Red Hat OpenShift Serverless 1.28

关于 Serverless

OpenShift Serverless 简介

Last Updated: 2023-09-19

Red Hat OpenShift Serverless 1.28 关于 Serverless

OpenShift Serverless 简介

法律通告

摘要

本文档概述了 OpenShift Serverless 的功能，包括功能、Serving 和 Eventing。它还包括发行注记以及如何获得支持。

目录

第 1 章 发行注记

注意

如需有关 OpenShift Serverless 生命周期和支持的平台的更多信息，请参阅 平台生命周期政策。

发行注记包含有关新的和已弃用的功能、破坏更改以及已知问题的信息。以下发行注记适用于 OpenShift Container Platform 的最新 OpenShift Serverless 版本。

如需了解 OpenShift Serverless 功能概述，请参阅 关于 OpenShift Serverless。

注意

OpenShift Serverless 基于开源的 Knative 项目。

有关最新 Knative 组件发行版本的详情，请参阅 Knative 博客。

1.1. 关于 API 版本

API 版本是 OpenShift Serverless 中特定函数和自定义资源的开发状态的重要因素。在没有使用正确 API 版本的集群中创建资源可能会导致部署出现问题。

OpenShift Serverless Operator 会自动升级使用已弃用 API 版本的旧资源以使用最新版本。例如，如果您在集群中创建了使用旧版本的 **ApiServerSource** API（如 **v1beta1** ）的资源，OpenShift Serverless Operator 会在可用时自动更新这些资源以使用 API 的 **v1** 版本，并弃用 **v1beta1** 版本。

弃用后，可能会在任何即将发布的发行版本中删除旧版本的 API。使用已弃用的 API 版本不会导致资源失败。但是，如果您尝试使用已删除的 API 版本，则会导致资源失败。确保您的清单已更新为使用最新版本以避免出现问题。

1.2. 正式发布（GA）和技术预览（TP）功能

正式发布（GA）的功能被完全支持，并适用于生产环境。技术预览功能为实验性功能，不适用于生产环境。有关 TP 功能的更多信息，请参阅红帽客户门户网站中的技术支持范围。

下表提供了有关哪些 OpenShift Serverless 功能是 GA 以及 TP 的信息：

表 1.1. OpenShift Container Platform 和 OpenShift Dedicated 正式发布和技术预览功能

功能	1.26	1.27	1.28
kn func	GA	GA	GA
Quarkus 功能	GA	GA	GA
Node.js 功能	TP	TP	GA
TypeScript 功能	TP	TP	GA
Python 功能	-	-	TP

功能	1.26	1.27	1.28
Service Mesh mTLS	GA	GA	GA
emptyDir 卷	GA	GA	GA
HTTPS 重定向	GA	GA	GA
Kafka 代理	GA	GA	GA
Kafka 接收器	GA	GA	GA
init 容器支持 Knative 服务	GA	GA	GA
Knative 服务的 PVC 支持	GA	GA	GA
TLS 用于内部流量	TP	TP	TP
命名空间范围的代理	-	TP	TP
多容器支持	-	-	TP

表 1.2. Red Hat OpenShift Service on AWS 正式发布和技术预览功能

功能	1.26	1.27	1.28
kn func	GA	GA	GA
Service Mesh mTLS	GA	GA	GA
emptyDir 卷	GA	GA	GA
HTTPS 重定向	GA	GA	GA
Kafka 代理	GA	GA	GA
Kafka 接收器	TP	TP	TP
init 容器支持 Knative 服务	GA	GA	GA
Knative 服务的 PVC 支持	GA	GA	GA
TLS 用于内部流量	TP	TP	TP
命名空间范围的代理	-	TP	TP

功能	1.26	1.27	1.28
多容器支持	-	-	TP

1.3. 弃用和删除的功能

一些在以前发行本中正式发布 (GA) 或技术预览 (TP) 的功能已被弃用或删除。弃用的功能仍然包含在 OpenShift Serverless 中，并且仍然被支持。但是，弃用的功能可能会在以后的发行版本中被删除，且不建议在新的部署中使用。

有关 OpenShift Serverless 中已弃用并删除的主要功能的最新列表，请参考下表：

表 1.3. 已弃用和删除 OpenShift Container Platform 和 OpenShift Dedicated 的功能

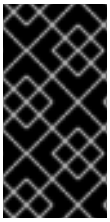
功能	1.20	1.21	1.22 到 1.26	1.27	1.28
KafkaBinding API	已弃用	已弃用	删除	删除	删除
kn func emit (**kn func invoke** 在 1.21+ 中)	已弃用	删除	删除	删除	删除
Serving 和 Eventing **v1alpha1** API	-	-	-	已弃用	已弃用
enable-secret-informer-filtering 注解	-	-	-	-	已弃用

表 1.4. 已弃用和删除 Red Hat OpenShift Service on AWS 的功能

功能	1.23 到 1.26	1.27	1.28
KafkaBinding API	删除	删除	删除
kn func emit (**kn func invoke** 在 1.21+ 中)	删除	删除	删除
Serving 和 Eventing **v1alpha1** API	-	已弃用	已弃用

1.4. RED HAT OPENSHIFT SERVERLESS 1.28 发行注记

OpenShift Serverless 1.28 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。



重要

OpenShift Container Platform 4.13 基于 Red Hat Enterprise Linux (RHEL) 9.2。RHEL 9.2 尚未提交用于联邦信息处理标准(FIPS)验证。虽然红帽无法提交到特定的时间段，但我们希望获得 RHEL 9.0 和 RHEL 9.2 模块的 FIPS 验证，之后甚至是 RHEL 9.x 的次版本。有关更新的信息，请参阅 Compliance Activities 和 Government Standards 知识库文章 。

1.4.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.7。
- OpenShift Serverless 现在使用 Knative Eventing 1.7。
- OpenShift Serverless 现在使用 Kourier 1.7。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.7。
- OpenShift Serverless 现在为 Apache Kafka 1.7 使用 Knative 代理实现。
- **kn func** CLI 插件现在使用 **func** 1.9.1 版本。
- Node.js 和 TypeScript 运行时现在正式发布 (GA)。
- OpenShift Serverless 功能的 Python 运行时现在作为技术预览提供。
- Knative Serving 的多容器支持现在作为技术预览提供。此功能允许您使用单个 Knative 服务来部署多容器 pod。
- 在 OpenShift Serverless 1.29 或更高版本中，Knative Eventing 的以下组件将从两个 pod 缩减为一个：
  - **imc-controller**
  - **imc-dispatcher**
  - **mt-broker-controller**
  - **mt-broker-filter**
  - **mt-broker-ingress**
- Serving CR 的 **serverless.openshift.io/enable-secret-informer-filtering** 注解现已弃用。该注解仅适用于 Istio，不适用于 Kourier。
  在 OpenShift Serverless 1.28 中，OpenShift Serverless Operator 允许注入 **net-istio** 和 **net-kourier** 的环境变量 **ENABLE_SECRET_INFORMER_FILTERING_BY_CERT_UID**。

  如果启用 secret 过滤，则所有 secret 都需要使用 **networking.internal.knative.dev/certificate-uid: "<id>"**。否则，Knative Serving 不会检测到它们，这会导致失败。您必须标记新的和现有的 secret。

  在以下 OpenShift Serverless 版本中，secret 过滤将默认启用。要防止失败，请预先标记您的 secret。

1.4.2. 已知问题

- 目前，IBM Power、IBM zSystems 和 IBM® LinuxONE 上的 OpenShift Serverless 功能不支持 Python 的运行时。
  Node.js、typetype 和 Quarkus 功能在这些架构中被支持。
- 在 Windows 平台上，因为 **app.sh** 文件权限，无法使用 Source-to-Image 构建器在本地构建、运行或部署 Python 功能。
  要临时解决这个问题，对 Linux 使用 Windows 子系统。

- 在 Red Hat OpenShift Serverless 1.27 被删除前创建的 **KafkaSource** 资源会卡住。要临时解决这个问题，在开始删除 **KafkaSource** 后，从资源中删除终结器。

1.5. RED HAT OPENSHIFT SERVERLESS 1.27 发行注记

OpenShift Serverless 1.27 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

重要

OpenShift Serverless 1.26 是 OpenShift Container Platform 4.12 完全支持的最早版本。OpenShift Serverless 1.25 和更早的版本不会在 OpenShift Container Platform 4.12 上部署。

因此，在将 OpenShift Container Platform 升级到 4.12 之前，首先将 OpenShift Serverless 升级到 1.26 或 1.27。

1.5.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.6。
- OpenShift Serverless 现在使用 Knative Eventing 1.6。
- OpenShift Serverless 现在使用 Kourier 1.6。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.6。
- OpenShift Serverless 现在使用 Knative Kafka 1.6。
- **kn func** CLI 插件现在使用 **func** 1.8.1。
- 命名空间范围的代理现在作为技术预览提供。例如，此类代理可用于实施基于角色的访问控制 (RBAC) 策略。
- **KafkaSink** 现在使用 **CloudEvent** 二进制内容模式。二进制内容模式比结构化模式更高效，因为它使用其正文中的标头而不是 **CloudEvent**。例如，对于 HTTP 协议，它使用 HTTP 标头。
- 现在，您可以使用 OpenShift Container Platform 4.10 及更新的版本中的 OpenShift Route 通过 HTTP/2 协议使用 gRPC 框架。这提高了客户端和服务器间的通信效率和速度。
- Knative Operator Serving 和 Eventings CRD 的 API 版本 **v1alpha1** 在 1.27 中弃用。它将在以后的版本中删除。红帽强烈建议使用 **v1beta1** 版本。这不会影响现有安装，因为在升级 Serverless Operator 时 CRD 会被自动更新。
- 现在默认启用交付超时功能。它允许您指定每个发送的 HTTP 请求的超时时间。这个功能仍是一个技术预览。

1.5.2. 修复的问题

- 在以前的版本中，Knative 服务有时没有处于 **Ready** 状态，报告等待负载均衡器就绪。这个问题已被解决。

1.5.3. 已知问题

- 当集群中存在太多 secret 时，将 OpenShift Serverless 与 Red Hat OpenShift Service Mesh 集成会导致 **net-kourier** pod 在启动时耗尽内存。

- 命名空间范围的代理可能会在用户命名空间中保留 **ClusterRoleBindings**，即使删除了命名空间范围的代理。
  如果发生这种情况，删除用户命名空间中名为 **rbac-proxy-reviews-prom-rb-knative-kafka-broker-data-plane-{{.Namespace}}** 的 **ClusterRoleBinding**。

- 如果您将 **net-istio** 用于 Ingress，并使用 **security.dataPlane.mtls: true** 通过 SMCP 启用 mTLS，Service Mesh 为 ***.local** 主机部署 **DestinationRule**，这代表不允许对 OpenShift Serverless 使用 **DomainMapping**。
  要临时解决这个问题，部署 **PeerAuthentication** 来启用 mTLS，而不是使用 **security.dataPlane.mtls: true**。

1.6. RED HAT OPENSHIFT SERVERLESS 1.26 发行注记

OpenShift Serverless 1.26 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.6.1. 新功能

- 带有 Quarkus 的 OpenShift Serverless 功能现在是 GA。

- OpenShift Serverless 现在使用 Knative Serving 1.5。

- OpenShift Serverless 现在使用 Knative Eventing 1.5。

- OpenShift Serverless 现在使用 Kourier 1.5。

- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.5。

- OpenShift Serverless 现在使用 Knative Kafka 1.5。

- OpenShift Serverless 现在使用 Knative Operator 1.3。

- **kn func** CLI 插件现在使用 **func** 1.8.1。

- 持久性卷声明 (PVC) 现在为 GA。PVC 为 Knative 服务提供持久性数据存储。

- 新的触发器过滤器功能现在作为技术预览提供。它允许用户指定一组过滤器表达式，其中每个表达式都会为每个事件评估为 true 或 false。
  要启用新的触发器过滤器，请在 operator 配置映射中 **KnativeEventing** 类型的部分中添加 **new-trigger-filters: enabled** 条目：

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeEventing
...
...
spec:
  config:
    features:
      new-trigger-filters: enabled
...
```

- Knative Operator 1.3 为 **operator.knative.dev** 添加了 API 的更新 **v1beta1** 版本。

要从 **KnativeServing** 和 **KnativeEventing** 自定义资源配置映射中的 **v1alpha1** 更新至 **v1beta1**，请编辑 **apiVersion** 键：

KnativeServing 自定义资源配置映射示例

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

KnativeEventing 自定义资源配置映射示例

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeEventing
...
```

1.6.2. 修复的问题

- 在以前的版本中，Kafka 代理、Kafka 源和 Kafka sink 禁用联邦信息处理标准 (FIPS) 模式。这个问题已被解决，现在 FIPS 模式可用。

1.6.3. 已知问题

- 如果您将 **net-istio** 用于 Ingress，并使用 **security.dataPlane.mtls: true** 通过 SMCP 启用 mTLS，Service Mesh 为 ***.local** 主机部署 **DestinationRule**，这代表不允许对 OpenShift Serverless 使用 **DomainMapping**。
  要临时解决这个问题，部署 **PeerAuthentication** 来启用 mTLS，而不是使用 **security.dataPlane.mtls: true**。

其他资源

- 有关新触发器过滤器的 Knative 文档

1.7. RED HAT OPENSHIFT SERVERLESS 1.25.0 发行注记

OpenShift Serverless 1.25.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.7.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.4。
- OpenShift Serverless 现在使用 Knative Eventing 1.4。
- OpenShift Serverless 现在使用 Kourier 1.4。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.4。
- OpenShift Serverless 现在使用 Knative Kafka 1.4。
- **kn func** CLI 插件现在使用 **func** 1.7.0。
- 用于创建和部署功能的集成开发环境(IDE)插件现在可用于 Visual Studio Code 和 IntelliJ。

- Knative Kafka 代理现在是 GA。Knative Kafka 代理是 Knative 代理 API 的高性能实现，直接以 Apache Kafka 为目标。
  建议您不要使用 MT-Channel-Broker，而是使用 Knative Kafka 代理。

- Knative Kafka sink 现在为 GA。**KafkaSink** 使用 **CloudEvent**，并将其发送到 Apache Kafka 主题。事件可以在结构化或二进制内容模式中指定。

- 为内部流量启用 TLS 现在作为技术预览提供。

1.7.2. 修复的问题

- 在以前的版本中，如果在存活度探测失败后重启容器，Knative Serving 会出现一个就绪度探测失败。这个问题已被解决。

1.7.3. 已知问题

- Kafka 代理、Kafka 源和 Kafka sink 禁用 Federal Information Processing Standards (FIPS) 模式。

- **SinkBinding** 对象不支持服务的自定义修订名称。

- Knative Serving Controller pod 添加了一个新的 informer 来监视集群中的 secret。informer 在缓存中包含 secret，这会增加控制器 pod 的内存消耗。
  如果 pod 内存不足，您可以通过增加部署的内存限值来解决此问题。

- 如果您将 **net-istio** 用于 Ingress，并使用 **security.dataPlane.mtls: true** 通过 SMCP 启用 mTLS，Service Mesh 为 ***.local** 主机部署 **DestinationRule**，这代表不允许对 OpenShift Serverless 使用 **DomainMapping**。
  要临时解决这个问题，部署 **PeerAuthentication** 来启用 mTLS，而不是使用 **security.dataPlane.mtls: true**。

OpenShift Container Platform 的其他资源

- 配置 TLS 身份验证

1.8. RED HAT OPENSHIFT SERVERLESS 1.24.0 发行注记

OpenShift Serverless 1.24.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.8.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.3。

- OpenShift Serverless 现在使用 Knative Eventing 1.3。

- OpenShift Serverless 现在使用 Kourier 1.3。

- OpenShift Serverless 现在使用 Knative **kn** CLI 1.3。

- OpenShift Serverless 现在使用 Knative Kafka 1.3。

- **kn func** CLI 插件现在使用 **func** 0.24。

- 现在，提供了对 Knative 服务的 init 容器支持 (GA)。

- OpenShift Serverless 逻辑现在作为技术预览提供。它启用了定义声明工作流模型来管理无服务器应用程序。
- 对于 OpenShift Container Platform，您可以在 OpenShift Serverless 中使用成本管理服务。

1.8.2. 修复的问题

- 当集群中存在太多 secret 时，将 OpenShift Serverless 与 Red Hat OpenShift Service Mesh 集成会导致 **net-istio-controller** pod 在启动时耗尽内存。
  现在，可以启用 secret 过滤，这会导致 **net-istio-controller** 只考虑带有 **networking.internal.knative.dev/certificate-uid** 标签的 secret，从而减少所需的内存量。
- OpenShift Serverless 功能技术预览现在默认使用 Cloud Native Buildpacks 构建容器镜像。

1.8.3. 已知问题

- Kafka 代理、Kafka 源和 Kafka sink 禁用 Federal Information Processing Standards (FIPS) 模式。
- 在 OpenShift Serverless 1.23 中，删除了 KafkaBindings 和 **kafka-binding** Webhook 的支持。但是，现有 **kafkabindings.webhook.sources.knative.dev MutatingWebhookConfiguration** 可能保留，指向 **kafka-source-webhook** 服务，该服务不再存在。
  对于集群上 KafkaBindings 的某些规格，**kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** 可能会被配置，将任何创建和更新事件传递给各种资源，如 Deployment、Knative Services 或 Jobs，然后 Webhook 会失败。

  要临时解决这个问题，请在升级到 OpenShift Serverless 1.23 后从集群中删除 **kafkabindings.webhook.sources.knative.dev MutatingWebhookConfiguration**：

  ```
  $ oc delete mutatingwebhookconfiguration kafkabindings.webhook.kafka.sources.knative.dev
  ```

- 如果您将 **net-istio** 用于 Ingress，并使用 **security.dataPlane.mtls: true** 通过 SMCP 启用 mTLS，Service Mesh 为 ***.local** 主机部署 **DestinationRule**，这代表不允许对 OpenShift Serverless 使用 **DomainMapping**。
  要临时解决这个问题，部署 **PeerAuthentication** 来启用 mTLS，而不是使用 **security.dataPlane.mtls: true**。

1.9. RED HAT OPENSHIFT SERVERLESS 1.23.0 发行注记

OpenShift Serverless 1.23.0 现已发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.9.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.2。
- OpenShift Serverless 现在使用 Knative Eventing 1.2。
- OpenShift Serverless 现在使用 Kourier 1.2。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.2。
- OpenShift Serverless 现在使用 Knative Kafka 1.2。

- **kn func** CLI 插件现在使用 **func** 0.24。
- 现在，可以在 Kafka 代理中使用 **kafka.eventing.knative.dev/external.topic** 注解。此注解可以使用现有的外部管理主题，而不是代理自行创建内部主题。
- **kafka-ch-controller** 和 **kafka-webhook** Kafka 组件不再存在。这些组件已被 **kafka-webhook-eventing** 组件替代。
- OpenShift Serverless 功能技术预览现在默认使用 Source-to-Image (S2I) 来构建容器镜像。

1.9.2. 已知问题

- Kafka 代理、Kafka 源和 Kafka sink 禁用 Federal Information Processing Standards (FIPS) 模式。
- 如果您要删除包括 Kafka 代理的命名空间，当 **auth.secret.ref.name** secret 在代理被删除之前被删除，则命名空间终结器（finalizer）可能无法删除。
- 使用大量 Knative 服务运行 OpenShift Serverless 时，可能会导致运行的 Knativeivator pod 接近其默认的内存限值 600MB。如果使用的内存达到这个限值，则这些 pod 可能会重启。通过修改 **KnativeServing** 自定义资源，可以配置激活器部署的请求和限值：

```
apiVersion: operator.knative.dev/v1beta1
kind: KnativeServing
metadata:
  name: knative-serving
  namespace: knative-serving
spec:
  deployments:
  - name: activator
    resources:
    - container: activator
      requests:
        cpu: 300m
        memory: 60Mi
      limits:
        cpu: 1000m
        memory: 1000Mi
```

- 如果您使用 Cloud Native Buildpacks 作为一个函数的本地构建策略，**kn func** 将无法自动启动 podman，或使用 SSH 隧道到远程守护进程。这些问题的解决方法是，在部署函数前，在本地开发计算机上已在运行 Docker 或 podman 守护进程。
- On-cluster 函数构建当前针对 Quarkus 和 Golang 运行时会失败。它们适用于 Node、Typescript、Python 和 Springboot 运行时。
- 如果您将 **net-istio** 用于 Ingress，并使用 **security.dataPlane.mtls: true** 通过 SMCP 启用 mTLS，Service Mesh 为 ***.local** 主机部署 **DestinationRule**，这代表不允许对 OpenShift Serverless 使用 **DomainMapping**。
  要临时解决这个问题，部署 **PeerAuthentication** 来启用 mTLS，而不是使用 **security.dataPlane.mtls: true**。

OpenShift Container Platform 的其他资源

- Source-to-Image

1.10. RED HAT OPENSHIFT SERVERLESS 1.22.0 发行注记

OpenShift Serverless 1.22.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.10.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.1。
- OpenShift Serverless 现在使用 Knative Eventing 1.1。
- OpenShift Serverless 现在使用 Kourier 1.1。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.1。
- OpenShift Serverless 现在使用 Knative Kafka 1.1。
- **kn func** CLI 插件现在使用 **func** 0.23。
- init 容器支持 Knative 服务现在作为技术预览提供。
- 持久性卷声明 (PVC) 对 Knative 服务的支持现在作为技术预览提供。
- **knative-serving**、**knative-serving-ingress**、**knative-eventing** 和 **knative-kafka** 系统命名空间现在默认具有 **knative.openshift.io/part-of: "openshift-serverless"** 标签。
- 添加了 **Knative Eventing - Kafka Broker/Trigger** 仪表板，它允许在 web 控制台中视觉化 Kafka 代理并触发指标。
- 添加了 **Knative Eventing - KafkaSink** 仪表板，它允许在 web 控制台中视觉化 KafkaSink 指标。
- **Knative Eventing - Broker/Trigger** 仪表板现在被称为 **Knative Eventing - 基于频道的代理/触发器**。
- **knative.openshift.io/part-of: "openshift-serverless"** 标签已替换 **knative.openshift.io/system-namespace** 标签。
- Knative Serving YAML 配置文件中的命名样式从 camel 格式（**ExampleName**）改为连字符格式 (**example-name**) 。从这个版本开始，在创建或编辑 Knative Serving YAML 配置文件时使用连字符格式表示法。

1.10.2. 已知问题

- Kafka 代理、Kafka 源和 Kafka sink 禁用 Federal Information Processing Standards (FIPS) 模式。

1.11. RED HAT OPENSHIFT SERVERLESS 1.21.0 发行注记

OpenShift Serverless 1.21.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.11.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 1.0

- OpenShift Serverless 现在使用 Knative Eventing 1.0。
- OpenShift Serverless 现在使用 Kourier 1.0.
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 1.0。
- OpenShift Serverless 现在使用 Knative Kafka 1.0。
- **kn func** CLI 插件现在使用 **func** 0.21。
- Kafka sink 现在作为技术预览提供。
- Knative 开源项目已开始弃用发现配置密钥，而是统一使用 kebab-cased 键。因此，OpenShift Serverless 1.18.0 发行注记中提到的 **defaultExternalScheme** 键现已弃用，并被 **default-external-scheme** 键替代。键的使用说明保持不变。

1.11.2. 修复的问题

- 在 OpenShift Serverless 1.20.0 中，存在一个与发送事件相关的问题，会影响使用 **kn event send** 向服务发送事件。这个问题现已解决。
- 在 OpenShift Serverless 1.20.0 (**func** 0.20) 中，使用 **http** 模板创建的 TypeScript 功能无法在集群中部署。这个问题现已解决。
- 在 OpenShift Serverless 1.20.0 (**func** 0.20) 中，使用 **gcr.io** registry 部署功能会失败，并出现错误。这个问题现已解决。
- 在 OpenShift Serverless 1.20.0 (**func** 0.20) 中，使用 **kn func create** 命令创建 Springboot 功能项目目录，然后运行 **kn func build** 命令失败并显示错误消息。这个问题现已解决。
- 在 OpenShift Serverless 1.19.0 (**func** 0.19) 中，一些运行时无法使用 podman 来构建功能。这个问题现已解决。

1.11.3. 已知问题

- 目前，域映射控制器无法处理包含当前不支持的路径的代理 URI。
  这意味着，如果要使用 **DomainMapping** 自定义资源 (CR) 将自定义域映射到代理，则必须使用代理的 ingress 服务配置 **DomainMapping** CR，并将代理的确切路径附加到自定义域：

  DomainMapping CR 示例

  ```
  apiVersion: serving.knative.dev/v1alpha1
  kind: DomainMapping
  metadata:
    name: <domain-name>
    namespace: knative-eventing
  spec:
    ref:
      name: broker-ingress
      kind: Service
      apiVersion: v1
  ```

  代理的 URI 为 **<domain-name>/<broker-namespace>/<broker-name>**。

1.12. RED HAT OPENSHIFT SERVERLESS 1.20.0 发行注记

OpenShift Serverless 1.20.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.12.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 0.26。
- OpenShift Serverless 现在使用 Knative Eventing 0.26。
- OpenShift Serverless 现在使用 Kourier 0.26。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 0.26。
- OpenShift Serverless 现在使用 Knative Kafka 0.26。
- **kn func** CLI 插件现在使用 **func** 0.20。
- Kafka 代理现在作为技术预览提供。

重要

FIPS 不支持 Kafka 代理（当前为技术预览）。

- **kn event** 插件现在作为技术预览提供。
- **kn service create** 命令的 **--min-scale** 和 **--max-scale** 标志已弃用。使用 **--scale-min** 和 **--scale-max** 标志。

1.12.2. 已知问题

- OpenShift Serverless 使用 HTTPS 的默认地址部署 Knative 服务。将事件发送到集群中的资源时，发件人不会配置集群证书颁发机构 (CA) 。这会导致事件交付失败，除非集群使用全局接受的证书。
  例如，向公开访问的地址发送事件可正常工作：

```
$ kn event send --to-url https://ce-api.foo.example.com/
```

  另一方面，如果服务使用由自定义 CA 发布的 HTTPS 证书的公共地址，则此交付会失败：

```
$ kn event send --to Service:serving.knative.dev/v1:event-display
```

  将事件发送到其他可寻址的对象（如代理或频道）不受此问题的影响，并可以正常工作。

- Kafka 代理目前无法在启用了联邦信息处理标准 (FIPS) 模式的集群中工作。
- 如果您使用 **kn func create** 命令创建 Springboot 功能项目目录，后续的 **kn func build** 命令运行会失败并显示以下错误消息：

```
[analyzer] no stack metadata found at path ''
[analyzer] ERROR: failed to : set API for buildpack 'paketo-buildpacks/ca-certificates@3.0.2': buildpack API version '0.7' is incompatible with the lifecycle
```

  作为临时解决方案，您可以在函数配置文件 **func.yaml** 中将 **builder** 属性更改为 **gcr.io/paketo-buildpacks/builder:base**。

- 使用 **gcr.io** registry 部署函数会失败并显示以下错误消息：

  ```
  Error: failed to get credentials: failed to verify credentials: status code: 404
  ```

  作为临时解决方案，请使用与 **gcr.io** 不同的 registry，如 **quay.io** 或 **docker.io**。

- 使用 **http** 模板创建的 TypeScript 函数无法在集群中部署。
  作为临时解决方案，在 **func.yaml** 文件中替换以下部分：

  ```
  buildEnvs: []
  ```

  使用这个：

  ```
  buildEnvs:
  - name: BP_NODE_RUN_SCRIPTS
    value: build
  ```

- 在 **func** 版本 0.20 中，一些运行时可能无法使用 podman 构建函数。您可能会看到类似如下的错误消息：

  ```
  ERROR: failed to image: error during connect: Get
  "http://%2Fvar%2Frun%2Fdocker.sock/v1.40/info": EOF
  ```

  - 这个问题存在以下临时解决方案：

    a. 通过在 service **ExecStart** 定义中添加 **--time=0** 来更新 podman 服务：

    服务配置示例

    ```
    ExecStart=/usr/bin/podman $LOGGING system service --time=0
    ```

    b. 运行以下命令来重启 podman 服务：

    ```
    $ systemctl --user daemon-reload
    ```

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

  - 或者，您可以使用 TCP 公开 podman API：

    ```
    $ podman system service --time=0 tcp:127.0.0.1:5534 &
    export DOCKER_HOST=tcp://127.0.0.1:5534
    ```

1.13. RED HAT OPENSHIFT SERVERLESS 1.19.0 发行注记

OpenShift Serverless 1.19.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.13.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 0.25。
- OpenShift Serverless 现在使用 Knative Eventing 0.25。

- OpenShift Serverless 现在使用 Kourier 0.25。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 0.25。
- OpenShift Serverless 现在使用 Knative Kafka 0.25。
- **kn func** CLI 插件现在使用 **func** 0.19。
- **KafkaBinding** API 在 OpenShift Serverless 1.19.0 中已弃用，并将在以后的发行版本中删除。
- HTTPS 重定向现在被支持，并可以为集群或每个 Knative 服务配置。

1.13.2. 修复的问题

- 在以前的版本中，Kafka 频道分配程序仅在响应前等待本地提交成功，这可能会在 Apache Kafka 节点失败时导致事件丢失。Kafka 频道分配程序现在会在响应前等待所有同步副本提交。

1.13.3. 已知问题

- 在 **func** 版本 0.19 中，一些运行时可能无法使用 podman 构建函数。您可能会看到类似如下的错误消息：

```
ERROR: failed to image: error during connect: Get
"http://%2Fvar%2Frun%2Fdocker.sock/v1.40/info": EOF
```

  - 这个问题存在以下临时解决方案：
    a. 通过在 service **ExecStart** 定义中添加 **--time=0** 来更新 podman 服务：

    服务配置示例

    ```
    ExecStart=/usr/bin/podman $LOGGING system service --time=0
    ```

    b. 运行以下命令来重启 podman 服务：

    ```
    $ systemctl --user daemon-reload
    ```

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

  - 或者，您可以使用 TCP 公开 podman API：

    ```
    $ podman system service --time=0 tcp:127.0.0.1:5534 &
    export DOCKER_HOST=tcp://127.0.0.1:5534
    ```

1.14. RED HAT OPENSHIFT SERVERLESS 1.18.0 发行注记

OpenShift Serverless 1.18.0 现已正式发布。OpenShift Container Platform 上的 OpenShift Serverless 的新功能、改变以及已知的问题包括在此文档中。

1.14.1. 新功能

- OpenShift Serverless 现在使用 Knative Serving 0.24.0。

- OpenShift Serverless 现在使用 Knative Eventing 0.24.0。
- OpenShift Serverless 现在使用 Kourier 0.24.0。
- OpenShift Serverless 现在使用 Knative (**kn**) CLI 0.24.0。
- OpenShift Serverless 现在使用 Knative Kafka 0.24.7。
- **kn func** CLI 插件现在使用 **func** 0.18.0。
- 在即将发布的 OpenShift Serverless 1.19.0 发行版本中，外部路由的 URL 方案将默认为 HTTPS 以增强安全性。
  如果您不希望此更改应用到工作负载，您可以在升级到 1.19.0 前覆盖默认设置，方法是将以下 YAML 添加到 **KnativeServing** 自定义资源 (CR)：

```
...
spec:
  config:
    network:
      defaultExternalScheme: "http"
...
```

  如果您想在 1.18.0 中应用更改，请添加以下 YAML：

```
...
spec:
  config:
    network:
      defaultExternalScheme: "https"
...
```

- 在接下来的 OpenShift Serverless 1.19.0 发行版本中，公开 Kourier 网关的默认服务类型将是 **ClusterIP**，而不是 **LoadBalancer**。
  如果您不希望此更改应用到工作负载，您可以在升级到 1.19.0 前覆盖默认设置，方法是将以下 YAML 添加到 **KnativeServing** 自定义资源 (CR)：

```
...
spec:
  ingress:
    kourier:
      service-type: LoadBalancer
...
```

- 现在，您可以在 OpenShift Serverless 中使用 **emptyDir** 卷。详情请参阅 OpenShift Serverless 文档中的 Knative Serving 文档。
- 现在，当您使用 **kn func** 创建函数时，可以使用 Rust 模板。

1.14.2. 修复的问题

- 1.4 之前的 Camel-K 版本与 OpenShift Serverless 1.17.0 不兼容。Camel-K 中的问题已被解决，Camel-K 版本 1.4.1 可以用于 OpenShift Serverless 1.17.0。
- 在以前的版本中，如果您为 Kafka 频道或新 Kafka 源创建新订阅，则 Kafka 数据平面可能会在新创建的订阅或 sink 报告就绪状态后准备好发送信息。

因此，数据平面没有报告就绪状态时发送的信息可能没有传送到订阅者或 sink。

在 OpenShift Serverless 1.18.0 中，这个问题已被解决，初始消息不再丢失。有关此问题的更多信息，请参阅 知识库文章 #6343981。

1.14.3. 已知问题

- 较旧版本的 Knative **kn** CLI 可能会使用较旧版本的 Knative Serving 和 Knative Eventing API。例如，**kn** CLI 版本 0.23.2 使用 **v1alpha1** API 版本。
  另一方面，较新的 OpenShift Serverless 发行版本可能不再支持旧的 API 版本。例如，OpenShift Serverless 1.18.0 不再支持 **kafkasources.sources.knative.dev API** 的版本 **v1alpha1**。

  因此，使用带有较新的 OpenShift Serverless 的 Knative **kn** CLI 的旧版本可能会产生错误，因为 **kn** 无法找到过时的 API。例如，kn **CLI** 的 0.23.2 版本无法用于 OpenShift Serverless 1.18.0。

  为避免出现问题，请使用适用于 OpenShift Serverless 发行版本的最新 **kn** CLI 版本。对于 OpenShift Serverless 1.18.0，使用 Knative **kn** CLI 0.24.0。

第 2 章 OPENSHIFT SERVERLESS 概述

OpenShift Serverless 提供 Kubernetes 原生构建块，供开发人员在 OpenShift Container Platform 中创建和部署无服务器、事件驱动的应用程序。OpenShift Serverless 基于开源 Knative 项目，通过启用企业级无服务器平台为混合和多云环境提供可移植性和一致性。

注意

因为 OpenShift Serverless 的发行节奏与 Red Hat OpenShift Serverless 不同，所以 OpenShift Serverless 文档现在作为每个产品的次版本单独提供。

OpenShift Serverless 文档位于 https://docs.openshift.com/serverless/。

特定版本的文档使用版本选择器下拉菜单，或者直接添加版本到 URL，例如 https://docs.openshift.com/serverless/1.28。

另外，红帽门户网站中也提供了 OpenShift Serverless 文档，网址为 https://access.redhat.com/documentation/zh-cn/red_hat_openshift_serverless/。

如需有关 OpenShift Serverless 生命周期和支持的平台的更多信息，请参阅 平台生命周期政策。

2.1. 其他资源

- 使用自定义资源定义来扩展 Kubernetes API
- 管理自定义资源定义中的资源
- 什么是无服务器？

第 3 章 KNATIVE SERVING

Knative Serving 可以帮助需要创建、部署和管理云原生应用程序的开发人员。它以 Kubernetes 自定义资源定义 (CRD) 的形式提供一组对象，用于定义和控制 OpenShift Container Platform 集群上无服务器工作负载的行为。

开发人员使用这些 CRD 创建自定义资源（CR）实例，这些实例可作为构建块用于处理复杂用例。例如：

- 快速部署无服务器容器。
- 自动缩放 pod。

3.1. KNATIVE SERVING 资源

服务

service.serving.knative.dev CRD 会自动管理工作负载的生命周期，以确保应用程序通过网络部署并可访问。每次用户创建的服务或自定义资源发生变化时，它都会创建一个路由、配置和新修订。Knative 中进行的大多数开发人员交互都是通过修改服务进行的。

Revision（修订）

revision.serving.knative.dev CRD 是每次对工作负载进行修改所涉及代码和配置的时间点快照。所有修订均为不可变对象，可以根据需要保留。

Route（路由）

route.serving.knative.dev CRD 可将网络端点映射到一个或多个修订。您可通过多种方式管理流量，包括部分流量和指定路由。

Configuration（配置）

configuration.serving.knative.dev CRD 可保持部署所需状态。它可在使编程过程和配置配置过程相互分离。修改配置则会创建一个新修订。

第 4 章 KNATIVE EVENTING

OpenShift Container Platform 上的 Knative Eventing 可让开发人员使用 事件驱动的架构和无服务器应用程序。事件驱动的体系结构是基于事件和事件用户间分离关系的概念。

事件生成者创建事件，事件 *sink*、或消费者接收事件。Knative Eventing 使用标准 HTTP POST 请求来发送和接收事件制作者和 sink 之间的事件。这些事件符合 CloudEvents 规范，它允许在任何编程语言中创建、解析、发送和接收事件。

Knative Eventing 支持以下用例：

在不创建消费者的情况下发布事件

您可以将事件作为 HTTP POST 发送到代理，并使用绑定分离生成事件的应用程序的目标配置。

在不创建发布程序的情况下消费事件

您可以使用 Trigger 来根据事件属性消费来自代理的事件。应用程序以 HTTP POST 的形式接收事件。

要启用多种 sink 类型的交付，Knative Eventing 会定义以下通用接口，这些接口可由多个 Kubernetes 资源实现：

可寻址的资源

能够接收和确认通过 HTTP 发送的事件到 Event 的 **status.address.url** 字段中定义的地址。Kubernetes **Service** 资源也满足可寻址的接口。

可调用的资源

能够通过 HTTP 接收事件并转换它，并在 HTTP 响应有效负载中返回 **0** 或 **1** 新事件。这些返回的事件可能会象处理外部事件源中的事件一样进一步处理。

4.1. 为 APACHE KAFKA 使用 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。

Apache Kafka 的 Knative 代理提供附加选项，例如：

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

4.2. 其他资源

- 安装 **KnativeKafka** 自定义资源。
- Red Hat AMQ Streams 文档
- Apache Kafka 文档中的 Red Hat AMQ Streams TLS 和 SASL
- 事件交付

第 5 章 关于 OPENSHIFT SERVERLESS FUNCTIONS

OpenShift Serverless Functions 帮助开发人员在 OpenShift Container Platform 上创建和部署无状态、事件驱动的函数，作为 Knative 服务。**kn func** CLI 作为 Knative **kn** CLI 的插件提供。您可以使用 **kn func** CLI 在集群中创建、构建和部署容器镜像作为 Knative 服务。

5.1. 包括的运行时

OpenShift Serverless Functions 提供了一组模板，可用于为以下运行时创建基本函数：

- Quarkus
- Node.js
- TypeScript

5.2. 后续步骤

- 函数入门。

第 6 章 OPENSHIFT SERVERLESS 支持

如果您在执行本文档所述的某个流程时遇到问题，请访问红帽客户门户网站 http://access.redhat.com。您可以使用红帽客户门户网站搜索或浏览有关红帽产品的技术支持文章。您还可以向红帽全球支持服务 (GSS) 提交支持问题单，或者访问其他产品文档。

如果您对本文档有任何改进建议，或发现了任何错误，您可以提交一个与相关文档组件的 Jira 问题。请提供具体信息，如章节号、指南名称和 OpenShift Serverless 版本，以便我们可以快速地找到相关内容。

注意

如下部分用于定义集群大小要求适用于这些发行版本：

- OpenShift Container Platform
- OpenShift Dedicated

6.1. 关于红帽知识库

红帽知识库提供丰富的内容以帮助您最大程度地利用红帽的产品和技术。红帽知识库包括文章、产品文档和视频，概述了安装、配置和使用红帽产品的最佳实践。另外，您还可以搜索已知问题的解决方案，其提供简洁的根原因描述和补救措施。

6.2. 搜索红帽知识库

如果出现 Red Hat OpenShift Serverless 问题，您可以执行初始搜索来确定红帽知识库中是否已存在相应的解决方案。

先决条件

- 您有红帽客户门户网站帐户。

流程

1. 登录到 红帽客户门户网站。
2. 在主红帽客户门户网站搜索字段中，输入与问题相关的关键字和字符串，包括：
   - OpenShift Container Platform 组件（如 **etcd**）
   - 相关步骤（比如 **安装**）
   - 警告、错误消息和其他与输出与特定的问题相关
3. 点 **Search**。
4. 选择 **OpenShift Container Platform** 产品过滤器。
5. 在内容类型过滤中选择 **Knowledgebase**。

6.3. 提交支持问题单

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI（**oc**）。
- 您有红帽客户门户网站帐户。
- 您有红帽标准订阅或高级订阅。

流程

1. 登录到 红帽客户门户网站 并选择 **SUPPORT CASES → Open a case**。
2. 为您的问题选择适当的类别（如 **Defect / Bug**）、产品(**OpenShift Container Platform**)和产品版本（如果还没有自动填充则为 ）。
3. 查看推荐的红帽知识库解决方案列表，它们可能会与您要报告的问题相关。如果建议的文章没有解决这个问题，请点 **Continue**。
4. 输入一个简洁但描述性的问题概述，以及问题症状的详细信息，以及您预期的结果。
5. 查看更新的推荐红帽知识库解决方案列表，它们可能会与您要报告的问题相关。这个列表的范围会缩小，因为您在创建问题单的过程中提供了更多信息。如果建议的文章没有解决这个问题，请点 **Continue**。
6. 请确保提供的帐户信息是正确的，如果需要，请相应调整。
7. 检查自动填充的 Red Hat OpenShift Serverless 集群 ID 是否正确。如果不正确，请手动提供集群 ID。
   - 使用 OpenShift Container Platform Web 控制台手动获得集群 ID：
     a. 导航到 **Home → Dashboards → Overview**。
     b. 该值包括在 **Details** 中的 **Cluster ID** 项中。
   - 另外，也可以通过 OpenShift Container Platform Web 控制台直接创建新的支持问题单，并自动填充集群 ID。
     a. 从工具栏导航至 **（?） help → Open Support Case**。
     b. **Cluster ID** 的值会被自动填充 。
   - 要使用 OpenShift CLI（**oc**）获取集群 ID，请运行以下命令：

     ```
     $ oc get clusterversion -o jsonpath='{.items[].spec.clusterID}{"\n"}'
     ```

8. 完成以下提示的问题，点 **Continue**:
   - 您在哪里遇到了这个问题？什么环境？
   - 这个行为在什么时候发生？发生频率？重复发生？是否只在特定时间发生？
   - 请提供这个问题对您的业务的影响及与时间相关的信息？
9. 上传相关的诊断数据文件并点击 **Continue**。

建议您将使用 **oc adm must-gather** 命令收集的数据作为起点，并提供这个命令没有收集的与您的具体问题相关的其他数据。

1. 输入相关问题单管理详情，点 **Continue**。
2. 预览问题单详情，点 **Submit**。

6.4. 为支持收集诊断信息

在提交问题单时同时提供您的集群信息，可以帮助红帽支持为您进行排除故障。您可使用 **must-gather** 工具来收集有关 OpenShift Container Platform 集群的诊断信息，包括与 OpenShift Serverless 相关的数据。为了获得快速支持，请提供 OpenShift Container Platform 和 OpenShift Serverless 的诊断信息。

6.4.1. 关于 must-gather 工具

oc adm must-gather CLI 命令可收集最有助于解决问题的集群信息，包括：

- 资源定义
- 服务日志

默认情况下，**oc adm must-gather** 命令使用默认的插件镜像，并写入 **./must-gather.local**。

另外，您可以使用适当的参数运行命令来收集具体信息，如以下部分所述：

- 要收集与一个或多个特定功能相关的数据，请使用 **--image** 参数和镜像，如以下部分所述。
  例如：

  ```
  $ oc adm must-gather  --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.13.0
  ```

- 要收集审计日志，请使用 **-- /usr/bin/gather_audit_logs** 参数，如以下部分所述。
  例如：

  ```
  $ oc adm must-gather -- /usr/bin/gather_audit_logs
  ```

注意

作为默认信息集合的一部分，不会收集审计日志来减小文件的大小。

当您运行 **oc adm must-gather** 时，集群的新项目中会创建一个带有随机名称的新 pod。在该 pod 上收集数据，并保存至以 **must-gather.local** 开头的一个新目录中。此目录在当前工作目录中创建。

例如：

```
NAMESPACE                  NAME              READY   STATUS     RESTARTS     AGE
...
openshift-must-gather-5drcj    must-gather-bklx4    2/2     Running     0          72s
openshift-must-gather-5drcj    must-gather-s8sdh    2/2     Running     0          72s
...
```

另外，您可以使用 **--run-namespace** 选项在特定命名空间中运行 **oc adm must-gather** 命令。

例如：

```
$ oc adm must-gather --run-namespace <namespace> --image=registry.redhat.io/container-native-virtualization/cnv-must-gather-rhel8:v4.13.0
```

6.4.2. 关于收集 OpenShift Serverless 数据

您可使用 **oc adm must-gather** CLI 命令来收集有关集群的信息，包括与 OpenShift Serverless 相关的功能和对象。要使用 **must-gather** 来收集 OpenShift Serverless 数据，您必须为已安装的 OpenShift Serverless 版本指定 OpenShift Serverless 镜像和镜像标签。

先决条件

- 安装 OpenShift CLI (**oc**) 。

流程

- 使用 **oc adm must-gather** 命令收集数据：

```
$ oc adm must-gather --image=registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8:<image_version_tag>
```

示例命令

```
$ oc adm must-gather --image=registry.redhat.io/openshift-serverless-1/svls-must-gather-rhel8:1.14.0
```