Red Hat OpenShift Serverless 1.28

Serverless について

OpenShift Serverless の概要

Last Updated: 2023-09-19

Red Hat OpenShift Serverless 1.28 Serverless について

OpenShift Serverless の概要

法律上の通知

概要

このドキュメントでは、Functions、Serving、Eventing などの OpenShift Serverless 機能の概要を説明します。また、リリースノートおよびサポートを受ける方法の詳細も含まれています。

目次

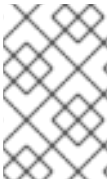
第1章 リリースノート

注記

OpenShift Serverless のライフサイクルとサポートされるプラットフォームに関する追加情報は、プラットフォームライフサイクルポリシー を参照してください。

リリースノートには、新機能、非推奨機能、互換性を損なう変更、既知の問題に関する情報が記載されています。以下のリリースノートは、OpenShift Container Platform 上の最新の OpenShift Serverless リリースに適用されます。

OpenShift Serverless 機能の概要については、OpenShift Serverless について を参照してください。



注記

OpenShift Serverless はオープンソースの Knative プロジェクトに基づいています。

最新の Knative コンポーネントリリースの詳細は、Knative ブログ を参照してください。

1.1. API バージョンについて

API バージョンは、OpenShift Serverless の特定の機能およびカスタムリソースの開発状況を示す重要な指標です。正しい API バージョンを使用していないリソースをクラスター上に作成すると、デプロイメントで問題が発生する可能性があります。

OpenShift Serverless Operator は、最新バージョンを使用するように非推奨の API を使用する古いリソースを自動的にアップグレードします。たとえば、**v1beta1** などの古いバージョンの **ApiServerSource** API を使用するクラスターにリソースを作成した場合、OpenShift Serverless Operator はこれらのリソースを自動的に更新し、これが利用可能な場合に API の **v1** バージョンを使用するように、**v1beta1** バージョンは非推奨になりました。

非推奨となった古いバージョンは、今後のリリースで削除される可能性があります。API の非推奨バージョンを使用すると、リソースが失敗することはありません。ただし、削除された API のバージョンを使用しようとすると、リソースが失敗します。問題を回避するために、マニフェストが最新バージョンを使用するように更新されていることを確認します。

1.2. 一般提供およびテクノロジープレビュー機能

一般提供 (GA) の機能は完全にサポートされており、実稼働での使用に適しています。テクノロジープレビュー (TP) 機能は実験的な機能であり、本番環境での使用を目的としたものではありません。TP 機能の詳細は、Red Hat Customer Portal の Technology Preview のサポート範囲 を参照してください。

次の表は、どの OpenShift Serverless 機能が GA であり、どの機能が TP であるかに関する情報を提供します。

表1.1 OpenShift Container Platform および OpenShift Dedicated の一般提供およびテクノロジープレビュー機能トラッカー

機能	1.26	1.27	1.28
kn func	GA	GA	GA

機能	1.26	1.27	1.28
Quarkus 関数	GA	GA	GA
Node.js 関数	TP	TP	GA
TypeScript 関数	TP	TP	GA
Python 関数	-	-	TP
Service Mesh mTLS	GA	GA	GA
emptyDir ボリューム	GA	GA	GA
HTTPS リダイレクト	GA	GA	GA
Kafka ブローカー	GA	GA	GA
Kafka シンク	GA	GA	GA
Knative サービスの init コンテナーのサポート	GA	GA	GA
Knative サービスの PVC サポート	GA	GA	GA
内部トラフィックの TLS	TP	TP	TP
namespace スコープのブローカー	-	TP	TP
multi-container support	-	-	TP

表1.2 Red Hat OpenShift Service on AWS の一般提供およびテクノロジープレビュー機能トラッカー

機能	1.26	1.27	1.28
kn func	GA	GA	GA
Service Mesh mTLS	GA	GA	GA
emptyDir ボリューム	GA	GA	GA
HTTPS リダイレクト	GA	GA	GA
Kafka ブローカー	GA	GA	GA
Kafka シンク	TP	TP	TP

機能	1.26	1.27	1.28
Knative サービスの init コンテナーのサポート	GA	GA	GA
Knative サービスの PVC サポート	GA	GA	GA
内部トラフィックの TLS	TP	TP	TP
namespace スコープのブローカー	-	TP	TP
multi-container support	-	-	TP

1.3. 非推奨および削除された機能

以前のリリースで一般提供 (GA) またはテクノロジープレビュー (TP) であった一部の機能は、非推奨または削除されました。非推奨の機能は依然として OpenShift Serverless に含まれており、引き続きサポートされますが、本製品の今後のリリースで削除されるため、新規デプロイメントでの使用は推奨されません。

OpenShift Serverless で非推奨となり、削除された主な機能の最新の一覧は、以下の表を参照してください。

表1.3 OpenShift Container Platform および OpenShift Dedicated の非推奨および削除された機能トラッカー

機能	1.20	1.21	1.22 から 1.26	1.27	1.28
KafkaBinding API	非推奨	非推奨	廃止	廃止	廃止
kn func emit (1.21 以降では **kn func invoke**)	非推奨	廃止	廃止	廃止	廃止
Serving および Eventing **v1alpha1** API	-	-	-	非推奨	非推奨
enable-secret-informer-filtering アノテーション	-	-	-	-	非推奨

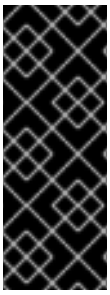
表1.4 Red Hat OpenShift Service on AWS の非推奨および削除された機能トラッカー

機能	1.23 から 1.26	1.27	1.28
KafkaBinding API	廃止	廃止	廃止
kn func emit (1.21 以降では **kn func invoke**)	廃止	廃止	廃止

機能	1.23 から 1.26	1.27	1.28
Serving および Eventing **v1alpha1** API	-	非推奨	非推奨

1.4. RED HAT OPENSHIFT SERVERLESS 1.28 のリリースノート

OpenShift Serverless 1.28 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。



重要

OpenShift Container Platform 4.13 は Red Hat Enterprise Linux (RHEL) 9.2 をベースにしています。RHEL 9.2 はまだ連邦情報処理標準 (FIPS) 検証のために提出されていません。Red Hat は特定の期限を約束することはできませんが、RHEL 9.0 および RHEL 9.2 モジュール、さらには RHEL 9.x のマイナーリリースでも FIPS 検証を取得する予定です。更新に関する情報は、ナレッジベースの記事 Compliance Activities and Government Standards で入手できます。

1.4.1. 新機能

- OpenShift Serverless は Knative Serving 1.7 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 1.7 を使用するようになりました。
- OpenShift Serverless は Kourier 1.7 を使用するようになりました。
- OpenShift Serverless は Knative (**kn**) CLI 1.7 を使用するようになりました。
- OpenShift Serverless は、Apache Kafka 1.7 に Knative ブローカー実装を使用するようになりました。
- **kn func** CLI プラグインは **func** 1.9.1 バージョンを使用するようになりました。
- OpenShift Serverless Functions の Node.js および TypeScript ランタイムが一般提供 (GA) になりました。
- OpenShift Serverless Functions の Python ランタイムがテクノロジープレビューとして利用可能になりました。
- Knative Serving のマルチコンテナーサポートがテクノロジープレビューとして利用可能になりました。この機能を使用すると、単一の Knative サービスを使用してマルチコンテナー Pod をデプロイできます。
- OpenShift Serverless 1.29 以降では、Knative Eventing の以下のコンポーネントが 2 つの Pod から 1 つにスケールダウンされます。
  - **imc-controller**
  - **imc-dispatcher**
  - **mt-broker-controller**
  - **mt-broker-filter**

- **mt-broker-ingress**

- Serving CR の **serverless.openshift.io/enable-secret-informer-filtering** アノテーションが非推奨になりました。アノテーションは Istio に対してのみ有効ですが、Kourier では有効ではありません。
  OpenShift Serverless 1.28 では、OpenShift Serverless Operator は **net-istio** と **net-kourier** の両方の環境変数 **ENABLE_SECRET_INFORMER_FILTERING_BY_CERT_UID** を挿入できます。

  シークレットフィルタリングを有効にする場合は、すべてのシークレットに **networking.internal.knative.dev/certificate-uid: "<id>"** というラベルを付ける必要があります。そうしないと、Knative Serving がシークレットを検出しないため、障害が発生します。新規および既存のシークレットの両方にラベルを付ける必要があります。

  次の OpenShift サーバーレスリリースのいずれかでは、シークレットフィルタリングがデフォルトで有効になります。障害が発生しないように、事前にシークレットにラベルを付けます。

1.4.2. 既知の問題

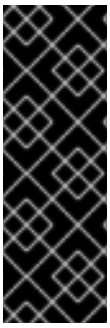
- 現在、Python のランタイムは、IBM Power、IBM zSystems、および IBM® LinuxONE の OpenShift Serverless Functions ではサポートされません。
  Node.js、TypeScript、および Quarkus 関数は、これらのアーキテクチャーでサポートされます。

- Windows プラットフォームでは、**app.sh** ファイルのパーミッションが原因で、Source-to-Image ビルダーを使用して Python 関数をローカルで構築、実行、またはデプロイできません。
  この問題を回避するには、Windows Subsystem for Linux を使用します。

- Red Hat OpenShift Serverless 1.27 より前に作成された **KafkaSource** リソースは、削除時にスタックします。この問題を回避するには、**KafkaSource** の削除を開始した後、リソースからファイナライザーを削除します。

1.5. RED HAT OPENSHIFT SERVERLESS 1.27 のリリースノート

OpenShift Serverless 1.27 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。



重要

OpenShift Serverless 1.26 は、OpenShift Container Platform 4.12 で完全にサポートされている最も古いリリースです。OpenShift Serverless 1.25 以前は、OpenShift Container Platform 4.12 にデプロイされません。

このため、OpenShift Container Platform をバージョン 4.12 にアップグレードする前に、OpenShift Serverless をバージョン 1.26 または 1.27 にアップグレードしてください。

1.5.1. 新機能

- OpenShift Serverless は Knative Serving 1.6 を使用するようになりました。

- OpenShift Serverless は Knative Eventing 1.6 を使用するようになりました。

- OpenShift Serverless は Kourier 1.6 を使用するようになりました。

- OpenShift Serverless は Knative (**kn**) CLI 1.6 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 1.6 を使用するようになりました。
- **kn func** CLI プラグインは **func** 1.8.1 を使用するようになりました。
- namespace スコープのブローカーがテクノロジープレビューとして利用できるようになりました。このブローカーは、たとえば、ロールベースのアクセス制御 (RBAC) ポリシーを実装するために使用できます。
- **KafkaSink は、** デフォルトで **CloudEvent** バイナリーコンテンツモードを使用するようになりました。バイナリーコンテンツモードは、**CloudEvent** の代わりにボディーのヘッダーを使用するため、構造化モードよりも効率的です。たとえば、HTTP プロトコルの場合、HTTP ヘッダーを使用します。
- OpenShift Container Platform 4.10 以降で OpenShift Route を使用して、外部トラフィックに HTTP/2 プロトコルを介して gRPC フレームワークを使用できるようになりました。これにより、クライアントとサーバー間の通信の効率と速度が向上します。
- Knative Operator Serving および Eventings CRD の API バージョン **v1alpha1** は、1.27 で非推奨になりました。これは今後のバージョンで削除される予定です。Red Hat は、代わりに **v1beta1** バージョンを使用することを強く推奨しています。Serverless Operator のアップグレード時に CRD が自動的に更新されるため、これは既存のインストールには影響しません。
- 配信タイムアウト機能がデフォルトで有効になりました。これを使用すると、送信される HTTP リクエストごとにタイムアウトを指定できます。この機能は、引き続きテクノロジープレビューです。

1.5.2. 修正された問題

- 以前は、Knative サービスが **Ready** 状態にならないことがあり、ロードバランサーの準備が整うのを待っていると報告されていました。この問題は修正されました。

1.5.3. 既知の問題

- OpenShift Serverless を Red Hat OpenShift Service Mesh と統合すると、クラスターに存在するシークレットが多すぎると、起動時に **net-kourier** Pod がメモリー不足になります。
- namespace スコープブローカーは、namespace スコープブローカーを削除した後でも、**ClusterRoleBindings** をユーザーの namespace に残す場合があります。
  これが発生した場合は、ユーザー namespace で **rbac-proxy-reviews-prom-rb-knative-kafka-broker-data-plane-{{.Namespace}}**という名前の **ClusterRoleBinding** を削除します。
- Ingress に **net-istio** を使用し、**security.dataPlane.mtls: true** を使用して SMCP 経由で mTLS を有効にする場合、Service Mesh は ***.local** ホストの **DestinationRules** をデプロイしますが、これは OpenShift Serverless の **DomainMapping** を許可しません。
  この問題を回避するには、**security.dataPlane.mtls: true** を使用する代わりに **PeerAuthentication** をデプロイして mTLS を有効にします。

1.6. RED HAT OPENSHIFT SERVERLESS 1.26 のリリースノート

OpenShift Serverless 1.26 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.6.1. 新機能

- Quarkus を使用した OpenShift Serverless Functions が GA になりました。
- OpenShift Serverless は Knative Serving 1.5 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 1.5 を使用するようになりました。
- OpenShift Serverless は Kourier 1.5 を使用するようになりました。
- OpenShift Serverless は Knative (**kn**) CLI 1.5 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 1.5 を使用するようになりました。
- OpenShift Serverless は Knative Operator 1.3 を使用するようになりました。
- **kn func** CLI プラグインは **func** 1.8.1 を使用するようになりました。
- Persistent Volume Claims (PVC) が GA になりました。PVC は、Knative サービスに永続的なデータストレージを提供します。
- 新しいトリガーフィルター機能が開発者プレビューとして利用できるようになりました。これにより、ユーザーはフィルター式のセットを指定できます。各式は、イベントごとに true または false に評価されます。
  新しいトリガーフィルターを有効にするには、Operator Config Map の **KnativeEventing** タイプのセクションに **new-trigger-filters: enabled** エントリーを追加します。

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

- Knative Operator 1.3 は、**operator.knative.dev** の更新された **v1beta1** バージョンの API を追加します。
  KnativeServing および **KnativeEventing** カスタムリソース Config Map で **v1alpha1** から **v1beta1** に更新するには、**apiVersion** キーを編集します。

  KnativeServing カスタムリソース Config Map の例

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

  KnativeEventing カスタムリソース Config Map の例

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

1.6.2. 修正された問題

- 以前、連邦情報処理標準 (FIPS) モードは、Kafka ブローカー、Kafka ソース、および Kafka シンクに対して無効になっていました。これは修正され、FIPS モードが利用できるようになりました。

1.6.3. 既知の問題

- Ingress に **net-istio** を使用し、**security.dataPlane.mtls: true** を使用して SMCP 経由で mTLS を有効にする場合、Service Mesh は ***.local** ホストの **DestinationRules** をデプロイしますが、これは OpenShift Serverless の **DomainMapping** を許可しません。
  この問題を回避するには、**security.dataPlane.mtls: true** を使用する代わりに **PeerAuthentication** をデプロイして mTLS を有効にします。

関連情報

- 新しいトリガーフィルターに関する Knative ドキュメント

1.7. RED HAT OPENSHIFT SERVERLESS 1.25.0 のリリースノート

OpenShift Serverless 1.25.0 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.7.1. 新機能

- OpenShift Serverless は Knative Serving 1.4 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 1.4 を使用するようになりました。
- OpenShift Serverless は Kourier 1.4 を使用するようになりました。
- OpenShift Serverless は Knative (**kn**) CLI 1.4 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 1.4 を使用するようになりました。
- **kn func** CLI プラグインは **func** 1.7.0 を使用するようになりました。
- 関数を作成およびデプロイするための統合開発環境 (IDE) プラグインが、Visual Studio Code および IntelliJ で利用できるようになりました。
- Knative Kafka ブローカーが一般提供されるようになりました。Knative Kafka ブローカーは、Apache Kafka を直接ターゲットとする、Knative ブローカー API の高性能な実装です。
  MT-Channel-Broker ではなく、Knative Kafka ブローカーを使用することをお勧めします。
- Knative Kafka シンクが一般提供されるようになりました。**KafkaSink** は **CloudEvent** を取得し、Apache Kafka トピックに送信します。イベントは、構造化コンテンツモードまたはバイナリーコンテンツモードのいずれかで指定できます。
- 内部トラフィックの TLS の有効化がテクノロジープレビューとして利用可能になりました。

1.7.2. 修正された問題

- 以前のバージョンでは、Knative Serving には liveness プローブの失敗後にコンテナーが再起動された場合に readiness プローブが失敗する問題がありました。この問題は修正されました。

1.7.3. 既知の問題

- 連邦情報処理標準 (FIPS) モードは、Kafka ブローカー、Kafka ソース、および Kafka シンクに対して無効になっています。
- **SinkBinding** オブジェクトは、サービスのカスタムリビジョン名をサポートしません。
- Knative Serving Controller Pod は、クラスター内のシークレットを監視するための新しいインフォーマーを追加します。インフォーマーはシークレットをキャッシュに含めるため、コントローラー Pod のメモリー消費量が増加します。
  Pod のメモリーが不足している場合は、デプロイのメモリー制限を増やすことで問題を回避できます。
- Ingress に **net-istio** を使用し、**security.dataPlane.mtls: true** を使用して SMCP 経由で mTLS を有効にする場合、Service Mesh は ***.local** ホストの **DestinationRules** をデプロイしますが、これは OpenShift Serverless の **DomainMapping** を許可しません。
  この問題を回避するには、**security.dataPlane.mtls: true** を使用する代わりに **PeerAuthentication** をデプロイして mTLS を有効にします。

OpenShift Container Platform の関連情報

- TLS 認証の設定

1.8. RED HAT OPENSHIFT SERVERLESS 1.24.0 のリリースノート

OpenShift Serverless 1.24.0 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.8.1. 新機能

- OpenShift Serverless は Knative Serving 1.3 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 1.3 を使用するようになりました。
- OpenShift Serverless は Kourier 1.3 を使用するようになりました。
- OpenShift Serverless は、Knative (**kn**) CLI 1.3 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 1.3 を使用するようになりました。
- **kn func** CLI プラグインが **func** 0.24 を使用するようになりました。
- Knative サービスの初期化コンテナーのサポートが一般提供 (GA) になりました。
- OpenShift Serverless ロジックが開発者プレビューとして利用できるようになりました。これにより、サーバーレスアプリケーションを管理するための宣言型ワークフローモデルを定義できます。
- OpenShift Container Platform では、OpenShift Serverless で Cost Management サービスを使用できるようになりました。

1.8.2. 修正された問題

- OpenShift Serverless を Red Hat OpenShift Service Mesh と統合すると、クラスターに存在するシークレットが多すぎると、起動時に **net-istio-controller** Pod がメモリー不足になります。

シークレットフィルタリングを有効にできるようになりました。これにより、**net-istio-controller** は、**networking.internal.knative.dev/certificate-uid** ラベルを持つシークレットのみを考慮するようになり、必要なメモリー量が削減されます。

- OpenShift Serverless Functions テクノロジープレビューは、デフォルトで Cloud Native Buildpacks を使用してコンテナーイメージをビルドするようになりました。

1.8.3. 既知の問題

- 連邦情報処理標準 (FIPS) モードは、Kafka ブローカー、Kafka ソース、および Kafka シンクに対して無効になっています。

- OpenShift Serverless 1.23 では、KafkaBindings および kafka **-binding** Webhook のサポートが削除されました。ただし、既存の **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** が残り、もはや存在しない **kafka-source-webhook** サービスを指している可能性があります。
クラスター上の KafkaBindings の特定の仕様については、**kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** を設定して、Webhook を介して Deployment、Knative Services、または Jobs などのさまざまなリソースに作成および更新イベントを渡すことができます。その後失敗します。

  この問題を回避するには、OpenShift Serverless 1.23 にアップグレードした後、クラスターから **kafkabindings.webhook.kafka.sources.knative.dev MutatingWebhookConfiguration** を手動で削除します。

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

- Ingress に **net-istio** を使用し、**security.dataPlane.mtls: true** を使用して SMCP 経由で mTLS を有効にする場合、Service Mesh は ***.local** ホストの **DestinationRules** をデプロイしますが、これは OpenShift Serverless の **DomainMapping** を許可しません。
この問題を回避するには、**security.dataPlane.mtls: true** を使用する代わりに **PeerAuthentication** をデプロイして mTLS を有効にします。

1.9. RED HAT OPENSHIFT SERVERLESS 1.23.0 のリリースノート

OpenShift Serverless 1.23.0 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.9.1. 新機能

- OpenShift Serverless は Knative Serving 1.2 を使用するようになりました。

- OpenShift Serverless は Knative Eventing 1.2 を使用するようになりました。

- OpenShift Serverless は Kourier 1.2 を使用するようになりました。

- OpenShift Serverless は Knative (**kn**) CLI 1.2 を使用するようになりました。

- OpenShift Serverless は Knative Kafka 1.2 を使用するようになりました。

- **kn func** CLI プラグインが **func** 0.24 を使用するようになりました。

- Kafka ブローカーで **kafka.eventing.knative.dev/external.topic** アノテーションを使用できるようになりました。このアノテーションを使用すると、ブローカー自体の内部トピックを作成する代わりに、既存の外部管理トピックを使用できます。

- **kafka-ch-controller** および **kafka-webhook** Kafka コンポーネントが存在しなくなりました。これらのコンポーネントは **kafka-webhook-eventing** コンポーネントに置き換えられました。

- OpenShift Serverless Functions テクノロジープレビューは、デフォルトで Source-to-Image (S2I) を使用してコンテナーイメージをビルドするようになりました。

1.9.2. 既知の問題

- 連邦情報処理標準 (FIPS) モードは、Kafka ブローカー、Kafka ソース、および Kafka シンクに対して無効になっています。

- Kafka ブローカーを含む namespace を削除する場合、ブローカーの **auth.secret.ref.name** シークレットがブローカーの前に削除されると、namespace ファイナライザーが削除されない可能性があります。

- 多数の Knative サービスで OpenShift Serverless を実行すると、Knative アクティベーター Pod がデフォルトのメモリー制限である 600MB 近くで実行される可能性があります。これらの Pod は、メモリー消費がこの制限に達すると再起動される可能性があります。アクティベーターデプロイメントの要求と制限は、**KnativeServing** カスタムリソースを変更することで設定できます。

```
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
```

- 関数のローカルビルド戦略として CloudNativeBuildpack を使用している場合、**kn func** は podman を自動的に起動したり、リモートデーモンへの SSH トンネルを使用したりすることはできません。これらの問題の回避策は、関数をデプロイする前に、ローカル開発コンピューターで Docker または podman デーモンを既に実行していることです。

- 現時点で、クラスター上の関数ビルドが Quarkus および Golang ランタイムで失敗します。これらは Node、Typescript、Python、および Springboot ランタイムで正常に機能します。

- Ingress に **net-istio** を使用し、**security.dataPlane.mtls: true** を使用して SMCP 経由で mTLS を有効にする場合、Service Mesh は ***.local** ホストの **DestinationRules** をデプロイしますが、これは OpenShift Serverless の **DomainMapping** を許可しません。
この問題を回避するには、**security.dataPlane.mtls: true** を使用する代わりに **PeerAuthentication** をデプロイして mTLS を有効にします。

OpenShift Container Platform の関連情報

- Source-to-Image

1.10. RED HAT OPENSHIFT SERVERLESS 1.22.0 のリリースノート

OpenShift Serverless 1.22.0 が公開されました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.10.1. 新機能

- OpenShift Serverless は Knative Serving 1.1 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 1.1 を使用するようになりました。
- OpenShift Serverless は Kourier 1.1 を使用するようになりました。
- OpenShift Serverless は Knative (**kn**) CLI 1.1 を使用するようになりました。
- OpenShift Serverless は KnativeKafka1.1 を使用するようになりました。
- **kn func** CLI プラグインは **func** 0.23 を使用するようになりました。
- Knative サービスの初期コンテナーサポートがテクノロジープレビューとして利用できるようになりました。
- Knative サービスの永続ボリュームクレーム (PVC) サポートが、テクノロジープレビューとして利用できるようになりました。
- **knative-serving**、**knative-serving-ingress**、**knative-eventing**、および **knative-kafka** システムネームボックスに、デフォルトで **knative.openshift.io/part-of:"openshift-serverless"** ラベルが付けられるようになりました。
- **Knative Eventing-Kafka Broker/Trigger** ダッシュボードが追加されました。これにより、Web コンソールで Kafka ブローカーとトリガーメトリックを視覚化できます。
- **Knative Eventing-KafkaSink** ダッシュボードが追加されました。これにより、Web コンソールで KafkaSink メトリックを視覚化できます。
- **Knative Eventing-Broker/Trigger** ダッシュボードは、**Knative Eventing-Channel-based Broker/Trigger** と呼ばれるようになりました。
- **knative.openshift.io/part-of: " openshift-serverless "** ラベルが、**knative.openshift.io/system-namespace** ラベルに置き換わりました。
- Knative Serving YAML 設定ファイルの命名スタイルがキャメルケース (**ExampleName**) からハイフンスタイル (**example-name**) に変更されました。このリリース以降、Knative Serving YAML 設定ファイルを作成または編集するときは、ハイフンスタイルの表記を使用してください。

1.10.2. 既知の問題

- 連邦情報処理標準 (FIPS) モードは、Kafka ブローカー、Kafka ソース、および Kafka シンクに対して無効になっています。

1.11. RED HAT OPENSHIFT SERVERLESS 1.21.0 のリリースノート

OpenShift Serverless 1.21.0 が利用可能になりました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.11.1. 新機能

- OpenShift Serverless は Knative Serving 1.0 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 1.0 を使用するようになりました。
- OpenShift Serverless は Kourier 1.0 を使用するようになりました。
- OpenShift Serverless は、Knative (**kn**) CLI 1.0 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 1.0 を使用するようになりました。
- **kn func** CLI プラグインが **func** 0.21 を使用するようになりました。
- Kafka シンクがテクノロジープレビューとして利用できるようになりました。
- Knative オープンソースプロジェクトは、camel-cased 設定キーを廃止し、kebab-cased キーを一貫して使用することを支持し始めました。その結果、OpenShift Serverless 1.18.0 リリースノートで前述した **defaultExternalScheme** キーは非推奨になり、**default-external-scheme** キーに置き換えられました。キーの使用方法は同じです。

1.11.2. 修正された問題

- OpenShift Serverless 1.20.0 では、サービスにイベントを送信するための **kn event send** の使用に影響するイベント配信の問題がありました。この問題は修正されています。
- OpenShift Serverless 1.20.0 (**func**0.20) では、**http** テンプレートを使用して作成された TypeScript 関数をクラスターにデプロイできませんでした。この問題は修正されています。
- OpenShift Serverless 1.20.0 (**func** 0.20) では、**gcr.io** レジストリーを使用した関数のデプロイがエラーで失敗しました。この問題は修正されています。
- OpenShift Serverless 1.20.0 (**func** 0.20) では、**kn func create** コマンドを使用して Springboot 関数プロジェクトディレクトリーを作成してから、**kn func build** コマンドを実行するとエラーメッセージが表示されて失敗しました。この問題は修正されています。
- OpenShift Serverless 1.19.0 (**func** 0.19) では、一部のランタイムが podman を使用して関数をビルドできませんでした。この問題は修正されています。

1.11.3. 既知の問題

- 現在、ドメインマッピングコントローラーは、現在サポートされていないパスを含むブローカーの URI を処理できません。
  つまり、**DomainMapping** カスタムリソース (CR) を使用してカスタムドメインをブローカーにマップする場合は、ブローカーの入力サービスを使用して **DomainMapping** CR を設定し、ブローカーの正確なパスをカスタムドメインに追加する必要があります。

  DomainMappingCR の例

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

```
name: broker-ingress
kind: Service
apiVersion: v1
```

その場合、ブローカーの URI は **<domain-name>/<broker-namespace>/<broker-name>** になります。

1.12. RED HAT OPENSHIFT SERVERLESS 1.20.0 のリリースノート

OpenShift Serverless 1.20.0 が利用可能になりました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.12.1. 新機能

- OpenShift Serverless は Knative Serving 0.26 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 0.26 を使用するようになりました。
- OpenShift Serverless は Kourier 0.26 を使用するようになりました。
- OpenShift Serverless は、Knative (**kn**) CLI 0.26 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 0.26 を使用するようになりました。
- **kn func** CLI プラグインが **func** 0.20 を使用するようになりました。
- Kafka ブローカーがテクノロジープレビュー機能として利用可能になりました。

重要

現在テクノロジープレビューにある Kafka ブローカーは、FIPS ではサポートされていません。

- **kn event** プラグインがテクノロジープレビューとして利用できるようになりました。
- **knservicecreate** コマンドの **--min-scale** フラグと **--max-scale** フラグは廃止されました。代わりに、**-scale-min** フラグと **--scale-max** フラグを使用してください。

1.12.2. 既知の問題

- OpenShift Serverless は、HTTPS を使用するデフォルトアドレスで Knative サービスをデプロイします。クラスター内のリソースにイベントを送信する場合、送信側にはクラスターの認証局 (CA) が設定されていません。これにより、クラスターがグローバルに受け入れ可能な証明書を使用しない限り、イベント配信は失敗します。
  たとえば、一般にアクセス可能なアドレスへのイベント配信は機能します。

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

一方、サービスがカスタム CA によって発行された HTTPS 証明書でパブリックアドレスを使用する場合、この配信は失敗します。

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

ブローカーやチャネルなどの他のアドレス指定可能なオブジェクトへのイベント送信は、この問題の影響を受けず、期待どおりに機能します。

- 現在、Kafka ブローカーは Federal Information Processing Standards (FIPS) モードが有効になっているクラスターでは動作しません。
- **kn func create** コマンドで Springboot 関数プロジェクトディレクトリーを作成する場合、それ以降の **kn func build** コマンドの実行は、以下のエラーメッセージと共に失敗します。

```
[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
```

回避策として、関数設定ファイル **func.yaml** で **builder** プロパティーを **gcr.io/paketo-buildpacks/builder:base** に変更します。

- **gcr.io** レジストリーを使用した関数のデプロイは、以下のエラーメッセージと共に失敗します。

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

回避策として、**quay.io** または **docker.io** などの **gcr.io** 以外のレジストリーを使用します。

- **http** テンプレートで作成された Typescript 関数は、クラスターへのデプロイに失敗します。回避策として、**func.yaml** ファイルで以下のセクションを置き換えます。

```
buildEnvs: []
```

上記を以下のように置き換えます。

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

- **func** バージョン 0.20 では、一部のランタイムが podman を使用して関数をビルドできない場合があります。以下のようなエラーメッセージが表示される場合があります。

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

  - この問題には、以下の回避策があります。
    a. **--time=0** をサービス **ExecStart** 定義に追加して、podman サービスを更新します。

サービス設定の例

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

    b. 以下のコマンドを実行して podman サービスを再起動します。

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

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

- または、TCP を使用して podman API を公開することもできます。

```
$ 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 のリリースノート

OpenShift Serverless 1.19.0 が利用可能になりました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.13.1. 新機能

- OpenShift Serverless は Knative Serving 0.25 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 0.25 を使用するようになりました。
- OpenShift Serverless は Kourier 0.25 を使用するようになりました。
- OpenShift Serverless は Knative (**kn**) CLI 0.25 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 0.25 を使用するようになりました。
- **kn func** CLI プラグインが **func** 0.19 を使用するようになりました。
- **KafkaBinding** API は OpenShift Serverless 1.19.0 で非推奨となり、今後のリリースで廃止される予定です。
- HTTPS リダイレクトがサポートされ、クラスターに対してグローバルに設定することも、各 Knative サービスごとに設定することもできるようになりました。

1.13.2. 修正された問題

- 以前のリリースでは、Kafka チャネルディスパッチャーは、ローカルコミットが成功するのを待ってからしか応答していませんでした。これにより、Apache Kafka ノードに障害が発生した場合に、イベントが失われる可能性がありました。Kafka チャネルディスパッチャーは、すべての同期レプリカがコミットするのを待ってから応答するようになりました。

1.13.3. 既知の問題

- **func** バージョン 0.19 では、一部のランタイムが podman を使用して関数をビルドできない場合があります。以下のようなエラーメッセージが表示される場合があります。

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

  - この問題には、以下の回避策があります。
    a. **--time=0** をサービス **ExecStart** 定義に追加して、podman サービスを更新します。

    サービス設定の例

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

    b. 以下のコマンドを実行して podman サービスを再起動します。

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

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

- または、TCP を使用して podman API を公開することもできます。

```
$ 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 のリリースノート

OpenShift Serverless 1.18.0 が利用可能になりました。以下では、OpenShift Container Platform 上の OpenShift Serverless に関連する新機能、変更点、および既知の問題を説明します。

1.14.1. 新機能

- OpenShift Serverless は Knative Serving 0.24.0 を使用するようになりました。
- OpenShift Serverless は Knative Eventing 0.24.0 を使用するようになりました。
- OpenShift Serverless は Kourier 0.24.0 を使用するようになりました。
- OpenShift Serverless は Knative (**kn**) CLI 0.24.0 を使用するようになりました。
- OpenShift Serverless は Knative Kafka 0.24.7 を使用するようになりました。
- **kn func** CLI プラグインが **func** 0.18.0 を使用するようになりました。
- 今後の OpenShift Serverless 1.19.0 リリースでは、外部ルートの URL スキームはデフォルトで HTTPS になり、セキュリティーが強化されます。
  この変更をワークロードに適用する必要がない場合は、以下の YAML を **KnativeServing** カスタムリソース (CR) に追加してから 1.19.0 にアップグレードする前にデフォルト設定を上書きできます。

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

  変更を 1.18.0 ですでに適用する必要がある場合には、以下の YAML を追加します。

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

- 今後の OpenShift Serverless 1.19.0 リリースでは、Kourier ゲートウェイが公開されるデフォルトのサービスタイプは **ClusterIP** であり、**LoadBalancer** ではありません。

この変更をワークロードに適用する必要がない場合は、以下の YAML を **KnativeServing** カスタムリソース定義 (CRD) に追加してから 1.19.0 にアップグレードする前にデフォルト設定を上書きできます。

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

- OpenShift Serverless で **emptyDir** ボリュームを使用できるようになりました。詳細は、Knative Serving に関する OpenShift Serverless ドキュメントを参照してください。
- **kn func** を使用して関数を作成すると、Rust テンプレートが利用できるようになりました。

1.14.2. 修正された問題

- 以前の 1.4 バージョンの Camel-K は OpenShift Serverless 1.17.0 と互換性がありませんでした。Camel-K の問題が修正され、Camel-K バージョン 1.4.1 を OpenShift Serverless 1.17.0 で使用できます。
- 以前のバージョンでは、Kafka チャネルまたは新しい Kafka ソースの新しいサブスクリプションを作成する場合は、新しく作成されたサブスクリプションまたはシンクが準備完了ステータスを報告した後、Kafka データプレーンがメッセージをディスパッチする準備ができるまでに遅延が生じる可能性があります。
その結果、データプレーンが準備完了ステータスを報告していないときに送信されたメッセージは、サブスクライバーまたはシンクに配信されない可能性があります。

  OpenShift Serverless 1.18.0 では、問題が修正され、初期メッセージが失われなくなりました。この問題の詳細は、ナレッジベースの記事 #6343981 を参照してください。

1.14.3. 既知の問題

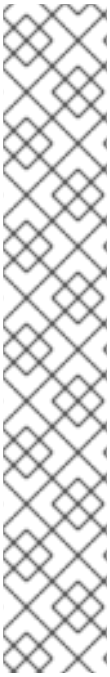
- Knative **kn** CLI の古いバージョンは、Knative Serving および Knative Eventing API の古いバージョンを使用する可能性があります。たとえば、**kn** CLI のバージョン 0.23.2 は **v1alpha1** API バージョンを使用します。
一方、OpenShift Serverless の新しいリリースでは、古い API バージョンをサポートしない可能性があります。たとえば、OpenShift Serverless 1.18.0 は **kafkasources.sources.knative.dev** API のバージョン **v1alpha1** をサポートしなくなりました。

  そのため、**kn** が古い API を検出できないため、新しい OpenShift Serverless で古いバージョンの Knative **kn** CLI を使用するとエラーが発生する可能性がありました。たとえば、**kn** CLI のバージョン 0.23.2 は OpenShift Serverless 1.18.0 では機能しません。

  問題を回避するには、OpenShift Serverless リリースで利用可能な最新の **kn** CLI バージョンを使用します。OpenShift Serverless 1.18.0 については、Knative **kn** CLI 0.24.0 を使用します。

第2章 OPENSHIFT SERVERLESS の概要

OpenShift Serverless は、Kubernetes ネイティブなビルディングブロックを提供します。開発者はこれらを使用して、OpenShift Container Platform 上でサーバーレスのイベント駆動型アプリケーションを作成およびデプロイできます。OpenShift Serverless はオープンソースの Knative プロジェクト をベースとし、エンタープライズレベルのサーバーレスプラットフォームを有効にすることで、ハイブリッドおよびマルチクラウド環境に対して移植性と一貫性をもたらします。



注記

OpenShift Serverless は Red Hat OpenShift Serverless とは異なる頻度でリリースされるため、OpenShift Serverless ドキュメントは製品のマイナーバージョンごとに個別のドキュメントセットとして利用できるようになりました。

OpenShift Serverless ドキュメントは https://docs.openshift.com/serverless/ で利用できます。

特定のバージョンのドキュメントは、バージョンセレクターのドロップダウンを使用するか、URL にバージョン (https://docs.openshift.com/serverless/1.28 など) を直接追加することによって利用できます。

さらに、OpenShift Serverless ドキュメントは、Red Hat カスタマーポータル (https://access.redhat.com/documentation/ja-jp/red_hat_openshift_serverless/) でも利用できます。

OpenShift Serverless のライフサイクルとサポートされるプラットフォームに関する追加情報は、プラットフォームライフサイクルポリシー を参照してください。

2.1. 関連情報

- カスタムリソース定義による Kubernetes API の拡張
- カスタムリソース定義からのリソースの管理
- What is serverless?

第3章 KNATIVE SERVING

Knative Serving は、クラウドネイティブアプリケーション の作成、デプロイ、管理を希望する開発者をサポートします。これにより、オブジェクトのセットが OpenShift Container Platform クラスター上のサーバーレスワークロードの動作を定義し制御する Kubernetes カスタムリソース定義 (CRD) として提供されます。

開発者はこれらの CRD を使用して、複雑なユースケースに対応するためにビルディングブロックとして使用できるカスタムリソース (CR) インスタンスを作成します。以下に例を示します。

- サーバーレスコンテナーの迅速なデプロイ
- Pod の自動スケーリング

3.1. KNATIVE SERVING リソース

サービス

service.serving.knative.dev CRD はワークロードのライフサイクルを自動的に管理し、アプリケーションがネットワーク経由でデプロイされ、到達可能であることを確認します。これは、ユーザーが作成したサービスまたはカスタムリソースに対して加えられるそれぞれの変更についての ルート、設定、および新規リビジョンを作成します。Knative での開発者の対話のほとんどは、サービスを変更して実行されます。

Revision

revision.serving.knative.dev CRD は、ワークロードに対して加えられるそれぞれの変更についてのコードおよび設定の特定の時点におけるスナップショットです。Revision (リビジョン) はイミュータブル (変更不可) オブジェクトであり、必要な期間保持することができます。

Route

route.serving.knative.dev CRD は、ネットワークのエンドポイントを、1 つ以上のリビジョンにマップします。部分的なトラフィックや名前付きルートなどのトラフィックを複数の方法で管理することができます。

Configuration

configuration.serving.knative.dev CRD は、デプロイメントの必要な状態を維持します。これにより、コードと設定を明確に分離できます。設定を変更すると、新規リビジョンが作成されます。

第4章 KNATIVE EVENTING

OpenShift Container Platform 上の Knative Eventing を使用すると、開発者はサーバーレスアプリケーションと共に イベント駆動型のアーキテクチャー を使用できます。イベント駆動型のアーキテクチャーは、イベントプロデューサーとイベントコンシューマー間の関係を切り離すという概念に基づいています。

イベントプロデューサーはイベントを作成し、イベント**シンク**またはコンシューマーはイベントを受信します。Knative Eventing は、標準の HTTP POST リクエストを使用してイベントプロデューサーとシンク間でイベントを送受信します。これらのイベントは CloudEvents 仕様 に準拠しており、すべてのプログラミング言語でのイベントの作成、解析、および送受信を可能にします。

Knative Eventing は以下のユースケースをサポートします。

コンシューマーを作成せずにイベントを公開する

イベントを HTTP POST としてブローカーに送信し、バインディングを使用してイベントを生成するアプリケーションから宛先設定を分離できます。

パブリッシャーを作成せずにイベントを消費

Trigger を使用して、イベント属性に基づいて Broker からイベントを消費できます。アプリケーションはイベントを HTTP POST として受信します。

複数のタイプのシンクへの配信を有効にするために、Knative Eventing は複数の Kubernetes リソースで実装できる以下の汎用インターフェイスを定義します。

アドレス指定可能なリソース

HTTP 経由でイベントの **status.address.url** フィールドに定義されるアドレスに配信されるイベントを受信し、確認することができます。Kubernetes **Service** リソースはアドレス指定可能なインターフェイスにも対応します。

呼び出し可能なリソース

HTTP 経由で配信されるイベントを受信し、これを変換できます。HTTP 応答ペイロードで **0** または **1** の新規イベントを返します。返されるイベントは、外部イベントソースからのイベントが処理されるのと同じ方法で処理できます。

4.1. APACHE KAFKA の KNATIVE ブローカーの使用

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。

Apache Kafka の Knative ブローカーは、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

4.2. 関連情報

- **KnativeKafka** カスタムリソースのインストール

- Red Hat AMQ Streams のドキュメント
- Apache Kafka での Red Hat AMQ Streams TLS および SASL のドキュメント
- イベント配信

第5章 OPENSHIFT SERVERLESS FUNCTIONS について

OpenShift Serverless Functions により、開発者は OpenShift Container Platform で Knative サービスとしてステートレスでイベント駆動型の関数を作成およびデプロイできます。**kn func CLI** は Knative **kn** CLI のプラグインとして提供されます。**kn func** CLI を使用して、クラスター上の Knative サービスとしてコンテナーイメージを作成、ビルド、デプロイできます。

5.1. 含まれるランタイム

OpenShift Serverless Functions は、以下のランタイムの基本機能を作成するために使用できるテンプレートを提供します。

- Quarkus
- Node.js
- TypeScript

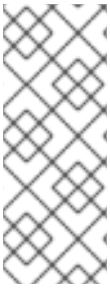
5.2. 次のステップ

- Getting started with functions

第6章 OPENSHIFT SERVERLESS のサポート

本書で説明されている手順で問題が発生した場合は、Red Hat カスタマーポータル (http://access.redhat.com) にアクセスしてください。Red Hat Customer Portal を使用して、Red Hat 製品に関するテクニカルサポート記事の Red Hat ナレッジベースを検索または閲覧できます。Red Hat Global Support Services (GSS) にサポートケースを送信したり、他の製品ドキュメントにアクセスしたりすることもできます。

このガイドを改善するための提案がある場合、またはエラーを見つけた場合は、最も関連性の高いドキュメントコンポーネントの Jira イシュー を送信できます。コンテンツを簡単に見つけられるよう、セクション番号、ガイド名、OpenShift Serverless のバージョンなどの詳細情報を記載してください。



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

6.1. RED HAT ナレッジベースについて

Red Hat ナレッジベース は、お客様が Red Hat の製品やテクノロジーを最大限に活用できるようにするための豊富なコンテンツを提供します。Red Hat ナレッジベースは、Red Hat 製品のインストール、設定、および使用に関する記事、製品ドキュメント、および動画で設定されています。さらに、簡潔な根本的な原因についての説明や修正手順を説明した既知の問題のソリューションを検索できます。

6.2. RED HAT ナレッジベースの検索

Red Hat OpenShift Serviceless の問題が発生した場合には、初期検索を実行して、Red Hat ナレッジベースにソリューションがすでに存在しているかどうかを確認できます。

前提条件

- Red Hat カスタマーポータルのアカウントがある。

手順

1. Red Hat カスタマーポータル にログインします。
2. 主な Red Hat カスタマーポータルの検索フィールドには、問題に関連する入力キーワードおよび文字列を入力します。これらには、以下が含まれます。
   - OpenShift Container Platform コンポーネント (**etcd** など)
   - 関連する手順 (**installation** など)
   - 明示的な失敗に関連する警告、エラーメッセージ、およびその他の出力
3. **Search** をクリックします。
4. **OpenShift Container Platform** 製品フィルターを選択します。
5. **ナレッジベース** のコンテンツタイプフィルターを選択します。

6.3. サポートケースの送信

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- OpenShift CLI (**oc**) がインストールされている。
- Red Hat カスタマーポータルのアカウントがある。
- Red Hat の標準またはプレミアムサブスクリプションがある。

手順

1. Red Hat カスタマーポータル にログインし、**SUPPORT CASES → Open a case** を選択します。
2. 問題の該当するカテゴリー (**Defect / Bug** など)、製品 (**OpenShift Container Platform**)、およびすでに自動入力されていない場合は製品バージョンを選択します。
3. Red Hat ナレッジベースで推奨されるソリューション一覧を確認してください。この一覧に上げられているソリューションは、報告しようとしている問題に適用される可能性があります。提案されている記事が問題に対応していない場合は、**Continue** をクリックします。
4. 問題の簡潔で説明的な概要と、確認されている現象および予想される動作の詳細情報を入力します。
5. 報告している問題に対する一致に基づいて推奨される Red Hat ナレッジベースソリューションの一覧が更新されることを確認してください。ケース作成プロセスでより多くの情報を提供すると、この一覧の絞り込みが行われます。提案されている記事が問題に対応していない場合は、**Continue** をクリックします。
6. アカウント情報が予想通りに表示されていることを確認し、そうでない場合は適宜修正します。
7. 自動入力された Red Hat OpenShift Serverless クラスター ID が正しいことを確認します。正しくない場合は、クラスター ID を手動で取得します。
   - OpenShift Container Platform Web コンソールを使用してクラスター ID を手動で取得するには、以下を実行します。
     a. **Home → Dashboards → Overview** に移動します。
     b. **Details** セクションの **Cluster ID** フィールドで値を見つけます。
   - または、OpenShift Container Platform Web コンソールで新規サポートケースを作成し、クラスター ID を自動的に入力することができます。
     a. ツールバーから、**(?) Help → Open Support Case** に移動します。
     b. **Cluster ID** 値が自動的に入力されます。
   - OpenShift CLI (**oc**) を使用してクラスター ID を取得するには、以下のコマンドを実行します。

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

8. プロンプトが表示されたら、以下の質問に入力し、**Continue** をクリックします。
   - 動作はどこで発生しているか？どの環境を使用しているか。
   - 動作はいつ発生するか。頻度は。繰り返し発生するか。特定のタイミングで発生するか。
   - 時間枠およびビジネスへの影響について提供できるどのような情報があるか？
9. 関連する診断データファイルをアップロードし、**Continue** をクリックします。

まず **oc adm must-gather** コマンドを使用して収集されるデータと、そのコマンドによって収集されない問題に固有のデータを含めることが推奨されます。

1. 関連するケース管理の詳細情報を入力し、**Continue** をクリックします。
2. ケースの詳細をプレビューし、**Submit** をクリックします。

6.4. サポート用の診断情報の収集

サポートケースを作成する際、ご使用のクラスターについてのデバッグ情報を Red Hat サポートに提供していただくと Red Hat のサポートに役立ちます。**must-gather** ツールを使用すると、OpenShift Serverless に関連するデータを含む、 OpenShift Container Platform クラスターについての診断情報を収集できます。迅速なサポートを得るには、OpenShift Container Platform と OpenShift Serverless の両方の診断情報を提供してください。

6.4.1. must-gather ツールについて

oc adm must-gather CLI コマンドは、以下のような問題のデバッグに必要となる可能性のあるクラスターからの情報を収集します。

- リソース定義
- サービスログ

デフォルトで、**oc adm must-gather** コマンドはデフォルトのプラグインイメージを使用し、**./must-gather.local** に書き込みを行います。

または、以下のセクションで説明されているように、適切な引数を指定してコマンドを実行すると、特定の情報を収集できます。

- 1 つ以上の特定の機能に関連するデータを収集するには、以下のセクションに示すように、イメージと共に **--image** 引数を使用します。
  以下に例を示します。

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

- 監査ログを収集するには、以下のセクションで説明されているように **--/usr/bin/gather_audit_logs** 引数を使用します。
  以下に例を示します。

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

注記

ファイルのサイズを小さくするために、監査ログはデフォルトの情報セットの一部として収集されません。

oc adm must-gather を実行すると、ランダムな名前を持つ新規 Pod がクラスターの新規プロジェクトに作成されます。データは Pod で収集され、**must-gather.local** で始まる新規ディレクトリーに保存されます。このディレクトリーは、現行の作業ディレクトリーに作成されます。

以下に例を示します。

```
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
...
```

オプションで、**--run-namespace** オプションを使用して、特定の namespace で **oc adm must-gather** コマンドを実行できます。

以下に例を示します。

```
$ 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 データの収集について

oc adm must-gather CLI コマンドを使用してクラスターについての情報を収集できます。これには、OpenShift Serverless に関連する機能およびオブジェクトが含まれます。**must-gather** を使用して OpenShift Serverless データを収集するには、インストールされたバージョンの OpenShift Serverless イメージおよびイメージタグを指定する必要があります。

前提条件

- OpenShift CLI (**oc**) がインストールされている。

手順

- **oc adm must-gather** コマンドを使用してデータを収集します。

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

コマンドの例

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