Red Hat OpenShift Serverless 1.28

Serverless 정보

OpenShift Serverless 소개

Last Updated: 2023-09-19

Red Hat OpenShift Serverless 1.28 Serverless 정보

OpenShift Serverless 소개

법적 공지

초록

이 문서에서는 Functions, Serving, Eventing을 포함한 OpenShift Serverless 기능에 대한 개요를 제공합니다. 또한 릴리스 노트와 지원을 받는 방법에 대한 세부 사항이 포함되어 있습니다.

차례

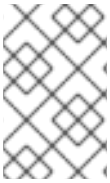
1장. 릴리스 노트

참고

OpenShift Serverless 라이프 사이클 및 지원되는 플랫폼에 대한 자세한 내용은 플랫폼 라이프 사이클 정책을 참조하십시오.

릴리스 노트에는 새로운 기능 및 더 이상 사용되지 않는 기능, 변경 사항 중단 및 알려진 문제에 대한 정보가 포함되어 있습니다. 다음 릴리스 노트는 OpenShift Container Platform의 최신 OpenShift Serverless 릴리스에 적용됩니다.

OpenShift Serverless 기능에 대한 개요는 OpenShift Serverless 정보를 참조하십시오.



참고

OpenShift Serverless는 오픈 소스 Knative 프로젝트를 기반으로 합니다.

최신 Knative 구성 요소 릴리스에 대한 자세한 내용은 Knative 블로그를 참조하십시오.

1.1. API 버전 정보

API 버전은 OpenShift Serverless에서 특정 기능 및 사용자 정의 리소스의 개발 상태에 대한 중요한 척도입니다. 올바른 API 버전을 사용하지 않는 클러스터에 리소스를 생성하면 배포에 문제가 발생할 수 있습니다.

OpenShift Serverless Operator는 최신 버전을 사용하기 위해 더 이상 사용되지 않는 API를 사용하는 이전 리소스를 자동으로 업그레이드합니다. 예를 들어 **v1beta1**과 같은 이전 **ApiServerSource** API 버전을 사용하는 리소스를 클러스터에 생성한 경우 OpenShift Serverless Operator는 사용 가능하고 **v1beta1** 버전이 더 이상 사용되지 않을 때 API의 **v1** 버전을 사용하도록 이러한 리소스를 자동으로 업데이트합니다.

더 이상 사용되지 않는 API의 이전 버전은 향후 릴리스에서 제거될 수 있습니다. 더 이상 사용되지 않는 API 버전을 사용해도 리소스가 실패하지 않습니다. 그러나 제거된 API 버전을 사용하려고 하면 리소스가 실패합니다. 문제를 방지하기 위해 최신 버전을 사용하도록 매니페스트가 업데이트되었는지 확인합니다.

1.2. 일반적으로 사용 가능한 기술 프리뷰 기능

GA(Generally Available) 기능은 완전하게 지원되며 프로덕션 용도에 적합합니다. TP(기술 프리뷰) 기능은 실험적 기능이며 프로덕션용이 아닙니다. TP 기능에 대한 자세한 내용은 Red Hat Customer Portal의 기술 프리뷰 지원 범위를 참조하십시오.

다음 표에서는 GA 및 TP인 OpenShift Serverless 기능에 대한 정보를 제공합니다.

표 1.1. 일반적으로 OpenShift Container Platform 및 OpenShift Dedicated의 사용 가능한 기술 프리뷰 기능 추적기

기능	1.26	1.27	1.28
kn func	GA	GA	GA
Quarkus 함수	GA	GA	GA
Node.js 함수	TP	TP	GA

기능	1.26	1.27	1.28
TypeScript 함수	TP	TP	GA
Python 함수	-	-	TP
서비스 메시 mTLS	GA	GA	GA
emptyDir 볼륨	GA	GA	GA
HTTPS 리디렉션	GA	GA	GA
Kafka 브로커	GA	GA	GA
Kafka 싱크	GA	GA	GA
Knative 서비스에 대한 Init 컨테이너 지원	GA	GA	GA
Knative 서비스에 대한 PVC 지원	GA	GA	GA
내부 트래픽의 TLS	TP	TP	TP
네임스페이스 범위 브로커	-	TP	TP
멀티컨테이너 지원	-	-	TP

표 1.2. 일반적으로 AWS에서 Red Hat OpenShift Service의 사용 가능 및 기술 프리뷰 기능 추적기

기능	1.26	1.27	1.28
kn func	GA	GA	GA
서비스 메시 mTLS	GA	GA	GA
emptyDir 볼륨	GA	GA	GA
HTTPS 리디렉션	GA	GA	GA
Kafka 브로커	GA	GA	GA
Kafka 싱크	TP	TP	TP
Knative 서비스에 대한 Init 컨테이너 지원	GA	GA	GA
Knative 서비스에 대한 PVC 지원	GA	GA	GA

기능	1.26	1.27	1.28
내부 트래픽의 TLS	TP	TP	TP
네임스페이스 범위 브로커	-	TP	TP
멀티컨테이너 지원	-	-	TP

1.3. 사용되지 않거나 삭제된 기능

GA(Generally Available) 또는 이전 릴리스의 TP(Technology Preview) 기능은 더 이상 사용되지 않거나 삭제되었습니다. 더 이상 사용되지 않는 기능은 여전히 OpenShift Serverless에 포함되어 있으며 계속 지원됩니다. 그러나 이 기능은 향후 릴리스에서 제거될 예정이므로 새로운 배포에는 사용하지 않는 것이 좋습니다.

OpenShift Serverless에서 더 이상 사용되지 않고 삭제된 주요 기능의 최신 목록은 다음 표를 참조하십시오.

표 1.3. OpenShift Container Platform 및 OpenShift Dedicated에서 더 이상 사용되지 않거나 삭제된 기능 추적기

기능	1.20	1.21	1.22에서 1.26 사이	1.27	1.28
KafkaBinding API	더 이상 사용되지 않음	더 이상 사용되지 않음	삭제됨	삭제됨	삭제됨
kn func emit (1.21 이상에서**kn func 호출**)	더 이상 사용되지 않음	삭제됨	삭제됨	삭제됨	삭제됨
서비스 및 Eventing **v1alpha1** API	-	-	-	더 이상 사용되지 않음	더 이상 사용되지 않음
enable-secret-informer-filtering 주석	-	-	-	-	더 이상 사용되지 않음

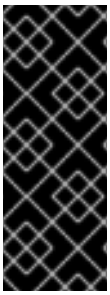
표 1.4. AWS에서 Red Hat OpenShift Service에 대해 더 이상 사용되지 않거나 삭제된 기능 추적기

기능	1.23에서 1.26으로	1.27	1.28
KafkaBinding API	삭제됨	삭제됨	삭제됨
kn func emit (1.21 이상에서**kn func 호출**)	삭제됨	삭제됨	삭제됨

기능	1.23에서 1.26으로	1.27	1.28
서비스 및 Eventing **v1alpha1** API	-	더 이상 사용되지 않음	더 이상 사용되지 않음

1.4. RED HAT OPENSHIFT SERVERLESS 1.28 릴리스 노트

OpenShift Serverless 1.28이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.



중요

OpenShift Container Platform 4.13은 RHEL (Red Hat Enterprise Linux) 9.2를 기반으로 합니다. RHEL 9.2는 FIPS(Federal Information Processing Standards) 검증을 위해 제출되지 않았습니다. Red Hat은 특정 기간 동안 커밋할 수 없지만 RHEL 9.0 및 RHEL 9.2 모듈에 대한 FIPS 검증을 받을 것으로 예상되며 나중에 RHEL 9.x의 마이너 릴리스도 제공됩니다. 업데이트에 대한 정보는 규정 준수 활동 및 정부 표준 지식 베이스 문서에서 확인할 수 있습니다.

1.4.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.7을 사용합니다.
- OpenShift Serverless에서 Knative Eventing 1.7을 사용합니다.
- OpenShift Serverless에서 Kourier 1.7을 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 1.7을 사용합니다.
- OpenShift Serverless에서 Apache Kafka 1.7에 Knative 브로커 구현을 사용합니다.
- **kn func** CLI 플러그인은 이제 **func** 1.9.1 버전을 사용합니다.
- 이제 OpenShift Serverless Functions의 node.js 및 TypeScript 런타임을 GA(일반 사용 가능)로 사용할 수 있습니다.
- OpenShift Serverless Functions의 Python 런타임은 이제 기술 프리뷰로 사용할 수 있습니다.
- Knative Serving에 대한 멀티컨테이너 지원을 기술 프리뷰로 사용할 수 있습니다. 이 기능을 사용하면 단일 Knative 서비스를 사용하여 멀티컨테이너 Pod를 배포할 수 있습니다.
- OpenShift Serverless 1.29 이상에서는 Knative Eventing의 다음 구성 요소가 두 Pod에서 하나로 축소됩니다.
  - **imc-controller**
  - **imc-dispatcher**
  - **mt-broker-controller**
  - **mt-broker-filter**

- **mt-broker-ingress**

- Serving CR에 대한 **서버리스.openshift.io/enable-secret-informer-filtering** 주석이 더 이상 사용되지 않습니다. 주석은 Istio에만 유효하며 Kourier에는 유효하지 않습니다.
  OpenShift Serverless 1.28을 사용하면 OpenShift Serverless Operator에서 **net-istio** 및 **net-kourier** 모두에 환경 변수 **ENABLE_SECRET_INFORMER_FILTERING_BY_CERT_UID** 를 삽입할 수 있습니다.

  시크릿 필터링을 활성화하면 모든 시크릿에 **networking.internal.knative.dev/certificate-uid: "<id>"** 로 레이블이 지정됩니다. 그러지 않으면 Knative Serving에서 이를 탐지하지 않아 오류가 발생합니다. 새 시크릿과 기존 보안의 라벨을 모두 지정해야 합니다.

  다음 OpenShift Serverless 릴리스 중 하나에서 시크릿 필터링은 기본적으로 활성화됩니다. 오류를 방지하려면 시크릿에 미리 레이블을 지정합니다.

1.4.2. 확인된 문제

- 현재 IBM Power, IBM zSystems 및 IBM® LinuxONE의 OpenShift Serverless Functions에서는 Python에 대한 런타임이 지원되지 않습니다.
  이러한 아키텍처에서 node.js, TypeScript 및 Quarkus 함수가 지원됩니다.

- Windows 플랫폼에서는 **app.sh** 파일 권한으로 Source-to-Image 빌더를 사용하여 Python 함수를 로컬로 빌드, 실행 또는 배포할 수 없습니다.
  이 문제를 해결하려면 Linux용 Windows를 사용하십시오.

- Red Hat OpenShift Serverless 1.27 이전에 생성된 **KafkaSource** 리소스는 삭제 시 중단됩니다.
  이 문제를 해결하려면 **KafkaSource** 를 삭제한 후 리소스에서 종료자를 제거합니다.

1.5. RED HAT OPENSHIFT SERVERLESS 1.27 릴리스 노트

OpenShift Serverless 1.27이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

중요

OpenShift Serverless 1.26은 OpenShift Container Platform 4.12에서 완전하게 지원되는 초기 릴리스입니다. OpenShift Serverless 1.25 이상은 OpenShift Container Platform 4.12에 배포되지 않습니다.

따라서 OpenShift Container Platform을 버전 4.12로 업그레이드하기 전에 먼저 OpenShift Serverless를 버전 1.26 또는 1.27로 업그레이드합니다.

1.5.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.6을 사용합니다.

- OpenShift Serverless에서 Knative Eventing 1.6을 사용합니다.

- OpenShift Serverless에서 Kourier 1.6을 사용합니다.

- OpenShift Serverless에서 Knative(**kn**) CLI 1.6을 사용합니다.

- OpenShift Serverless에서 Knative Kafka 1.6을 사용합니다.

- **kn func** CLI 플러그인은 이제 **func** 1.8.1을 사용합니다.

- 네임스페이스 범위 브로커를 이제 기술 프리뷰로 사용할 수 있습니다. 예를 들어 이러한 브로커를 사용하여 RBAC(역할 기반 액세스 제어) 정책을 구현할 수 있습니다.

- **KafkaSink** 는 기본적으로 **CloudEvent** 바이너리 콘텐츠 모드를 사용합니다. 바이너리 콘텐츠 모드는 **CloudEvent** 대신 본문에서 헤더를 사용하므로 구조화된 모드보다 효율적입니다. 예를 들어 HTTP 프로토콜의 경우 HTTP 헤더를 사용합니다.

- OpenShift Container Platform 4.10 이상의 OpenShift 경로를 사용하여 외부 트래픽에 HTTP/2 프로토콜을 통해 gRPC 프레임워크를 사용할 수 있습니다. 이를 통해 클라이언트와 서버 간의 통신 효율성과 속도가 향상됩니다.

- Knative Operator Serving 및 Eventings CRD의 API 버전 **v1alpha1** 은 1.27에서 더 이상 사용되지 않습니다. 이는 향후 버전에서 제거될 예정입니다. Red Hat은 대신 **v1beta1** 버전을 사용할 것을 권장합니다. Serverless Operator를 업그레이드할 때 CRD가 자동으로 업데이트되므로 기존 설치에 영향을 미치지 않습니다.

- 이제 제공 시간 초과 기능이 기본적으로 활성화되어 있습니다. 이를 통해 전송된 각 HTTP 요청에 대한 시간 초과를 지정할 수 있습니다. 이 기능은 기술 프리뷰로 남아 있습니다.

1.5.2. 해결된 문제

- 이전 버전에서는 Knative 서비스가 **Ready** 상태가 되어 로드 밸런서가 준비될 때까지 대기 중인 것으로 보고하는 경우가 있었습니다. 이 문제가 해결되었습니다.

1.5.3. 확인된 문제

- OpenShift Serverless를 Red Hat OpenShift Service Mesh와 통합하면 클러스터에 보안이 너무 많으면 **net-kourier** Pod가 시작 시 메모리 부족이 발생합니다.

- 네임스페이스 범위 브로커는 네임스페이스 범위 브로커를 삭제한 후에도 사용자 네임스페이스에서 **ClusterRoleBinding**을 종료할 수 있습니다.
  이 경우 사용자 네임스페이스에서 **rbac-proxy-reviews-prom-rb-knative-kafka-broker-data-plane-{{.Namespace}}** 이라는 **ClusterRoleBinding** 을 삭제합니다.

- **security.dataPlane.mtls: true** 를 사용하여 SMCP에 **net-istio** 를 사용하고 mTLS를 활성화하면 서비스 메시는 OpenShift Serverless의 **DomainMapping** 을 허용하지 않는 ***.local** 호스트에 대한 **DestinationRules** 를 배포합니다.
  이 문제를 해결하려면 **security.dataPlane.mtls: true** 를 사용하는 대신 **PeerAuthentication** 을 배포하여 mTLS를 활성화합니다.

1.6. RED HAT OPENSHIFT SERVERLESS 1.26 릴리스 노트

OpenShift Serverless 1.26이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.6.1. 새로운 기능

- OpenShift Serverless Functions with Quarkus가 이제 GA입니다.

- OpenShift Serverless에서 Knative Serving 1.5를 사용합니다.

- OpenShift Serverless에서 Knative Eventing 1.5를 사용합니다.

- OpenShift Serverless에서 Kourier 1.5를 사용합니다.

- OpenShift Serverless에서 Knative(**kn**) CLI 1.5를 사용합니다.
- OpenShift Serverless에서 Knative Kafka 1.5를 사용합니다.
- OpenShift Serverless에서 Knative Operator 1.3을 사용합니다.
- **kn func** CLI 플러그인에서 **func** 1.8.1을 사용합니다.
- PVC(영구 볼륨 클레임)는 이제 GA입니다. PVC는 Knative 서비스에 영구 데이터 스토리지를 제공합니다.
- 이제 새 트리거 필터 기능을 개발자 프리뷰로 사용할 수 있습니다. 이를 통해 사용자는 각 이벤트에 대해 각 표현식이 true 또는 false로 평가되는 필터 표현식 세트를 지정할 수 있습니다.
  새 트리거 필터를 활성화하려면 Operator 구성 맵에서 **KnativeEventing** 유형의 섹션에 **new-trigger-filters: enabled** 항목을 추가합니다.

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

- Knative Operator 1.3은 **operator.knative.dev** 용으로 API의 업데이트된 **v1beta1** 버전을 추가합니다.
  KnativeServing 및 **KnativeEventing** 사용자 정의 리소스 구성 맵의 **v1alpha1** 에서 **v1beta1** 으로 업데이트하려면 **apiVersion** 키를 편집합니다.

  KnativeServing 사용자 정의 리소스 구성 맵의 예

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

  KnativeEventing 사용자 정의 리소스 구성 맵의 예

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

1.6.2. 해결된 문제

- 이전에는 Kafka 브로커, Kafka 소스 및 Kafka 싱크에 대해 FIPS(Federal Information Processing Standards) 모드가 비활성화되었습니다. 이 문제는 해결되었으며 FIPS 모드를 사용할 수 있습니다.

1.6.3. 확인된 문제

- **security.dataPlane.mtls: true** 를 사용하여 SMCP에 **net-istio** 를 사용하고 mTLS를 활성화하면 서비스 메시는 OpenShift Serverless의 **DomainMapping** 을 허용하지 않는 ***.local** 호스트에 대한 **DestinationRules** 를 배포합니다.

이 문제를 해결하려면 **security.dataPlane.mtls: true** 를 사용하는 대신 **PeerAuthentication** 을 배포하여 mTLS를 활성화합니다.

추가 리소스

- 새 트리거 필터에 대한 Knative 설명서

1.7. RED HAT OPENSHIFT SERVERLESS 1.25.0 릴리스 정보

OpenShift Serverless 1.25.0이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.7.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.4를 사용합니다.
- OpenShift Serverless에서 Knative Eventing 1.4를 사용합니다.
- OpenShift Serverless에서 Kourier 1.4를 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 1.4를 사용합니다.
- OpenShift Serverless에서 Knative Kafka 1.4를 사용합니다.
- **kn func** CLI 플러그인에서 **func** 1.7.0을 사용합니다.
- 함수를 만들고 배포하기 위한 IDE(통합 개발 환경) 플러그인을 이제 Visual Studio Code 및 IntelliJ 에서 사용할 수 있습니다.
- Knative Kafka 브로커는 이제 GA입니다. Knative Kafka 브로커는 Apache Kafka를 직접 대상으로 하는 Knative 브로커 API의 고성능 구현입니다.
  대신 MT-Channel-Broker를 사용하지 않는 것이 좋지만 Knative Kafka 브로커를 대신 사용하는 것이 좋습니다.
- Knative Kafka 싱크는 이제 GA입니다. **KafkaSink** 은 **CloudEvent** 를 가져와서 Apache Kafka 주제로 보냅니다. 이벤트는 구조화된 또는 바이너리 콘텐츠 모드로 지정할 수 있습니다.
- 내부 트래픽에 TLS를 활성화하는 것은 이제 기술 프리뷰로 사용할 수 있습니다.

1.7.2. 해결된 문제

- 이전 버전에서는 활성 상태 프로브가 실패한 후 컨테이너가 다시 시작되면 Knative Serving에 준비 프로브가 실패한 문제가 있었습니다. 이 문제가 해결되었습니다.

1.7.3. 확인된 문제

- Kafka 브로커, Kafka 소스 및 Kafka 싱크에 대해 FIPS(Federal Information Processing Standards) 모드가 비활성화됩니다.
- **SinkBinding** 오브젝트는 서비스에 대한 사용자 정의 버전 이름을 지원하지 않습니다.
- Knative Serving 컨트롤러 Pod는 클러스터에서 시크릿을 감시하기 위해 새 정보를 추가합니다. 정보 제공자에는 캐시에 시크릿이 포함되어 있어 컨트롤러 Pod의 메모리 사용량이 증가합니다. Pod가 메모리가 부족하면 배포에 대한 메모리 제한을 늘려 문제를 해결할 수 있습니다.

- **security.dataPlane.mtls: true** 를 사용하여 SMCP에 **net-istio** 를 사용하고 mTLS를 활성화하면 서비스 메시는 OpenShift Serverless의 **DomainMapping** 을 허용하지 않는 ***.local** 호스트에 대한 **DestinationRules** 를 배포합니다.
  이 문제를 해결하려면 **security.dataPlane.mtls: true** 를 사용하는 대신 **PeerAuthentication** 을 배포하여 mTLS를 활성화합니다.

OpenShift Container Platform의 추가 리소스

- TLS 인증 구성

1.8. RED HAT OPENSHIFT SERVERLESS 1.24.0 릴리스 노트

OpenShift Serverless 1.24.0이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.8.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.3을 사용합니다.
- OpenShift Serverless에서 Knative Eventing 1.3을 사용합니다.
- OpenShift Serverless에서 Kourier 1.3을 사용합니다.
- OpenShift Serverless에서 Knative **kn** CLI 1.3을 사용합니다.
- OpenShift Serverless에서 Knative Kafka 1.3을 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.24를 사용합니다.
- 이제 Knative 서비스에 대한 init 컨테이너 지원을 일반적으로 사용할 수 있습니다(GA).
- OpenShift Serverless 논리를 개발자 프리뷰로 사용할 수 있습니다. 서버리스 애플리케이션을 관리하기 위해 선언적 워크플로 모델을 정의할 수 있습니다.
- OpenShift Container Platform의 경우 OpenShift Serverless에서 비용 관리 서비스를 사용할 수 있습니다.

1.8.2. 해결된 문제

- OpenShift Serverless를 Red Hat OpenShift Service Mesh와 통합하면 클러스터에 보안이 너무 많으면 **net-istio-controller** Pod가 시작 시 메모리 부족이 발생합니다.
  이제 보안 필터링을 활성화하여 **net-istio-controller** 가 **networking.internal.knative.dev/certificate-uid** 라벨이 있는 시크릿만 고려하여 필요한 메모리 양을 줄일 수 있습니다.
- OpenShift Serverless Functions 기술 프리뷰에서는 기본적으로 Cloud Native Buildpacks 를 사용하여 컨테이너 이미지를 빌드합니다.

1.8.3. 확인된 문제

- Kafka 브로커, Kafka 소스 및 Kafka 싱크에 대해 FIPS(Federal Information Processing Standards) 모드가 비활성화됩니다.
- OpenShift Serverless 1.23에서는 KafkaBindings 및 **kafka-binding** webhook에 대한 지원이 제거되었습니다. 그러나 기존 **kafkabindings.webhook.kafka.sources.knative.dev**

MutatingWebhookConfiguration 은 더 이상 존재하지 않는 **kafka-source-webhook** 서비스를 가리키는 남아 있을 수 있습니다.
클러스터에서 KafkaBindings의 특정 사양에 대해 **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** 을 구성하여 Deployments, Knative Services 또는 Jobs와 같은 다양한 리소스에 이벤트를 업데이트하고, Webhook를 통해 실패할 수 있습니다.

이 문제를 해결하려면 OpenShift Serverless 1.23으로 업그레이드한 후 **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** 을 수동으로 삭제합니다.

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

- **security.dataPlane.mtls: true** 를 사용하여 SMCP에 **net-istio** 를 사용하고 mTLS를 활성화하면 서비스 메시는 OpenShift Serverless의 **DomainMapping** 을 허용하지 않는 ***.local** 호스트에 대한 **DestinationRules** 를 배포합니다.
이 문제를 해결하려면 **security.dataPlane.mtls: true** 를 사용하는 대신 **PeerAuthentication** 을 배포하여 mTLS를 활성화합니다.

1.9. RED HAT OPENSHIFT SERVERLESS 1.23.0 릴리스 노트

OpenShift Serverless 1.23.0을 사용할 수 있습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.9.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.2를 사용합니다.
- OpenShift Serverless에서 Knative Eventing 1.2를 사용합니다.
- OpenShift Serverless에서 Kourier 1.2를 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 1.2를 사용합니다.
- OpenShift Serverless에서 Knative Kafka 1.2를 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.24를 사용합니다.
- Kafka 브로커와 함께 **kafka.eventing.knative.dev/external.topic** 주석을 사용할 수 있습니다. 이 주석을 사용하면 브로커가 자체 내부 주제를 생성하는 대신 기존 외부 관리 주제를 사용할 수 있습니다.
- **kafka-ch-controller** 및 **kafka-webhook** Kafka 구성 요소가 더 이상 존재하지 않습니다. 이러한 구성 요소는 **kafka-webhook-eventing** 구성 요소로 교체되었습니다.
- OpenShift Serverless Functions 기술 프리뷰에서는 기본적으로 S2I(Source-to-Image)를 사용하여 컨테이너 이미지를 빌드합니다.

1.9.2. 확인된 문제

- Kafka 브로커, Kafka 소스 및 Kafka 싱크에 대해 FIPS(Federal Information Processing Standards) 모드가 비활성화됩니다.
- Kafka 브로커를 포함하는 네임스페이스를 삭제하면 브로커 전에 브로커의 **auth.secret.ref.name** 시크릿이 삭제되면 네임스페이스 종료자가 제거되지 못할 수 있습니다.

- 많은 Knative 서비스에서 OpenShift Serverless를 실행하면 Knative 활성화기 Pod가 기본 메모리 제한 600MB에 가깝게 실행될 수 있습니다. 메모리 사용량이 이 제한에 도달하면 이러한 Pod가 재시작될 수 있습니다. 활성화기 배포에 대한 요청 및 제한은 **KnativeServing** 사용자 정의 리소스를 수정하여 구성할 수 있습니다.

```
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
```

- Cloud Native Buildpacks 를 함수의 로컬 빌드 전략으로 사용하는 경우 **kn func** 는 podman을 자동으로 시작하거나 원격 데몬에 대한 SSH 터널을 사용할 수 없습니다. 이러한 문제에 대한 해결 방법은 함수를 배포하기 전에 로컬 개발 컴퓨터에서 Docker 또는 podman 데몬이 이미 실행되고 있어야 하는 것입니다.

- On-cluster 함수 빌드는 현재 Quarkus 및 Golang 런타임에 실패합니다. Node, Typescript, Python 및 Springboot 런타임에 대해 올바르게 작동합니다.

- **security.dataPlane.mtls: true** 를 사용하여 SMCP에 **net-istio** 를 사용하고 mTLS를 활성화하면 서비스 메시는 OpenShift Serverless의 **DomainMapping** 을 허용하지 않는 ***.local** 호스트에 대한 **DestinationRules** 를 배포합니다.
  이 문제를 해결하려면 **security.dataPlane.mtls: true** 를 사용하는 대신 **PeerAuthentication** 을 배포하여 mTLS를 활성화합니다.

OpenShift Container Platform의 추가 리소스

- S2I(Source-to-Image)

1.10. RED HAT OPENSHIFT SERVERLESS 1.22.0 릴리스 노트

OpenShift Serverless 1.22.0이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.10.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.1을 사용합니다.

- OpenShift Serverless에서 Knative Eventing 1.1을 사용합니다.

- OpenShift Serverless에서 Kourier 1.1을 사용합니다.

- OpenShift Serverless에서 Knative(**kn**) CLI 1.1을 사용합니다.

- OpenShift Serverless에서 Knative Kafka 1.1을 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.23을 사용합니다.
- Knative 서비스에 대한 Init 컨테이너 지원을 기술 프리뷰로 사용할 수 있습니다.
- Knative 서비스에 대한 PVC(영구 볼륨 클레임) 지원을 기술 프리뷰로 사용할 수 있습니다.
- **knative-serving**,**knative-serving-ingress**,**knative-eventing** 및 **knative-kafka** 시스템 네임스페이스에 기본적으로 **knative.openshift.io/part-of: "openshift-serverless"** 레이블이 있습니다.
- **Knative Eventing - Kafka Broker/Trigger** 대시보드가 추가되어 웹 콘솔에서 Kafka 브로커를 시각화하고 트리거할 수 있습니다.
- **Knative Eventing - KafkaSink** 대시보드가 추가되어 웹 콘솔에서 KafkaSink 지표를 시각화할 수 있습니다.
- **Knative Eventing - Broker/Trigger** 대시보드를 **Knative Eventing - 채널 기반 브로커/Trigger** 라고 합니다.
- **knative.openshift.io/part-of: "openshift-serverless"** 레이블은 **knative.openshift.io/system-namespace** 라벨을 대체합니다.
- Knative Serving YAML 구성 파일의 이름 지정 스타일은 camel case(**ExampleName**)에서 하이픈 스타일(**example-name**)으로 변경되었습니다. 이 릴리스부터 Knative Serving YAML 구성 파일을 생성하거나 편집할 때 하이픈 스타일 표기법을 사용합니다.

1.10.2. 확인된 문제

- Kafka 브로커, Kafka 소스 및 Kafka 싱크에 대해 FIPS(Federal Information Processing Standards) 모드가 비활성화됩니다.

1.11. RED HAT OPENSHIFT SERVERLESS 1.21.0 릴리스 정보

OpenShift Serverless 1.21.0이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.11.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 1.0 사용
- OpenShift Serverless에서 Knative Eventing 1.0을 사용합니다.
- OpenShift Serverless에서 Kourier 1.0을 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 1.0을 사용합니다.
- OpenShift Serverless에서 Knative Kafka 1.0을 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.21을 사용합니다.
- Kafka 싱크를 기술 프리뷰로 사용할 수 있습니다.
- Knative 오픈 소스 프로젝트는 kebab-cased 키를 일관되게 사용하는 것이 좋습니다. 결과적으로 이전에 OpenShift Serverless 1.18.0 릴리스 노트에 언급된 **defaultExternalScheme** 키는 더 이상 사용되지 않으며 **default-external-scheme** 키로 대체됩니다. 키에 대한 사용 지침은 동일하게 유

지됩니다.

1.11.2. 해결된 문제

- OpenShift Serverless 1.20.0에는 **kn 이벤트 전송 사용에 영향을 미치는 이벤트 전달 문제가 있어 서비스에 이벤트를 보냅니다**. 이제 이 문제가 해결되었습니다.
- OpenShift Serverless 1.20.0(**func** 0.20)에서는 **http** 템플릿으로 생성된 TypeScript 함수가 클러스터에 배포되지 않았습니다. 이제 이 문제가 해결되었습니다.
- OpenShift Serverless 1.20.0(**func** 0.20)에서는 **gcr.io** 레지스트리를 사용하여 함수를 배포할 때 오류와 함께 실패했습니다. 이제 이 문제가 해결되었습니다.
- OpenShift Serverless 1.20.0(**func** 0.20)에서 **kn func create** 명령을 사용하여 Springboot 함수 프로젝트 디렉터리를 생성한 다음 **kn func build** 명령을 실행하면 오류 메시지와 함께 실패했습니다. 이제 이 문제가 해결되었습니다.
- OpenShift Serverless 1.19.0(**func** 0.19)에서는 podman을 사용하여 일부 런타임에서 함수를 빌드할 수 없었습니다. 이제 이 문제가 해결되었습니다.

1.11.3. 확인된 문제

- 현재 도메인 매핑 컨트롤러는 현재 지원되지 않는 경로가 포함된 브로커의 URI를 처리할 수 없습니다.
  즉, **DomainMapping** CR(사용자 정의 리소스)을 사용하여 사용자 정의 도메인을 브로커에 매핑하려면 브로커의 수신 서비스를 사용하여 **DomainMapping** CR을 구성하고 브로커의 정확한 경로를 사용자 정의 도메인에 추가해야 합니다.

  DomainMapping CR의 예

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

  브로커의 URI는 < **domain-name>/<broker-namespace>/<broker-name>입니다**.

1.12. RED HAT OPENSHIFT SERVERLESS 1.20.0 릴리스 노트

OpenShift Serverless 1.20.0이 출시되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.12.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 0.26을 사용합니다.
- OpenShift Serverless에서 Knative Eventing 0.26을 사용합니다.

- OpenShift Serverless에서 Kourier 0.26을 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 0.26을 사용합니다.
- OpenShift Serverless에서 Knative Kafka 0.26을 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.20을 사용합니다.
- Kafka 브로커를 기술 프리뷰로 사용할 수 있습니다.

중요

현재 기술 프리뷰에 있는 Kafka 브로커는 FIPS에서 지원되지 않습니다.

- **kn 이벤트** 플러그인은 이제 기술 프리뷰로 사용할 수 있습니다.
- **kn service create** 명령의 **--min-scale** 및 **--max-scale** 플래그가 더 이상 사용되지 않습니다. 대신 **--scale-min** 및 **--scale-max** 플래그를 사용합니다.

1.12.2. 확인된 문제

- OpenShift Serverless는 HTTPS를 사용하는 기본 주소로 Knative 서비스를 배포합니다. 이벤트를 클러스터 내부의 리소스로 전송할 때 발신자에 클러스터 인증 기관(CA)이 구성되어 있지 않습니다. 이로 인해 클러스터가 전역에서 허용되는 인증서를 사용하지 않는 한 이벤트 전달이 실패합니다.
  예를 들어 공개적으로 액세스 가능한 주소로 이벤트 전달은 다음과 같습니다.

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

  반면, 서비스에서 사용자 정의 CA에서 발행한 HTTPS 인증서가 있는 공용 주소를 사용하는 경우 이 전달이 실패합니다.

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

  브로커 또는 채널과 같은 다른 주소 지정 가능한 개체에 이벤트를 보내는 것은 이 문제의 영향을 받지 않으며 예상대로 작동합니다.

- Kafka 브로커는 현재 FIPS(Federal Information Processing Standards) 모드가 활성화된 클러스터에서 작동하지 않습니다.
- kn func create 명령을 사용하여 Springboot 함수 프로젝트 디렉터리를 **생성하면 kn func build** 명령을 나중에 실행하면 이 오류 메시지와 함께 실패합니다.

```
[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
```

  이 문제를 해결하려면 함수 구성 파일 **func.yaml**.yaml에서 **builder** 속성을 **gcr.io/paketo-buildpacks/builder:base** 로 변경할 수 있습니다.

- **gcr.io** 레지스트리를 사용하여 함수를 배포하면 다음 오류 메시지와 함께 실패합니다.

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

이 문제를 해결하려면 **quay.io** 또는 **docker.io** 와 같은 **gcr.io** 와 다른 레지스트리를 사용합니다.

- **http** 템플릿으로 생성된 TypeScript 함수는 클러스터에 배포하지 못합니다.
이 문제를 해결하려면 **func.yaml** 파일에서 다음 섹션을 교체합니다.

```
buildEnvs: []
```

다음과 같이 수행합니다.

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

- **func** 버전 0.20에서 일부 런타임은 podman을 사용하여 함수를 빌드할 수 없습니다. 다음과 유사한 오류 메시지가 표시될 수 있습니다.

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

  - 이 문제에 대한 해결방법은 다음과 같습니다.

    a. service **ExecStart** 정의에 **--time=0** 을 추가하여 podman 서비스를 업데이트합니다.

    서비스 구성 예

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

    b. 다음 명령을 실행하여 podman 서비스를 다시 시작하십시오.

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

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

  - 또는 TCP를 사용하여 podman API를 노출할 수 있습니다.

    ```
    $ 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 릴리스 정보

OpenShift Serverless 1.19.0이 공개되었습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.13.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 0.25를 사용합니다.
- OpenShift Serverless에서 Knative Eventing 0.25를 사용합니다.
- OpenShift Serverless에서 Kourier 0.25를 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 0.25를 사용합니다.

- OpenShift Serverless에서 Knative Kafka 0.25를 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.19를 사용합니다.
- **KafkaBinding** API는 OpenShift Serverless 1.19.0에서 더 이상 사용되지 않으며 향후 릴리스에서 제거됩니다.
- 이제 HTTPS 리디렉션이 지원되며 클러스터 또는 각 Knative 서비스에 대해 전역적으로 구성할 수 있습니다.

1.13.2. 해결된 문제

- 이전 릴리스에서는 Kafka 채널 디스패처가 응답하기 전에 로컬 커밋이 성공할 때까지만 대기했으며 Apache Kafka 노드 실패의 경우 이벤트가 손실되었을 수 있었습니다. Kafka 채널 디스패처에서 응답하기 전에 모든 동기화 복제본이 커밋될 때까지 기다립니다.

1.13.3. 확인된 문제

- **func** 버전 0.19에서 일부 런타임은 podman을 사용하여 함수를 빌드할 수 없습니다. 다음과 유사한 오류 메시지가 표시될 수 있습니다.

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

  - 이 문제에 대한 해결방법은 다음과 같습니다.
    a. service **ExecStart** 정의에 **--time=0** 을 추가하여 podman 서비스를 업데이트합니다.

    서비스 구성 예

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

    b. 다음 명령을 실행하여 podman 서비스를 다시 시작하십시오.

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

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

  - 또는 TCP를 사용하여 podman API를 노출할 수 있습니다.

```
$ 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 릴리스 정보

OpenShift Serverless 1.18.0을 사용할 수 있습니다. 이에는 OpenShift Container Platform의 OpenShift Serverless와 관련된 새로운 기능, 변경 사항, 알려진 문제가 포함되어 있습니다.

1.14.1. 새로운 기능

- OpenShift Serverless에서 Knative Serving 0.24.0을 사용합니다.
- OpenShift Serverless에서 Knative Eventing 0.24.0을 사용합니다.

- OpenShift Serverless에서 Kourier 0.24.0을 사용합니다.
- OpenShift Serverless에서 Knative(**kn**) CLI 0.24.0을 사용합니다.
- OpenShift Serverless에서 Knative Kafka 0.24.7을 사용합니다.
- **kn func** CLI 플러그인에서 **func** 0.18.0을 사용합니다.
- 향후 OpenShift Serverless 1.19.0 릴리스에서는 보안을 강화하기 위해 외부 경로의 URL 체계가 기본적으로 HTTPS로 설정됩니다.
  이 변경 사항을 워크로드에 적용하지 않으려면 1.19.0으로 업그레이드하기 전에 **KnativeServing** CR(사용자 정의 리소스)에 다음 YAML을 추가하여 기본 설정을 덮어쓸 수 있습니다.

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

  1.18.0에서 변경 사항을 이미 적용하려면 다음 YAML을 추가합니다.

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

- 예정된 OpenShift Serverless 1.19.0 릴리스에서 Kourier Gateway가 노출되는 기본 서비스 유형은 **LoadBalancer** 가 아닌 **ClusterIP** 가 됩니다.
  이 변경 사항을 워크로드에 적용하지 않으려면 1.19.0으로 업그레이드하기 전에 **KnativeServing** CR(사용자 정의 리소스)에 다음 YAML을 추가하여 기본 설정을 덮어쓸 수 있습니다.

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

- OpenShift Serverless에서 **emptyDir** 볼륨을 사용할 수 있습니다. 자세한 내용은 Knative Serving에 대한 OpenShift Serverless 설명서를 참조하십시오.
- **kn func** 를 사용하여 함수를 생성할 때rust 템플릿을 사용할 수 있습니다.

1.14.2. 해결된 문제

- Camel-K의 이전 1.4 버전은 OpenShift Serverless 1.17.0과 호환되지 않았습니다. Camel-K의 문제가 해결되었으며 Camel-K 버전 1.4.1은 OpenShift Serverless 1.17.0에서 사용할 수 있습니다.
- 이전에는 Kafka 채널 또는 새 Kafka 소스에 대한 새 서브스크립션을 생성한 경우 새로 생성된 서브스크립션 또는 싱크에서 준비 상태를 보고한 후 Kafka 데이터 플레인에서 메시지를 디스패치할 준비가 된 지연이 있었습니다.

그 결과 데이터 플레인에서 준비 상태를 보고하지 않은 시간 동안 전송된 메시지가 구독자 또는 싱크로 전달되지 않았을 수 있었습니다.

OpenShift Serverless 1.18.0에서는 문제가 수정되어 초기 메시지가 더 이상 손실되지 않습니다. 이 문제에 대한 자세한 내용은 Knowledgebase 문서 #6343981 을 참조하십시오.

1.14.3. 확인된 문제

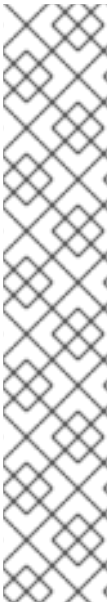
- 이전 버전의 Knative **kn** CLI에서는 이전 버전의 Knative Serving 및 Knative Eventing API를 사용할 수 있습니다. 예를 들어 **kn** CLI의 버전 0.23.2에서는 **v1alpha1** API 버전을 사용합니다. 반면 최신 OpenShift Serverless 릴리스에서는 이전 API 버전을 더 이상 지원하지 않을 수 있습니다. 예를 들어 OpenShift Serverless 1.18.0은 **kafkasources.sources.knative.dev** API의 버전 **v1alpha1** 을 더 이상 지원하지 않습니다.

  결과적으로 이전 버전의 Knative **kn** CLI를 최신 OpenShift Serverless에서 사용하면 **kn** 에서 오래된 API를 찾을 수 없기 때문에 오류가 발생할 수 있습니다. 예를 들어 **kn** CLI 버전 0.23.2는 OpenShift Serverless 1.18.0에서 작동하지 않습니다.

  문제를 방지하려면 OpenShift Serverless 릴리스에 사용 가능한 최신 **kn** CLI 버전을 사용하십시오. OpenShift Serverless 1.18.0의 경우 Knative **kn** CLI 0.24.0을 사용합니다.

2장. OPENSHIFT SERVERLESS 개요

OpenShift Serverless에서는 개발자가 OpenShift Container Platform에서 서버리스 이벤트 중심 애플리케이션을 생성하고 배포할 수 있는 Kubernetes 기본 구성 블록을 제공합니다. OpenShift Serverless는 엔터프라이즈급 서버리스 플랫폼을 활성화하여 하이브리드 및 멀티 클라우드 환경에 이식성과 일관성을 제공하는 오픈 소스 Knative 프로젝트를 기반으로 합니다.



참고

OpenShift Serverless는 Red Hat OpenShift Serverless의 다른 주기로 릴리스되므로 OpenShift Serverless 설명서는 이제 제품의 각 마이너 버전에 대해 별도의 문서 세트로 제공됩니다.

OpenShift Serverless 문서는 https://docs.openshift.com/serverless/ 에서 확인할 수 있습니다.

특정 버전에 대한 문서는 버전 선택기 드롭다운을 사용하거나 URL에 버전을 추가하여 직접 확인할 수 있습니다(예: https://docs.openshift.com/serverless/1.28).

또한 OpenShift Serverless 설명서는 https://access.redhat.com/documentation/en-us/red_hat_openshift_serverless/ 의 Red Hat 포털에서도 사용할 수 있습니다.

OpenShift Serverless 라이프 사이클 및 지원되는 플랫폼에 대한 자세한 내용은 플랫폼 라이프 사이클 정책을 참조하십시오.

2.1. 추가 리소스

- 사용자 정의 리소스 정의를 사용하여 Kubernetes API 확장
- 사용자 정의 리소스 정의에서 리소스 관리
- 서버리스란 무엇인가?

3장. KNATIVE SERVING

Knative Serving에서는 클라우드 네이티브 애플리케이션을 생성, 배포, 관리하려는 개발자를 지원합니다. OpenShift Container Platform 클러스터에서 서버리스 워크로드의 동작을 정의하고 제어하는 Kubernetes CRD(사용자 정의 리소스 정의)로 오브젝트 세트를 제공합니다.

개발자는 이러한 CRD를 사용하여 복잡한 사용 사례를 다룰 때 구성 요소로 사용할 수 있는 CR(사용자 정의 리소스) 인스턴스를 생성합니다. 예를 들면 다음과 같습니다.

- 신속한 서버리스 컨테이너 배포.
- Pod 자동 스케일링.

3.1. KNATIVE SERVING 리소스

Service

service.serving.knative.dev CRD를 사용하면 워크로드의 라이프사이클을 자동으로 관리하여 네트워크를 통해 애플리케이션을 배포하고 연결할 수 있습니다. 이 CRD는 사용자 생성 서비스 또는 사용자 정의 리소스가 변경될 때마다 경로, 구성, 새 버전을 생성합니다. Knative의 개발자 상호 작용은 대부분 서비스 수정을 통해 수행됩니다.

버전

revision.serving.knative.dev CRD는 워크로드에 수행된 각 수정의 코드 및 구성에 대한 시점 스냅샷입니다. 버전은 변경할 수 없는 오브젝트이며 필요한 기간에 유지할 수 있습니다.

Route

route.serving.knative.dev CRD는 네트워크 끝점을 하나 이상의 버전에 매핑합니다. 부분 트래픽 및 이름이 지정된 경로를 포함하여 다양한 방법으로 트래픽을 관리할 수 있습니다.

설정

configuration.serving.knative.dev CRD는 배포에 필요한 상태를 유지 관리합니다. 코드와 구성을 명확하게 분리합니다. 구성을 수정하면 새 버전이 생성됩니다.

4장. KNATIVE EVENTING

OpenShift Container Platform에서 Knative Eventing을 사용하면 개발자가 서버리스 애플리케이션에 이벤트 중심 아키텍처를 사용할 수 있습니다. 이벤트 중심 아키텍처는 이벤트 생산자와 이벤트 소비자 간의 분리된 관계에 대한 개념을 기반으로 합니다.

이벤트 생산자가 이벤트를 생성하고 이벤트 *싱크* 또는 소비자는 이벤트를 수신합니다. Knative Eventing은 표준 HTTP POST 요청을 사용하여 이벤트 프로듀서와 싱크 사이에서 이벤트를 전송하고 수신합니다. 이러한 이벤트는 이벤트를 모든 프로그래밍 언어로 생성, 구문 분석, 전송, 수신할 수 있도록 CloudEvents 사양을 준수합니다.

Knative Eventing에서는 다음 유스 케이스를 지원합니다.

소비자를 생성하지 않고 이벤트 게시

이벤트를 HTTP POST에 브로커로 보내고 바인딩을 사용하여 이벤트를 생성하는 애플리케이션에서 대상 구성을 분리할 수 있습니다.

게시자를 생성하지 않고 이벤트 사용

트리거를 사용하면 이벤트 특성을 기반으로 브로커의 이벤트를 사용할 수 있습니다. 애플리케이션은 이벤트를 HTTP POST로 수신합니다.

Knative Eventing에서는 다양한 싱크 유형으로 전달할 수 있도록 다음과 같이 여러 Kubernetes 리소스에서 구현할 수 있는 일반 인터페이스를 정의합니다.

주소 지정 가능 리소스

HTTP를 통해 전달되는 이벤트를 이벤트의 **status.address.url** 필드에 정의된 주소로 수신 및 승인할 수 있습니다. Kubernetes **Service** 리소스도 주소 지정 가능 인터페이스의 조건을 충족합니다.

호출 가능한 리소스

HTTP를 통해 전달되는 이벤트를 수신하고 변환하여 HTTP 응답 페이로드에서 **0** 또는 **1**개의 새 이벤트를 반환합니다. 반환된 이벤트는 외부 이벤트 소스의 이벤트를 처리하는 것과 동일한 방식으로 추가로 처리할 수 있습니다.

4.1. APACHE KAFKA용 KNATIVE 브로커 사용

Apache Kafka의 Knative 브로커 구현에서는 지원되는 Apache Kafka 메시지 스트리밍 플랫폼을 OpenShift Serverless와 함께 사용할 수 있는 통합 옵션을 제공합니다. Kafka는 이벤트 소스, 채널, 브로커 및 이벤트 싱크 기능에 대한 옵션을 제공합니다.

Apache Kafka용 Knative 브로커는 다음과 같은 추가 옵션을 제공합니다.

- Kafka 소스
- Kafka 채널
- Kafka 브로커
- Kafka 싱크

4.2. 추가 리소스

- **KnativeKafka** 사용자 정의 리소스 설치.
- Red Hat AMQ Streams documentation

- Apache Kafka 문서의 Red Hat AMQ Streams TLS 및 SASL
- 이벤트 전달

5장. OPENSHIFT SERVERLESS FUNCTIONS 정보

OpenShift Serverless Functions를 사용하면 개발자가 상태 비저장 이벤트 중심 함수를 OpenShift Container Platform에서 Knative 서비스로 생성하고 배포할 수 있습니다. **kn func** CLI는 Knative **kn** CLI의 플러그인으로 제공됩니다. **kn func** CLI를 사용하여 컨테이너 이미지를 클러스터에서 Knative 서비스로 생성, 빌드, 배포할 수 있습니다.

5.1. 포함된 런타임

OpenShift Serverless Functions는 다음 런타임에 대한 기본 기능을 생성하는 데 사용할 수 있는 템플릿을 제공합니다.

- Quarkus
- Node.js
- TypeScript

5.2. 다음 단계

- 함수 시작하기

6장. OPENSHIFT SERVERLESS 지원

이 설명서에 설명된 절차를 수행하는 데 어려움이 있는 경우 Red Hat Customer Portal(http://access.redhat.com)을 방문하십시오. Red Hat 고객 포털을 사용하여 Red Hat 제품에 대한 기술 지원 문서의 Red Hat 지식베이스를 검색하거나 검색할 수 있습니다. Red Hat GSS(글로벌 지원 서비스)에 지원 케이스를 제출하거나 기타 제품 문서에 액세스할 수도 있습니다.

이 가이드를 개선하기 위한 제안이 있거나 오류를 발견한 경우 가장 관련 있는 문서 구성 요소에 대한 Jira 문제를 제출할 수 있습니다. 콘텐츠를 쉽게 찾을 수 있도록 섹션 번호, 가이드 이름, OpenShift Serverless 버전과 같은 구체적인 세부 사항을 입력하십시오.

참고

클러스터 크기 요구 사항을 정의하는 다음 섹션에서는 이러한 배포에 적용됩니다.

- OpenShift Container Platform
- OpenShift Dedicated

6.1. RED HAT 지식베이스 정보

Red Hat 지식베이스는 Red Hat의 제품과 기술을 최대한 활용할 수 있도록 풍부한 콘텐츠를 제공합니다. Red Hat 지식베이스는 Red Hat 제품 설치, 설정 및 사용에 대한 기사, 제품 문서 및 동영상으로 구성되어 있습니다. 또한 알려진 문제에 대한 솔루션을 검색할 수 있으며, 간결한 근본 원인 설명 및 해결 단계를 제공합니다.

6.2. RED HAT 지식베이스 검색

Red Hat OpenShift Serverless 문제가 발생한 경우 초기 검색을 수행하여 솔루션이 이미 Red Hat 지식베이스에 있는지 확인할 수 있습니다.

사전 요구 사항

- Red Hat 고객 포털 계정이 있어야 합니다.

프로세스

1. Red Hat 고객 포털 에 로그인합니다.
2. 기본 Red Hat 고객 포털 검색 필드에 다음과 같이 문제와 관련된 키워드 및 문자열을 입력하십시오.
   - OpenShift Container Platform 구성 요소 (**etcd** 등)
   - 관련 절차 (예: **installation** 등)
   - 명시적 실패와 관련된 경고, 오류 메시지 및 기타 출력
3. **Search**를 클릭합니다
4. **OpenShift Container Platform** 제품 필터를 선택합니다.
5. **Knowledgebase** 콘텐츠 유형 필터를 선택합니다.

6.3. 지원 케이스 제출

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- OpenShift CLI(**oc**)가 설치되어 있습니다.
- Red Hat 고객 포털 계정이 있어야 합니다.
- Red Hat 표준 또는 프리미엄 서브스크립션이 있습니다.

프로세스

1. Red Hat 고객 포털 에 로그인하고 **SUPPORT CASES → Open a case**를 선택합니다.
2. 문제에 대한 적절한 카테고리(예: **Defect/Bug**), 제품(**OpenShift Container Platform**) 및 아직 자동 입력되지 않은 경우 제품 버전을 선택합니다.
3. 보고되는 문제와 관련이 있을 수 있는 권장 Red Hat 지식베이스 솔루션 목록을 확인합니다. 제안된 문서로 문제가 해결되지 않으면 **Continue**을 클릭합니다.
4. 문제의 증상 및 예상 동작에 대한 자세한 정보와 함께 간결하지만 구체적인 문제 요약을 입력합니다.
5. 보고되는 문제와 관련있는 제안된 Red Hat 지식베이스 솔루션 목록을 확인하십시오. 케이스 작성 과정에서 더 많은 정보를 제공하면 목록이 구체화됩니다. 제안된 문서로 문제가 해결되지 않으면 **Continue**을 클릭합니다.
6. 제시된 계정 정보가 정확한지 확인하고 필요한 경우 적절하게 수정합니다.
7. Red Hat OpenShift Serverless Cluster ID가 자동으로 입력되었는지 확인합니다. 그렇지 않은 경우 클러스터 ID를 수동으로 가져옵니다.
   - OpenShift Container Platform 웹 콘솔을 사용하여 클러스터 ID를 수동으로 가져오려면 다음을 수행합니다.
     a. **Home → Dashboards → Overview**로 이동합니다.
     b. **Details** 섹션의 **Cluster ID** 필드에서 값을 찾습니다.
   - 또는 OpenShift Container Platform 웹 콘솔을 통해 새 지원 케이스를 열고 클러스터 ID를 자동으로 입력할 수 있습니다.
     a. 툴바에서 **(?) Help → Open Support Case**로 이동합니다.
     b. **Cluster ID** 값이 자동으로 입력됩니다.
   - OpenShift CLI (**oc**)를 사용하여 클러스터 ID를 얻으려면 다음 명령을 실행합니다.

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

8. 프롬프트가 표시되면 다음 질문을 입력한 후 **Continue**를 클릭합니다.
   - 이 문제가 어디에서 발생했습니까? 어떤 시스템 환경을 사용하고 있습니까?

- 이 동작이 언제 발생했습니까? 발생 빈도는 어떻게 됩니까? 반복적으로 발생합니까? 특정 시간에만 발생합니까?
- 이 문제의 발생 기간 및 비즈니스에 미치는 영향에 대한 정보를 제공해주십시오.

9. 관련 진단 데이터 파일을 업로드하고 **Continue**를 클릭합니다.

oc adm must-gather 명령을 사용하여 수집된 데이터와 해당 명령으로 수집되지 않은 특정 문제와 관련된 데이터를 제공하는 것이 좋습니다

1. 관련 케이스 관리 세부 정보를 입력하고 **Continue**를 클릭합니다.
2. 케이스 세부 정보를 미리보고 **Submit**을 클릭합니다.

6.4. 지원을 위한 진단 정보 수집

지원 케이스를 여는 경우 클러스터에 대한 디버깅 정보를 Red Hat 지원에 제공하는 것이 좋습니다. **must-gather** 툴을 사용하면 OpenShift Serverless 관련 데이터를 포함하여 OpenShift Container Platform 클러스터에 대한 진단 정보를 수집할 수 있습니다. 즉각 지원을 받을 수 있도록 OpenShift Container Platform 및 OpenShift Serverless 둘 다에 대한 진단 정보를 제공하십시오.

6.4.1. must-gather 툴 정보

oc adm must-gather CLI 명령은 다음과 같은 문제를 디버깅하는 데 필요한 정보를 클러스터에서 수집합니다.

- 리소스 정의
- 서비스 로그

기본적으로 **oc adm must-gather** 명령은 기본 플러그인 이미지를 사용하여 **./must-gather.local** 에 씁니다.

또는 다음 섹션에 설명된 대로 적절한 인수를 사용하여 명령을 실행하여 특정 정보를 수집할 수 있습니다.

- 하나 이상의 특정 기능과 관련된 데이터를 수집하려면 다음 섹션에 나열된 대로 이미지에 **--image** 인수를 사용합니다.
  예를 들면 다음과 같습니다.

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

- 감사 로그를 수집하려면 다음 섹션에 설명된 대로 **-- /usr/bin/gather_audit_logs** 인수를 사용하십시오.
  예를 들면 다음과 같습니다.

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

참고

감사 로그는 파일 크기를 줄이기 위해 기본 정보 세트의 일부로 수집되지 않습니다.

oc adm must-gather 를 실행하면 클러스터의 새 프로젝트에서 임의의 이름의 새 Pod가 생성됩니다. 해당 Pod에 대한 데이터가 수집되어 **must-gather.local**로 시작하는 새 디렉터리에 저장됩니다. 이 디렉터리는 현재 작업 중인 디렉터리에 생성되어 있습니다.

예를 들면 다음과 같습니다.

```
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
...
```

필요한 경우 **--run-namespace** 옵션을 사용하여 특정 네임스페이스에서 **oc adm must-gather** 명령을 실행할 수 있습니다.

예를 들면 다음과 같습니다.

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

6.4.2. OpenShift Serverless 데이터 수집 정보

oc adm must-gather CLI 명령을 사용하면 OpenShift Serverless와 연관된 기능 및 오브젝트를 포함하여 클러스터에 대한 정보를 수집할 수 있습니다. **must-gather**로 OpenShift Serverless 데이터를 수집하려면 OpenShift Serverless 이미지 및 설치된 OpenShift Serverless 버전의 이미지 태그를 지정해야 합니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.

절차

- **oc adm must-gather** 명령을 사용하여 데이터 수집

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

  명령 예

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