Red Hat 3scale API Management 2.9

3scale の移行

3scale API Management およびそのコンポーネントの移行またはアップグレード

Last Updated: 2023-03-11

Red Hat 3scale API Management 2.9 3scale の移行

3scale API Management およびそのコンポーネントの移行またはアップグレード

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

3scale をテンプレートベースから operator ベースのインストールに移行する方法を説明します。3scale およびそのコンポーネントを最新バージョンにアップグレードするための情報についても説明します。

目次

前書き

本ガイドでは、Red Hat 3scale API Management をテンプレートベースから operator ベースのインストールに移行するための情報、3scale インストールを 2.8 から 2.9 にアップグレードするのに必要な詳細情報、および operator ベースのデプロイメントで APIcast をアップグレードする手順について説明します。

テンプレートベースから operator ベースのデプロイメントに移行するには、3scale の移行ガイド に記載の手順を参照してください。

オンプレミス型 3scale のデプロイメントを 2.8 から 2.9 にアップグレードするには、インストールのタイプに応じて以下のガイドのいずれかを参照してください。

- テンプレートベースの 3scale のアップグレードガイド
- operator ベースの 3scale のアップグレードガイド

Developer Portal で API をプロビジョニングするためのアップグレード後のステップ。

- 3scale 2.9 へのアップグレードに成功した後、3scale 2.8 の Developer Portal で OpenAPI Specification 3.0 (OAS 3.0) が設定済みで、引き続き OAS 3.0 を使用する場合は、次を参照してください。OAS 3.0 が設定されたデベロッパーポータルの更新

operator ベースのデプロイメントで APIcast をアップグレードするには、operator ベースの APIcast のアップグレードガイド に記載の手順を参照してください。

第1章 3SCALE の移行ガイド: テンプレートベースから OPERATOR ベースのデプロイメントへ

本セクションでは、Red Hat 3scale API Management を Red Hat OpenShift 3.11 を使用するテンプレートベースのデプロイメントから Red Hat OpenShift 4.x を使用する operator ベースのデプロイメントに移行する方法について説明します。

警告

必要な条件および手順を理解するために、記載の手順を適用する前に、移行ガイド全体を読んでください。移行プロセスの手順が完了するまで、サービスの提供が中断されます。このサービス中断が生じるため、メンテナンス期間を設けるようにしてください。

1.1. 移行を行うための前提条件

3scale インストールをテンプレートベースから operator ベースのデプロイメントに移行する前に、以下のガイドを参照してデプロイメントがサポートされていることを確認してください。

- テンプレートベースの 3scale デプロイメントのバックアップ
- operator ベースのデプロイメントにおけるバックアップの復元

1.2. 3SCALE のテンプレートベースから OPERATOR ベースのデプロイメントへの移行

移行前の基本セットアップは、3scale が OCP3 ドメインをポイントする設定です。**3scale.example.com → ocp3.example.com**

3scale を Red Hat OpenShift 3.11 を使用するテンプレートベースのデプロイメントから Red Hat OpenShift 4.1 を使用する operator ベースのデプロイメントに移行するには、以下の手順に従います。

1. テンプレートベースのデプロイメントから 3scale のバックアップを作成する。
2. operator を使用して 3scale をデプロイする。
3. operator ベースのデプロイメントでバックアップを復元する。
4. 3scale WILDCARD_DOMAIN (ここでは **3scale.example.com**) を **ocp4.example.com** にポイントする。

上記の手順をすべて実施すると、3scale のテンプレートベースから operator ベースのデプロイメントへの移行が完了します。

第2章 テンプレートベースの 3SCALE のアップグレードガイド: 2.8 から 2.9.1 へ

本セクションでは、テンプレートベースのデプロイメントにおいて、Red Hat 3scale API Management をバージョン 2.8 から 2.9.1 にアップグレードする方法について説明します。

重要な留意事項

- 必要な条件および手順を理解するために、記載の手順を適用する前に、アップグレードガイド全体を読んでください。アップグレードプロセスの手順が完了するまで、サービスの提供が中断されます。このサービス中断が生じるため、メンテナーンス期間を設けるようにしてください。
- 3scale のテンプレートベースのインストール環境が Oracle Database と共に動作している場合は、3scale 2.9.1 リリースノートの Oracle データベースのサポート を参照してください。

2.1. アップグレードを行うための前提条件

本セクションでは、テンプレートベースのインストール環境において、3scale を 2.8 から 2.9.1 にアップグレードするのに必要な設定、タスク、およびツールについて説明します。

2.1.1. 設定

- OpenShift 3.11 上の 3scale では、テンプレートを使用した 2.8 から 2.9.1 へのアップグレードパスがサポートされます。

2.1.2. 主要なタスク

- OpenShift CLI ツールが 3scale をデプロイしたプロジェクトに設定されるようにする。
- 3scale で使用しているデータベースのバックアップを行う。バックアップの手順は、データベースのタイプや設定により異なります。

2.1.3. ツール

アップグレードを行うには、以下のツールが必要です。

- テンプレートを使用して OpenShift 3.11 のプロジェクトにデプロイされた 3scale 2.8
- Bash シェル: アップグレード手順で詳細を説明するコマンドを実行します。
- base64: シークレットの情報をエンコードおよびデコードします。
- jq: JSON 変換用です。

2.2. テンプレートベースのインストール環境における 2.8 から 2.9.1 へのアップグレード

テンプレートベースのインストール環境において、3scale 2.8 を 2.9.1 にアップグレードするには、本セクションに記載の手順に従います。

アップグレードを開始するには、3scale がデプロイされているプロジェクトに移動します。

```
$ oc project <3scale-project>
```

続いて、以下の順序で手順を実行します。

1. 「3scale プロジェクトのバックアップの作成」
2. 「BASE_URL フィールドの削除」
3. 「3scale のバージョン番号の更新」
4. 「3scale イメージのアップグレード」

2.2.1. 3scale プロジェクトのバックアップの作成

前の手順

なし

現在の手順

3scale プロジェクトのバックアップを作成するのに必要なアクションを以下のリストに示します。

1. 3scale で使用するデータベースに応じて、${SYSTEM_DB} を以下のいずれかの値に設定します。
   - データベースが MySQL の場合: **SYSTEM_DB=system-mysql**
   - データベースが PostgreSQL の場合: **SYSTEM_DB=system-postgresql**
2. 既存の DeploymentConfigs でバックアップファイルを作成します。

```
THREESCALE_DC_NAMES="apicast-production apicast-staging backend-cron backend-listener backend-redis backend-worker system-app system-memcache ${SYSTEM_DB} system-redis system-sidekiq system-sphinx zync zync-database zync-que"

for component in ${THREESCALE_DC_NAMES}; do oc get --export -o yaml dc ${component} > ${component}_dc.yml ; done
```

3. **export all** コマンドでエクスポートされるプロジェクト内のすべての既存 OpenShift リソースをバックアップします。

```
oc get -o yaml --export all > threescale-project-elements.yaml
```

4. **export all** コマンドでエクスポートされない追加の要素でバックアップファイルを作成します。

```
for object in rolebindings serviceaccounts secrets imagestreamtags cm rolebindingrestrictions limitranges resourcequotas pvc templates cronjobs statefulsets hpa deployments replicasets poddisruptionbudget endpoints
do
  oc get -o yaml --export $object > $object.yaml
done
```

5. 生成されたすべてのファイルが空ではないこと、およびそれらすべての内容が予想どおりであることを確認します。

次の手順

「BASE_URL フィールドの削除」

2.2.2. BASE_URL フィールドの削除

前の手順

「3scale プロジェクトのバックアップの作成」

現在の手順

本セクションでは、**system-master-apicast** シークレットから BASE_URL フィールドを削除する方法について説明します。3scale 2.9 ではこのフィールドを使用するコンポーネントはないので、このフィールドを削除します。

1. 以下のコマンドを実行します。

```
oc patch secret system-master-apicast --type=json -p='[{"op": "remove", "path": "/data/BASE_URL"}]'
```

2. BASE_URL フィールドが正しく削除されたことを確認します。

```
oc get secret system-master-apicast -o json | jq .data
```

次のステップ

「3scale のバージョン番号の更新」

2.2.3. 3scale のバージョン番号の更新

前の手順

「BASE_URL フィールドの削除」

現在の手順

この手順では、**system-environment** ConfigMap の 3scale のリリースバージョン番号を **2.8** から **2.9** に更新します。AMP_RELEASE は、一部の DeploymentConfig コンテナー環境で参照される ConfigMap のエントリーです。

1. AMP_RELEASE にパッチを適用するには、以下のコマンドを実行します。

```
oc patch cm system-environment --patch '{"data": {"AMP_RELEASE": "2.9"}}'
```

2. system-environment ConfigMap の AMP_RELEASE キーの値が **2.9** であることを確認します。

```
oc get cm system-environment -o json | jq '.data["AMP_RELEASE"]'
```

次のステップ

「3scale イメージのアップグレード」

2.2.4. 3scale イメージのアップグレード

前の手順

「3scale のバージョン番号の更新」

現在の手順

この手順では、アップグレードプロセスに必要な 3scale イメージを更新します。

2.2.4.1. system イメージのパッチ

1. 新しいイメージストリームタグを作成します。

```
oc patch imagestream/amp-system --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "AMP system 2.9"}, "from": { "kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/system-rhel7:3scale2.9"}, "name": "2.9", "referencePolicy": {"type": "Source"}}}]'
```

2. 手順を進めるには、ご自分の 3scale デプロイメントで使用しているデータベースを考慮してください。

   - データベースが Oracle DB の場合は、「システムイメージのパッチ適用: Oracle データベースを使用する 3Scale」に記載の手順に従ってください。
   - データベースが Oracle DB 以外の場合は、「システムイメージのパッチ適用: 他のデータベースを使用する 3scale」に記載の手順に従ってください。

2.2.4.1.1. システムイメージのパッチ適用: Oracle データベースを使用する 3Scale

1. Oracle Database を使用して 3scale のシステムイメージのパッチ適用を開始するには、データベースバージョンに応じて以下の手順のいずれかを実行します。

   a. Oracle 12c を使用して 3scale 2.8 から 2.9.1 に移行する

   b. Oracle 19c を使用して 3scale 2.8 から 2.9.1 に移行する

2. system-app ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

```
oc set triggers dc/system-app --from-image=amp-system:latest --containers=system-master,system-developer,system-provider --remove
```

   b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-app --from-image=amp-system:2.9-oracle --containers=system-master,system-developer,system-provider
```

   これがトリガーとなり **system-app** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

   c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d amp-system:latest
```

3. **system-sidekiq** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

```
oc set triggers dc/system-sidekiq --from-image=amp-system:latest --containers=system-sidekiq,check-svc --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-sidekiq --from-image=amp-system:2.9-oracle --containers=system-sidekiq,check-svc
```

これがトリガーとなり **system-sidekiq** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

4. **system-sphinx** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/system-sphinx --from-image=amp-system:latest --containers=system-sphinx,system-master-svc --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-sphinx --from-image=amp-system:2.9-oracle --containers=system-sphinx,system-master-svc
```

これがトリガーとなり **system-sphinx** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

5. 3scale をスケールダウンした場合は、元に戻します。

2.2.4.1.2. システムイメージのパッチ適用: 他のデータベースを使用する 3scale

1. **system-app** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/system-app --from-image=amp-system:latest --containers=system-master,system-developer,system-provider --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-app --from-image=amp-system:2.9 --containers=system-master,system-developer,system-provider
```

これがトリガーとなり **system-app** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

2. **system-sidekiq** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/system-sidekiq --from-image=amp-system:latest --containers=system-sidekiq,check-svc --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-sidekiq --from-image=amp-system:2.9 --containers=system-sidekiq,check-svc
```

これがトリガーとなり **system-sidekiq** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

3. **system-sphinx** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

   ```
   oc set triggers dc/system-sphinx --from-image=amp-system:latest --containers=system-sphinx,system-master-svc --remove
   ```

   b. 新しいバージョン固有のトリガーを追加します。

   ```
   oc set triggers dc/system-sphinx --from-image=amp-system:2.9 --containers=system-sphinx,system-master-svc
   ```

   これがトリガーとなり **system-sphinx** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

2.2.4.2. apicast イメージのパッチ

1. **amp-apicast** イメージストリームにパッチを適用します。

```
oc patch imagestream/amp-apicast --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "AMP APIcast 2.9"}, "from": { "kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.9"}, "name": "2.9", "referencePolicy": {"type": "Source"}}}]'
```

2. **apicast-staging** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

   ```
   oc set triggers dc/apicast-staging --from-image=amp-apicast:latest --containers=apicast-staging --remove
   ```

   b. 新しいバージョン固有のトリガーを追加します。

   ```
   oc set triggers dc/apicast-staging --from-image=amp-apicast:2.9 --containers=apicast-staging
   ```

   これがトリガーとなり **apicast-staging** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

3. **apicast-production** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

   ```
   oc set triggers dc/apicast-production --from-image=amp-apicast:latest --containers=apicast-production,system-master-svc --remove
   ```

   b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/apicast-production --from-image=amp-apicast:2.9 --
containers=apicast-production,system-master-svc
```

これがトリガーとなり **apicast-production** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d amp-apicast:latest
```

2.2.4.3. backend イメージのパッチ

1. **amp-backend** イメージストリームにパッチを適用します。

```
oc patch imagestream/amp-backend --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "AMP Backend 2.9"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/3scale-amp2/backend-rhel7:3scale2.9"}, "name":
"2.9", "referencePolicy": {"type": "Source"}}}]'
```

2. **backend-listener** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/backend-listener --from-image=amp-backend:latest --
containers=backend-listener --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/backend-listener --from-image=amp-backend:2.9 --
containers=backend-listener
```

これがトリガーとなり **backend-listener** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

3. **backend-worker** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/backend-worker --from-image=amp-backend:latest --
containers=backend-worker,backend-redis-svc --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/backend-worker --from-image=amp-backend:2.9 --
containers=backend-worker,backend-redis-svc
```

これがトリガーとなり **backend-worker** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d amp-backend:latest
```

4. **backend-cron** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

   ```
   oc set triggers dc/backend-cron --from-image=amp-backend:latest --
   containers=backend-cron,backend-redis-svc --remove
   ```

   b. 新しいバージョン固有のトリガーを追加します。

   ```
   oc set triggers dc/backend-cron --from-image=amp-backend:2.9 --containers=backend-
   cron,backend-redis-svc
   ```

   これがトリガーとなり **backend-cron** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

2.2.4.4. zync イメージへのパッチ適用

1. **amp-zync** イメージストリームにパッチを適用します。

```
oc patch imagestream/amp-zync --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value":
{"annotations": {"openshift.io/display-name": "AMP Zync 2.9"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/3scale-amp2/zync-rhel7:3scale2.9"}, "name":
"2.9", "referencePolicy": {"type": "Source"}}}]'
```

2. **zync** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

   ```
   oc set triggers dc/zync --from-image=amp-zync:latest --containers=zync,zync-db-svc --
   remove
   ```

   b. 新しいバージョン固有のトリガーを追加します。

   ```
   oc set triggers dc/zync --from-image=amp-zync:2.9 --containers=zync,zync-db-svc
   ```

   これがトリガーとなり **zync** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

3. **zync-que** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

   ```
   oc set triggers dc/zync-que --from-image=amp-zync:latest --containers=que --remove
   ```

   b. 新しいバージョン固有のトリガーを追加します。

   ```
   oc set triggers dc/zync-que --from-image=amp-zync:2.9 --containers=que
   ```

   これがトリガーとなり **zync-que** が再デプロイされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

   c. **ImageStream** から **:latest** タグを削除します。

   ```
   oc tag -d amp-zync:latest
   ```

2.2.4.5. system-memcached イメージのパッチ

1. **system-memcached** ImageStream にパッチを適用します。

```
oc patch imagestream/system-memcached --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "System 2.9 Memcached"}, "from": { "kind": "DockerImage", "name": "registry.redhat.io/3scale-amp2/memcached-rhel7:3scale2.9"}, "name": "2.9", "referencePolicy": {"type": "Source"}}}]'
```

2. **system-memcache** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

```
oc set triggers dc/system-memcache --from-image=system-memcached:latest --containers=memcache --remove
```

   b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-memcache --from-image=system-memcached:2.9 --containers=memcache
```

   これにより、**system-memcache** DeploymentConfig の再デプロイメントがトリガーされます。再デプロイが完了し、対応する新規 Pod が使用できる状態になり、古い Pod が終了するまで待ちます。

   c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d system-memcached:latest
```

2.2.4.6. zync-database-postgresql イメージのパッチ

1. **zync-database-postgresql** イメージストリームにパッチを適用します。

```
oc patch imagestream/zync-database-postgresql --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "Zync 2.9 PostgreSQL"}, "from": { "kind": "DockerImage", "name": "registry.redhat.io/rhscl/postgresql-10-rhel7"}, "name": "2.9", "referencePolicy": {"type": "Source"}}}]'
```

   - このパッチコマンドにより **zync-database-postgresql** イメージストリームが更新され、**2.9** タグが含まれるようになります。以下の手順により、**2.9** タグが作成されていることを確認することができます。

     a. 以下のコマンドを実行します。

```
oc get is zync-database-postgresql
```

     b. **Tags** 欄に **2.9** タグが表示されていることを確認します。

2. **zync-database** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

```
oc set triggers dc/zync-database --from-image=zync-database-postgresql:latest --containers=postgresql --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/zync-database --from-image=zync-database-postgresql:2.9 --
containers=postgresql
```

イメージに新しい更新があれば、このパッチがトリガーとなり **zync-database** DeploymentConfig も再デプロイされます。その場合は、新規 Pod の再デプロイが完了して使用できる状態になり、古い Pod が終了するまで待ちます。

c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d zync-database-postgresql:latest
```

2.2.4.7. その他のイメージの変更

3scale 2.8 のインストール環境で以下の DeploymentConfig の 1 つまたは複数が利用可能な場合は、該当するリンクをクリックして詳細な操作手順を確認してください。

- 「**backend-redis** DeploymentConfig」
- 「**system-redis** DeploymentConfig」
- 「**system-mysql** DeploymentConfig」
- 「**system-postgresql** DeploymentConfig」

2.2.4.7.1. backend-redis DeploymentConfig

現在の 3scale インストール環境に **backend-redis** DeploymentConfig が存在する場合は、**backend-redis** 用の **redis** イメージにパッチを適用します。

1. **backend-redis** イメージストリームにパッチを適用します。

```
oc patch imagestream/backend-redis --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "Backend 2.9 Redis"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/rhscl/redis-32-rhel7:3.2"}, "name": "2.9",
"referencePolicy": {"type": "Source"}}}]'
```

このパッチにより backend-redis イメージストリームが更新され、2.9 タグが含まれるようになります。以下のコマンドにより **Tags** 欄に **2.9** が表示されれば、タグが作成されていることを確認することができます。

```
oc get is backend-redis
```

2. **backend-redis** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/backend-redis --from-image=backend-redis:latest --
containers=backend-redis --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/backend-redis --from-image=backend-redis:2.9 --containers=backend-redis
```

イメージに新しい更新があれば、このパッチがトリガーとなり **backend-redis** DeploymentConfig も再デプロイされます。その場合は、新規 Pod の再デプロイが完了して使用できる状態になり、古い Pod が終了するまで待ちます。

c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d backend-redis:latest
```

2.2.4.7.2. system-redis DeploymentConfig

現在の 3scale インストール環境に **system-redis** DeploymentConfig が存在する場合は、 **system-redis** 用の **redis** イメージにパッチを適用します。

1. **system-redis** イメージストリームにパッチを適用します。

```
oc patch imagestream/system-redis --type=json -p '[{"op": "add", "path": "/spec/tags/-", "value": {"annotations": {"openshift.io/display-name": "System 2.9 Redis"}, "from": { "kind": "DockerImage", "name": "registry.redhat.io/rhscl/redis-32-rhel7:3.2"}, "name": "2.9", "referencePolicy": {"type": "Source"}}}]'
```

このパッチにより **system-redis** イメージストリームが更新され、2.9 タグが含まれるようになります。以下のコマンドにより **Tags** 欄に **2.9** が表示されれば、タグが作成されていることを確認することができます。

```
oc get is system-redis
```

2. **system-redis** ImageChange トリガーにパッチを適用します。

a. 最新のトリガーを削除します。

```
oc set triggers dc/system-redis --from-image=system-redis:latest --containers=system-redis --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-redis --from-image=system-redis:2.9 --containers=system-redis
```

イメージに新しい更新があれば、このパッチがトリガーとなり **system-redis** DeploymentConfig も再デプロイされます。その場合は、新規 Pod の再デプロイが完了して使用できる状態になり、古い Pod が終了するまで待ちます。

c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d system-redis:latest
```

2.2.4.7.3. system-mysql DeploymentConfig

現在の 3scale インストール環境に **system-mysql** DeploymentConfig が存在する場合は、 **system-mysql** 用の MySQL イメージにパッチを適用します。

1. **system-mysql** イメージストリームにパッチを適用します。

```
oc patch imagestream/system-mysql --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "System 2.9 MySQL"}, "from": { "kind":
"DockerImage", "name": "registry.redhat.io/rhscl/mysql-57-rhel7:5.7"}, "name": "2.9",
"referencePolicy": {"type": "Source"}}}]'
```

このパッチにより **system-mysql** イメージストリームが更新され、2.9 タグが含まれるようになります。以下のコマンドにより **Tags** 欄に 2.9 が表示されれば、タグが作成されていることを確認することができます。

```
oc get is system-mysql
```

2. **system-mysql** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

```
oc set triggers dc/system-mysql --from-image=system-mysql:latest --containers=system-
mysql --remove
```

   b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-mysql --from-image=system-mysql:2.9 --containers=system-
mysql
```

   イメージに新しい更新があれば、このパッチがトリガーとなり **system-mysql** DeploymentConfig も再デプロイされます。その場合は、新規 Pod の再デプロイが完了して使用できる状態になり、古い Pod が終了するまで待ちます。

   c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d system-mysql:latest
```

2.2.4.7.4. system-postgresql DeploymentConfig

現在の 3scale インストール環境に **system-postgresql** DeploymentConfig が存在する場合は、**system-postgresql** 用の PostgreSQL イメージにパッチを適用します。

1. **system-postgresql** イメージストリームにパッチを適用します。

```
oc patch imagestream/system-postgresql --type=json -p '[{"op": "add", "path": "/spec/tags/-",
"value": {"annotations": {"openshift.io/display-name": "System 2.9 PostgreSQL"}, "from": {
"kind": "DockerImage", "name": "registry.redhat.io/rhscl/postgresql-10-rhel7"}, "name": "2.9",
"referencePolicy": {"type": "Source"}}}]'
```

このパッチにより **system-postgresql** イメージストリームが更新され、2.9 タグが含まれるようになります。以下のコマンドにより **Tags** 欄に 2.9 が表示されれば、タグが作成されていることを確認することができます。

```
oc get is system-postgresql
```

2. **system-postgresql** ImageChange トリガーにパッチを適用します。

   a. 最新のトリガーを削除します。

```
oc set triggers dc/system-postgresql --from-image=system-postgresql:latest --
containers=system-postgresql --remove
```

b. 新しいバージョン固有のトリガーを追加します。

```
oc set triggers dc/system-postgresql --from-image=system-postgresql:2.9 --
containers=system-postgresql
```

イメージに新しい更新があれば、このパッチがトリガーとなり **system-postgresql** DeploymentConfig も再デプロイされます。その場合は、新規 Pod の再デプロイが完了して使用できる状態になり、古い Pod が終了するまで待ちます。

c. **ImageStream** から **:latest** タグを削除します。

```
oc tag -d system-postgresql:latest
```

2.2.4.8. イメージの URL の確認

前の手順

DeploymentConfig のすべてのイメージ URL に、各 URL アドレスの最後に追加されたハッシュと共に新しいイメージレジストリーの URL が含まれていることを確認します。

```
THREESCALE_DC_NAMES="apicast-production apicast-staging backend-cron backend-listener
backend-redis backend-worker system-app system-memcache system-mysql system-redis system-
sidekiq system-sphinx zync zync-database zync-que"
for component in ${THREESCALE_DC_NAMES}; do echo -n "${component} image: " && oc get dc
$component -o json | jq .spec.template.spec.containers[0].image ; done
```

次のステップ

なし。上記の手順をすべて実施すると、テンプレートベースのデプロイメントにおける 3scale 2.8 から 2.9.1 へのアップグレードが完了します。

2.3. テンプレートベースのインストール環境での ORACLE DATABASE を使用した 3SCALE のアップグレード

本セクションでは、OpenShift 3.11 とテンプレートベースのインストール環境の組み合わせにおいて、Oracle Database で 3scale システムイメージを使用している場合に、Red Hat 3scale API Management をアップグレードする方法について説明します。

前提条件

Oracle Database を使用した 3scale インストール環境。Oracle Database を使用した 3scale システムイメージの設定 を参照してください。

テンプレートベースのインストール環境で Oracle Database を使用して 3scale のシステムイメージをアップグレードするには、データベースバージョンに応じて以下の手順のいずれかを実行します。

- 「Oracle 19c を使用した 3scale のアップグレード」
- 「Oracle 12c を使用した 3scale のアップグレード」

2.3.1. Oracle 19c を使用した 3scale のアップグレード

以下の手順では、Oracle Database 12c から Oracle Database 19c へ、既存の 3scale 2.8 インストール環境から 3scale 2.9.1 への変更について説明します。

重要:データベースへの接続が失われると、3scale が破損する可能性があります。アップグレードを進める前にバックアップを作成します。Oracle データベースの公式ドキュメントを参照してください。Oracle Database Backup and Recovery User's Guide を参照してください。

前提条件

- 3scale 2.8 のインストール
- Oracle Database 12c のインストール
  - Oracle を使用した 3scale の設定に関する詳細は、Oracle Database の準備 を参照してください。

手順

1. 以下の考慮事項に注意して、3scale を 0 (ゼロ) にスケールダウンします。
   a. Oracle 12c から切断されているため、3scale のデータベースへの接続に失敗します。
2. Oracle 12c を Oracle 19c にアップグレードします。
   a. アップグレードを実行するには、公式の Oracle ドキュメント Database Upgrade Guide に従ってください。
3. 3scale 2.9.1 用の OpenShift テンプレート のクローンを作成します。

```
$ git clone --branch 2.9.1.GA https://github.com/3scale/3scale-amp-openshift-templates.git
```

4. Oracle Database の Instant Client パッケージファイルを **3scale-amp-openshift-templates/amp/system-oracle/oracle-client-files** ディレクトリーに置きます。
5. **-f** オプションで **build.yml** OpenShift テンプレートを指定して、**oc process** コマンドを実行します。

```
$ oc process -f build.yml | oc apply -f -
```

6. **oc start-build** コマンドを入力し、新しいシステムイメージをビルドします。

```
$ oc start-build 3scale-amp-system-oracle --from-dir=.
```

7. ビルドが完了するまで待ちます。ビルドの状態を確認するには、以下のコマンドを実行します。

```
$ oc get build <build-name> -o jsonpath="{.status.phase}"
```

   a. ビルドが Complete の状態になるまで待ちます。

2.3.2. Oracle 12c を使用した 3scale のアップグレード

以下の手順では、Oracle Database 12c の更新、既存の 3scale 2.8 インストール環境から 3scale 2.9.1 への変更について説明します。

重要:データベースへの接続が失われると、3scale が破損する可能性があります。アップグレードを進める前にバックアップを作成します。Oracle データベースの公式ドキュメントを参照してください。Oracle Database Backup and Recovery User's Guide を参照してください。

前提条件

- 3scale 2.8 のインストール
- Oracle Database 12c のインストール
  - Oracle を使用した 3scale の設定に関する詳細は、Oracle Database の準備 を参照してください。

手順

1. 3scale 2.9.1 用の OpenShift テンプレート のクローンを作成します。

```
$ git clone --branch 2.9.1.GA https://github.com/3scale/3scale-amp-openshift-templates.git
```

2. Oracle Database の Instant Client パッケージファイルを **3scale-amp-openshift-templates/amp/system-oracle/oracle-client-files** ディレクトリーに置きます。
3. **-f** オプションで **build.yml** OpenShift テンプレートを指定して、**oc process** コマンドを実行します。

```
$ oc process -f build.yml | oc apply -f -
```

4. **oc start-build** コマンドを入力し、新しいシステムイメージをビルドします。

```
$ oc start-build 3scale-amp-system-oracle --from-dir=.
```

5. ビルドが完了するまで待ちます。ビルドの状態を確認するには、以下のコマンドを実行します。

```
$ oc get build <build-name> -o jsonpath="{.status.phase}"
```

   a. ビルドが Complete の状態になるまで待ちます。

関連情報

3scale と Oracle Database のサポートについては、Red Hat 3scale API Management Supported Configurations を参照してください。

第3章 OPERATOR ベースの 3SCALE のアップグレードガイド: 2.8 から 2.9 へ

本セクションでは、operator ベースのデプロイメントにおいて、Red Hat 3scale API Management をバージョン 2.8 から 2.9 にアップグレードする方法について説明します。

3scale のマイクロリリースを自動的に取得するには、自動更新が有効であることを確認してください。これを確認するには、Setting up the 3scale operator for micro releases を参照してください。

重要

必要な条件および手順を理解するために、記載の手順を適用する前に、アップグレードガイド全体を読んでください。アップグレードプロセスの手順が完了するまで、サービスの提供が中断されます。このサービス中断が生じるため、メンテナンス期間を設けるようにしてください。

3.1. アップグレードを行うための前提条件

本セクションでは、operator ベースのインストール環境において、3scale を 2.8 から 2.9 にアップグレードするのに必要な設定について説明します。

- 3scale operator によりデプロイされている 3scale 2.8
- OpenShift Container Platform (OCP) 4.x クラスターおよびその管理者アクセス

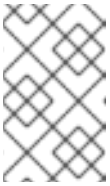
3.2. OPERATOR ベースのインストール環境における 2.8 から 2.9 へのアップグレード

operator ベースのデプロイメントにおいて、3scale をバージョン 2.8 から 2.9 にアップグレードするには、以下の手順を実施します。

1. 管理者権限を持つアカウントを使用して OCP コンソールにログインします。
2. **3scale-operator** がデプロイされているプロジェクトを選択します。
3. **Operators > Installed Operators**の順にクリックします。
4. **Red Hat Integration - 3scale > Subscription > Channel**の順に選択します。
5. **threescale-2.9** を選択してサブスクリプションのチャンネルを編集し、変更を保存します。
   - これによりアップグレードプロセスが開始されます。
   - **APIManager** のアップグレードプロセスが終了するまで待ちます。
6. プロジェクトの Pod ステータスのクエリーを行います。

```
oc get pods
```

   - すべての新しいバージョンが動作して使用できる状態になり、エラーが無くなるまで待ちます。
   - アップグレードプロセス中、一時的にエラーが発生する場合があります。



注記

所要時間はおよそ 5 - 10 分間の範囲で幅があります。すべての Pod が動作して使用できる状態になり、エラーが無くなるまで、Pod の状態確認を続けてください。

3scale 管理ポータルにログインして期待通りに動作することを確認し、アップグレードプロセスが正常に完了したことを確認します。

1. **APIManager** オブジェクトのステータスを確認し、以下のコマンドを実行して **YAML** のコンテンツを取得します。

```
oc get apimanager <myapimanager> -o yaml
```

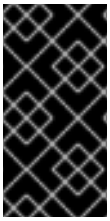
- 新しいアノテーションおよび値は以下のようになります。

```
apps.3scale.net/apimanager-threescale-version: "2.9"
apps.3scale.net/threescale-operator-version: "0.6.0"
```

上記の手順をすべて実施すると、operator ベースのデプロイメントにおける 3scale 2.8 から 2.9 へのアップグレードが完了します。

第4章 OPERATOR ベースの APICAST のアップグレードガイド: 2.8 から 2.9 へ

本セクションでは、operator ベースのデプロイメントにおいて、APIcast をバージョン 2.8 から 2.9 にアップグレードする方法について説明します。



重要

必要な条件および手順を理解するために、記載の手順を適用する前に、アップグレードガイド全体を読んでください。アップグレードプロセスの手順が完了するまで、サービスの提供が中断されます。このサービス中断が生じるため、メンテナンス期間を設けるようにしてください。

4.1. アップグレードを行うための前提条件

本セクションでは、operator ベースのインストール環境において、APIcast を 2.8 から 2.9 にアップグレードするのに必要な設定について説明します。

- APIcast operator によりデプロイされている APIcast 2.8
- OpenShift Container Platform (OCP) 4.x クラスターおよびその管理者アクセス

4.2. OPERATOR ベースのインストール環境における APICAST 2.8 から 2.9 へのアップグレード

operator ベースのデプロイメントにおいて、APIcast をバージョン 2.8 から 2.9 にアップグレードするには、以下の手順を実施します。

1. 管理者権限を持つアカウントを使用して OCP コンソールにログインします。
2. **APIcast operator** がデプロイされているプロジェクトを選択します。
3. **Operators > Installed Operators**の順にクリックします。
4. **Subscription > Channel**の順に移動し、**Red Hat Integration - 3scale APIcast gateway**を選択します。
5. **threescale-2.9** チャンネルを選択してサブスクリプションのチャンネルを編集し、変更を保存します。
   - これによりアップグレードプロセスが開始されます。
   - **APIcast** のアップグレードプロセスが終了するまで待ちます。
6. プロジェクトの Pod ステータスのクエリーを行います。

```
oc get pods
```

   - すべての新しいバージョンが動作して使用できる状態になり、エラーが無くなるまで待ちます。
   - アップグレードプロセス中、一時的にエラーが発生する場合があります。

注記

所要時間はおよそ 5 - 10 分間の範囲で幅があります。すべての Pod が動作して使用できる状態になり、エラーが無くなるまで、Pod の状態確認を続けてください。

7. **APIcast** オブジェクトのステータスを確認し、以下のコマンドを実行して **YAML** のコンテンツを取得します。

```
oc get apicast <myapicast> -o yaml
```

- 新しいアノテーションおよび値は以下のようになります。

```
apicast.apps.3scale.net/operator-version: “0.3.0”
```

上記の手順をすべて実施すると、operator ベースのデプロイメントにおける APIcast 2.8 から 2.9 へのアップグレードが完了します。