Serverless Operator 时可能会遇到错误，例如执行手动卸载和重新安装时。如果您遇到错误，您必须手动重新安装 OpenShift Serverless Operator。

流程

1. 通过在 OpenShift Serverless 发行注记中搜索最初安装的 OpenShift Serverless Operator 版本。例如，尝试升级过程中的错误消息可能包含以下字符串：

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

   在本例中，KnativeServing **MAJOR.MINOR** 版本为 **1.5**，它已在 OpenShift Serverless 1.26 的发行注记中介绍：*OpenShift Serverless 现在使用 Knative Serving 1.5*。

2. 卸载 OpenShift Serverless Operator 及其所有安装计划。

3. 手动安装您在第一步中发现的 OpenShift Serverless Operator 版本。要安装，首先创建一个 **serverless-subscription.yaml** 文件，如下例所示：

   ```
   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. 然后，运行以下命令来安装订阅：

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

5. 在出现升级时手动批准升级安装计划进行升级。

其他资源

- OpenShift Serverless 发行注记
- 使用 Web 控制台从集群中删除 Operator

- 通过 Web 控制台安装 OpenShift Serverless Operator