Red Hat OpenShift Serverless 1.32

서버리스 설치

Serverless Operator, Knative CLI, Knative Serving 및 Knative Eventing 설치

Last Updated: 2024-06-27

Red Hat OpenShift Serverless 1.32 서버리스 설치

Serverless Operator, Knative CLI, Knative Serving 및 Knative Eventing 설치

법적 공지

초록

이 문서에서는 Serverless Operator, Knative CLI, Knative Serving 및 Knative Eventing을 설치하는 방법을 설명합니다. 또한 Apache Kafka의 Knative 구성 및 Serverless Functions 구성에 대한 세부 정보도 제공합니다.

차례

1장. OPENSHIFT SERVERLESS 설치 준비

OpenShift Serverless를 설치하기 전에 지원되는 구성 및 사전 요구 사항에 대한 다음 정보를 확인하십시오.

OpenShift Container Platform의 경우:

- OpenShift Serverless는 제한된 네트워크 환경에서 설치할 수 있습니다.
- OpenShift Serverless는 현재 단일 클러스터의 다중 테넌트 구성에서 사용할 수 없습니다.

1.1. 지원되는 구성

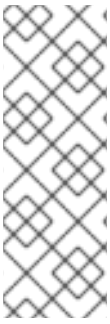
OpenShift Serverless에 지원되는 일련의 기능, 구성, 통합과 현재 버전 및 이전 버전은 지원되는 구성 페이지에서 확인할 수 있습니다.

1.2. OPENSHIFT CONTAINER PLATFORM의 확장성 및 성능

OpenShift Serverless는 3개의 기본 노드와 3개의 작업자 노드로 구성된 구성으로 테스트되었으며 각각 64개의 CPU, 457GB의 메모리, 각각 394GB의 스토리지가 있습니다.

이 구성을 사용하여 생성할 수 있는 최대 Knative 서비스 수는 3000입니다. 1개의 Knative 서비스에서 3개의 Kubernetes 서비스를 생성하기 때문에 OpenShift Container Platform Kubernetes 서비스 제한은 10,000 입니다.

응답 시간이 0인 평균 규모는 약 3.4초이며, 최대 응답 시간은 8초이고 간단한 Quarkus 애플리케이션의 경우 4.5초의 99.9%가 됩니다. 이러한 시간은 애플리케이션 및 애플리케이션의 런타임에 따라 다를 수 있습니다.



참고

클러스터 크기 요구 사항을 정의하는 방법에 대한 다음 섹션은 다음 배포에 적용됩니다.

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

1.3. 클러스터 크기 요구 사항 정의

OpenShift Serverless를 설치하고 사용하려면 클러스터의 크기를 올바르게 조정해야 합니다.

참고

다음 요구 사항은 클러스터의 작업자 시스템 풀에만 관련이 있습니다. 컨트롤 플레인 노드는 일반 스케줄링에 사용되지 않으며 요구 사항에서 생략됩니다.

OpenShift Serverless를 사용해야 하는 최소 요구 사항은 CPU 10개 및 40GB 메모리가 있는 클러스터입니다. 기본적으로 각 Pod는 CPU ~ 400m을 요청하므로 최소 요구 사항은 이 값을 기반으로 합니다.

OpenShift Serverless를 실행하기 위한 총 크기 요구 사항은 설치된 구성 요소와 배포된 애플리케이션에 따라 다르며 배포에 따라 다를 수 있습니다.

1.4. OPENSHIFT CONTAINER PLATFORM의 컴퓨팅 머신 세트를 사용하여 클러스터 스케일링

OpenShift Container Platform **MachineSet** API를 사용하여 클러스터를 원하는 크기로 수동으로 확장할 수 있습니다. 최소 요구 사항은 일반적으로 기본 컴퓨팅 머신 세트 중 하나를 두 개의 추가 시스템으로 확장해야 함을 의미합니다. 컴퓨팅 머신 세트 수동 스케일링을 참조하십시오.

1.4.1. 고급 사용 사례에 대한 추가 요구 사항

OpenShift Container Platform에서 로깅 또는 미터링과 같은 고급 사용 사례의 경우 더 많은 리소스를 배포해야 합니다. 이러한 사용 사례에 대한 권장 요구 사항은 CPU 24개 및 96GB 메모리입니다.

클러스터에 HA(고가용성)를 활성화하면 Knative Serving 컨트롤 플레인을 복제할 때마다 0.5 ~ 1.5코어 및 200MB ~ 2GB의 메모리가 필요합니다. HA는 기본적으로 일부 Knative Serving 구성 요소에 사용됩니다. "고가용성 복제본 구성" 설명서에 따라 HA를 비활성화할 수 있습니다.

1.5. OPENSHIFT CONTAINER PLATFORM 설명서의 추가 리소스

- 제한된 네트워크에서 Operator Lifecycle Manager 사용
- OperatorHub 이해
- 클러스터 기능

2장. OPENSHIFT SERVERLESS OPERATOR 설치

OpenShift Serverless Operator를 설치하면 OpenShift Container Platform 클러스터에서 Knative Serving, Knative Eventing 및 Knative Kafka를 설치 및 사용할 수 있습니다. OpenShift Serverless Operator는 클러스터에 대한 Knative CRD(사용자 정의 리소스 정의)를 관리하고 각 구성 요소에 대한 개별 구성 맵을 직접 수정하지 않고도 구성할 수 있습니다.

2.1. OPENSHIFT SERVERLESS OPERATOR 리소스 요구 사항

다음 샘플 설정은 OpenShift Serverless Operator 설치에 대한 최소 리소스 요구 사항을 추정하는 데 도움이 될 수 있습니다. 특정 요구 사항이 크게 다를 수 있으며 OpenShift Serverless 사용이 증가함에 따라 증가할 수 있습니다.

샘플 설정에 사용되는 테스트 모음에는 다음과 같은 매개변수가 있습니다.

- 다음을 사용하는 OpenShift Container Platform 클러스터
  - 10개의 작업자 (8 vCPU, 16GiB 메모리)
  - Kafka 전용 작업자 3개
  - Prometheus 전용 작업자 2개
  - 서버리스 및 테스트 배포에 남아 있는 작업자 5개
- 89 테스트 시나리오는 주로 컨트롤 플레인을 사용하는 데 중점을 두고 병렬로 실행되는 테스트 시나리오입니다. 테스트 시나리오에는 일반적으로 메모리 내 채널, Kafka 채널, 메모리 내 브로커 또는 Kafka 브로커를 통해 배포 또는 Knative 서비스로 이벤트를 전송하는 Knative 서비스가 있습니다.
- 48 다시 생성 시나리오로, 테스트 시나리오가 반복적으로 삭제되고 다시 생성됩니다.
- 41 테스트 전체에서 이벤트가 전송되는 안정적인 시나리오는 천천히 실행되지만 지속적으로 실행됩니다.
- 테스트 설정에는 전체적으로 다음이 포함됩니다.
  - 170 Knative 서비스
  - 20 메모리 내 채널
  - 24개의 Kafka 채널
  - 52 서브스크립션
  - 42 브로커
  - 68 트리거

다음 표에서는 테스트 모음에서 검색한 Highly-Available (HA) 설정에 대한 최소 리소스 요구 사항을 자세히 설명합니다.

Component	RAM 리소스	CPU 리소스
OpenShift Serverless Operator	1GB	0.2 코어

Component	RAM 리소스	CPU 리소스
Knative Serving	5GB	2.5 코어
Knative Eventing	2GB	0.5 코어
Apache Kafka용 Knative 브로커	6GB	코어 1개
합계	14GB	4.2 코어

다음 표에서는 테스트 모음에서 검색한 비HA 설정에 대한 최소 리소스 요구 사항을 자세히 설명합니다.

Component	RAM 리소스	CPU 리소스
OpenShift Serverless Operator	1GB	0.2 코어
Knative Serving	2.5GB	1.2 코어
Knative Eventing	1GB	0.2 코어
Apache Kafka용 Knative 브로커	6GB	코어 1개
합계	10.5GB	2.6 코어

2.2. 웹 콘솔에서 OPENSHIFT SERVERLESS OPERATOR 설치

OpenShift Container Platform 웹 콘솔을 사용하여 OperatorHub에서 OpenShift Serverless Operator를 설치할 수 있습니다. 이 Operator를 설치하면 Knative 구성 요소를 설치하고 사용할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Container Platform의 경우 클러스터에 Marketplace 기능이 활성화되어 있거나 Red Hat Operator 카탈로그 소스가 수동으로 구성되어 있습니다.
- 웹 콘솔에 로그인했습니다.

프로세스

1. 웹 콘솔에서 **Operator → OperatorHub** 페이지로 이동합니다.
2. 스크롤하거나 **키워드로 필터링 상자**에 키워드 **서버리스**를 입력하여 OpenShift Serverless Operator를 찾습니다.
3. Operator에 대한 정보를 확인하고 **설치**를 클릭합니다.
4. **Operator 설치** 페이지에서 다음을 수행합니다.

a. **설치 모드**는 **클러스터의 모든 네임스페이스(기본값)**입니다. 이 모드에서는 클러스터의 모든 네임스페이스를 감시하고 사용할 수 있도록 기본 **openshift-serverless** 네임스페이스에 Operator를 설치합니다.

b. **설치된 네임스페이스**는 **openshift-serverless** 입니다.

c. **Update Channel**을 선택합니다.

- **stable** 채널을 사용하면 OpenShift Serverless Operator의 안정적인 최신 릴리스를 설치할 수 있습니다. **stable** 채널이 기본값입니다.
- 다른 버전을 설치하려면 해당 **stable-x.y** 채널을 지정합니다(예: **stable-1.29**).

d. **stable** 채널을 **업데이트 채널**로 선택합니다. **stable** 채널을 사용하면 안정적인 최신 릴리스의 OpenShift Pipelines Operator를 설치할 수 있습니다.

e. **자동** 또는 **수동** 승인 전략을 선택합니다.

5. 이 OpenShift Container Platform 클러스터에서 선택한 네임스페이스에서 Operator를 사용할 수 있도록 하려면 **설치**를 클릭합니다.

6. **카탈로그 → Operator 관리** 페이지에서 OpenShift Serverless Operator 서브스크립션 설치 및 업그레이드 진행 상황을 모니터링할 수 있습니다.

a. **수동** 승인 전략을 선택한 경우 설치 계획을 검토하고 승인할 때까지 서브스크립션의 업그레이드 상태가 **업그레이드 중**으로 유지됩니다. **설치 계획** 페이지에서 승인한 후 subscription 업그레이드 상태가 **최신**으로 이동합니다.

b. **자동** 승인 전략을 선택한 경우 업그레이드 상태가 개입 없이 **최신** 상태로 확인되어야 합니다.

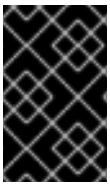
검증

서브스크립션의 업그레이드 상태가 **최신**인 경우 **카탈로그 → 설치된 Operator**를 선택하여 OpenShift Serverless Operator가 표시되고 관련 네임스페이스에서 **상태**가 최종적으로 **InstallSucceeded**로 표시되는지 확인합니다.

그러지 않은 경우 다음을 수행합니다.

1. **카탈로그 → Operator 관리** 페이지로 전환한 후 **Operator 서브스크립션** 및 **설치 계획** 탭의 **상태**에 장애나 오류가 있는지 검사합니다.

2. **워크로드 → Pod** 페이지의 **openshift-serverless** 프로젝트에서 문제를 보고하는 Pod의 로그를 확인하여 추가로 문제를 해결합니다.



중요

OpenShift Serverless에서 Red Hat OpenShift distributed tracing을 사용하려면 Knative Serving 또는 Knative Eventing을 설치하기 전에 Red Hat OpenShift distributed tracing을 설치하고 구성해야 합니다.

2.3. CLI에서 OPENSHIFT SERVERLESS OPERATOR 설치

CLI를 사용하여 OperatorHub에서 OpenShift Serverless Operator를 설치할 수 있습니다. 이 Operator를 설치하면 Knative 구성 요소를 설치하고 사용할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Container Platform의 경우 클러스터에 Marketplace 기능이 활성화되어 있거나 Red Hat Operator 카탈로그 소스가 수동으로 구성되어 있습니다.
- 클러스터에 로그인했습니다.

프로세스

1. **Namespace**,**OperatorGroup**, **Subscription** 오브젝트가 포함된 YAML 파일을 생성하여 OpenShift Serverless Operator에 네임스페이스를 서브스크립션합니다. 예를 들어 다음 콘텐츠를 사용하여 **서버리스-subscription.yaml** 파일을 생성합니다.

   서브스크립션 예

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

   (1) Operator의 채널 이름입니다. **stable** 채널을 사용하면 OpenShift Serverless Operator의 최신 안정적인 버전을 설치할 수 있습니다. 다른 버전을 설치하려면 해당 **stable-x.y** 채널을 지정합니다(예: **stable-1.29**).

   (2) 등록할 Operator의 이름입니다. OpenShift Serverless Operator의 경우 이는 항상 **서버리스 운영자** 입니다.

   (3) Operator를 제공하는 CatalogSource의 이름입니다. 기본 OperatorHub 카탈로그 소스에 **redhat-operators** 를 사용합니다.

   (4) CatalogSource의 네임스페이스입니다. 기본 OperatorHub 카탈로그 소스에는 **openshift-marketplace**를 사용합니다.

2. **Subscription** 오브젝트를 생성합니다.

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

검증

CSV(클러스터 서비스 버전)가 **성공** 단계에 도달했는지 확인합니다.

명령 예

```
$ oc get csv
```

출력 예

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

중요

OpenShift Serverless에서 Red Hat OpenShift distributed tracing을 사용하려면 Knative Serving 또는 Knative Eventing을 설치하기 전에 Red Hat OpenShift distributed tracing을 설치하고 구성해야 합니다.

2.4. 글로벌 구성

OpenShift Serverless Operator는 **KnativeServing** 및 **KnativeEventing** 사용자 정의 리소스에서 시스템 구성 맵에 값을 전파하는 등 Knative 설치의 글로벌 구성을 관리합니다. 수동으로 적용되는 구성 맵에 대한 업데이트는 Operator에서 덮어씁니다. 그러나 Knative 사용자 정의 리소스를 수정하면 이러한 구성 맵의 값을 설정할 수 있습니다.

Knative에는 접두사 **config-** 로 이름이 지정된 여러 구성 맵이 있습니다. 모든 Knative 구성 맵은 적용되는 사용자 정의 리소스와 동일한 네임스페이스에 생성됩니다. 예를 들어 **knative-serving** 네임스페이스에 **KnativeServing** 사용자 정의 리소스가 생성되면 이 네임스페이스에 모든 Knative Serving 구성 맵도 생성됩니다.

Knative 사용자 정의 리소스의 **spec.config** 에는 **config-< name >이라는 각 구성 맵에 대해 하나의 <name** > 항목이 있으며 구성 맵 **데이터에** 사용되는 값이 있습니다.

2.5. OPENSHIFT CONTAINER PLATFORM에 대한 추가 리소스

- 사용자 정의 리소스 정의에서 리소스 관리
- 영구저장장치 이해
- 사용자 정의 PKI 구성

2.6. 다음 단계

- OpenShift Serverless Operator를 설치한 후 Knative Serving을 설치하거나 Knative Eventing을 설치할 수 있습니다.

3장. KNATIVE CLI 설치

Knative(**kn**) CLI에는 자체 로그인 메커니즘이 없습니다. 클러스터에 로그인하려면 OpenShift CLI(**oc**)를 설치하고 **oc login** 명령을 사용해야 합니다. CLI의 설치 옵션은 운영 체제에 따라 다를 수 있습니다.

운영 체제에 OpenShift CLI(**oc**)를 설치하고 **oc** 로 로그인하는 방법에 대한 자세한 내용은 OpenShift CLI 시작하기 설명서를 참조하십시오.

OpenShift Serverless는 Knative(**kn**) CLI를 사용하여 설치할 수 없습니다. 클러스터 관리자는 OpenShift Serverless Operator 설치 설명서에 설명된 대로 OpenShift Serverless Operator 를 설치하고 Knative 구성 요소를 설정해야 합니다.

중요

최신 OpenShift Serverless 릴리스에서 이전 버전의 Knative(**kn**) CLI를 사용하려는 경우 API를 찾을 수 없으며 오류가 발생합니다.

예를 들어 Knative Serving 및 Knative Eventing API의 1.3 버전을 사용하는 1.24.0 OpenShift Serverless 릴리스와 함께 버전 1.2를 사용하는 Knative (**kn**) CLI의 1.23.0 릴리스를 사용하는 경우 CLI는 오래된 1.2 API 버전을 계속 찾기 때문에 CLI가 작동하지 않습니다.

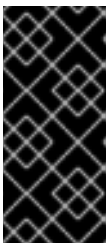
문제를 방지하려면 OpenShift Serverless 릴리스에 최신 Knative (**kn**) CLI 버전을 사용하고 있는지 확인하십시오.

3.1. OPENSHIFT CONTAINER PLATFORM 웹 콘솔을 통한 KNATIVE CLI 설치

OpenShift Container Platform 웹 콘솔을 사용하면 간소화되고 직관적인 사용자 인터페이스를 통해 Knative(**kn**) CLI를 설치할 수 있습니다. OpenShift Serverless Operator가 설치되면 OpenShift Container Platform 웹 콘솔의 **명령줄 툴** 페이지에서 Linux (amd64, s390x, ppc64le), macOS 또는 Windows용 Knative (**kn**) CLI를 다운로드할 수 있는 링크가 표시됩니다.

사전 요구 사항

- OpenShift Container Platform 웹 콘솔에 로그인했습니다.
- OpenShift Serverless Operator 및 Knative Serving이 OpenShift Container Platform 클러스터에 설치되어 있습니다.



중요

libc를 사용할 수 없는 경우 CLI 명령을 실행할 때 다음과 같은 오류가 표시될 수 있습니다.

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

- 이 프로세스에 확인 단계를 사용하려면 OpenShift(**oc**) CLI를 설치해야 합니다.

프로세스

1. **명령줄 툴** 페이지에서 Knative(**kn**) CLI를 다운로드합니다. 웹 콘솔의 오른쪽 상단에 있는  아이콘을 클릭하고 목록에서 **명령줄 툴을 선택하여 명령줄** 툴에 액세스할 수 있습니다.

2. 아카이브의 압축을 풉니다.

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

3. **kn** 바이너리를 **PATH**의 디렉터리로 이동합니다.

4. **PATH**를 확인하려면 다음을 실행합니다.

```
$ echo $PATH
```

검증

- 다음 명령을 실행하여 올바른 Knative CLI 리소스 및 경로가 생성되었는지 확인합니다.

```
$ oc get ConsoleCLIDownload
```

출력 예

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

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

출력 예

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

3.2. RPM 패키지 관리자를 사용하여 LINUX용 KNATIVE CLI 설치

RHEL(Red Hat Enterprise Linux)의 경우 **yum** 또는 **dnf** 와 같은 패키지 관리자를 사용하여 Knative(**kn**) CLI를 RPM으로 설치할 수 있습니다. 이를 통해 시스템에서 Knative CLI 버전을 자동으로 관리할 수 있습니다. 예를 들어 **dnf upgrade** 와 같은 명령을 사용하면 새 버전을 사용할 수 있는 경우 **kn** 을 포함한 모든 패키지를 업그레이드합니다.

사전 요구 사항

- Red Hat 계정에 활성 OpenShift Container Platform 서브스크립션이 있어야 합니다.

프로세스

1. Red Hat Subscription Manager에 등록합니다.

```
# subscription-manager register
```

2. 최신 서브스크립션 데이터를 가져옵니다.

```
# subscription-manager refresh
```

3. 등록된 시스템에 서브스크립션을 연결합니다.

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

1 활성 OpenShift Container Platform 서브스크립션의 풀 ID

4. Knative (**kn**) CLI에 필요한 리포지토리를 활성화합니다.
   - Linux (x86_64, amd64)

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

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

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

   - Linux on IBM Power (ppc64le)

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

5. 패키지 관리자를 사용하여 Knative(**kn**) CLI를 RPM으로 설치합니다.

yum 명령 예

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

3.3. RPM 패키지 관리자와 함께 설치된 KNATIVE CLI의 버전 잠금

최신은 아니지만 유지 관리 단계에 있는 Knative(**kn**) CLI 버전을 사용해야 할 수 있습니다. **kn** 버전을 잠그면 업그레이드할 수 없습니다.

프로세스

1. DNF 패키지 관리자의 **versionlock** 플러그인을 설치합니다.

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

2. 다음을 실행하여 **kn** 버전을 잠급니다.

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

이 명령은 OpenShift Serverless 1.29 릴리스 시 유지 관리 단계에 있는 Knative 1.7 기반 버전으로 **kn** 을 잠급니다.

3.4. 잠긴 버전으로 KNATIVE CLI 업그레이드

이전에 version-locked된 Knative (**kn**) CLI 버전을 수동으로 업그레이드할 수 있습니다.

프로세스

프로세스

1. 업그레이드된 패키지를 사용할 수 있는지 확인합니다.

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

2. **kn** 의 버전 잠금을 제거합니다.

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

1. **kn** 을 새 버전으로 잠급니다.

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

+ 이 예제에서는 Knative 1.8을 기반으로 하는 **kn** 버전을 사용합니다.

1. 사용 가능한 새 버전으로 업그레이드:

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

3.5. LINUX의 KNATIVE CLI 설치

RPM 또는 다른 패키지 관리자가 설치되지 않은 Linux 배포를 사용하는 경우 Knative(**kn**) CLI를 바이너리 파일로 설치할 수 있습니다. 이렇게 하려면 **tar.gz** 아카이브의 압축을 풀고 **PATH** 의 디렉터리에 바이너리를 추가해야 합니다.

사전 요구 사항

- RHEL 또는 Fedora를 사용하지 않는 경우 **libc** 가 라이브러리 경로의 디렉터리에 설치되어 있는지 확인합니다.

중요

libc를 사용할 수 없는 경우 CLI 명령을 실행할 때 다음과 같은 오류가 표시될 수 있습니다.

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

프로세스

1. 관련 Knative (**kn**) CLI **tar.gz** 아카이브를 다운로드합니다.
   - Linux (x86_64, amd64)
   - Linux on IBM zSystems and IBM® LinuxONE (s390x)
   - Linux on IBM Power (ppc64le)

   Serverless 클라이언트 다운로드 미러의 해당 버전으로 이동하여 **kn** 버전을 다운로드할 수도 있습니다.

2. 아카이브의 압축을 풉니다.

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

3. **kn** 바이너리를 **PATH**의 디렉터리로 이동합니다.
4. **PATH**를 확인하려면 다음을 실행합니다.

```
$ echo $PATH
```

3.6. MACOS용 KNATIVE CLI 설치

macOS를 사용하는 경우 Knative (**kn**) CLI를 바이너리 파일로 설치할 수 있습니다. 이렇게 하려면 **tar.gz** 아카이브의 압축을 풀고 **PATH** 의 디렉터리에 바이너리를 추가해야 합니다.

프로세스

1. Knative(**kn**) CLI **tar.gz** 아카이브를 다운로드합니다.
   Serverless 클라이언트 다운로드 미러의 해당 버전으로 이동하여 **kn** 버전을 다운로드할 수도 있습니다.
2. 아카이브의 압축을 해제하고 압축을 풉니다.
3. **kn** 바이너리를 **PATH**의 디렉터리로 이동합니다.
4. **PATH**를 확인하려면 터미널 창을 열고 다음을 실행합니다.

```
$ echo $PATH
```

3.7. WINDOWS용 KNATIVE CLI 설치

Windows를 사용하는 경우 Knative (**kn**) CLI를 바이너리 파일로 설치할 수 있습니다. 이렇게 하려면 ZIP 아카이브의 압축을 풀고 **PATH** 의 디렉터리에 바이너리를 추가해야 합니다.

프로세스

1. Knative (**kn**) CLI ZIP 아카이브를 다운로드합니다.
   Serverless 클라이언트 다운로드 미러의 해당 버전으로 이동하여 **kn** 버전을 다운로드할 수도 있습니다.
2. ZIP 프로그램으로 아카이브의 압축을 풉니다.
3. **kn** 바이너리를 **PATH**의 디렉터리로 이동합니다.
4. **PATH**를 확인하려면 명령 프롬프트를 열고 다음 명령을 실행합니다.

```
C:\> path
```

4장. KNATIVE SERVING 설치

Knative Serving을 설치하면 클러스터에서 Knative 서비스 및 기능을 생성할 수 있습니다. 또한 애플리케이션의 자동 스케일링 및 네트워킹 옵션과 같은 추가 기능을 사용할 수 있습니다.

OpenShift Serverless Operator를 설치한 후 기본 설정을 사용하여 Knative Serving을 설치하거나 **KnativeServing** 사용자 정의 리소스(CR)에서 고급 설정을 구성할 수 있습니다. **KnativeServing** CR의 구성 옵션에 대한 자세한 내용은 글로벌 구성 을 참조하십시오.

중요

OpenShift Serverless에서 Red Hat OpenShift distributed tracing을 사용하려면 Knative Serving을 설치하기 전에 Red Hat OpenShift distributed tracing을 설치하고 구성해야 합니다.

4.1. 웹 콘솔을 사용하여 KNATIVE SERVING 설치

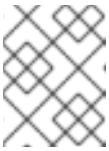
OpenShift Serverless Operator를 설치한 후 OpenShift Container Platform 웹 콘솔을 사용하여 Knative Serving을 설치합니다. 기본 설정을 사용하여 Knative Serving을 설치하거나 **KnativeServing** 사용자 정의 리소스(CR)에서 고급 설정을 구성할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Container Platform 웹 콘솔에 로그인했습니다.
- OpenShift Serverless Operator를 설치했습니다.

프로세스

1. OpenShift Container Platform 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator**로 이동합니다.
2. 페이지 상단에 있는 **프로젝트** 드롭다운이 **Project: knative-serving**으로 설정되어 있는지 확인합니다.
3. OpenShift Serverless Operator의 **제공되는 API** 목록에서 **Knative Serving**을 클릭하여 **Knative Serving** 탭으로 이동합니다.
4. **Knative Serving 생성** 을 클릭합니다.
5. **Knative Serving 생성** 페이지에서 **생성**을 클릭하면 기본 설정을 사용하여 Knative Serving을 설치할 수 있습니다.
   제공된 양식을 사용하여 **KnativeServing** 오브젝트를 편집하거나 YAML을 편집하여 Knative Serving 설치에 대한 설정을 수정할 수도 있습니다.
   - 해당 양식은 **KnativeServing** 오브젝트 생성을 완전히 제어할 필요가 없는 단순한 구성에 사용하는 것이 좋습니다.
   - **KnativeServing** 오브젝트 생성을 완전히 제어해야 하는 복잡한 구성의 경우 YAML을 편집하는 것이 좋습니다. **Knative Serving 생성** 페이지의 오른쪽 상단에 있는 **YAML 편집** 링크를 클릭하여 YAML에 액세스할 수 있습니다.
     양식을 작성하거나 YAML을 수정한 후 **생성**을 클릭합니다.



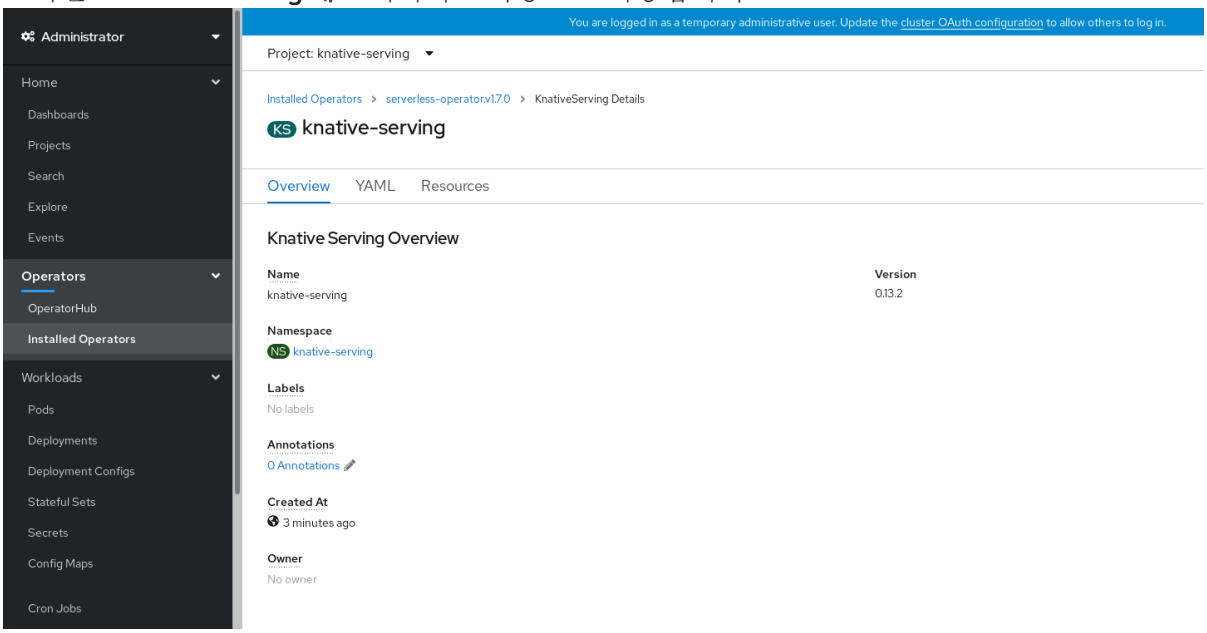
참고

KnativeServing 사용자 정의 리소스 정의의 구성 옵션에 대한 자세한 내용은 *고급 설치 구성 옵션* 설명서를 참조하십시오.

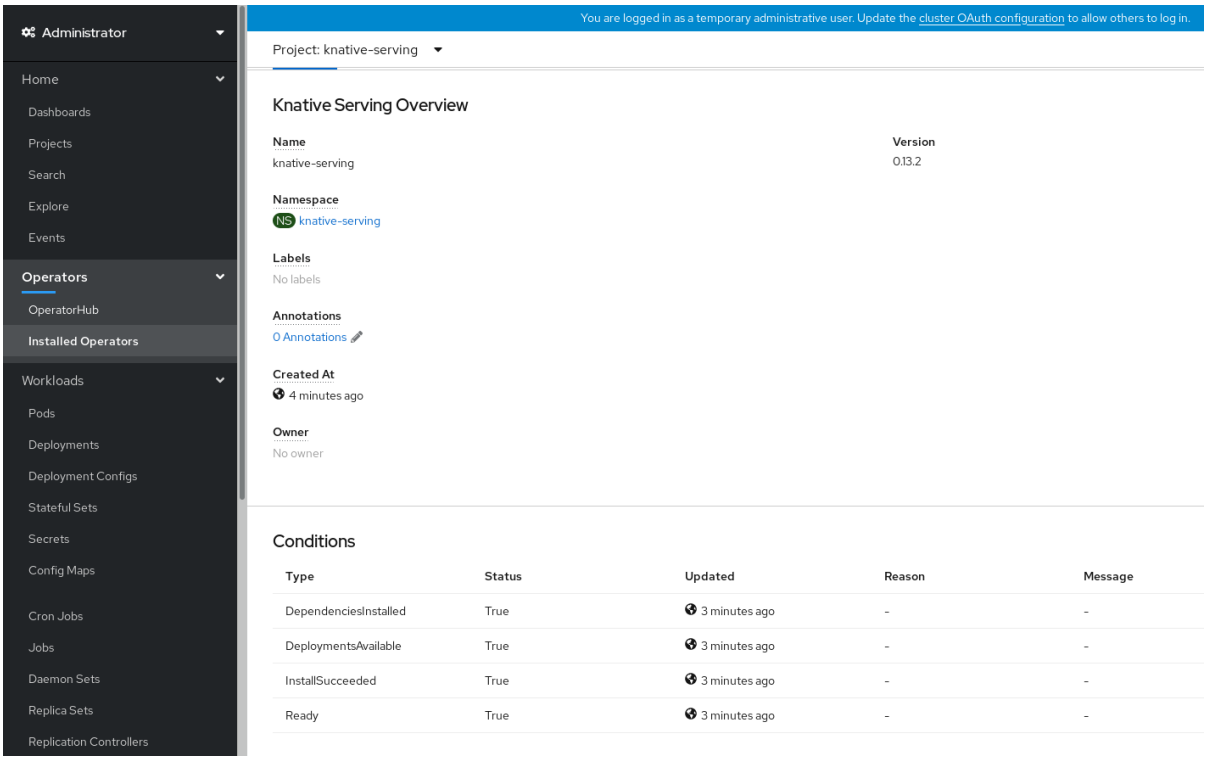
6. Knative Serving을 설치하면 **KnativeServing** 오브젝트가 생성되고 **Knative Serving** 탭으로 자동으로 이동합니다. 리소스 목록에 **knative-serving** 사용자 정의 리소스가 표시됩니다.

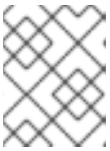
검증

1. **Knative Serving** 탭에서 **knative-serving** 사용자 정의 리소스를 클릭합니다.
2. 그러면 **Knative Serving 개요** 페이지로 자동으로 이동합니다.



3. **조건** 목록을 보려면 아래로 스크롤합니다.
4. 예제 이미지에 표시된 대로 상태가 **True**인 조건 목록이 표시되어야 합니다.





참고

Knative Serving 리소스를 생성하는 데 몇 초가 걸릴 수 있습니다. **리소스** 탭에서 해당 상태를 확인할 수 있습니다.

5. 조건이 **알 수 없음** 또는 **False** 상태인 경우 몇 분 정도 기다렸다가 리소스가 생성된 것을 확인한 후 다시 확인하십시오.

4.2. YAML을 사용하여 KNATIVE SERVING 설치

OpenShift Serverless Operator를 설치한 후 기본 설정을 사용하여 Knative Serving을 설치하거나 **KnativeServing** 사용자 정의 리소스(CR)에서 고급 설정을 구성할 수 있습니다. 다음 절차에 따라 YAML 파일 및 **oc** CLI를 사용하여 Knative Serving을 설치할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Serverless Operator를 설치했습니다.
- OpenShift CLI(**oc**)를 설치합니다.

프로세스

1. **serving.yaml**이라는 파일을 생성하고 다음 예제 YAML을 이 파일에 복사합니다.

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

2. **serving.yaml** 파일을 적용합니다.

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

검증

1. 설치가 완료되었는지 확인하려면 다음 명령을 입력합니다.

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

출력 예

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

참고

Knative Serving 리소스를 생성하는 데 몇 초가 걸릴 수 있습니다.

조건이 **알 수 없음** 또는 **False** 상태인 경우 몇 분 정도 기다렸다가 리소스가 생성된 것을 확인한 후 다시 확인하십시오.

2. Knative Serving 리소스가 생성되었는지 확인합니다.

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

출력 예

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

3. 필요한 네트워킹 구성 요소가 자동으로 생성된 **knative-serving-ingress** 네임스페이스에 설치되어 있는지 확인합니다.

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

출력 예

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

4.3. 추가 리소스

- Kourier 및 Istio 인그레스

4.4. 다음 단계

- Knative 이벤트 중심 아키텍처를 사용하려면 Knative Eventing을 설치할 수 있습니다.

5장. KNATIVE EVENTING 설치

클러스터에서 이벤트 중심 아키텍처를 사용하려면 Knative Eventing을 설치합니다. 이벤트 소스, 브로커 및 채널과 같은 Knative 구성 요소를 생성한 다음 이를 사용하여 이벤트를 애플리케이션 또는 외부 시스템으로 보낼 수 있습니다.

OpenShift Serverless Operator를 설치한 후 기본 설정을 사용하여 Knative Eventing을 설치하거나 **KnativeEventing** 사용자 정의 리소스(CR)에서 고급 설정을 구성할 수 있습니다. **KnativeEventing** CR의 구성 옵션에 대한 자세한 내용은 글로벌 구성 을 참조하십시오.

중요

OpenShift Serverless에서 Red Hat OpenShift distributed tracing을 사용하려면 Knative Eventing을 설치하기 전에 Red Hat OpenShift distributed tracing을 설치하고 구성해야 합니다.

5.1. 웹 콘솔을 사용하여 KNATIVE EVENTING 설치

OpenShift Serverless Operator를 설치한 후 OpenShift Container Platform 웹 콘솔을 사용하여 Knative Eventing을 설치합니다. 기본 설정을 사용하거나 **KnativeEventing** 사용자 정의 리소스(CR)에서 고급 설정을 구성하여 Knative Eventing을 설치할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Container Platform 웹 콘솔에 로그인했습니다.
- OpenShift Serverless Operator를 설치했습니다.

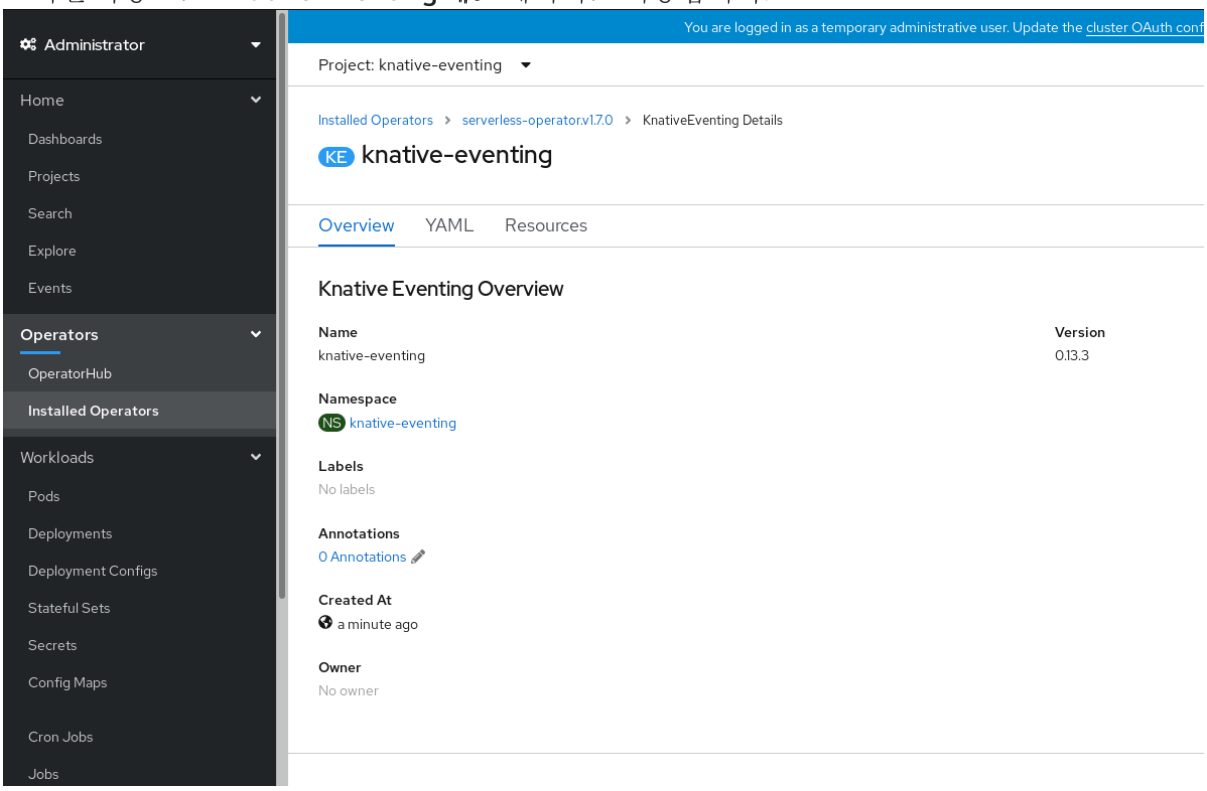
프로세스

1. OpenShift Container Platform 웹 콘솔의 **관리자** 화면에서 **Operator → 설치된 Operator**로 이동합니다.
2. 페이지 상단에 있는 **프로젝트** 드롭다운이 **Project: knative-eventing**으로 설정되어 있는지 확인합니다.
3. OpenShift Serverless Operator의 **제공되는 API** 목록에서 **Knative Eventing**을 클릭하여 **Knative Eventing** 탭으로 이동합니다.
4. **Knative Eventing 생성**을 클릭합니다.
5. **Knative Eventing 생성** 페이지에서 제공된 양식을 사용하거나 YAML 파일을 편집하여 **KnativeEventing** 오브젝트를 구성할 수 있습니다.
   - **KnativeEventing** 오브젝트 생성을 완전히 제어할 필요가 없는 간단한 구성은 이 양식을 사용합니다.
6. **생성**을 클릭합니다.
   - **KnativeEventing** 오브젝트 생성을 완전히 제어해야 하는 복잡한 구성을 위해 YAML 파일을 편집합니다. YAML 편집기에 액세스하려면 **Knative Eventing 생성** 페이지에서 **YAML 편집을** 클릭합니다.

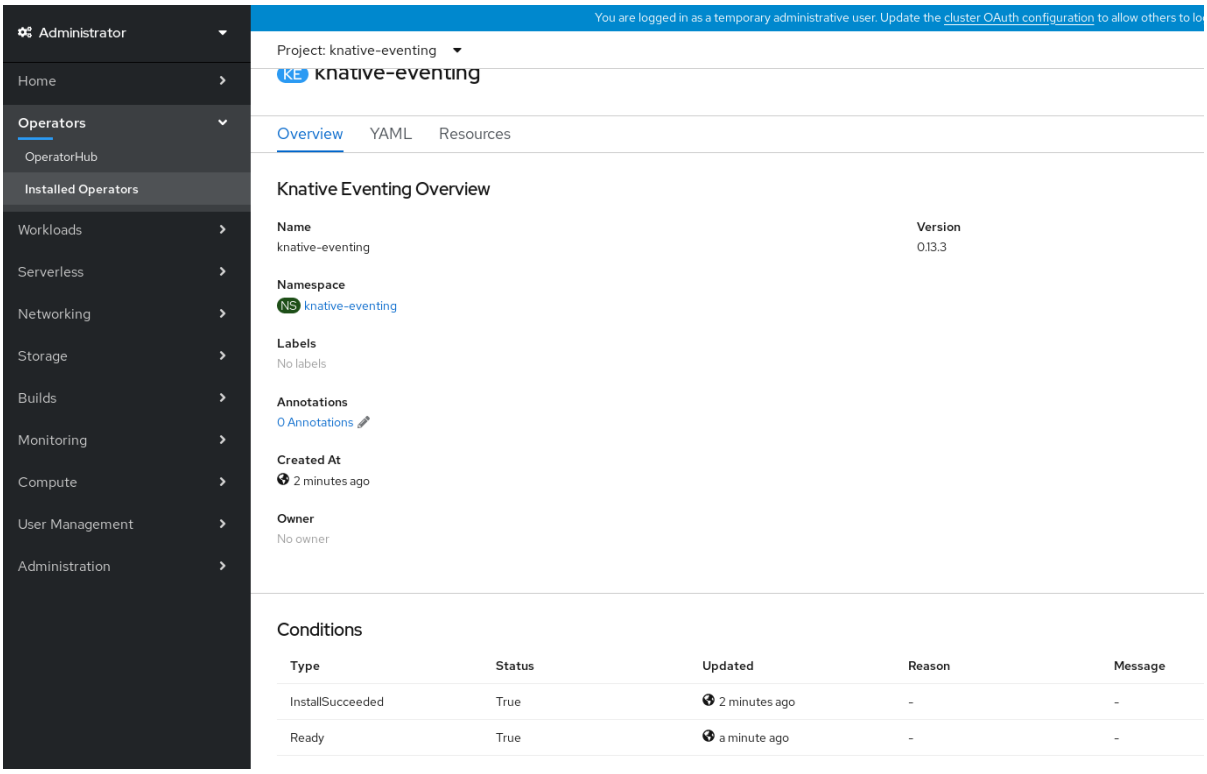
7. Knative Eventing을 설치하면 **KnativeEventing** 오브젝트가 생성되고 **Knative Eventing** 탭으로 자동으로 이동합니다. 리소스 목록에 **knative-eventing** 사용자 정의 리소스가 표시됩니다.

검증

1. **Knative Eventing** 탭에서 **knative-eventing** 사용자 정의 리소스를 클릭합니다.
2. 그러면 자동으로 **Knative Eventing 개요** 페이지로 이동합니다.



3. **조건** 목록을 보려면 아래로 스크롤합니다.
4. 예제 이미지에 표시된 대로 상태가 **True**인 조건 목록이 표시되어야 합니다.



참고

Knative Eventing 리소스를 생성하는 데 몇 초가 걸릴 수 있습니다. **리소스** 탭에서 해당 상태를 확인할 수 있습니다.

5. 조건이 **알 수 없음** 또는 **False** 상태인 경우 몇 분 정도 기다렸다가 리소스가 생성된 것을 확인한 후 다시 확인하십시오.

5.2. YAML을 사용하여 KNATIVE EVENTING 설치

OpenShift Serverless Operator를 설치한 후 기본 설정을 사용하여 Knative Eventing을 설치하거나 **KnativeEventing** 사용자 정의 리소스(CR)에서 고급 설정을 구성할 수 있습니다. 다음 절차에 따라 YAML 파일 및 **oc** CLI를 사용하여 Knative Eventing을 설치할 수 있습니다.

사전 요구 사항

- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Serverless Operator를 설치했습니다.
- OpenShift CLI(**oc**)를 설치합니다.

프로세스

1. **eventing.yaml**이라는 파일을 생성합니다.
2. 다음 샘플 YAML을 **eventing.yaml**에 복사합니다.

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

3. 선택 사항입니다. Knative Eventing 배포를 위해 구현하려는 YAML을 변경합니다.
4. 다음을 입력하여 **eventing.yaml** 파일을 적용합니다.

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

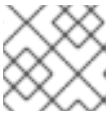
검증

1. 다음 명령을 입력하고 출력을 관찰하여 설치가 완료되었는지 확인합니다.

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

출력 예

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



참고

Knative Eventing 리소스를 생성하는 데 몇 초가 걸릴 수 있습니다.

2. 조건이 **알 수 없음** 또는 **False** 상태인 경우 몇 분 정도 기다렸다가 리소스가 생성된 것을 확인한 후 다시 확인하십시오.

3. 다음을 입력하여 Knative Eventing 리소스가 생성되었는지 확인합니다.

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

출력 예

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

5.3. APACHE KAFKA용 KNATIVE 브로커 설치

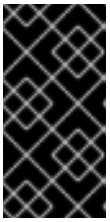
Apache Kafka의 Knative 브로커 구현은 지원되는 버전의 Apache Kafka 메시지 스트리밍 플랫폼을 OpenShift Serverless와 함께 사용할 수 있는 통합 옵션을 제공합니다. Apache Kafka 기능에 대한 Knative 브로커는 **KnativeKafka** 사용자 정의 리소스를 설치한 경우 OpenShift Serverless 설치에서 사용할 수 있습니다.

사전 요구 사항

- OpenShift Serverless Operator 및 Knative Eventing을 클러스터에 설치했습니다.
- Red Hat AMQ Streams 클러스터에 액세스할 수 있습니다.
- 확인 단계를 사용하려면 OpenShift CLI(**oc**)를 설치합니다.
- OpenShift Container Platform에 대한 클러스터 관리자 권한이 있거나 AWS 또는 OpenShift Dedicated의 Red Hat OpenShift Service에 대한 클러스터 또는 전용 관리자 권한이 있습니다.
- OpenShift Container Platform 웹 콘솔에 로그인되어 있습니다.

프로세스

1. **관리자** 화면에서 **Operator → 설치된 Operator**로 이동합니다.
2. 페이지 상단에 있는 **프로젝트** 드롭다운이 **Project: knative-eventing**으로 설정되어 있는지 확인합니다.
3. OpenShift Serverless Operator의 **Provided APIs** 목록에서 **Knative Kafka**를 검색하여 **Create Instance**를 클릭합니다.
4. **Knative Kafka 생성** 페이지에서 **KnativeKafka** 오브젝트를 구성합니다.



중요

클러스터에서 Kafka 채널, 소스, 브로커 또는 싱크를 사용하려면 사용할 옵션에 대해 **활성화된** 스위치를 **true** 로 전환해야 합니다. 이러한 스위치는 기본적으로 **false**로 설정됩니다. 또한 Kafka 채널, 브로커 또는 싱크를 사용하려면 부트스트랩 서버를 지정해야 합니다.

- **KnativeKafka** 오브젝트 생성을 완전히 제어할 필요가 없는 간단한 구성은 이 양식을 사용합니다.
- **KnativeKafka** 오브젝트 생성을 완전히 제어해야 하는 복잡한 구성을 위해 YAML을 편집합니다. **Knative Kafka 생성** 페이지에서 YAML **편집 링크를 클릭하여 YAML** 에 액세스할 수 있습니다.

KnativeKafka 사용자 정의 리소스의 예

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

1. 개발자가 클러스터에서 **KafkaChannel** 채널 유형을 사용할 수 있습니다.
2. AMQ Streams 클러스터에 있는 쉼표로 구분된 부트스트랩 서버 목록입니다.
3. 개발자가 클러스터에서 **KafkaSource** 이벤트 소스 유형을 사용할 수 있습니다.
4. 개발자가 클러스터에서 Apache Kafka에 Knative 브로커 구현을 사용할 수 있습니다.
5. Red Hat AMQ Streams 클러스터의 쉼표로 구분된 부트스트랩 서버 목록입니다.
6. **Broker** 오브젝트에서 지원하는 Kafka 주제의 파티션 수를 정의합니다. 기본값은 **10**입니다.
7. **Broker** 오브젝트에서 지원하는 Kafka 주제의 복제 요소를 정의합니다. 기본값은 **3** 입니다. **replicationFactor** 값은 Red Hat AMQ Streams 클러스터의 노드 수보다 작거나 같아야 합니다.

8 개발자가 클러스터에서 Kafka 싱크를 사용할 수 있습니다.

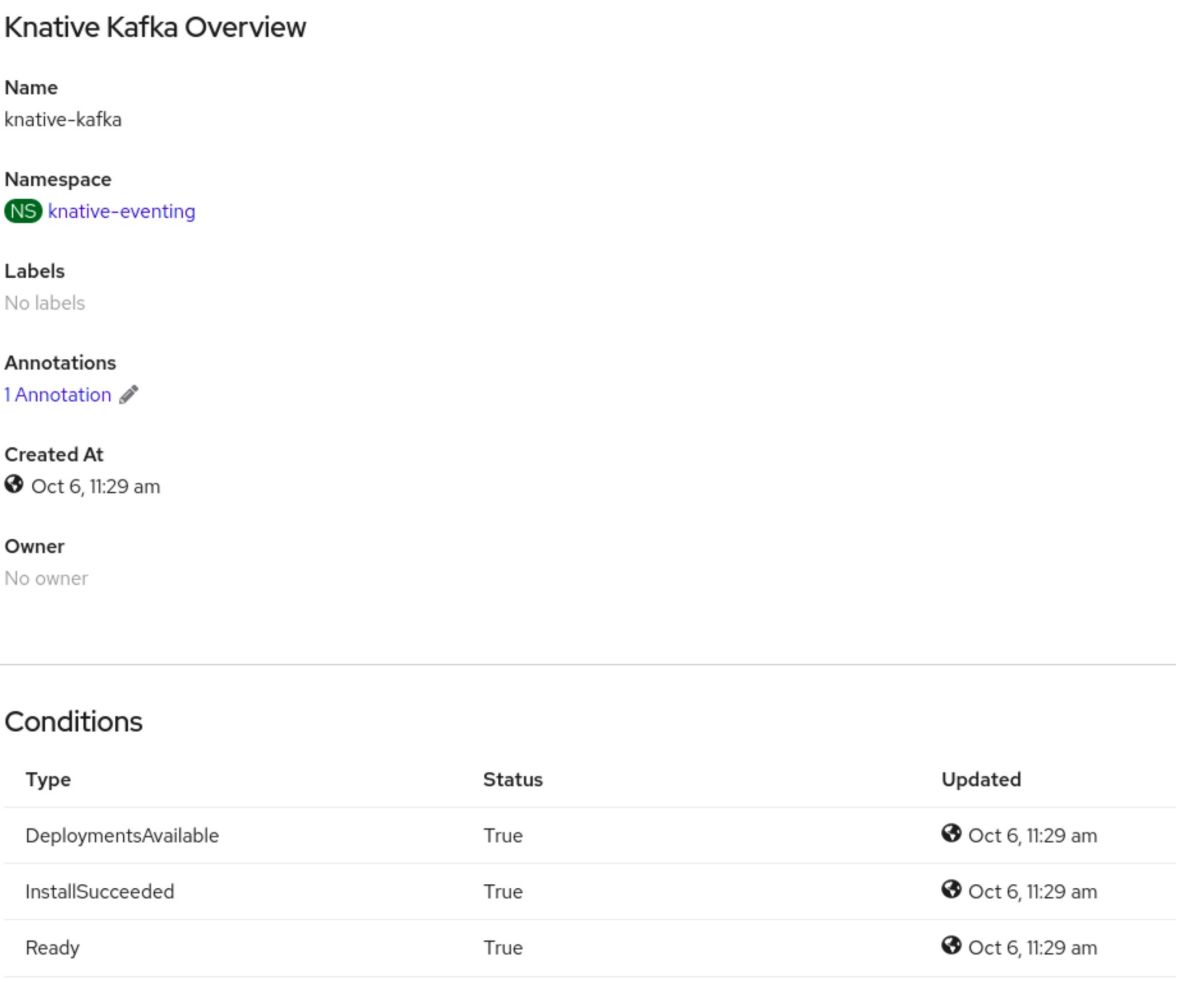
9 Kafka 데이터 플레인의 로그 수준을 정의합니다. 허용되는 값은 **TRACE**,**DEBUG**,**INFO**,**WARN** 및 **ERROR** 입니다. 기본값은 **INFO** 입니다.

> **주의**
>
> **DEBUG** 또는 **TRACE** 를 프로덕션 환경에서 로깅 수준으로 사용하지 마십시오. 이러한 로깅 수준의 출력은 상세하며 성능이 저하될 수 있습니다.

5. Kafka에 대한 선택적 구성을 완료한 후 **생성**을 클릭합니다. 그러면 리소스 목록에 **knative-kafka**가 있는 **Knative Kafka** 탭으로 자동으로 이동합니다.

검증

1. **Knative Kafka** 탭에서 **knative-kafka** 리소스를 클릭합니다. 그러면 자동으로 **Knative Kafka 개요** 페이지로 이동합니다.
2. 리소스에 대한 **조건** 목록을 확인하고 상태가 **True**인지 확인합니다.



조건 상태가 **알 수 없음** 또는 **False**인 경우 몇 분 정도 기다린 후 페이지를 새로 고칩니다.

3. Apache Kafka 리소스의 Knative 브로커가 생성되었는지 확인합니다.

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

출력 예

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

5.4. 다음 단계

- Knative 서비스를 사용하려면 Knative Serving을 설치할 수 있습니다.

6장. APACHE KAFKA용 KNATIVE 브로커 구성

Apache Kafka의 Knative 브로커 구현은 지원되는 버전의 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.

코어 OpenShift Serverless 설치의 일부로 제공되는 Knative Eventing 구성 요소 외에도 다음을 통해 **KnativeKafka** CR(사용자 정의 리소스)을 설치할 수 있습니다.

- OpenShift Container Platform용 클러스터 관리자
- AWS의 Red Hat OpenShift Service 또는 OpenShift Dedicated의 클러스터 또는 전용 관리자입니다.

KnativeKafka CR은 다음과 같은 추가 옵션을 사용자에게 제공합니다.

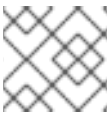
- Kafka 소스
- Kafka 채널
- Kafka 브로커
- Kafka 싱크

7장. APACHE KAFKA의 KNATIVE용 KUBE-RBAC-PROXY 구성

kube-rbac-proxy 구성 요소는 Apache Kafka에 Knative에 대한 내부 인증 및 권한 부여 기능을 제공합니다.

7.1. APACHE KAFKA용 KNATIVE에 대한 KUBE-RBAC-PROXY 리소스 구성

OpenShift Serverless Operator CR을 사용하여 **kube-rbac-proxy** 컨테이너의 리소스 할당을 전역적으로 덮어쓸 수 있습니다.



참고

특정 배포에 대한 리소스 할당을 덮어쓸 수도 있습니다.

다음 구성은 Knative Kafka **kube-rbac-proxy** 최소 및 최대 CPU 및 메모리 할당을 설정합니다.

KnativeKafka CR 예

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

1. 최소 CPU 할당을 설정합니다.
2. 최소 RAM 할당을 설정합니다.
3. 최대 CPU 할당을 설정합니다.
4. 최대 RAM 할당을 설정합니다.

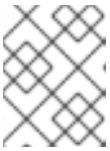
8장. OPENSHIFT SERVERLESS FUNCTIONS 구성

애플리케이션 코드 배포 프로세스를 개선하기 위해 OpenShift Serverless를 사용하여 OpenShift Container Platform에서 상태 비저장 이벤트 중심 함수를 Knative 서비스로 배포할 수 있습니다. 함수를 개발하려면 설정 단계를 완료해야 합니다.

8.1. 사전 요구 사항

클러스터에서 OpenShift Serverless Functions을 사용하려면 다음 단계를 완료해야 합니다.

- OpenShift Serverless Operator 및 Knative Serving이 클러스터에 설치되어 있습니다.



참고

함수는 Knative 서비스로 배포됩니다. 함수와 함께 이벤트 중심 아키텍처를 사용하려면 Knative Eventing도 설치해야 합니다.

- **oc** CLI 가 설치되어 있어야 합니다.
- Knative(**kn**) CLI 가 설치되어 있어야 합니다. Knative CLI를 설치하면 함수를 생성하고 관리하는 데 사용할 수 있는 **kn func** 명령을 사용할 수 있습니다.
- Docker Container Engine 또는 Podman 버전 3.4.7 이상을 설치했습니다.
- 사용 가능한 이미지 레지스트리(예: OpenShift Container Registry)에 액세스할 수 있습니다.
- Quay.io 를 이미지 레지스트리로 사용하는 경우 리포지토리가 비공개가 아니거나 Pod가 다른 보안 레지스트리의 이미지를 참조하도록 허용하는 OpenShift Container Platform 설명서를 따라야 합니다.
- OpenShift Container Registry를 사용하는 경우 클러스터 관리자가 레지스트리를 공개해야 합니다.

8.2. PODMAN 설정

고급 컨테이너 관리 기능을 사용하려면 OpenShift Serverless Functions에서 Podman을 사용할 수 있습니다. 이렇게 하려면 Podman 서비스를 시작하고 이를 연결하도록 Knative(**kn**) CLI를 구성해야 합니다.

프로세스

1. **${XDG_RUNTIME_DIR}/podman/podman.sock** 의 UNIX 소켓에서 Docker API를 제공하는 Podman 서비스를 시작합니다.

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

참고

대부분의 시스템에서 이 소켓은 **/run/user/$(id -u)/podman/podman.sock**에 있습니다.

2. 기능을 구축하는 데 사용되는 환경 변수를 설정합니다.

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

3. 자세한 출력을 보려면 **-v** 플래그를 사용하여 함수 프로젝트 디렉터리 내에서 build 명령을 실행합니다. 로컬 UNIX 소켓에 대한 연결이 표시됩니다.

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

8.3. MACOS에서 PODMAN 설정

고급 컨테이너 관리 기능을 사용하려면 OpenShift Serverless Functions에서 Podman을 사용할 수 있습니다. macOS에서 이 작업을 수행하려면 Podman 머신을 시작하고 연결하도록 Knative(**kn**) CLI를 구성해야 합니다.

프로세스

1. Podman 시스템을 생성합니다.

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

2. UNIX 소켓에서 Docker API를 제공하는 Podman 시스템을 시작합니다.

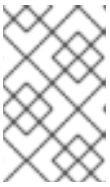
```
$ podman machine start
Starting machine "podman-machine-default"
Waiting for VM ...
Mounting volume... /Users/myuser:/Users/user

[...truncated output...]

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

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

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



참고

대부분의 macOS 시스템에서 이 소켓은 **/Users/myuser/.local/share/containers/podman/machine/podman-machine-default/podman.sock** 에 있습니다.

3. 기능을 구축하는 데 사용되는 환경 변수를 설정합니다.

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

4. 자세한 출력을 보려면 **-v** 플래그를 사용하여 함수 프로젝트 디렉터리 내에서 build 명령을 실행합니다. 로컬 UNIX 소켓에 대한 연결이 표시됩니다.

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

8.4. 다음 단계

- Docker Container Engine 또는 Podman에 대한 자세한 내용은 컨테이너 빌드 툴 옵션을 참조하십시오.
- 함수 시작하기를 참조하십시오.

9장. 서버리스 업그레이드

OpenShift Serverless는 릴리스 버전을 건너뛰지 않고 업그레이드해야 합니다. 이 섹션에서는 업그레이드 관련 문제를 해결하는 방법을 설명합니다.

9.1. SERVERLESS OPERATOR 유지 관리 릴리스 업그레이드

9.1.1. 최신 및 유지 관리 릴리스

OpenShift Serverless 1.29부터는 다음과 같이 다양한 제품 버전을 사용할 수 있습니다.

- 최신 릴리스는 **stable** 채널을 통해 제공됩니다.
- 유지 관리 릴리스는 버전 기반 채널(예: **stable-1.29**)을 통해 사용할 수 있습니다.

참고

유지 관리 릴리스는 최신 릴리스 이전 릴리스입니다. 예를 들어 **stable** 채널에 버전 1.30이 포함된 경우 **stable-1.29** 채널에서 유지 관리 릴리스를 사용할 수 있습니다.

버전 기반 채널을 사용하면 특정 **x.y** 스트림 내에 있을 수 있습니다. 또한 제품의 최신 버전으로 업그레이드할 수 없으므로 변경 사항이 손상될 수 있습니다.

유지 관리 릴리스로 전환하려면 서브스크립션 오브젝트 YAML 파일의 channel 매개변수를 **stable** 에서 **stable-1.29** 와 같은 해당 버전 기반 채널로 업데이트합니다.

9.1.2. 유지 관리 릴리스를 위한 패치 및 핫픽스

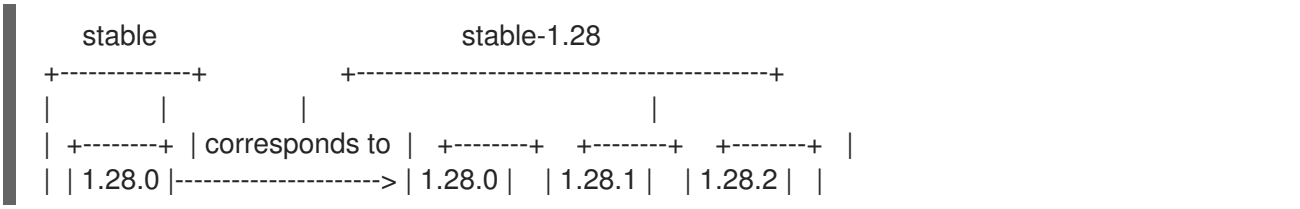
안정적인 릴리스와 마찬가지로 유지 관리 릴리스에는 패치 및 핫픽스가 적용되어 중요한 버그 및 보안 수정 사항으로 배포를 최신 상태로 유지할 수 있습니다.

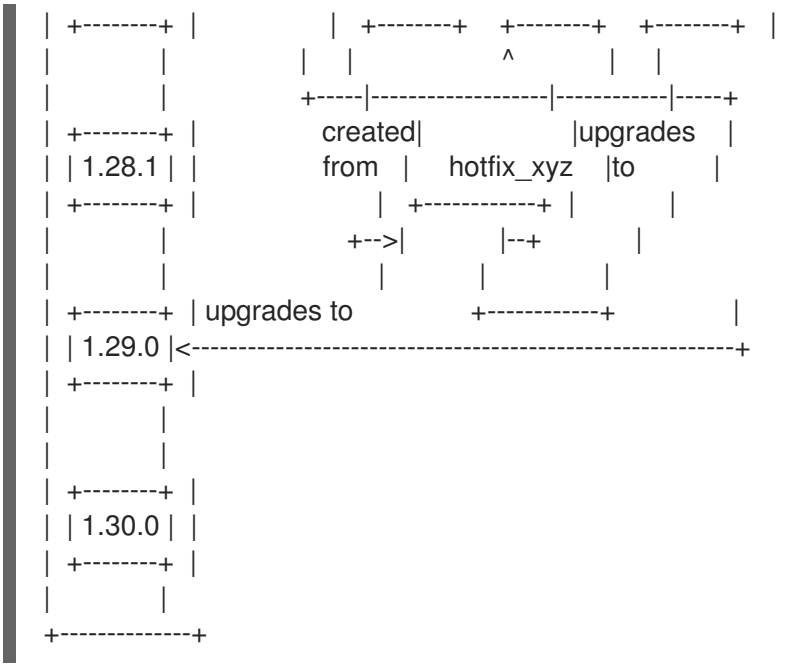
- 패치는 z-release로 배포되는 업데이트입니다(예: OpenShift Serverless 1.29.1은 버전 1.29.0 이후의 업데이트를 제공하는 패치입니다.
- 핫픽스는 다운타임이 필요하지 않고 프로덕션 환경에서 직접 사용되는 수정 프로그램입니다. 이는 최신 릴리스 버전이 아닌 고객의 배포된 버전을 업그레이드한다는 점에서 일반적인 업데이트와 다릅니다.
  모든 고객이 즉시 핫픽스를 사용할 수 없습니다. 그러나 핫픽스에서 도입한 변경 사항은 향후 릴리스의 일부로 모든 고객이 사용할 수 있습니다.

  배포와 관련된 핫픽스를 사용할 수 있는 경우 핫픽스 CatalogSource를 통해 서브스크립션을 업데이트하고 핫픽스를 받을 수 있습니다.

  새로운 Operator 릴리스를 사용할 수 있게 되면 핫픽스가 포함된 배포된 Operator도 업그레이드할 수 있습니다. 최신 GA 버전을 사용하려면 핫픽스 대신 공개 CatalogSource를 사용하도록 서브스크립션을 수정합니다.

다음 다이어그램은 패치 및 핫픽스가 작동하는 방법을 보여줍니다.





9.1.3. 유지 관리 릴리스의 업그레이드 경로

버전 기반 채널을 사용하는 경우 언제든지 채널의 최신 버전으로 업그레이드하거나 헤드로 업그레이드할 수 있습니다. 예를 들어 **stable-1.29** 채널에서 1.29.0에서 1.29.2로 업그레이드할 수 있습니다.

또한 채널 헤드에서 다음 **x.y** 릴리스로 업그레이드할 수 있습니다. 예를 들어 1.29.2가 **stable-1.29** 채널의 헤드인 경우 1.29.2에서 1.30으로 업그레이드할 수 있습니다. 이러한 채널 간 업데이트는 자동으로 수행되지 않으며 관리자는 서브스크립션을 업데이트하여 채널을 수동으로 전환해야 합니다.

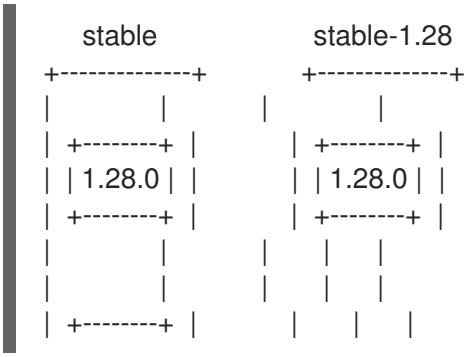
9.1.4. 업그레이드 예

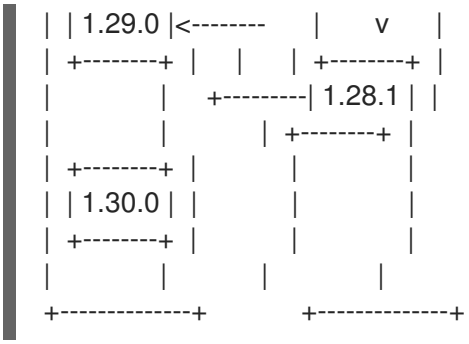
9.1.4.1. 시나리오 1

이 시나리오에서는 다음과 같은 상황이 적용됩니다.

- 채널은 **stable-1.28**
- **stable** 채널로 전환
- 현재 설치된 버전은 1.28.0입니다.
- 1.29.0이 1.28.1 이전에 릴리스되었습니다.
- 1.30.0은 **stable** 채널의 헤드입니다.

이 시나리오에서 **stable -1.28** 의 1.28.0에서 1.29.0으로의 업그레이드 경로는 1.28.0 ~ 1.29.0입니다.



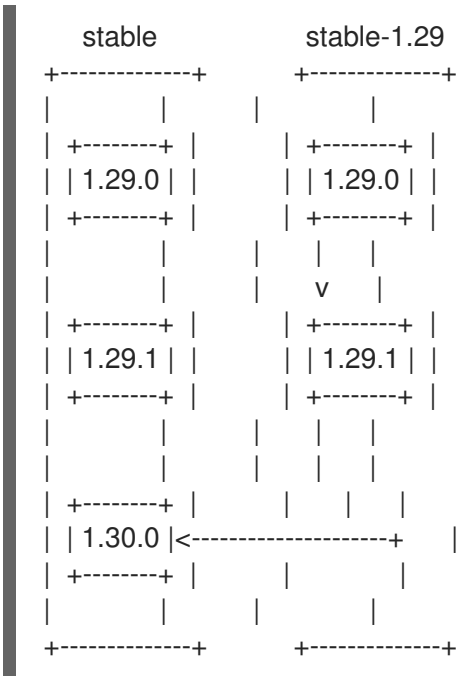


9.1.4.2. 시나리오 2

이 시나리오에서는 다음과 같은 상황이 적용됩니다.

- 채널은 **stable-1.29**입니다.
- 현재 설치된 버전은 1.29.0입니다.
- 1.29.1이 **1.30.0** 이전의 **stable -1.29** 및 **stable** 채널 모두에 릴리스되었습니다.

이 시나리오에서 **stable -1.29** 에서 1.30.0으로 1.29.0에서 1.30.0으로의 업그레이드 경로는 1.29.0에서 1.29.1에서 1.30.0으로입니다.

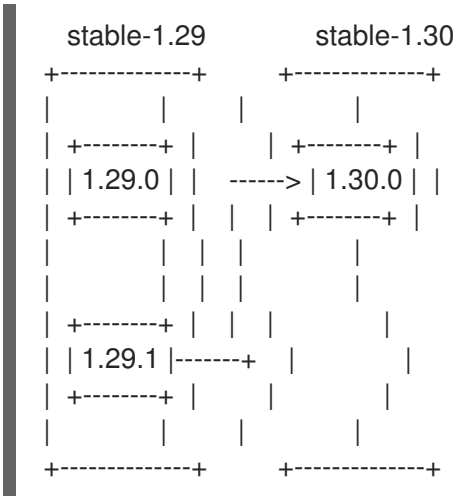


9.1.4.3. 시나리오 3

이 시나리오에서는 다음과 같은 상황이 적용됩니다.

- 채널은 **stable-1.29**입니다.
- **stable-1.30** 채널로 전환하고 있습니다.
- 현재 설치된 버전은 1.29.1입니다.
- 1.29.1은 **stable-1.29** 채널의 헤드입니다.

이 시나리오의 **stable-1.29** 에서 1.30.0으로 1.29.1에서 1.30.0으로의 업그레이드 경로는 1.29.1 ~ 1.30.0입니다.



9.2. OPENSHIFT SERVERLESS OPERATOR 업그레이드 실패 해결

예를 들어 수동 제거를 수행하고 다시 설치할 때 OpenShift Serverless Operator를 업그레이드할 때 오류가 발생할 수 있습니다. 오류가 발생하면 OpenShift Serverless Operator를 수동으로 다시 설치해야 합니다.

프로세스

1. **OpenShift Serverless 릴리스 노트에서 검색하여 원래 설치된 OpenShift Serverless Operator 버전을 확인합니다.**

   예를 들어 업그레이드 시도 중 오류 메시지에 다음 문자열이 포함될 수 있습니다.

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

   이 예에서 KnativeServing MAJOR.MINOR 버전은 1.5 로, OpenShift Serverless 1.26 릴리스 노트에서 적용됩니다. OpenShift Serverless에서 *Knative Serving 1.5를 사용합니다*.

2. **OpenShift Serverless Operator 및 모든 설치 계획을 설치 제거합니다.**

3. **첫 번째 단계에서 검색한 OpenShift Serverless Operator 버전을 수동으로 설치합니다. 설치하려면 다음 예와 같이 먼저 서버리스-subscription.yaml 파일을 생성합니다.**

   ```
   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   ```

```
metadata:
  name: serverless-operator
  namespace: openshift-serverless
spec:
  channel: stable
  name: serverless-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  installPlanApproval: Manual
  startingCSV: serverless-operator.v1.26.0
```

4. 그런 다음 다음 명령을 실행하여 서브스크립션을 설치합니다.

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

5. 표시되는 대로 업그레이드 설치 계획을 수동으로 승인하여 업그레이드합니다.

추가 리소스

- OpenShift Serverless 릴리스 정보
- 웹 콘솔을 사용하여 클러스터에서 Operator 삭제
- 웹 콘솔에서 OpenShift Serverless Operator 설치