Red Hat OpenShift Serverless 1.28

About Serverless

Introduction to OpenShift Serverless

Last Updated: 2023-07-03

Red Hat OpenShift Serverless 1.28 About Serverless

Introduction to OpenShift Serverless

Legal Notice

Abstract

This document provides an overview of OpenShift Serverless features, including Functions, Serving and Eventing. It also includes release notes and details on how to get support.

Table of Contents

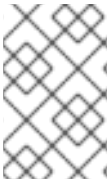
CHAPTER 1. RELEASE NOTES

NOTE

For additional information about the OpenShift Serverless life cycle and supported platforms, refer to the Platform Life Cycle Policy .

Release notes contain information about new and deprecated features, breaking changes, and known issues. The following release notes apply for the most recent OpenShift Serverless releases on OpenShift Container Platform.

For an overview of OpenShift Serverless functionality, see About OpenShift Serverless.



NOTE

OpenShift Serverless is based on the open source Knative project.

For details about the latest Knative component releases, see the Knative blog.

1.1. ABOUT API VERSIONS

API versions are an important measure of the development status of certain features and custom resources in OpenShift Serverless. Creating resources on your cluster that do not use the correct API version can cause issues in your deployment.

The OpenShift Serverless Operator automatically upgrades older resources that use deprecated versions of APIs to use the latest version. For example, if you have created resources on your cluster that use older versions of the **ApiServerSource** API, such as **v1beta1**, the OpenShift Serverless Operator automatically updates these resources to use the **v1** version of the API when this is available and the **v1beta1** version is deprecated.

After they have been deprecated, older versions of APIs might be removed in any upcoming release. Using deprecated versions of APIs does not cause resources to fail. However, if you try to use a version of an API that has been removed, it will cause resources to fail. Ensure that your manifests are updated to use the latest version to avoid issues.

1.2. GENERALLY AVAILABLE AND TECHNOLOGY PREVIEW FEATURES

Features which are Generally Available (GA) are fully supported and are suitable for production use. Technology Preview (TP) features are experimental features and are not intended for production use. See the Technology Preview scope of support on the Red Hat Customer Portal for more information about TP features.

The following table provides information about which OpenShift Serverless features are GA and which are TP:

Table 1.1. Generally Available and Technology Preview features tracker for OpenShift Container Platform and OpenShift Dedicated

Feature	1.26	1.27	1.28
kn func	GA	GA	GA

Feature	1.26	1.27	1.28
Quarkus functions	GA	GA	GA
Node.js functions	TP	TP	GA
TypeScript functions	TP	TP	GA
Python functions	-	-	TP
Service Mesh mTLS	GA	GA	GA
emptyDir volumes	GA	GA	GA
HTTPS redirection	GA	GA	GA
Kafka broker	GA	GA	GA
Kafka sink	GA	GA	GA
Init containers support for Knative services	GA	GA	GA
PVC support for Knative services	GA	GA	GA
TLS for internal traffic	TP	TP	TP
Namespace-scoped brokers	-	TP	TP
multi-container support	-	-	TP

Table 1.2. Generally Available and Technology Preview features tracker for Red Hat OpenShift Service on AWS

Feature	1.26	1.27	1.28
kn func	GA	GA	GA
Service Mesh mTLS	GA	GA	GA
emptyDir volumes	GA	GA	GA
HTTPS redirection	GA	GA	GA
Kafka broker	GA	GA	GA

Feature	1.26	1.27	1.28
Kafka sink	TP	TP	TP
Init containers support for Knative services	GA	GA	GA
PVC support for Knative services	GA	GA	GA
TLS for internal traffic	TP	TP	TP
Namespace-scoped brokers	-	TP	TP
multi-container support	-	-	TP

1.3. DEPRECATED AND REMOVED FEATURES

Some features that were Generally Available (GA) or a Technology Preview (TP) in previous releases have been deprecated or removed. Deprecated functionality is still included in OpenShift Serverless and continues to be supported; however, it will be removed in a future release of this product and is not recommended for new deployments.

For the most recent list of major functionality deprecated and removed within OpenShift Serverless, refer to the following table:

Table 1.3. Deprecated and removed features tracker for OpenShift Container Platform and OpenShift Dedicated

Feature	1.20	1.21	1.22 to 1.26	1.27	1.28
KafkaBinding API	Deprecated	Deprecated	Removed	Removed	Removed
kn func emit (**kn func invoke** in 1.21+)	Deprecated	Removed	Removed	Removed	Removed
Serving and Eventing **v1alpha1** API	-	-	-	Deprecated	Deprecated
enable-secret-informer-filtering annotation	-	-	-	-	Deprecated

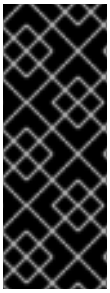
Table 1.4. Deprecated and removed features tracker for Red Hat OpenShift Service on AWS

Feature	1.23 to 1.26	1.27	1.28
KafkaBinding API	Removed	Removed	Removed

Feature	1.23 to 1.26	1.27	1.28
kn func emit (**kn func invoke** in 1.21+)	Removed	Removed	Removed
Serving and Eventing **v1alpha1** API	-	Deprecated	Deprecated

1.4. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.28

OpenShift Serverless 1.28 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.



IMPORTANT

OpenShift Container Platform 4.13 is based on Red Hat Enterprise Linux (RHEL) 9.2. RHEL 9.2 is yet to be submitted for Federal Information Processing Standards (FIPS) validation. Although Red Hat cannot commit to a specific timeframe, we expect to obtain FIPS validation for RHEL 9.0 and RHEL 9.2 modules, and later even minor releases of RHEL 9.x. Information on updates will be available in the Compliance Activities and Government Standards Knowledgebase article.

1.4.1. New features

- OpenShift Serverless now uses Knative Serving 1.7.
- OpenShift Serverless now uses Knative Eventing 1.7.
- OpenShift Serverless now uses Kourier 1.7.
- OpenShift Serverless now uses Knative (**kn**) CLI 1.7.
- OpenShift Serverless now uses Knative broker implementation for Apache Kafka 1.7.
- The **kn func** CLI plug-in now uses **func** 1.9.1 version.
- Node.js and TypeScript runtimes for OpenShift Serverless Functions are now Generally Available (GA).
- Python runtime for OpenShift Serverless Functions is now available as a Technology Preview.
- Multi-container support for Knative Serving is now available as a Technology Preview. This feature allows you to use a single Knative service to deploy a multi-container pod.
- In OpenShift Serverless 1.29 or later, the following components of Knative Eventing will be scaled down from two pods to one:
  - **imc-controller**
  - **imc-dispatcher**
  - **mt-broker-controller**
  - **mt-broker-filter**

- **mt-broker-ingress**

- The **serverless.openshift.io/enable-secret-informer-filtering** annotation for the Serving CR is now deprecated. The annotation is valid only for Istio, and not for Kourier.
  With OpenShift Serverless 1.28, the OpenShift Serverless Operator allows injecting the environment variable **ENABLE_SECRET_INFORMER_FILTERING_BY_CERT_UID** for both **net-istio** and **net-kourier**.

  If you enable secret filtering, all of your secrets need to be labeled with **networking.internal.knative.dev/certificate-uid: "<id>"**. Otherwise, Knative Serving does not detect them, which leads to failures. You must label both new and existing secrets.

  In one of the following OpenShift Serverless releases, secret filtering will become enabled by default. To prevent failures, label your secrets in advance.

1.4.2. Known issues

- Currently, runtimes for Python are not supported for OpenShift Serverless Functions on IBM Power, IBM zSystems, and IBM® LinuxONE.
  Node.js, TypeScript, and Quarkus functions are supported on these architectures.

- On the Windows platform, Python functions cannot be locally built, run, or deployed using the Source-to-Image builder due to the **app.sh** file permissions.
  To work around this problem, use the Windows Subsystem for Linux.

- **KafkaSource** resources created before Red Hat OpenShift Serverless 1.27 get stuck when being deleted. To work around the issue, after starting to delete a **KafkaSource**, remove the finalizer from the resource.

1.5. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.27

OpenShift Serverless 1.27 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

IMPORTANT

OpenShift Serverless 1.26 is the earliest release that is fully supported on OpenShift Container Platform 4.12. OpenShift Serverless 1.25 and older does not deploy on OpenShift Container Platform 4.12.

For this reason, before upgrading OpenShift Container Platform to version 4.12, first upgrade OpenShift Serverless to version 1.26 or 1.27.

1.5.1. New features

- OpenShift Serverless now uses Knative Serving 1.6.

- OpenShift Serverless now uses Knative Eventing 1.6.

- OpenShift Serverless now uses Kourier 1.6.

- OpenShift Serverless now uses Knative (**kn**) CLI 1.6.

- OpenShift Serverless now uses Knative Kafka 1.6.

- The **kn func** CLI plug-in now uses **func** 1.8.1.
- Namespace-scoped brokers are now available as a Technology Preview. Such brokers can be used, for instance, to implement role-based access control (RBAC) policies.
- **KafkaSink** now uses the **CloudEvent** binary content mode by default. The binary content mode is more efficient than the structured mode because it uses headers in its body instead of a **CloudEvent**. For example, for the HTTP protocol, it uses HTTP headers.
- You can now use the gRPC framework over the HTTP/2 protocol for external traffic using the OpenShift Route on OpenShift Container Platform 4.10 and later. This improves efficiency and speed of the communications between the client and server.
- API version **v1alpha1** of the Knative Operator Serving and Eventings CRDs is deprecated in 1.27. It will be removed in future versions. Red Hat strongly recommends to use the **v1beta1** version instead. This does not affect the existing installations, because CRDs are updated automatically when upgrading the Serverless Operator.
- The delivery timeout feature is now enabled by default. It allows you to specify the timeout for each sent HTTP request. The feature remains a Technology Preview.

1.5.2. Fixed issues

- Previously, Knative services sometimes did not get into the **Ready** state, reporting waiting for the load balancer to be ready. This issue has been fixed.

1.5.3. Known issues

- Integrating OpenShift Serverless with Red Hat OpenShift Service Mesh causes the **net-kourier** pod to run out of memory on startup when too many secrets are present on the cluster.
- Namespace-scoped brokers might leave **ClusterRoleBindings** in the user namespace even after deletion of namespace-scoped brokers.
  If this happens, delete the **ClusterRoleBinding** named **rbac-proxy-reviews-prom-rb-knative-kafka-broker-data-plane-{{.Namespace}}** in the user namespace.
- If you use **net-istio** for Ingress and enable mTLS via SMCP using **security.dataPlane.mtls: true**, Service Mesh deploys **DestinationRules** for the ***.local** host, which does not allow **DomainMapping** for OpenShift Serverless.
  To work around this issue, enable mTLS by deploying **PeerAuthentication** instead of using **security.dataPlane.mtls: true**.

1.6. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.26

OpenShift Serverless 1.26 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.6.1. New features

- OpenShift Serverless Functions with Quarkus is now GA.
- OpenShift Serverless now uses Knative Serving 1.5.
- OpenShift Serverless now uses Knative Eventing 1.5.

- OpenShift Serverless now uses Kourier 1.5.
- OpenShift Serverless now uses Knative (**kn**) CLI 1.5.
- OpenShift Serverless now uses Knative Kafka 1.5.
- OpenShift Serverless now uses Knative Operator 1.3.
- The **kn func** CLI plugin now uses **func** 1.8.1.
- Persistent volume claims (PVCs) are now GA. PVCs provide permanent data storage for your Knative services.
- The new trigger filters feature is now available as a Developer Preview. It allows users to specify a set of filter expressions, where each expression evaluates to either true or false for each event.
  To enable new trigger filters, add the **new-trigger-filters: enabled** entry in the section of the **KnativeEventing** type in the operator config map:

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

- Knative Operator 1.3 adds the updated **v1beta1** version of the API for **operator.knative.dev**. To update from **v1alpha1** to **v1beta1** in your **KnativeServing** and **KnativeEventing** custom resource config maps, edit the **apiVersion** key:

  Example KnativeServing custom resource config map

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

  Example KnativeEventing custom resource config map

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

1.6.2. Fixed issues

- Previously, Federal Information Processing Standards (FIPS) mode was disabled for Kafka broker, Kafka source, and Kafka sink. This has been fixed, and FIPS mode is now available.

1.6.3. Known issues

- If you use **net-istio** for Ingress and enable mTLS via SMCP using **security.dataPlane.mtls: true**, Service Mesh deploys **DestinationRules** for the ***.local** host, which does not allow **DomainMapping** for OpenShift Serverless.
  To work around this issue, enable mTLS by deploying **PeerAuthentication** instead of using **security.dataPlane.mtls: true**.

Additional resources

- Knative documentation on new trigger filters

1.7. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.25.0

OpenShift Serverless 1.25.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.7.1. New features

- OpenShift Serverless now uses Knative Serving 1.4.
- OpenShift Serverless now uses Knative Eventing 1.4.
- OpenShift Serverless now uses Kourier 1.4.
- OpenShift Serverless now uses Knative (**kn**) CLI 1.4.
- OpenShift Serverless now uses Knative Kafka 1.4.
- The **kn func** CLI plugin now uses **func** 1.7.0.
- Integrated development environment (IDE) plugins for creating and deploying functions are now available for Visual Studio Code and IntelliJ.
- Knative Kafka broker is now GA. Knative Kafka broker is a highly performant implementation of the Knative broker API, directly targeting Apache Kafka.
  It is recommended to not use the MT-Channel-Broker, but the Knative Kafka broker instead.
- Knative Kafka sink is now GA. A **KafkaSink** takes a **CloudEvent** and sends it to an Apache Kafka topic. Events can be specified in either structured or binary content modes.
- Enabling TLS for internal traffic is now available as a Technology Preview.

1.7.2. Fixed issues

- Previously, Knative Serving had an issue where the readiness probe failed if the container was restarted after a liveness probe fail. This issue has been fixed.

1.7.3. Known issues

- The Federal Information Processing Standards (FIPS) mode is disabled for Kafka broker, Kafka source, and Kafka sink.
- The **SinkBinding** object does not support custom revision names for services.

- The Knative Serving Controller pod adds a new informer to watch secrets in the cluster. The informer includes the secrets in the cache, which increases memory consumption of the controller pod.
  If the pod runs out of memory, you can work around the issue by increasing the memory limit for the deployment.

- If you use **net-istio** for Ingress and enable mTLS via SMCP using **security.dataPlane.mtls: true**, Service Mesh deploys **DestinationRules** for the ***.local** host, which does not allow **DomainMapping** for OpenShift Serverless.
  To work around this issue, enable mTLS by deploying **PeerAuthentication** instead of using **security.dataPlane.mtls: true**.

Additional resources for OpenShift Container Platform

- Configuring TLS authentication

1.8. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.24.0

OpenShift Serverless 1.24.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.8.1. New features

- OpenShift Serverless now uses Knative Serving 1.3.
- OpenShift Serverless now uses Knative Eventing 1.3.
- OpenShift Serverless now uses Kourier 1.3.
- OpenShift Serverless now uses Knative **kn** CLI 1.3.
- OpenShift Serverless now uses Knative Kafka 1.3.
- The **kn func** CLI plugin now uses **func** 0.24.
- Init containers support for Knative services is now generally available (GA).
- OpenShift Serverless logic is now available as a Developer Preview. It enables defining declarative workflow models for managing serverless applications.
- For OpenShift Container Platform, you can now use the cost management service with OpenShift Serverless.

1.8.2. Fixed issues

- Integrating OpenShift Serverless with Red Hat OpenShift Service Mesh causes the **net-istio-controller** pod to run out of memory on startup when too many secrets are present on the cluster.
  It is now possible to enable secret filtering, which causes **net-istio-controller** to consider only secrets with a **networking.internal.knative.dev/certificate-uid** label, thus reducing the amount of memory needed.

- The OpenShift Serverless Functions Technology Preview now uses Cloud Native Buildpacks by default to build container images.

1.8.3. Known issues

- The Federal Information Processing Standards (FIPS) mode is disabled for Kafka broker, Kafka source, and Kafka sink.

- In OpenShift Serverless 1.23, support for KafkaBindings and the **kafka-binding** webhook were removed. However, an existing **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** might remain, pointing to the **kafka-source-webhook** service, which no longer exists.
  For certain specifications of KafkaBindings on the cluster, **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** might be configured to pass any create and update events to various resources, such as Deployments, Knative Services, or Jobs, through the webhook, which would then fail.

  To work around this issue, manually delete **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** from the cluster after upgrading to OpenShift Serverless 1.23:

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

- If you use **net-istio** for Ingress and enable mTLS via SMCP using **security.dataPlane.mtls: true**, Service Mesh deploys **DestinationRules** for the ***.local** host, which does not allow **DomainMapping** for OpenShift Serverless.
  To work around this issue, enable mTLS by deploying **PeerAuthentication** instead of using **security.dataPlane.mtls: true**.

1.9. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.23.0

OpenShift Serverless 1.23.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.9.1. New features

- OpenShift Serverless now uses Knative Serving 1.2.

- OpenShift Serverless now uses Knative Eventing 1.2.

- OpenShift Serverless now uses Kourier 1.2.

- OpenShift Serverless now uses Knative (**kn**) CLI 1.2.

- OpenShift Serverless now uses Knative Kafka 1.2.

- The **kn func** CLI plugin now uses **func** 0.24.

- It is now possible to use the **kafka.eventing.knative.dev/external.topic** annotation with the Kafka broker. This annotation makes it possible to use an existing externally managed topic instead of the broker creating its own internal topic.

- The **kafka-ch-controller** and **kafka-webhook** Kafka components no longer exist. These components have been replaced by the **kafka-webhook-eventing** component.

- The OpenShift Serverless Functions Technology Preview now uses Source-to-Image (S2I) by default to build container images.

1.9.2. Known issues

- The Federal Information Processing Standards (FIPS) mode is disabled for Kafka broker, Kafka source, and Kafka sink.
- If you delete a namespace that includes a Kafka broker, the namespace finalizer may fail to be removed if the broker's **auth.secret.ref.name** secret is deleted before the broker.
- Running OpenShift Serverless with a large number of Knative services can cause Knative activator pods to run close to their default memory limits of 600MB. These pods might be restarted if memory consumption reaches this limit. Requests and limits for the activator deployment can be configured by modifying the **KnativeServing** custom resource:

```
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
```

- If you are using Cloud Native Buildpacks as the local build strategy for a function, **kn func** is unable to automatically start podman or use an SSH tunnel to a remote daemon. The workaround for these issues is to have a Docker or podman daemon already running on the local development computer before deploying a function.
- On-cluster function builds currently fail for Quarkus and Golang runtimes. They work correctly for Node, Typescript, Python, and Springboot runtimes.
- If you use **net-istio** for Ingress and enable mTLS via SMCP using **security.dataPlane.mtls: true**, Service Mesh deploys **DestinationRules** for the ***.local** host, which does not allow **DomainMapping** for OpenShift Serverless.
  To work around this issue, enable mTLS by deploying **PeerAuthentication** instead of using **security.dataPlane.mtls: true**.

Additional resources for OpenShift Container Platform

- Source-to-Image

1.10. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.22.0

OpenShift Serverless 1.22.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.10.1. New features

- OpenShift Serverless now uses Knative Serving 1.1.
- OpenShift Serverless now uses Knative Eventing 1.1.
- OpenShift Serverless now uses Kourier 1.1.
- OpenShift Serverless now uses Knative (**kn**) CLI 1.1.
- OpenShift Serverless now uses Knative Kafka 1.1.
- The **kn func** CLI plugin now uses **func** 0.23.
- Init containers support for Knative services is now available as a Technology Preview.
- Persistent volume claim (PVC) support for Knative services is now available as a Technology Preview.
- The **knative-serving**, **knative-serving-ingress**, **knative-eventing** and **knative-kafka** system namespaces now have the **knative.openshift.io/part-of: "openshift-serverless"** label by default.
- The **Knative Eventing - Kafka Broker/Trigger** dashboard has been added, which allows visualizing Kafka broker and trigger metrics in the web console.
- The **Knative Eventing - KafkaSink** dashboard has been added, which allows visualizing KafkaSink metrics in the web console.
- The **Knative Eventing - Broker/Trigger** dashboard is now called **Knative Eventing - Channel-based Broker/Trigger**.
- The **knative.openshift.io/part-of: "openshift-serverless"** label has substituted the **knative.openshift.io/system-namespace** label.
- Naming style in Knative Serving YAML configuration files changed from camel case (**ExampleName**) to hyphen style (**example-name**). Beginning with this release, use the hyphen style notation when creating or editing Knative Serving YAML configuration files.

1.10.2. Known issues

- The Federal Information Processing Standards (FIPS) mode is disabled for Kafka broker, Kafka source, and Kafka sink.

1.11. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.21.0

OpenShift Serverless 1.21.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.11.1. New features

- OpenShift Serverless now uses Knative Serving 1.0
- OpenShift Serverless now uses Knative Eventing 1.0.
- OpenShift Serverless now uses Kourier 1.0.
- OpenShift Serverless now uses Knative (**kn**) CLI 1.0.

- OpenShift Serverless now uses Knative Kafka 1.0.
- The **kn func** CLI plugin now uses **func** 0.21.
- The Kafka sink is now available as a Technology Preview.
- The Knative open source project has begun to deprecate camel-cased configuration keys in favor of using kebab-cased keys consistently. As a result, the **defaultExternalScheme** key, previously mentioned in the OpenShift Serverless 1.18.0 release notes, is now deprecated and replaced by the **default-external-scheme** key. Usage instructions for the key remain the same.

1.11.2. Fixed issues

- In OpenShift Serverless 1.20.0, there was an event delivery issue affecting the use of **kn event send** to send events to a service. This issue is now fixed.
- In OpenShift Serverless 1.20.0 (**func** 0.20), TypeScript functions created with the **http** template failed to deploy on the cluster. This issue is now fixed.
- In OpenShift Serverless 1.20.0 (**func** 0.20), deploying a function using the **gcr.io** registry failed with an error. This issue is now fixed.
- In OpenShift Serverless 1.20.0 (**func** 0.20), creating a Springboot function project directory with the **kn func create** command and then running the **kn func build** command failed with an error message. This issue is now fixed.
- In OpenShift Serverless 1.19.0 (**func** 0.19), some runtimes were unable to build a function by using podman. This issue is now fixed.

1.11.3. Known issues

- Currently, the domain mapping controller cannot process the URI of a broker, which contains a path that is currently not supported.
  This means that, if you want to use a **DomainMapping** custom resource (CR) to map a custom domain to a broker, you must configure the **DomainMapping** CR with the broker's ingress service, and append the exact path of the broker to the custom domain:

  Example DomainMapping CR

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

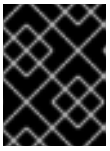
  The URI for the broker is then **<domain-name>/<broker-namespace>/<broker-name>**.

1.12. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.20.0

OpenShift Serverless 1.20.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.12.1. New features

- OpenShift Serverless now uses Knative Serving 0.26.
- OpenShift Serverless now uses Knative Eventing 0.26.
- OpenShift Serverless now uses Kourier 0.26.
- OpenShift Serverless now uses Knative (**kn**) CLI 0.26.
- OpenShift Serverless now uses Knative Kafka 0.26.
- The **kn func** CLI plugin now uses **func** 0.20.
- The Kafka broker is now available as a Technology Preview.



IMPORTANT

The Kafka broker, which is currently in Technology Preview, is not supported on FIPS.

- The **kn event** plugin is now available as a Technology Preview.
- The **--min-scale** and **--max-scale** flags for the **kn service create** command have been deprecated. Use the **--scale-min** and **--scale-max** flags instead.

1.12.2. Known issues

- OpenShift Serverless deploys Knative services with a default address that uses HTTPS. When sending an event to a resource inside the cluster, the sender does not have the cluster certificate authority (CA) configured. This causes event delivery to fail, unless the cluster uses globally accepted certificates.
  For example, an event delivery to a publicly accessible address works:

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

  On the other hand, this delivery fails if the service uses a public address with an HTTPS certificate issued by a custom CA:

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

  Sending an event to other addressable objects, such as brokers or channels, is not affected by this issue and works as expected.
- The Kafka broker currently does not work on a cluster with Federal Information Processing Standards (FIPS) mode enabled.
- If you create a Springboot function project directory with the **kn func create** command, subsequent running of the **kn func build** command fails with this error message:

```
[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
```

As a workaround, you can change the **builder** property to **gcr.io/paketo-buildpacks/builder:base** in the function configuration file **func.yaml**.

- Deploying a function using the **gcr.io** registry fails with this error message:

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

As a workaround, use a different registry than **gcr.io**, such as **quay.io** or **docker.io**.

- TypeScript functions created with the **http** template fail to deploy on the cluster.
As a workaround, in the **func.yaml** file, replace the following section:

```
buildEnvs: []
```

with this:

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

- In **func** version 0.20, some runtimes might be unable to build a function by using podman. You might see an error message similar to the following:

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

  - The following workaround exists for this issue:

    a. Update the podman service by adding **--time=0** to the service **ExecStart** definition:

    Example service configuration

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

    b. Restart the podman service by running the following commands:

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

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

  - Alternatively, you can expose the podman API by using TCP:

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

1.13. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.19.0

OpenShift Serverless 1.19.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.13.1. New features

- OpenShift Serverless now uses Knative Serving 0.25.
- OpenShift Serverless now uses Knative Eventing 0.25.
- OpenShift Serverless now uses Kourier 0.25.
- OpenShift Serverless now uses Knative (**kn**) CLI 0.25.
- OpenShift Serverless now uses Knative Kafka 0.25.
- The **kn func** CLI plugin now uses **func** 0.19.
- The **KafkaBinding** API is deprecated in OpenShift Serverless 1.19.0 and will be removed in a future release.
- HTTPS redirection is now supported and can be configured either globally for a cluster or per each Knative service.

1.13.2. Fixed issues

- In previous releases, the Kafka channel dispatcher waited only for the local commit to succeed before responding, which might have caused lost events in the case of an Apache Kafka node failure. The Kafka channel dispatcher now waits for all in-sync replicas to commit before responding.

1.13.3. Known issues

- In **func** version 0.19, some runtimes might be unable to build a function by using podman. You might see an error message similar to the following:

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

  - The following workaround exists for this issue:

    a. Update the podman service by adding **--time=0** to the service **ExecStart** definition:

    Example service configuration

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

    b. Restart the podman service by running the following commands:

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

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

  - Alternatively, you can expose the podman API by using TCP:

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

1.14. RELEASE NOTES FOR RED HAT OPENSHIFT SERVERLESS 1.18.0

OpenShift Serverless 1.18.0 is now available. New features, changes, and known issues that pertain to OpenShift Serverless on OpenShift Container Platform are included in this topic.

1.14.1. New features

- OpenShift Serverless now uses Knative Serving 0.24.0.
- OpenShift Serverless now uses Knative Eventing 0.24.0.
- OpenShift Serverless now uses Kourier 0.24.0.
- OpenShift Serverless now uses Knative (**kn**) CLI 0.24.0.
- OpenShift Serverless now uses Knative Kafka 0.24.7.
- The **kn func** CLI plugin now uses **func** 0.18.0.
- In the upcoming OpenShift Serverless 1.19.0 release, the URL scheme of external routes will default to HTTPS for enhanced security.
  If you do not want this change to apply for your workloads, you can override the default setting before upgrading to 1.19.0, by adding the following YAML to your **KnativeServing** custom resource (CR):

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

  If you want the change to apply in 1.18.0 already, add the following YAML:

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

- In the upcoming OpenShift Serverless 1.19.0 release, the default service type by which the Kourier Gateway is exposed will be **ClusterIP** and not **LoadBalancer**.
  If you do not want this change to apply to your workloads, you can override the default setting before upgrading to 1.19.0, by adding the following YAML to your **KnativeServing** custom resource (CR):

  ```
  ...
  spec:
    ingress:
  ```

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

- You can now use **emptyDir** volumes with OpenShift Serverless. See the OpenShift Serverless documentation about Knative Serving for details.

- Rust templates are now available when you create a function using **kn func**.

1.14.2. Fixed issues

- The prior 1.4 version of Camel-K was not compatible with OpenShift Serverless 1.17.0. The issue in Camel-K has been fixed, and Camel-K version 1.4.1 can be used with OpenShift Serverless 1.17.0.

- Previously, if you created a new subscription for a Kafka channel, or a new Kafka source, a delay was possible in the Kafka data plane becoming ready to dispatch messages after the newly created subscription or sink reported a ready status.
  As a result, messages that were sent during the time when the data plane was not reporting a ready status, might not have been delivered to the subscriber or sink.

  In OpenShift Serverless 1.18.0, the issue is fixed and the initial messages are no longer lost. For more information about the issue, see Knowledgebase Article #6343981.

1.14.3. Known issues

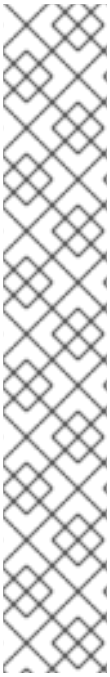
- Older versions of the Knative **kn** CLI might use older versions of the Knative Serving and Knative Eventing APIs. For example, version 0.23.2 of the **kn** CLI uses the **v1alpha1** API version. On the other hand, newer releases of OpenShift Serverless might no longer support older API versions. For example, OpenShift Serverless 1.18.0 no longer supports version **v1alpha1** of the **kafkasources.sources.knative.dev** API.

  Consequently, using an older version of the Knative **kn** CLI with a newer OpenShift Serverless might produce an error because the **kn** cannot find the outdated API. For example, version 0.23.2 of the **kn** CLI does not work with OpenShift Serverless 1.18.0.

  To avoid issues, use the latest **kn** CLI version available for your OpenShift Serverless release. For OpenShift Serverless 1.18.0, use Knative **kn** CLI 0.24.0.

CHAPTER 2. OPENSHIFT SERVERLESS OVERVIEW

OpenShift Serverless provides Kubernetes native building blocks that enable developers to create and deploy serverless, event-driven applications on OpenShift Container Platform. OpenShift Serverless is based on the open source Knative project, which provides portability and consistency for hybrid and multi-cloud environments by enabling an enterprise-grade serverless platform.



NOTE

Because OpenShift Serverless releases on a different cadence from Red Hat OpenShift Serverless, the OpenShift Serverless documentation is now available as separate documentation sets for each minor version of the product.

The OpenShift Serverless documentation is available at https://docs.openshift.com/serverless/.

Documentation for specific versions is available using the version selector dropdown, or directly by adding the version to the URL, for example, https://docs.openshift.com/serverless/1.28.

In addition, the OpenShift Serverless documentation is also available on the Red Hat Portal at https://access.redhat.com/documentation/en-us/red_hat_openshift_serverless/.

For additional information about the OpenShift Serverless life cycle and supported platforms, refer to the Platform Life Cycle Policy.

2.1. ADDITIONAL RESOURCES

- Extending the Kubernetes API with custom resource definitions
- Managing resources from custom resource definitions
- What is serverless?

CHAPTER 3. KNATIVE SERVING

Knative Serving supports developers who want to create, deploy, and manage cloud-native applications. It provides a set of objects as Kubernetes custom resource definitions (CRDs) that define and control the behavior of serverless workloads on an OpenShift Container Platform cluster.

Developers use these CRDs to create custom resource (CR) instances that can be used as building blocks to address complex use cases. For example:

- Rapidly deploying serverless containers.
- Automatically scaling pods.

3.1. KNATIVE SERVING RESOURCES

Service

The **service.serving.knative.dev** CRD automatically manages the life cycle of your workload to ensure that the application is deployed and reachable through the network. It creates a route, a configuration, and a new revision for each change to a user created service, or custom resource. Most developer interactions in Knative are carried out by modifying services.

Revision

The **revision.serving.knative.dev** CRD is a point-in-time snapshot of the code and configuration for each modification made to the workload. Revisions are immutable objects and can be retained for as long as necessary.

Route

The **route.serving.knative.dev** CRD maps a network endpoint to one or more revisions. You can manage the traffic in several ways, including fractional traffic and named routes.

Configuration

The **configuration.serving.knative.dev** CRD maintains the desired state for your deployment. It provides a clean separation between code and configuration. Modifying a configuration creates a new revision.

CHAPTER 4. KNATIVE EVENTING

Knative Eventing on OpenShift Container Platform enables developers to use an event-driven architecture with serverless applications. An event-driven architecture is based on the concept of decoupled relationships between event producers and event consumers.

Event producers create events, and event *sinks*, or consumers, receive events. Knative Eventing uses standard HTTP POST requests to send and receive events between event producers and sinks. These events conform to the CloudEvents specifications, which enables creating, parsing, sending, and receiving events in any programming language.

Knative Eventing supports the following use cases:

Publish an event without creating a consumer

You can send events to a broker as an HTTP POST, and use binding to decouple the destination configuration from your application that produces events.

Consume an event without creating a publisher

You can use a trigger to consume events from a broker based on event attributes. The application receives events as an HTTP POST.

To enable delivery to multiple types of sinks, Knative Eventing defines the following generic interfaces that can be implemented by multiple Kubernetes resources:

Addressable resources

Able to receive and acknowledge an event delivered over HTTP to an address defined in the **status.address.url** field of the event. The Kubernetes **Service** resource also satisfies the addressable interface.

Callable resources

Able to receive an event delivered over HTTP and transform it, returning **0** or **1** new events in the HTTP response payload. These returned events may be further processed in the same way that events from an external event source are processed.

4.1. USING THE 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.

Knative broker for Apache Kafka provides additional options, such as:

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

4.2. ADDITIONAL RESOURCES

- Installing the **KnativeKafka** custom resource.
- Red Hat AMQ Streams documentation

- Red Hat AMQ Streams TLS and SASL on Apache Kafka documentation
- Event delivery

CHAPTER 5. ABOUT OPENSHIFT SERVERLESS FUNCTIONS

OpenShift Serverless Functions enables developers to create and deploy stateless, event-driven functions as a Knative service on OpenShift Container Platform. The **kn func** CLI is provided as a plugin for the Knative **kn** CLI. You can use the **kn func** CLI to create, build, and deploy the container image as a Knative service on the cluster.

5.1. INCLUDED RUNTIMES

OpenShift Serverless Functions provides templates that can be used to create basic functions for the following runtimes:

- Quarkus
- Node.js
- TypeScript

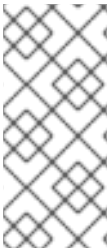
5.2. NEXT STEPS

- Getting started with functions.

CHAPTER 6. OPENSHIFT SERVERLESS SUPPORT

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at http://access.redhat.com. You can use the Red Hat Customer Portal to search or browse through the Red Hat Knowledgebase of technical support articles about Red Hat products. You can also submit a support case to Red Hat Global Support Services (GSS), or access other product documentation.

If you have a suggestion for improving this guide or have found an error, you can submit a Jira issue for the most relevant documentation component. Provide specific details, such as the section number, guide name, and OpenShift Serverless version so we can easily locate the content.



NOTE

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

- OpenShift Container Platform
- OpenShift Dedicated

6.1. ABOUT THE RED HAT KNOWLEDGEBASE

The Red Hat Knowledgebase provides rich content aimed at helping you make the most of Red Hat's products and technologies. The Red Hat Knowledgebase consists of articles, product documentation, and videos outlining best practices on installing, configuring, and using Red Hat products. In addition, you can search for solutions to known issues, each providing concise root cause descriptions and remedial steps.

6.2. SEARCHING THE RED HAT KNOWLEDGEBASE

In the event of an Red Hat OpenShift Serverless issue, you can perform an initial search to determine if a solution already exists within the Red Hat Knowledgebase.

Prerequisites

- You have a Red Hat Customer Portal account.

Procedure

1. Log in to the Red Hat Customer Portal .
2. In the main Red Hat Customer Portal search field, input keywords and strings relating to the problem, including:
   - OpenShift Container Platform components (such as **etcd**)
   - Related procedure (such as **installation**)
   - Warnings, error messages, and other outputs related to explicit failures
3. Click **Search**.
4. Select the **OpenShift Container Platform** product filter.
5. Select the **Knowledgebase** content type filter.

6.3. SUBMITTING A SUPPORT CASE

Prerequisites

- You have access to the cluster as a user with the **cluster-admin** role.
- You have installed the OpenShift CLI (**oc**).
- You have a Red Hat Customer Portal account.
- You have a Red Hat standard or premium Subscription.

Procedure

1. Log in to the Red Hat Customer Portal and select **SUPPORT CASES → Open a case**.
2. Select the appropriate category for your issue (such as **Defect / Bug**), product (**OpenShift Container Platform**), and product version if this is not already autofilled).
3. Review the list of suggested Red Hat Knowledgebase solutions for a potential match against the problem that is being reported. If the suggested articles do not address the issue, click **Continue**.
4. Enter a concise but descriptive problem summary and further details about the symptoms being experienced, as well as your expectations.
5. Review the updated list of suggested Red Hat Knowledgebase solutions for a potential match against the problem that is being reported. The list is refined as you provide more information during the case creation process. If the suggested articles do not address the issue, click **Continue**.
6. Ensure that the account information presented is as expected, and if not, amend accordingly.
7. Check that the autofilled Red Hat OpenShift Serverless Cluster ID is correct. If it is not, manually obtain your cluster ID.
   - To manually obtain your cluster ID using the OpenShift Container Platform web console:
     a. Navigate to **Home → Dashboards → Overview**.
     b. Find the value in the **Cluster ID** field of the **Details** section.
   - Alternatively, it is possible to open a new support case through the OpenShift Container Platform web console and have your cluster ID autofilled.
     a. From the toolbar, navigate to **(?) Help → Open Support Case**.
     b. The **Cluster ID** value is autofilled.
   - To obtain your cluster ID using the OpenShift CLI (**oc**), run the following command:

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

8. Complete the following questions where prompted and then click **Continue**:
   - Where are you experiencing the behavior? What environment?

- When does the behavior occur? Frequency? Repeatedly? At certain times?
- What information can you provide around time-frames and the business impact?

9. Upload relevant diagnostic data files and click **Continue**.

It is recommended to include data gathered using the **oc adm must-gather** command as a starting point, plus any issue specific data that is not collected by that command.

1. Input relevant case management details and click **Continue**.
2. Preview the case details and click **Submit**.

6.4. GATHERING DIAGNOSTIC INFORMATION FOR SUPPORT

When you open a support case, it is helpful to provide debugging information about your cluster to Red Hat Support. The **must-gather** tool enables you to collect diagnostic information about your OpenShift Container Platform cluster, including data related to OpenShift Serverless. For prompt support, supply diagnostic information for both OpenShift Container Platform and OpenShift Serverless.

6.4.1. About the must-gather tool

The **oc adm must-gather** CLI command collects the information from your cluster that is most likely needed for debugging issues, including:

- Resource definitions
- Service logs

By default, the **oc adm must-gather** command uses the default plugin image and writes into **./must-gather.local**.

Alternatively, you can collect specific information by running the command with the appropriate arguments as described in the following sections:

- To collect data related to one or more specific features, use the **--image** argument with an image, as listed in a following section.
  For example:

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

- To collect the audit logs, use the **-- /usr/bin/gather_audit_logs** argument, as described in a following section.
  For example:

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

NOTE

Audit logs are not collected as part of the default set of information to reduce the size of the files.

When you run **oc adm must-gather**, a new pod with a random name is created in a new project on the cluster. The data is collected on that pod and saved in a new directory that starts with **must-gather.local**. This directory is created in the current working directory.

For example:

```
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
...
```

Optionally, you can run the **oc adm must-gather** command in a specific namespace by using the **--run-namespace** option.

For example:

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

6.4.2. About collecting OpenShift Serverless data

You can use the **oc adm must-gather** CLI command to collect information about your cluster, including features and objects associated with OpenShift Serverless. To collect OpenShift Serverless data with **must-gather**, you must specify the OpenShift Serverless image and the image tag for your installed version of OpenShift Serverless.

Prerequisites

- Install the OpenShift CLI (**oc**).

Procedure

- Collect data by using the **oc adm must-gather** command:

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

Example command

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