Red Hat OpenShift Serverless 1.30

Serverless のインストール

Serverless Operator、Knative CLI、Knative Serving、および Knative Eventing のインストール

Last Updated: 2023-09-20

Red Hat OpenShift Serverless 1.30 Serverless のインストール

Serverless Operator、Knative CLI、Knative Serving、および Knative Eventing のインストール

法律上の通知

概要

このドキュメントでは、Serverless Operator、Knative CLI、Serving、および Eventing のインストールを説明します。また、Apache Kafka 用の Knative の設定、および Serverless 機能の設定も詳しく説明します。

目次

第1章 OPENSHIFT SERVERLESS のインストールの準備

OpenShift Serverless をインストールする前に、サポートされる設定および前提条件に関する以下の情報を確認してください。

OpenShift Container Platform の場合:

- OpenShift Serverless は、ネットワークが制限された環境でのインストールに対してサポートされません。
- 現時点で、OpenShift Serverless は単一クラスター上でのマルチテナント設定で使用することはできません。

1.1. サポートされる構成

OpenShift Serverless (最新バージョンおよび以前のバージョン) のサポートされる機能、設定、および統合のセットは、サポートされる設定 ページで確認できます。

1.2. OPENSHIFT CONTAINER PLATFORM でのスケーラビリティーおよびパフォーマンス

OpenShift Serverless は、3 つのメインノードと 3 つのワーカーノードの設定でテストされています。各ノードには、64 個の CPU、457 GB のメモリー、および 394 GB のストレージがあります。

この設定を使用して作成できる Knative サービスの最大数は 3,000 です。これは、OpenShift Container Platform の Kubernetes サービスの制限である 10,000 に相当します。これは、1 つの Knative サービスが 3 つの Kubernetes サービスを作成するためです。

ゼロ応答時間からの平均スケールは約 3.4 秒で、最大応答時間は 8 秒で、単純な Quarkus アプリケーションの 99.9 パーセンタイルは 4.5 秒でした。これらの時間は、アプリケーションとアプリケーションの実行時間によって異なる場合があります。

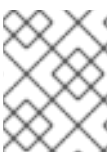
注記

クラスターサイズ要件の定義に関する次のセクションは、これらのディストリビューションに適用されます。

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

1.3. クラスターサイズ要件の定義

OpenShift Serverless をインストールして使用するには、クラスターのサイズを適切に設定する必要があります。



注記

以下の要件は、クラスターのワーカーマシンのプールだけを対象とします。コントロールプレーンは一般的なスケジューリングには使用されず、要件から省略されます。

OpenShift Serverless を使用する最小要件は、10 CPU および 40GB メモリーを持つクラスターです。デフォルトで、各 Pod は CPU ~400m を要求し、推奨値のベースはこの値になります。

OpenShift Serverless を実行するために必要な総サイズは、インストールされているコンポーネントとデプロイされているアプリケーションに依存し、デプロイメントによって異なる場合があります。

1.4. OPENSHIFT CONTAINER PLATFORM でコンピュートマシンセットを使用したクラスターのスケーリング

OpenShift Container Platform **MachineSet** API を使用して、クラスターを必要なサイズに手動でスケールアップすることができます。最小要件は、通常 2 つのマシンを追加することによってデフォルトのコンピュートマシンセットのいずれかをスケールアップする必要があることを意味します。コンピューティングマシンセットの手動スケーリング を参照してください。

1.4.1. 高度なユースケースの追加要件

OpenShift Container Platform でのロギングまたはメータリングなどの高度なユースケースの場合は、追加のリソースをデプロイする必要があります。このようなユースケースで推奨される要件は 24 vCPU および 96GB メモリーです。

クラスターで高可用性 (HA) を有効にしている場合、これには Knative Serving コントロールプレーンの各レプリカについて 0.5 - 1.5 コアおよび 200MB - 2GB のメモリーが必要です。HA は、デフォルトで一部の Knative Serving コンポーネントについて有効になります。高可用性コンポーネントレプリカの設定のドキュメントに従って HA を無効にできます。

1.5. OPENSHIFT CONTAINER PLATFORM ドキュメントのその他のリソース

- ネットワークが制限された環境での Operator Lifecycle Manager の使用
- OperatorHub について
- クラスター機能

第2章 OPENSHIFT SERVERLESS OPERATOR のインストール

OpenShift Serverless Operator をインストールすると、OpenShift Container Platform クラスターに Knative Serving、Knative Eventing、および Apache Kafka 用の Knative ブローカーをインストールし、使用できます。OpenShift Serverless Operator は、クラスターの Knative カスタムリソース定義 (CRD) を管理し、各コンポーネントの個別の Config Map を直接修正することなくそれらを設定できるようにします。

2.1. WEB コンソールからの OPENSHIFT SERVERLESS OPERATOR のインストール

OpenShift Container Platform Web コンソールを使用して、OperatorHub から OpenShift Serverless Operator をインストールできます。この Operator をインストールすると、Knative コンポーネントをインストールして使用できるようになります。

前提条件

- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Container Platform の場合は、クラスターで Marketplace 機能が有効になっているか、Red Hat Operator カタログソースが手動で設定されている。
- Web コンソールにログインしている。

手順

1. Web コンソールで、**Operators → OperatorHub** ページに移動します。
2. スクロールするか、またはこれらのキーワード **Serverless** を **Filter by keyword ボックス** に入力して OpenShift Serverless Operator を検索します。
3. Operator についての情報を確認してから、**Install** をクリックします。
4. **Install Operator** ページで以下を行います。
    a. **Installation Mode** は **All namespaces on the cluster (default)**になります。このモードは、デフォルトの **openshift-serverless** namespace で Operator をインストールし、クラスターのすべての namespace を監視し、Operator をこれらの namespace に対して利用可能にします。
    b. **Installed Namespace** は **openshift-serverless** です。
    c. **Update Channel** を選択します。
        - **stable** チャネルは、OpenShift Serverless Operator の最新の安定したリリースのインストールを可能にします。**stable** チャネルがデフォルトです。
        - 別のバージョンをインストールするには、対応する **stable-x.y** チャネル (たとえば、**stable-1.29**) を指定します。
    d. **Update Channel** として **stable** チャネルを選択します。**stable** チャネルは、OpenShift Serverless Operator の最新の安定したリリースのインストールを可能にします。
    e. **Automatic** または **Manual** 承認ストラテジーを選択します。

5. **Install** をクリックし、Operator をこの OpenShift Container Platform クラスターの選択した namespace で利用可能にします。

6. **Catalog → Operator Management** ページから、OpenShift Serverless Operator サブスクリプションのインストールおよびアップグレードの進捗をモニターできます。

   a. **手動** の承認ストラテジーを選択している場合、サブスクリプションのアップグレードステータスは、その Install Plan を確認し、承認するまで **Upgrading** のままになります。**Install Plan** ページでの承認後に、サブスクリプションのアップグレードステータスは **Up to date** に移行します。

   b. **自動** の承認ストラテジーを選択している場合、アップグレードステータスは、介入なしに **Up to date** に解決するはずです。

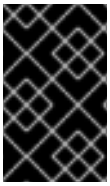
検証

サブスクリプションのアップグレードステータスが **Up to date** に移行したら、**Catalog → Installed Operators** を選択し、OpenShift Serverless Operator が表示され、その **Status** が最終的に関連する namespace で **InstallSucceeded** に解決することを確認します。

上記通りにならない場合:

1. **Catalog → Operator Management** ページに切り替え、**Operator Subscriptions** および **Install Plans** タブで **Status** の下の失敗またはエラーの有無を確認します。

2. さらにトラブルシューティングを行うために、**Workloads → Pods** ページで問題を報告している **openshift-serverless** プロジェクト内の Pod のログを確認します。



重要

OpenShift Serverless で Red Hat 分散トレースを使用する 場合は、KnativeServing または KnativeEventing をインストールする前に、Red Hat 分散トレースをインストールして設定する必要があります。

2.2. CLI からの OPENSHIFT SERVERLESS OPERATOR のインストール

CLI を使用して、OperatorHub から OpenShift Serverless Operator をインストールできます。この Operator をインストールすると、Knative コンポーネントをインストールして使用できるようになります。

前提条件

- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Container Platform の場合は、クラスターで Marketplace 機能が有効になっているか、Red Hat Operator カタログソースが手動で設定されている。
- クラスターにログインしている。

手順

1. **Namespace**、**OperatorGroup**、および **Subscription** オブジェクトを含む YAML ファイルを作成して、namespace を OpenShift Serverless Operator にサブスクライブします。たとえば、次の内容でファイル **serverless-subscription.yaml** を作成します。

Subscription の例

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

(1) Operator のチャネル名。**stable** チャンネルを使用すると、OpenShift Serverless Operator の最新の安定したバージョンをインストールできます。別のバージョンをインストールするには、対応する **stable-x.y** チャネル (たとえば、**stable-1.29**) を指定します。

(2) サブスクライブする Operator の名前。OpenShift Serverless Operator の場合、これは常に **serverless-operator** です。

(3) Operator を提供する CatalogSource の名前。デフォルトの OperatorHub カタログソースには **redhat-operators** を使用します。

(4) CatalogSource の namespace。デフォルトの OperatorHub カタログソースには **openshift-marketplace** を使用します。

2. **Subscription** オブジェクトを作成します。

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

検証

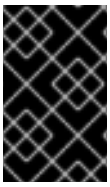
クラスターサービスバージョン (CSV) が **Succeeded** フェーズに達したことを確認します。

コマンドの例

```
$ oc get csv
```

出力例

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



重要

OpenShift Serverless で Red Hat 分散トレースを使用する 場合は、KnativeServing または KnativeEventing をインストールする前に、Red Hat 分散トレースをインストールして設定する必要があります。

2.3. グローバル設定

OpenShift Serverless Operator は、**KnativeServing** および **KnativeEventing** カスタムリソースからシステムの Config Map への値の反映を含む Knative インストールのグローバル設定を管理します。手動で適用される Config Map の更新は Operator によって上書きされます。ただし、Knative カスタムリソースを変更すると、これらの Config Map の値を設定できます。

Knative には、名前に接頭辞 **config-** が付けられた複数の Config Map があります。すべての Knative Config Map は、適用するカスタムリソースと同じ namespace に作成されます。たとえば、**KnativeServing** カスタムリソースが **knative-serving** namespace に作成される場合は、すべての Knative Serving Config Map もこの namespace に作成されます。

Knative カスタムリソースの **spec.config** には、Config Map ごとに **config-<name>** という名前の **<name>** エントリーが 1 つあり、Config Map **data** で使用される値を持ちます。

2.4. OPENSHIFT CONTAINER PLATFORM の関連情報

- カスタムリソース定義か定義定義らのリソースの管理
- 永続ストレージについて
- カスタム PKI の設定

2.5. 次のステップ

- OpenShift Serverless Operator のインストール後に、Knative Serving をインストールするか、Knative Eventing をインストールする ことができます。

第3章 KNATIVE CLI のインストール

Knative (**kn**) CLI には、独自のログインメカニズムがありません。クラスターにログインするには、OpenShift (**oc**) CLI をインストールし、**oc login** コマンドを使用する必要があります。CLI のインストールオプションは、オペレーティングシステムによって異なる場合があります。

ご使用のオペレーティングシステム用に OpenShift CLI (**oc**) をインストールする方法および **oc** でのログイン方法の詳細は、OpenShift CLI の使用開始 に関するドキュメントを参照してください。

Knative (**kn**) CLI を使用して OpenShift Serverless をインストールすることはできません。クラスター管理者は、OpenShift Serverless Operator のインストール のドキュメントで説明されているように、OpenShift Serverless Operator をインストールし、Knative コンポーネントをセットアップする必要があります。

重要

新しい OpenShift Serverless リリースで古いバージョンの Knative (**kn**) CLI の使用を試行する場合は、API が見つからないとエラーが発生します。

たとえば、バージョン 1.2 を使用する Knative (**kn**) CLI の 1.23.0 リリースと、Knative Serving および Knative Eventing API の 1.3 バージョンを使用する 1.24.0 OpenShift Serverless リリースを使用する場合、CLI は古い 1.2 API バージョンを探し続けるため、機能しません。

問題を回避するために、OpenShift Serverless リリースの最新の Knative (**kn**) CLI バージョンを使用していることを確認してください。

3.1. OPENSHIFT CONTAINER PLATFORM WEB コンソールを使用した KNATIVE CLI のインストール

OpenShift Container Platform Web コンソールを使用すると、Knative (**kn**) CLI をインストールするための合理化された直感的なユーザーインターフェイスが提供されます。OpenShift Serverless Operator をインストールすると、OpenShift Container PlatformWeb コンソールの **コマンドラインツール** ページから Linux (amd64、s390x、ppc64le)、macOS、または Windows 用の Knative (**kn**) CLI をダウンロードするためのリンクが表示されます。

前提条件

- OpenShift Container Platform Web コンソールにログインしている。
- OpenShift Serverless Operator および Knative Serving が OpenShift Container Platform クラスターにインストールされている。

重要

libc が利用できない場合は、CLI コマンドの実行時に以下のエラーが表示される場合があります。

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

- この手順の検証手順を使用する場合は、OpenShift (**oc**) CLI をインストールする必要がある。

手順

1. **Command Line Tools** ページから Knative (**kn**) CLI をダウンロードします。**Command Line Tools** ページには、Web コンソールの右上の  アイコンをクリックして、リストの **Command Line Tools** を選択します。

2. アーカイブを展開します。

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

3. **kn** バイナリーを **PATH** にあるディレクトリーに移動します。

4. **PATH** を確認するには、以下を実行します。

```
$ echo $PATH
```

検証

- 以下のコマンドを実行して、正しい Knative CLI リソースおよびルートが作成されていることを確認します。

```
$ oc get ConsoleCLIDownload
```

出力例

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

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

出力例

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

3.2. RPM パッケージマネージャーを使用した LINUX 用の KNATIVE CLI のインストール

Red Hat Enterprise Linux (RHEL) の場合、**yum** や **dnf** などのパッケージマネージャーを使用して、Knative (**kn**) CLI を RPM としてインストールできます。これにより、Knative CLI バージョンをシステムで自動的に管理できます。たとえば、**dnf upgrade** のようなコマンドを使用すると、新しいバージョンが利用可能な場合は、**kn** を含むすべてのパッケージがアップグレードされます。

前提条件

- お使いの Red Hat アカウントに有効な OpenShift Container Platform サブスクリプションがある。

手順

手順

1. Red Hat Subscription Manager に登録します。

```
# subscription-manager register
```

2. 最新のサブスクリプションデータをプルします。

```
# subscription-manager refresh
```

3. 登録済みのシステムにサブスクリプションを添付します。

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

1 有効な OpenShift Container Platform サブスクリプションのプール ID

4. Knative (**kn**) CLI に必要なリポジトリーを有効にします。
   - Linux (x86_64, amd64)

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

   - Linux on IBM zSystems および 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. パッケージマネージャーを使用して、Knative (**kn**) CLI を RPM としてインストールします。

yum コマンドの例

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

3.3. RPM パッケージマネージャーとともにインストールされる KNATIVE CLI のロックバージョン

最新ではなく、メンテナンスフェーズにある Knative (**kn**) CLI バージョンの使用が必要になる場合があります。これを実現するには、**kn** のバージョンをロックしてアップグレードできないようにします。

手順

1. DNF パッケージマネージャーの **versionlock** プラグインをインストールします。

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

2. 次のコマンドを実行して、**kn** のバージョンをロックします。

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

このコマンドは、**kn** を Knative 1.7 に基づくバージョンにロックします。Knative 1.7 は、OpenShift Serverless 1.29 のリリース時にメンテナンスフェーズにありました。

3.4. ロックされたバージョンでの KNATIVE CLI のアップグレード

以前にバージョンがロックされていた Knative (**kn**) CLI バージョンを手動でアップグレードできます。

手順

1. アップグレードされたパッケージが利用可能かどうかを確認します。

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

2. **kn** のバージョンロックを削除します。

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

1. 新しいバージョンに **kn** をロックします。

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

+ この例では、Knative 1.8 に基づく **kn** バージョンを使用します。

1. 新しい利用可能なバージョンにアップグレードします。

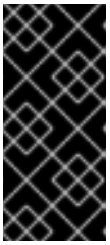
```
# dnf install --upgrade openshift-serverless-clients
```

3.5. LINUX の KNATIVE CLI のインストール

RPM または別のパッケージマネージャーがインストールされていない Linux ディストリビューションを使用している場合は、Knative (**kn**) CLI をバイナリーファイルとしてインストールできます。これを行うには、**tar.gz** アーカイブをダウンロードして解凍し、バイナリーを **PATH** のディレクトリーに追加する必要があります。

前提条件

- RHEL または Fedora を使用していない場合は、ライブラリーパスのディレクトリーに **libc** がインストールされていることを確認してください。



重要

libc が利用できない場合は、CLI コマンドの実行時に以下のエラーが表示される場合があります。

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

手順

1. Knative (**kn**) CLI の **tar.gz** アーカイブをダウンロードします。
   - Linux (x86_64, amd64)

- Linux on IBM zSystems および IBM® LinuxONE (s390x)
- Linux on IBM Power (ppc64le)

また、サーバーレスクライアントダウンロードミラー 内のそのバージョンに対応するディレクトリーに移動して、任意のバージョンの **kn** をダウンロードすることもできます。

2. アーカイブを展開します。

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

3. **kn** バイナリーを **PATH** にあるディレクトリーに移動します。
4. **PATH** を確認するには、以下を実行します。

```
$ echo $PATH
```

3.6. MACOS の KNATIVE CLI のインストール

macOS を使用している場合は、Knative (**kn**) CLI をバイナリーファイルとしてインストールできます。これを行うには、**tar.gz** アーカイブをダウンロードして解凍し、バイナリーを **PATH** のディレクトリーに追加する必要があります。

手順

1. Knative (**kn**) **CLItar.gz** アーカイブ をダウンロードします。
また、サーバーレスクライアントダウンロードミラー 内のそのバージョンに対応するディレクトリーに移動して、任意のバージョンの **kn** をダウンロードすることもできます。
2. アーカイブを解凍して解凍します。
3. **kn** バイナリーを **PATH** にあるディレクトリーに移動します。
4. **PATH** を確認するには、ターミナルウィンドウを開き、以下を実行します。

```
$ echo $PATH
```

3.7. WINDOWS の KNATIVE CLI のインストール

Windows を使用している場合は、Knative (**kn**) CLI をバイナリーファイルとしてインストールできます。これを行うには、ZIP アーカイブをダウンロードして解凍し、バイナリーを **PATH** のディレクトリーに追加する必要があります。

手順

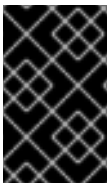
1. Knative (**kn**) CLIZIP アーカイブ をダウンロードします。
また、サーバーレスクライアントダウンロードミラー 内のそのバージョンに対応するディレクトリーに移動して、任意のバージョンの **kn** をダウンロードすることもできます。
2. ZIP プログラムでアーカイブを展開します。
3. **kn** バイナリーを **PATH** にあるディレクトリーに移動します。
4. **PATH** を確認するには、コマンドプロンプトを開いて以下のコマンドを実行します。

```
C:\> path
```

第4章 KNATIVE SERVING のインストール

Knative Serving をインストールすると、クラスター上で Knative サービスや関数を作成できます。また、オートスケーリングやネットワークオプションなどの追加機能をアプリケーションに利用することも可能です。

OpenShift Serverless Operator をインストールした後、デフォルト設定を使用して Knative Serving をインストールすることも、**KnativeServing** カスタムリソース (CR) でより詳細な設定を設定することもできます。**KnativeServing** CR の設定オプションの詳細については、グローバル設定を参照してください。



重要

OpenShift Serverless で Red Hat 分散トレースを使用する 場合は、KnativeServing をインストールする前に、Red Hat 分散トレースをインストールして設定する必要があります。

4.1. WEB コンソールを使用した KNATIVE SERVING のインストール

OpenShift Serverless Operator をインストールした後、OpenShift Container Platform の Web コンソールを使用して Knative Serving をインストールします。デフォルトの設定を使用して Knative Serving をインストールすることも、**KnativeServing** カスタムリソース (CR) でより詳細な設定を設定することもできます。

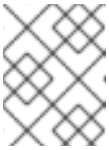
前提条件

- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Container Platform Web コンソールにログインしている。
- OpenShift Serverless Operator がインストールされている。

手順

1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. ページ上部の **Project** ドロップダウンメニューが **Project: knative-serving** に設定されていることを確認します。
3. OpenShift Serverless Operator の **Provided API** 一覧で **Knative Serving** をクリックし、**Knative Serving** タブに移動します。
4. **Create Knative Serving**をクリックします。
5. **Create Knative Serving**ページで、**Create** をクリックしてデフォルト設定を使用し、Knative Serving をインストールできます。
   また、Knative Serving インストールの設定を変更するには、提供されるフォームを使用するか、YAML を編集して **KnativeServing** オブジェクトを編集します。
   - **KnativeServing** オブジェクト作成を完全に制御する必要がない単純な設定には、このフォームの使用が推奨されます。

- **KnativeServing** オブジェクトの作成を完全に制御する必要のあるより複雑な設定には、YAML の編集が推奨されます。YAML にアクセスするには、**Create Knative Serving**ページの右上にある **edit YAML** リンクをクリックします。
フォームを完了するか、YAML の変更が完了したら、**Create** をクリックします。



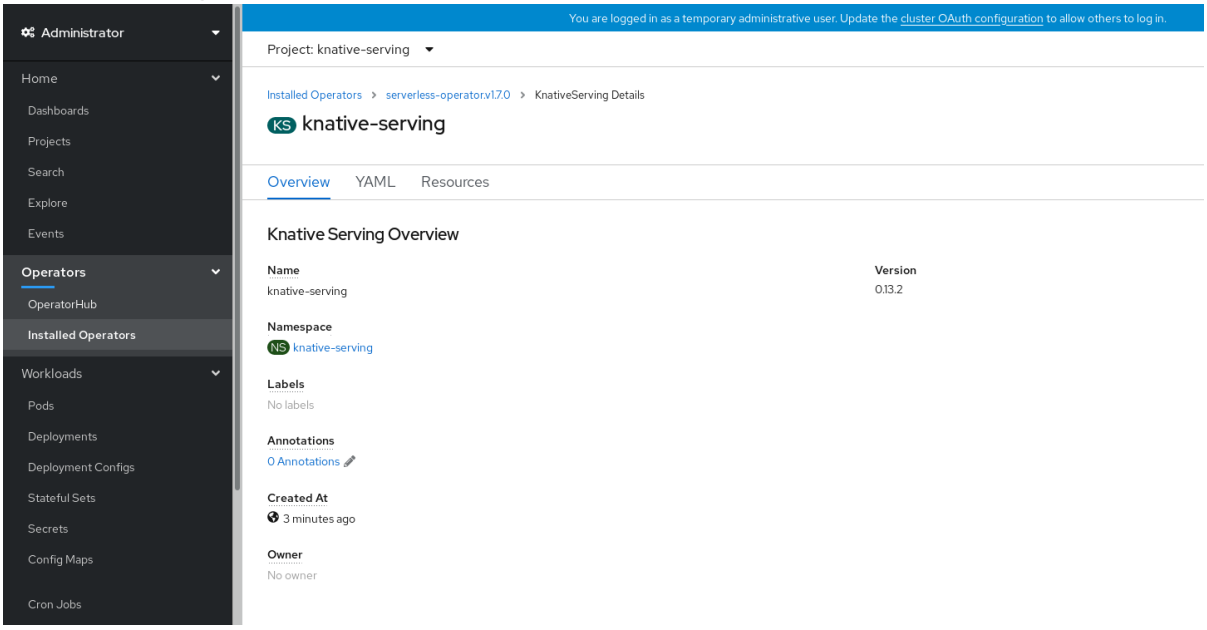
注記

KnativeServing カスタムリソース定義の設定オプションの詳細は、**高度なインストール設定オプション** に関するドキュメントを参照してください。

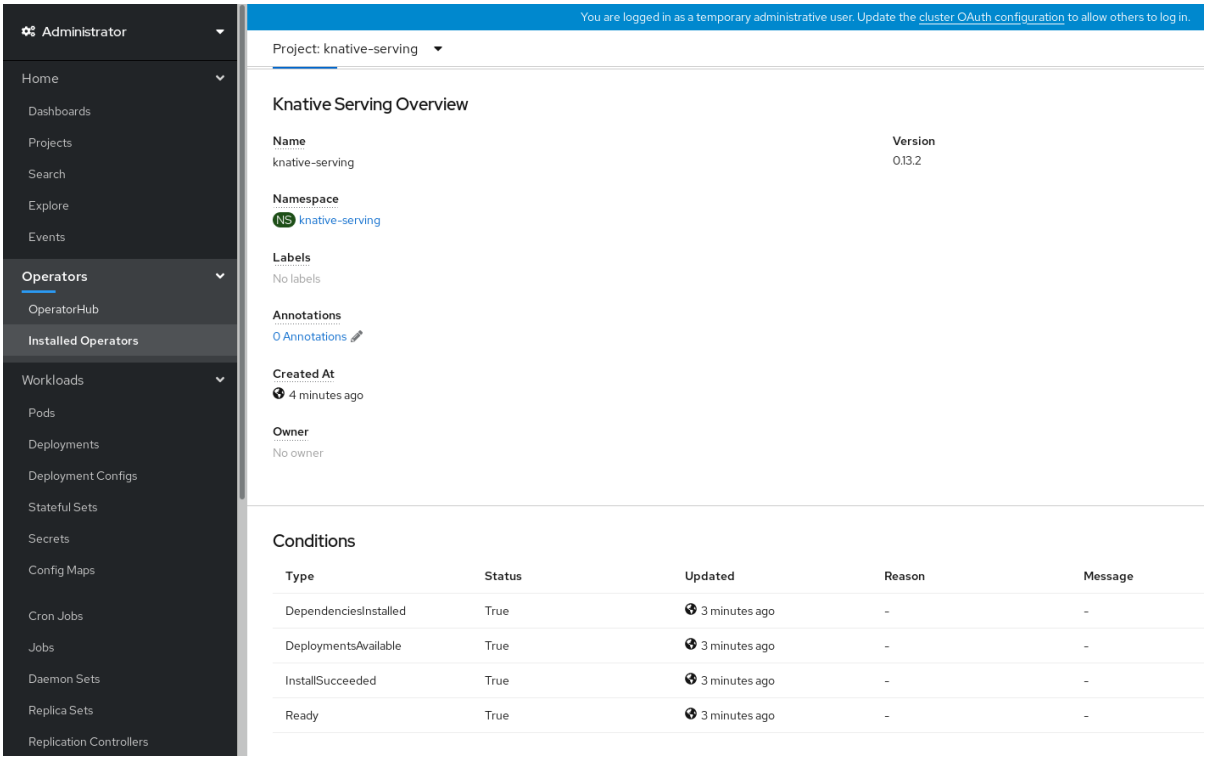
6. Knative Serving のインストール後に、**KnativeServing** オブジェクトが作成され、**Knative Serving** タブに自動的にダイレクトされます。リソースの一覧に **knative-serving** カスタムリソースが表示されます。

検証

1. **Knative Serving** タブで **knative-serving** カスタムリソースをクリックします。
2. **Knative Serving Overview** ページに自動的にダイレクトされます。



3. スクロールダウンして、**Conditions** のリストを確認します。
4. ステータスが **True** の条件のリストが表示されます (例のイメージを参照)。



注記

Knative Serving リソースが作成されるまでに数分の時間がかかる場合があります。**Resources** タブでステータスを確認できます。

5. 条件のステータスが **Unknown** または **False** である場合は、しばらく待ってから、リソースが作成されたことを再度確認します。

4.2. YAML を使用した KNATIVE SERVING のインストール

OpenShift Serverless Operator をインストールした後、デフォルト設定を使用して Knative Serving をインストールすることも、**KnativeServing** カスタムリソース (CR) でより詳細な設定を設定することもできます。YAML ファイルと **oc** CLI を利用して、以下の手順で Knative Serving をインストールすることができます。

前提条件

- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Serverless Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **serving.yaml** という名前のファイルを作成し、以下の YAML サンプルをこれにコピーします。

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

```
    name: knative-serving
    namespace: knative-serving
```

2. **serving.yaml** ファイルを適用します。

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

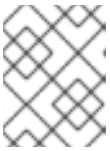
検証

1. インストールが完了したことを確認するには、以下のコマンドを実行します。

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

出力例

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



注記

Knative Serving リソースが作成されるまでに数分の時間がかかる場合があります。

条件のステータスが **Unknown** または **False** である場合は、しばらく待ってから、リソースが作成されたことを再度確認します。

2. Knative Serving リソースが作成されていることを確認します。

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

出力例

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

3. 必要なネットワークコンポーネントが、自動的に作成された **knative-serving-ingress** namespace にインストールされていることを確認します。

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

出力例

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

4.3. 次のステップ

- Knative イベント駆動型アーキテクチャーを使用する必要がある場合は、Knative Eventing をインストール できます。

第5章 KNATIVE EVENTING のインストール

クラスターでイベント駆動型アーキテクチャーを使用するには、Knative Eventing をインストールします。イベントソース、ブローカー、チャネルなどの Knative コンポーネントを作成し、それらを使用してアプリケーションや外部システムにイベントを送信することができます。

OpenShift Serverless Operator をインストールした後、デフォルト設定を使用して Knative Eventing をインストールすることも、**KnativeEventing** カスタムリソース (CR) でより詳細な設定を設定することもできます。**KnativeEventing** CR の設定オプションの詳細については、 グローバル設定 を参照してください。

重要

OpenShift Serverless で Red Hat 分散トレースを使用する 場合は、Knative Eventing をインストールする前に、Red Hat 分散トレースをインストールして設定する必要があります。

5.1. WEB コンソールを使用した KNATIVE EVENTING のインストール

OpenShift Serverless Operator をインストールした後、OpenShift Container Platform の Web コンソールを使用して Knative Eventing をインストールします。デフォルトの設定で Knative Eventing をインストールするか、**KnativeEventing** カスタムリソース (CR) でより詳細な設定を行うことが可能です。

前提条件

- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Container Platform Web コンソールにログインしている。
- OpenShift Serverless Operator がインストールされている。

手順

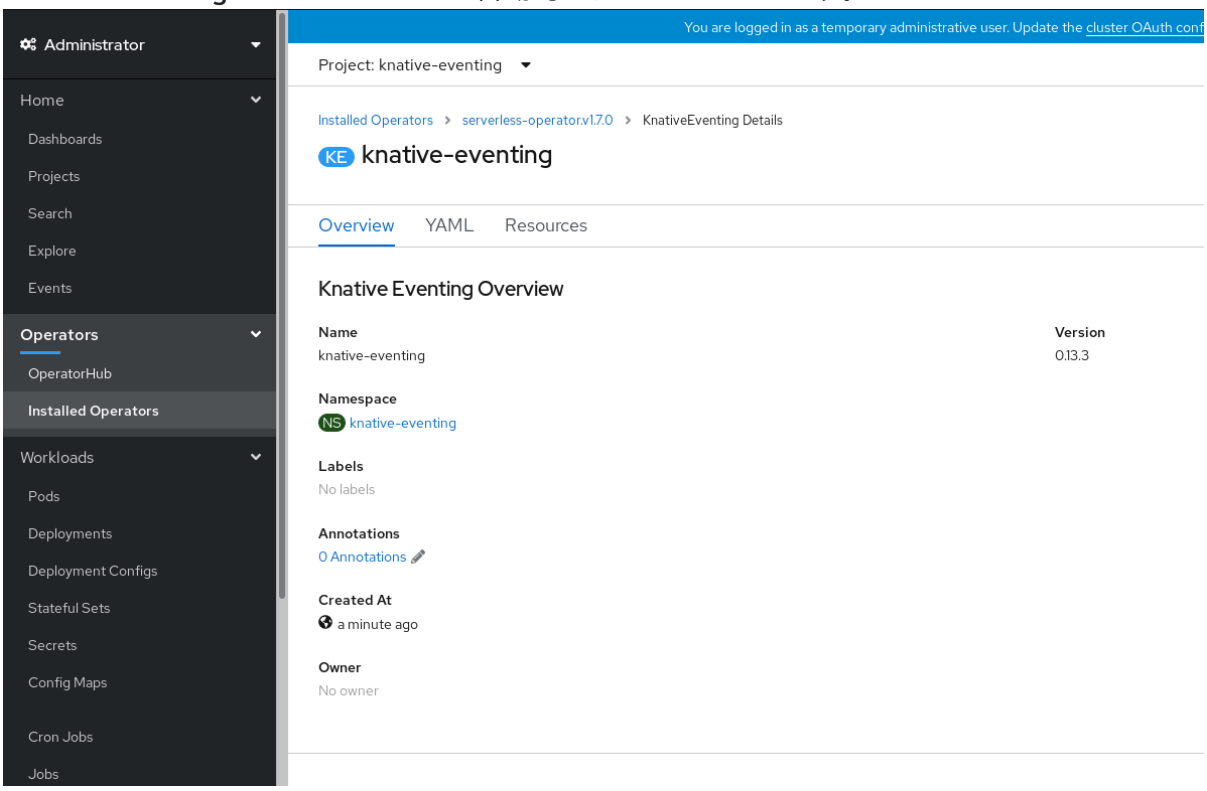
1. OpenShift Container Platform Web コンソールの **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. ページ上部の **Project** ドロップダウンメニューが **Project: knative-eventing** に設定されていることを確認します。
3. OpenShift Serverless Operator の **Provided API** 一覧で **Knative Eventing** をクリックし、**Knative Eventing** タブに移動します。
4. **Create Knative Eventing**をクリックします。
5. **Create Knative Eventing**ページでは、提供されたフォームを使用するか、YAML ファイルを編集して、**KnativeEventing** オブジェクトを設定できます。
    - **KnativeEventing** オブジェクトの作成を完全に制御する必要がない、より単純な設定には、このフォームを使用します。
6. **Create** をクリックします。

- **KnativeEventing** オブジェクトの作成を完全に制御する必要がある、より複雑な設定の場合は、YAML ファイルを編集します。YAML エディターにアクセスするには、**Create Knative Eventing** ページで **YAML の編集** をクリックします。

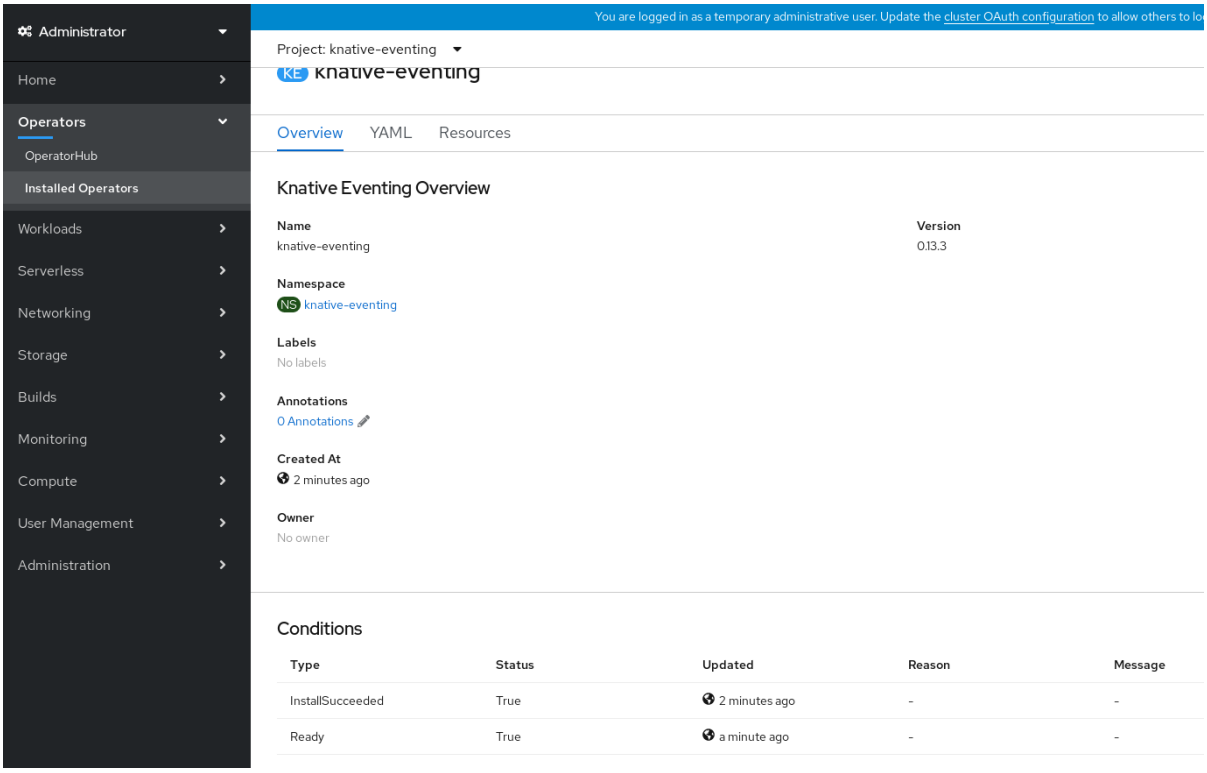
7. Knative Eventing のインストール後に、**KnativeEventing** オブジェクトが作成され、**Knative Eventing** タブに自動的にダイレクトされます。リソースの一覧に **knative-eventing** リソースが表示されます。

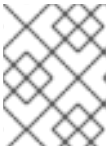
検証

1. **Knative Eventing** タブで **knative-eventing** カスタムリソースをクリックします。
2. **Knative Eventing Overview** ページに自動的にダイレクトされます。



3. スクロールダウンして、**Conditions** のリストを確認します。
4. ステータスが **True** の条件のリストが表示されます (例のイメージを参照)。





注記

Knative Eventing リソースが作成されるまでに数秒の時間がかかる場合があります。**Resources** タブでステータスを確認できます。

5. 条件のステータスが **Unknown** または **False** である場合は、しばらく待ってから、リソースが作成されたことを再度確認します。

5.2. YAML を使用した KNATIVE EVENTING のインストール

OpenShift Serverless Operator をインストールした後、デフォルト設定を使用して Knative Eventing をインストールすることも、**KnativeEventing** カスタムリソース (CR) でより詳細な設定を設定することもできます。YAML ファイルと **oc** CLI を利用して、以下の手順で Knative Eventing をインストールすることができます。

前提条件

- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Serverless Operator がインストールされている。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **eventing.yaml** という名前のファイルを作成します。
2. 以下のサンプル YAML を **eventing.yaml** にコピーします。

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

```
kind: KnativeEventing
metadata:
  name: knative-eventing
  namespace: knative-eventing
```

3. オプション: Knative Eventing デプロイメントについて実装する必要のある変更を YAML に加えます。

4. 以下を入力して **eventing.yaml** ファイルを適用します。

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

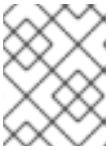
検証

1. 以下のコマンドを入力して出力を確認し、インストールが完了したことを確認します。

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

出力例

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



注記

Knative Eventing リソースが作成されるまでに数秒の時間がかかる場合があります。

2. 条件のステータスが **Unknown** または **False** である場合は、しばらく待ってから、リソースが作成されたことを再度確認します。

3. 以下のコマンドを実行して Knative Eventing リソースが作成されていることを確認します。

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

出力例

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

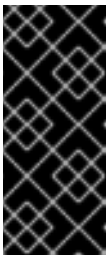
5.3. APACHE KAFKA の KNATIVE ブローカーのインストール

Apache Kafka の Knative ブローカー実装では、サポートされているバージョンの Apache Kafka メッセージストリーミングプラットフォームを OpenShift Serverless で使用できるように、統合オプションが提供されています。**KnativeKafka** カスタムリソースをインストールしている場合、Apache Kafka 機能の Knative ブローカーは OpenShift Serverless インストールで使用できます。

前提条件

- OpenShift Serverless Operator および Knative Eventing がクラスターにインストールされている。
- Red Hat AMQ Streams クラスターにアクセスできる。
- 検証手順を使用する場合は、OpenShift CLI (**oc**) をインストールしている。
- OpenShift Container Platform に対するクラスター管理者権限を持っているか、Red Hat OpenShift Service on AWS または OpenShift Dedicated に対するクラスターもしくは専用管理者権限を持っている。
- OpenShift Container Platform Web コンソールにログインしている。以下に手順を示します。

1. **Administrator** パースペクティブで、**Operators → Installed Operators** に移動します。
2. ページ上部の **Project** ドロップダウンメニューが **Project: knative-eventing** に設定されていることを確認します。
3. OpenShift Serverless Operator の **Provided APIs** のリストで **Knative Kafka** ボックスを見つけ、**Create Instance** をクリックします。
4. **Create Knative Kafka**ページで **KnativeKafka** オブジェクトを設定します。



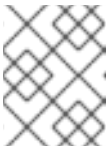
重要

クラスターで Kafka チャネル、ソース、ブローカー、またはシンクを使用するには、使用するオプションの **有効な** スイッチを **true** に切り替える必要があります。これらのスイッチは、デフォルトで **false** に設定されます。さらに、Kafka チャネル、ブローカー、またはシンクを使用するには、ブートストラップサーバーを指定する必要があります。

KnativeKafka カスタムリソースの例

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

1. 開発者はクラスターで **KafkaChannel** チャネルを使用できます。
2. AMQ Streams クラスターからのブートストラップサーバーのコンマ区切りのリスト。
3. 開発者はクラスターで **KafkaSource** イベントソースタイプを使用できます。
4. 開発者はクラスターで Apache Kafka 用の Knative ブローカー実装を使用できます。
5. Red Hat AMQ Streams クラスターからのブートストラップサーバーのコンマ区切りリスト。
6. **Broker** オブジェクトでサポートされる Kafka トピックのパーティション数を定義します。デフォルトは **10** です。
7. **Broker** オブジェクトでサポートされる Kafka トピックのレプリケーション係数を定義します。デフォルトは **3** です。
8. 開発者がクラスター内で Kafka シンクを使用できるようにします。



注記

replicationFactor の値は、Red Hat AMQ Streams クラスターのノード数以下である必要があります。

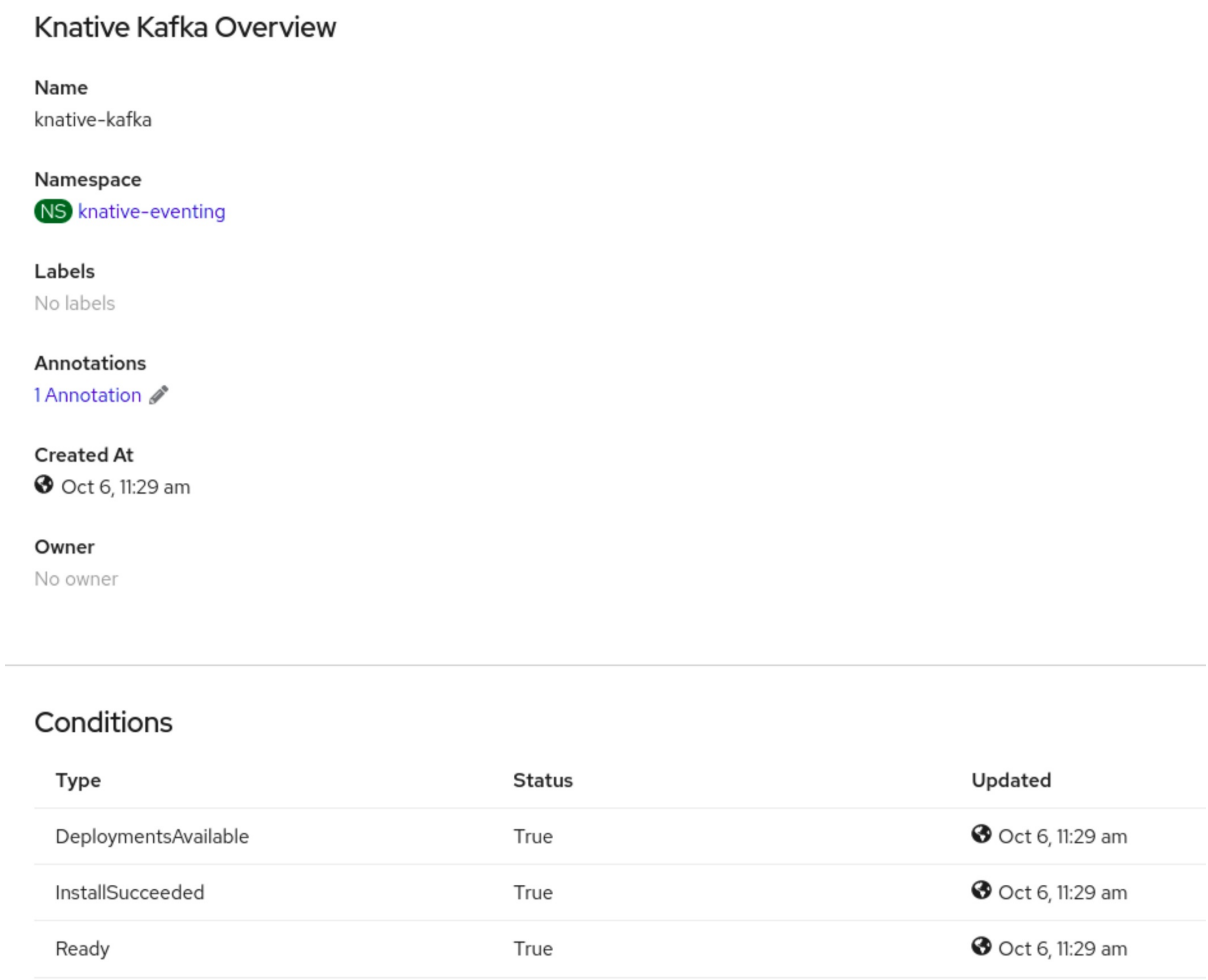
a. **KnativeKafka** オブジェクトの作成を完全に制御する必要がない単純な設定に、このフォームの使用が推奨されます。

b. **KnativeKafka** オブジェクトの作成を完全に制御する必要のあるより複雑な設定には、YAML の編集が推奨されます。YAML にアクセスするには、**Create Knative Kafka** ページの右上にある **Edit YAML** リンクをクリックします。

5. Kafka のオプションの設定が完了したら、**Create** をクリックします。**Knative Kafka** タブに自動的にダイレクトされます。ここで、**knative-kafka** はリソースのリストにあります。

検証

1. **Knative Kafka** タブで **knative-kafka** リソースをクリックします。**Knative Kafka Overview** ページに自動的にダイレクトされます。
2. リソースの **Conditions** (状態) のリストを表示し、それらのステータスが **True** であることを確認します。



状態のステータスが **Unknown** または **False** である場合は、ページを更新するためにしばらく待機します。

3. Apache Kafka リソースの Knative ブローカーが作成されたことを確認します。

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

出力例

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

5.4. 次のステップ

- Knative サービスを使用する場合は、Knative Serving をインストールできます。

第6章 APACHE KAFKA の KNATIVE ブローカーの設定

Apache Kafka の Knative ブローカー実装では、サポートされているバージョンの Apache Kafka メッセージストリーミングプラットフォームを OpenShift Serverless で使用できるように、統合オプションが提供されています。Kafka は、イベントソース、チャネル、ブローカー、およびイベントシンク機能のオプションを提供します。

コア OpenShift Serverless インストールの一部として提供される Knative Eventing コンポーネントに加えて、**KnativeKafka** カスタムリソース (CR) は次の方法でインストールできます。

- OpenShift Container Platform のクラスター管理者
- Red Hat OpenShift Service on AWS または OpenShift Dedicated のクラスターまたは Dedicated 管理者

KnativeKafka CR は、ユーザーに以下のような追加オプションを提供します。

- Kafka ソース
- Kafka チャネル
- Kafka ブローカー
- Kafka シンク

第7章 APACHE KAFKA の KNATIVE 用に KUBE-RBAC-PROXY を設定する

kube-rbac-proxy コンポーネントは、Apache Kafka の内部認証および認可機能を提供します。

7.1. APACHE KAFKA の KNATIVE 用に KUBE-RBAC-PROXY リソースを設定する

OpenShift Serverless Operator CR を使用して、**kube-rbac-proxy** コンテナーのリソース割り当てをグローバルにオーバーライドできます。

> You can also override resource allocation for a specific deployment.

次の設定では、Knative Kafka **kube-rbac-proxy** の最小および最大の CPU およびメモリー割り当てを設定します。

KnativeKafka CR の例

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

1. 最小 CPU 割り当てを設定します。
2. 最小 RAM 割り当てを設定します。
3. 最大 CPU 割り当てを設定します。
4. 最大 RAM 割り当てを設定します。

第8章 OPENSHIFT SERVERLESS FUNCTIONS の設定

アプリケーションコードのデプロイプロセスを改善するために、OpenShift Serverless を使用して、ステートレスでイベント駆動型の関数を Knative サービスとして OpenShift Container Platform にデプロイできます。関数を開発する場合は、セットアップ手順を完了する必要があります。

8.1. 前提条件

クラスターで OpenShift Serverless Functions の使用を有効にするには、以下の手順を実行する必要があります。

- OpenShift Serverless Operator および Knative Serving がクラスターにインストールされている。

注記

関数は Knative サービスとしてデプロイされます。関数でイベント駆動型のアーキテクチャーを使用する必要がある場合は、Knative Eventing もインストールする必要があります。

- **oc** CLI がインストールされている。
- Knative (**kn**) CLI がインストールされている。Knative CLI をインストールすると、関数の作成および管理に使用できる **kn func** コマンドを使用できます。
- Docker Container Engine または Podman バージョン 3.4.7 以降がインストールされている。
- OpenShift Container Registry などの利用可能なイメージレジストリーにアクセスできる。
- Quay.io をイメージレジストリーとして使用する場合は、リポジトリーがプライベートではないか確認するか、OpenShift Container Platform ドキュメント Pod が他のセキュアなレジストリーからイメージを参照できるようにする設定 に従っていることを確認している。
- OpenShift Container レジストリーを使用している場合は、クラスター管理者が レジストリーを公開する 必要があります。

8.2. PODMAN の設定

高度なコンテナー管理機能を使用するには、OpenShift Serverless Functions で Podman を使用することが推奨されます。そのためには、Podman サービスを開始し、それに接続するように Knative (**kn**) CLI を設定する必要があります。

手順

1. **${XDG_RUNTIME_DIR}/podman/podman.sock** で、UNIX ソケットで Docker API を提供する Podman サービスを起動します。

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

注記

多くのシステムでは、このソケットは **/run/user/$(id -u)/podman/podman.sock** にあります。

2. 関数のビルドに使用する環境変数を確立します。

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

3. **-v** フラグを指定して、関数プロジェクトディレクトリー内で build コマンドを実行し、詳細な出力を表示します。ローカルの UNIX ソケットへの接続が表示されるはずです。

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

8.3. MACOS での PODMAN のセットアップ

高度なコンテナー管理機能を使用するには、OpenShift Serverless Functions で Podman を使用することが推奨されます。macOS でこれを行うには、Podman マシンを起動し、それに接続するように Knative (**kn**) CLI を設定する必要があります。

手順

1. Podman マシンを作成します。

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

2. UNIX ソケットで Docker API を提供する Podman マシンを開始します。

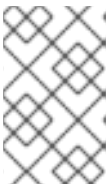
```
$ podman machine start
Starting machine "podman-machine-default"
Waiting for VM ...
Mounting volume... /Users/myuser:/Users/user

[...truncated output...]

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

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

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



注記

ほとんどの macOS システムでは、このソケットは **/Users/myuser/.local/share/containers/podman/machine/podman-machine-default/podman.sock** にあります。

3. 関数のビルドに使用する環境変数を確立します。

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

4. **-v** フラグを指定して、関数プロジェクトディレクトリー内で build コマンドを実行し、詳細な出力を表示します。ローカルの UNIX ソケットへの接続が表示されるはずです。

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

8.4. 次のステップ

- Docker Container Engine または Podman の詳細は、コンテナービルドツールのオプション を参照してください。
- 関数を使い始めるを参照してください。

第9章 SERVERLESS のアップグレード

OpenShift Serverless は、リリースバージョンをスキップせずにアップグレードする必要があります。本セクションでは、アップグレードに関する問題を解決する方法を説明します。

9.1. SERVERLESS OPERATOR メンテナンスリリースのアップグレード

9.1.1. 最新リリースとメンテナンスリリース

OpenShift Serverless 1.29 以降、次のようなさまざまな製品バージョンが利用できます。

- 最新リリースは **stable** チャネルから入手できます。
- メンテナンスリリースは、バージョンベースのチャネル (**stable-1.29** など) を通じて入手できます。

注記

メンテナンスリリースは、最新リリースよりも前のリリースです。たとえば、**stable** チャネルにバージョン 1.30 が含まれている場合、メンテナンスリリースは **stable-1.29** チャネルで利用可能になります。

バージョンベースのチャネルを使用すると、特定の **xy** ストリーム内にとどまることができます。さらに、重大な変更が含まれる可能性がある製品の最新バージョンへのアップグレードが妨げられます。

メンテナンスリリースに切り替えるには、サブスクリプションオブジェクト YAML ファイルのチャネルパラメーターを **stable** から対応するバージョンベースのチャネル (**stable-1.29** など) に更新します。

9.1.2. メンテナンスリリースのパッチとホットフィックス

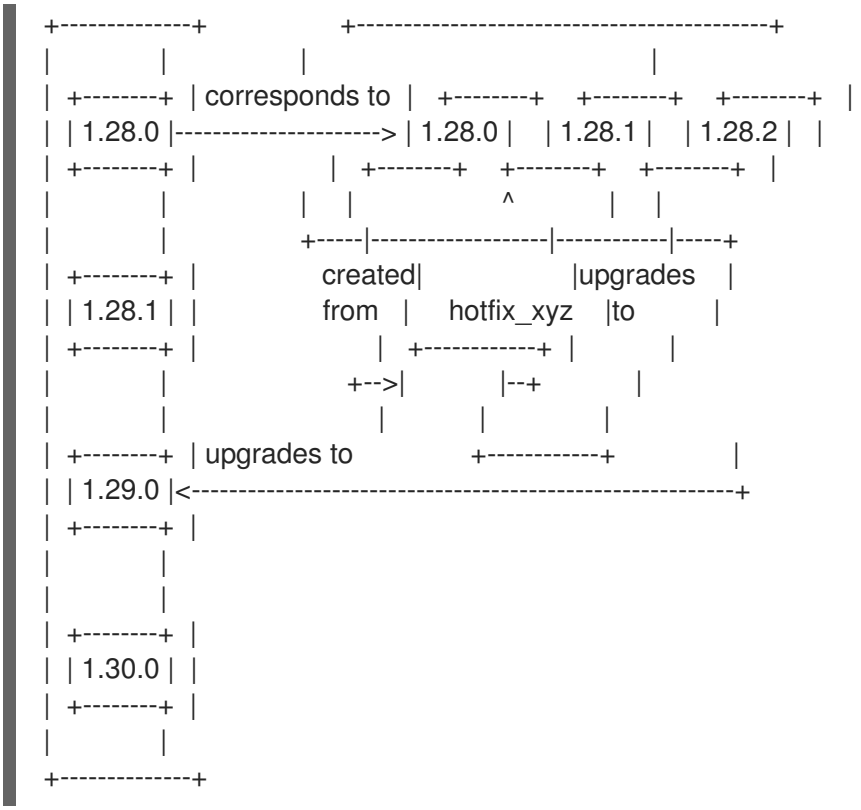
安定版リリースと同様に、メンテナンスリリースにはパッチとホットフィックスが適用されるため、重要なバグやセキュリティーの修正によりデプロイメントを最新の状態に保つことができます。

- パッチは、z-release として配布される更新です。たとえば、OpenShift Serverless 1.29.1 は、バージョン 1.29.0 以降に行われた更新を提供するパッチです。
- ホットフィックスは、ダウンタイムを必要とせず、運用環境で直接使用される修正です。これらは、最新のリリースされたバージョンではなく、顧客がデプロイメントしたバージョンをアップグレードするという点で、通常の更新とは異なります。
  ホットフィックスは、すべてのお客様にすぐに提供されるわけではありません。ただし、ホットフィックスによって導入された変更は、多くの場合、将来のリリースの一部としてすべての顧客が利用できるようになります。

  導入に関連するホットフィックスが利用可能になると、サブスクリプションを更新してホットフィックスを入手するためのホットフィックス CatalogSource が提供されます。

  新しいオペレーターのリリースが利用可能になった後、ホットフィックスを適用してデプロイされたオペレーターもアップグレードできます。最新の GA バージョンを使用するには、ホットフィックスの代わりにパブリック CatalogSource を使用するようにサブスクリプションを変更します。

次の図は、パッチとホットフィックスがどのように機能するかを示しています。



9.1.3. メンテナンスリリースのアップグレードパス

バージョンベースのチャネルを使用している場合は、チャネルまたはヘッドでいつでも最新バージョンにアップグレードできます。たとえば、**stable-1.29** チャネルでは 1.29.0 から 1.29.2 にアップグレードできます。

さらに、チャネルの先頭から、次の **xy** リリースにアップグレードできます。たとえば、1.29.2 が **stable-1.29** チャネルの先頭である場合、1.29.2 から 1.30 にアップグレードできます。このようなクロスチャネル更新は自動的には行われないため、管理者はサブスクリプションを更新してチャネルを手動で切り替える必要があります。

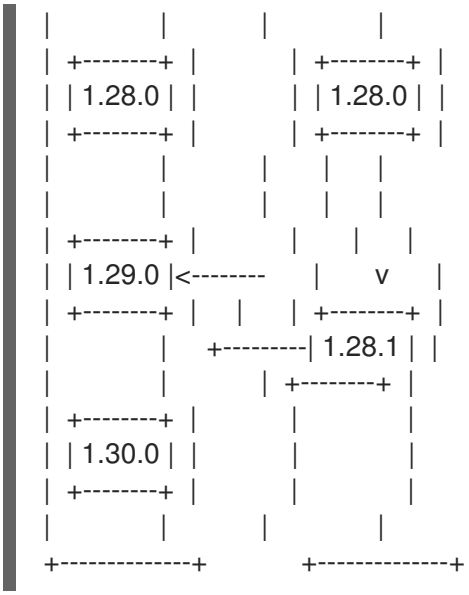
9.1.4. アップグレードの例

9.1.4.1. シナリオ 1

このシナリオでは、次の状況が当てはまります。

- チャネルが **stable-1.28**
- **stable** チャンネルに切り替えている
- 現在インストールされているバージョンは 1.28.0
- 1.29.0 は 1.28.1 より前にリリースされている
- 1.30.0 は **stable** チャネルのヘッド

このシナリオにおける **stable-1.28** の 1.28.0 から **stable** の 1.29.0 へのアップグレードパスは、1.28.0、1.28.1、1.29.0 です。

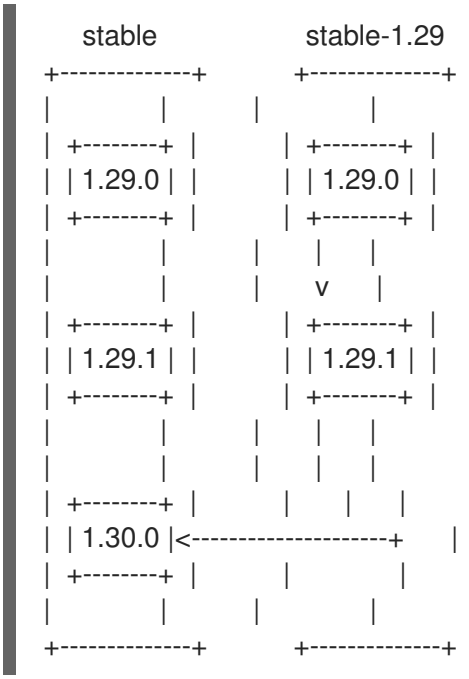


9.1.4.2. シナリオ 2

このシナリオでは、次の状況が当てはまります。

- チャネルは **stable-1.29**
- 現在インストールされているバージョンは 1.29.0
- 1.29.1 は、**1. 30.0** が **stable** チャネルにリリースされる前に **stable-1.29** チャネルと **stable** チャネルの両方にリリースされている

このシナリオでは、**stable-1.29** の 1.29.0 から **stable** の 1.30.0 へのアップグレードパスは、1.29.0 から 1.29.1 から 1.30.0 です。



9.1.4.3. シナリオ 3

このシナリオでは、次の状況が当てはまります。

- チャネルは **stable-1.29**

- **stable-1.30** チャネルに切り替えている
- 現在インストールされているバージョンは 1.29.1
- 1.29.1 は **stable-1.29** チャネルのヘッド

このシナリオでは、**stable-1.29** の 1.29.1 から **stable-1.30** の 1.30.0 へのアップグレードパスは、1.29.1 から 1.30.0 です。

```
  stable-1.29            stable-1.30
+--------------+       +--------------+
|              |       |              |
| +--------+   |       | +--------+   |
| | 1.29.0 |   |   ------> | 1.30.0 |   |
| +--------+   |   |   | +--------+   |
|              |   |   |              |
|              |   |   |              |
| +--------+   |   |   |              |
| | 1.29.1 |-------+   |              |
| +--------+   |       |              |
|              |       |              |
+--------------+       +--------------+
```

9.2. OPENSHIFT SERVERLESS OPERATOR のアップグレードの失敗の解決

たとえば、手動のアンインストールや再インストールの実行時に、OpenShift Serverless Operator のアップグレード時にエラーが発生する可能性があります。エラーが発生した場合は、OpenShift Serverless Operator を手動で再インストールする必要があります。

手順

1. 最初に OpenShift Serverless リリースノートを検索してインストールされた OpenShift Serverless Operator のバージョンを特定します。
   たとえば、アップグレードの試行時のエラーメッセージには以下の文字列が含まれる場合があります。

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

   この例では、KnativeServing **MAJOR.MINOR** バージョンは **1.5** です。これは、OpenShift Serverless 1.26 のリリースノートで説明しています。**OpenShift Serverless は Knative Serving 1.5 を使用するようになりました。**
2. OpenShift Serverless Operator とそのすべてのインストール計画をアンインストールします。
3. 最初の手順で検出された OpenShift Serverless Operator のバージョンを手動でインストールします。インストールするには、以下の例のように **serverless-subscription.yaml** ファイルを作成します。

   ```
   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. 次に、以下のコマンドを実行してサブスクリプションをインストールします。

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

5. アップグレードインストールプランが表示される際に手動で承認してアップグレードします。

関連情報

- OpenShift Serverless リリースノート
- Web コンソールの使用によるクラスターからの Operator の削除
- Web コンソールからの OpenShift Serverless Operator のインストール