Red Hat 3scale API Management 2.6

3scale のインストール

3scale API Management のインストールおよび設定

Last Updated: 2023-02-19

Red Hat 3scale API Management 2.6 3scale のインストール

3scale API Management のインストールおよび設定

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

本ガイドは、3scale API Management のインストールおよび設定に関する情報を提供します。

目次

はじめに

本ガイドは、3scale のインストールおよび設定に役立ちます。

第1章 3SCALE システムイメージと ORACLE DATABASE

デフォルトでは、Red Hat 3scale API Management 2.6 には **システム** と呼ばれるコンポーネントがあり、設定データを MySQL データベースに保存します。任意で、デフォルトのデータベースをオーバーライドし、情報を外部の Oracle Database に保管することができます。本章の手順に従って、独自の Oracle Database クライアントバイナリーでカスタムのシステムコンテナーイメージをビルドし、3scale を OpenShift にデプロイします。

1.1. 始める前に

1.1.1. Oracle ソフトウェアコンポーネントの取得

カスタムの 3scale システムコンテナーイメージをビルドする前に、以下の Oracle ソフトウェアコンポーネントの サポートされるバージョン を取得する必要があります。

- Oracle Instant Client Package Basic または Basic Light
- Oracle インスタントクライアントパッケージ SDK
- Oracle インスタントクライアントパッケージ ODBC

1.1.2. 前提条件

- OpenShift クラスターからアクセスできる、サポートされるバージョン の Oracle Database
- インストール手順に必要な Oracle Database の **system** ユーザーへのアクセス
- 3scale 2.6 **amp.yml** テンプレート

1.2. ORACLE DATABASE の準備

1. 新しいデータベースを作成します。
   Oracle Database が 3scale と動作するようにするには、以下の設定が必要です。

```
ALTER SYSTEM SET max_string_size=extended SCOPE=SPFILE;
ALTER SYSTEM SET compatible='12.2.0.1' SCOPE=SPFILE;
```

2. データベースの詳細を収集します。
   3scale の設定に必要な以下の情報を取得します。
   - Oracle Database の URL
   - Oracle Database の サービス名
   - Oracle Database の **システム** のパスワード
     DATABASE_URL パラメーターは次の形式に従う必要があります。

```
oracle-enhanced://USER:PASSWORD@HOST:PORT/DATABASE
```

例

```
DATABASE_URL="oracle-enhanced://USER:PASSWORD@my-oracle-
database.com:1521/threescalepdb"
```

-

関連情報

Oracle Database での新規データベース作成については、Oracle のドキュメント を参照してください。

1.3. システムイメージのビルド

注記

システムの Oracle **build.yml** ファイルをダウンロードした後、手動で **AMP_RELEASE** 値を **2.5.0** から **2.6.0** に変更する必要があります。

本セクションでは、システムイメージをビルドする手順について説明します。

前提条件

- Oracle Database が設定されている必要があります。詳細は、「Oracle Database の準備」 の手順に従ってください。

手順

1. 3scale OpenShift テンプレート GitHub リポジトリーのクローンを作成します。以下のコマンドを使用します。

```
$ git clone --branch 2.6.0.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. 3scale 2.6 **amp.yml** テンプレートをダウンロードします。

4. **-f** オプションで **build.yml** OpenShift テンプレートを指定して、**oc new-app** コマンドを実行します。

```
$ oc new-app -f build.yml
```

5. **-f** オプションで **amp.yml** OpenShift テンプレートを指定し、**-p** オプションで **WILDCARD_DOMAIN** パラメーターに OpenShift クラスターのドメインを指定して、**oc new-app** コマンドを実行します。

```
$ oc new-app -f amp.yml -p WILDCARD_DOMAIN=mydomain.com
```

6. 次の **oc patch** コマンドを入力し、**SYSTEM_PASSWORD** を 「Oracle Database の準備」 で設定した Oracle データベースの **システム** パスワードに置き換えます。

```
$ oc patch dc/system-app -p '[{"op": "add", "path": "/spec/strategy/rollingParams/pre/execNewPod/env/-", "value": {"name": "ORACLE_SYSTEM_PASSWORD", "value": "SYSTEM_PASSWORD"}}]' --type=json

$ oc patch dc/system-app -p '{"spec": {"strategy": {"rollingParams": {"post":{"execNewPod": {"env": [{"name": "ORACLE_SYSTEM_PASSWORD", "value": "SYSTEM_PASSWORD"}]}}}}}}'
```

7. 次のコマンドを入力します。ただし、。**DATABASE_URL** は、「Oracle Database の準備」 で指定した Oracle データベースに置き換えます。

```
$ oc patch secret/system-database -p '{"stringData": {"URL": "DATABASE_URL"}}'
```

8. 以下のコマンドを実行してプルシークレットをビルダーにリンクします。

```
$ oc secrets link builder threescale-registry-auth
```

9. **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=.
```

関連情報

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

第2章 3SCALE を OPENSHIFT にインストールするためのガイド

2.1. はじめに

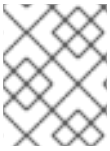
本ガイドは、OpenShift にオンプレミス型 Red Hat 3scale API Management 2.6 をデプロイする手順を説明します。

オンプレミスデプロイメントの 3scale ソリューションは、以下の要素で設定されています。

- 2 つの API ゲートウェイ: Embedded APIcast
- 1 つの 3scale 管理ポータルおよび永続ストレージを持つデベロッパーポータル

3scale ソリューションをデプロイする方法は 2 つあります。

- 「テンプレートを使用した OpenShift への 3scale のデプロイ」
- 「operator を使用した 3scale のデプロイ」



注記

3scale のデプロイに operator とテンプレートのどちらを使用するかにかかわらず、まず「OpenShift でのレジストリー認証の設定」 に記載の手順を実施する必要があります。

2.1.1. 前提条件

- 3scale サーバーを UTC (協定世界時) に設定する必要があります。

2.2. システム要件

本セクションでは、3scale - OpenShift テンプレートの要件を示します。

2.2.1. 環境要件

3scale には、Red Hat 3scale API Management Supported Configurations で規定される環境が必要です。

永続ボリューム

- Redis および MySQL の永続用の 3 つの RWO (ReadWriteOnce) 永続ボリューム
- CMS および System-app Assets 用の 1 つの RWX (ReadWriteMany) 永続ボリューム

RWX 永続ボリュームはグループ書き込みができるように設定する必要があります。必要なアクセスモードをサポートする永続ボリュームタイプのリストは、OpenShift のドキュメント を参照してください。

2.2.2. ハードウェア要件

ハードウェア要件は、用途のニーズによって異なります。Red Hat は、テストを行い個々の要件を満たすように環境を設定することを推奨します。OpenShift 上の 3scale の環境を設定する場合、以下が推奨されます。

- クラウド環境へのデプロイメントには、コンピュートタスクに最適化したノードを使用します (AWS c4.2xlarge または Azure Standard_F8)。
- メモリーの要件が現在のノードで使用できる RAM よりも大きい場合、非常に大きなインストールでは、Redis に別のノードが必要になることがある (AWS M4 シリーズまたは Azure Av2 シリーズ)。
- ルーティングタスクとコンピュートタスクには別のノードを使用する。
- 3scale 固有のタスクには専用のコンピュートノードを使用する。
- バックエンドリスナーの **PUMA_WORKERS** 変数をコンピュートノードのコア数に設定する。

2.3. ノードおよびエンタイトルメントの設定

3scale を OpenShift にデプロイする前に、ノードおよび環境が Red Hat Container Registry からイメージを取得するのに必要なエンタイトルメントを設定する必要があります。

以下の手順を実行し、エンタイトルメントを設定します。

1. 各ノードに Red Hat Enterprise Linux (RHEL) をインストールします 。
2. インターフェイス または コマンドライン で Red Hat Subscription Manager (RHSM) を使用し、Red Hat にノードを登録します。
3. RHSM を使用して ノードを 3scale サブスクリプションに割り当てます。
4. 以下の要件に準拠して、ノードに OpenShift をインストールします。
   - サポート対象バージョンの OpenShift を使用する。
   - 複数書き込みをサポートするファイルシステムで 永続ストレージ を設定する。
5. OpenShift コマンドラインインターフェイス をインストールします。
6. Subscription Manager を使用して、**rhel-7-server-3scale-amp-2.6-rpms** リポジトリーへのアクセスを有効にします。

```
sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2.6-rpms
```

7. 3scale テンプレート **3scale-amp-template** をインストールします。このテンプレートは **/opt/amp/templates** に保存されます。

```
sudo yum install 3scale-amp-template
```

2.4. テンプレートを使用した OPENSHIFT への 3SCALE のデプロイ

本セクションでは、テンプレートを使用して OpenShift に 3scale をデプロイする方法について説明します。

2.4.1. 前提条件

- 「ノードおよびエンタイトルメントの設定」 セクションで指定されたとおりに設定された OpenShift クラスター

- OpenShift クラスターに対して解決する ドメイン
  - **注記:** OpenShift Container Platform (OCP) 3.11 は、テンプレートを使用した 3scale のデプロイメントのみをサポートしています。
- Red Hat コンテナーカタログ へのアクセス
- (オプション) 稼働中の電子メール機能用 SMTP サーバー

以下の手順に従い、**.yml** テンプレートを使用して 3scale を OpenShift にインストールします。

- 「OpenShift でのレジストリー認証の設定」
- 「レジストリーサービスアカウントの作成」
- 「3scale テンプレートのインポート」
- 「SMTP 変数の設定 (任意)」

2.4.2. OpenShift でのレジストリー認証の設定

Red Hat 3scale API Management OpenShift イメージストリームを使用するには、Red Hat コンテナーレジストリーへのレジストリー認証を設定する必要があります。以下の手順に従って、コンテナーレジストリーへの登録を設定してください。

1. 以下のように、管理者として OpenShift サーバーにログインします。

```
oc login -u system:admin
```

2. イメージストリームをインストールする OpenShift プロジェクトにログインします。Red Hat では、3scale OpenShift イメージストリーム用に **openshift** プロジェクトを使用することを推奨します。
注記: プロジェクトには、ランダムな文字列からなる固有の接頭辞が付きます。

```
oc project your-openshift-project
```

3. 「レジストリーサービスアカウントの作成」 で作成するクレデンシャルを使用して、**docker-registry** シークレットを作成します。

注記

- **your-registry-service-account-username** を 12345678|username のフォーマットで作成したユーザー名に置き換えてください。
- **your-registry-service-account-password** を Token Information タブでユーザー名の下に表示されるパスワードの文字列に置き換えてください。
- イメージストリームが存在し **registry.redhat.io** を使用するすべての新規 **namespace** について、**docker-registry** シークレットを作成します。

```
$ oc create secret docker-registry threescale-registry-auth \
  --docker-server=registry.redhat.io \
  --docker-username="your-registry-service-account-username" \
  --docker-password="your-registry-service-account-password"
```

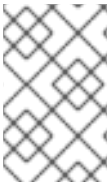
2.4.3. レジストリーサービスアカウントの作成

OpenShift 上にデプロイされた 3scale 2.6 と共に共有環境で **registry.redhat.io** からのコンテナーイメージを使用するには、個々のユーザーの **カスタマーポータル** のクレデンシャルではなく、**レジストリーサービスアカウント** を使用する必要があります。

注記

テンプレートまたは operator のどちらを使用して 3scale 2.6 を OpenShift にデプロイする場合でも、その前に以下に概略を示す手順に従う必要があります。両オプションともレジストリーの認証を使用するためです。

1. Registry Service Accounts のページに移動し、ログインします。
2. **New Service Account** をクリックします。
3. **Create a New Registry Service Account**のページに表示されるフォームに入力します。
    a. **サービスアカウント** の名前を追加します。
    注記: フォームのフィールドの前に、決められた桁数のランダムに生成された数字の文字列が表示されます。
4. **Description** を入力します。
5. **Create** をクリックします。
6. **Registry Service Accounts**のページに戻ります。
7. 作成した **サービスアカウント** をクリックします。
8. 接頭辞の文字列を含めたユーザー名 (例: **12345678|username**) およびパスワードを書き留めます。
    a. このユーザー名およびパスワードは、**registry.redhat.io** へのログインに使用されます。



注記

Token Information のページには、認証トークンの使用方法を説明したタブがあります。たとえば、**Token Information** タブには、**12345678|username** フォーマットのユーザー名およびその下にパスワードの文字列が表示されます。

2.4.4. レジストリーサービスアカウントの変更

サービスアカウントを変更または削除することができます。**Registry Service Account**ページの表中の各認証トークン右側のポップアップメニューを使用して、その操作を行うことができます。

警告

トークンを再生成したり **サービスアカウント** を削除したりすると、そのトークンを用いて認証および **registry.redhat.io** からコンテンツを取得しているシステムに影響を及ぼします。

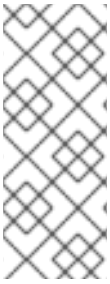
各機能の説明は以下のとおりです。

- **Regenerate Token:** 許可されたユーザーは、**サービスアカウント** に関連付けられたパスワードをリセットすることができます。
  注記: サービスアカウント のユーザー名を変更することはできません。
- **Update Description:** 許可されたユーザーは、**サービスアカウント** の説明を更新することができます。
- **Delete Account:** 許可されたユーザーは、**サービスアカウント** を削除することができます。

詳細は、以下のドキュメントを参照してください。

- Red Hat コンテナーレジストリーの認証
- Authentication enabled Red Hat registry

2.4.5. 3scale テンプレートのインポート



注記

- 3scale 2.6 では、ワイルドカードルートは 廃止されています。
  - この機能は、バックグラウンドで Zync により処理されます。
- API プロバイダーが作成、更新、または削除されると、これらの変更が自動的にルートに反映されます。

3scale テンプレートを OpenShift クラスターにインポートするには、以下の手順を実施します。

1. ターミナルセッションから、OpenShift にクラスター管理者としてログインします。

```
oc login
```

2. プロジェクトを選択するか新しいプロジェクトを作成します。

```
oc project <project_name>
```

```
oc new-project <project_name>
```

3. **oc new-app** コマンドを入力します。
   a. **--file** オプションを使用して、「ノードおよびエンタイトルメントの設定」 でダウンロードした **amp.yml** ファイルへのパスを指定します。
   b. **--param** オプションを使用して、**WILDCARD_DOMAIN** パラメーターに OpenShift クラスターのドメインを設定します。

```
oc new-app --file /opt/amp/templates/amp.yml --param WILDCARD_DOMAIN=
<WILDCARD_DOMAIN>
```

ターミナルには、マスターおよびテナント URL と、新たに作成された 3scale 管理ポータルのクレデンシャルが表示されます。この出力には以下の情報が含まれます。

- マスター管理者のユーザー名
- マスターのパスワード
- マスターのトークン情報
- テナントのユーザー名
- テナントのパスワード
- テナントのトークン情報

4. https://user-admin.3scale-project.example.com に admin/xXxXyz123 としてログインします。

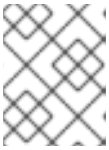
```
* With parameters:

 * ADMIN_PASSWORD=xXxXyz123 # generated
 * ADMIN_USERNAME=admin
 * TENANT_NAME=user

 * MASTER_NAME=master
 * MASTER_USER=master
 * MASTER_PASSWORD=xXxXyz123 # generated

--> Success
Access your application via route 'user-admin.3scale-project.example.com'
Access your application via route 'master-admin.3scale-project.example.com'
Access your application via route 'backend-user.3scale-project.example.com'
Access your application via route 'user.3scale-project.example.com'
Access your application via route 'api-user-apicast-staging.3scale-project.example.com'
Access your application via route 'api-user-apicast-production.3scale-project.example.com'
```

5. 後で確認できるようにするため、詳細を書き留めておきます。



注記

3scale が OpenShift に完全にデプロイされ、ログインとクレデンシャルが有効になるまで待ちます。

2.4.6. 管理ポータルの URL の取得

テンプレートを使用して 3scale をデプロイすると、固定 URL (**3scale-admin.${wildcardDomain}**) のデフォルトテナントが作成されます。

3scale の Dashboard には、テナントの新しいポータル URL が表示されます。たとえば、**<wildCardDomain>** が **3scale-project.example.com** の場合、管理ポータル URL は **https://3scale-admin.3scale-project.example.com** となります。

wildcardDomain は、インストール中に指定した **<wildCardDomain>** パラメーターです。以下のコマンドを使用し、ブラウザーでこの一意の URL を開きます。

```
xdg-open https://3scale-admin.3scale-project.example.com
```

オプションとして、**マスターポータル URL** (**master.${wildcardDomain}**) に新しいテナントを作成できます。

2.4.7. SMTP 変数の設定 (任意)

OpenShift は 通知の送信 および 新規ユーザーの招待 に電子メールを使用します。これらの機能を使用する場合は、独自の SMTP サーバーを提供し、SMTP 設定マップで SMTP 変数を設定する必要があります。

SMTP ConfigMap で SMTP 変数を設定するには、以下の手順を実施します。

1. OpenShift にログインしていない場合はログインします。

```
oc login
```

   a. SMTP ConfigMap の変数を設定します。**oc patch** コマンドを使用して **configmap** および **smtp** オブジェクトを指定し、続いて **-p** オプションを指定し、以下の変数に対して JSON 形式で新しい値を指定します。

変数	説明
address	リモートメールサーバーをリレーとして指定できます。
username	メールサーバーのユーザー名を指定します。
password	メールサーバーのパスワードを指定します。
domain	HELO ドメインを指定します。
port	メールサーバーが新しい接続をリッスンするポートを指定します。
authentication	メールサーバーの認証タイプを指定します。指定できる値は **plain** (パスワードをクリアテキストで送信)、**login** (パスワードを Base64 エンコードで送信)、または **cram_md5** (ハッシュ関数に Message Digest 5 アルゴリズムを使用し認証情報を交換) です。
openssl.verify.mode	TLS の使用時に OpenSSL が証明書をチェックする方法を指定します。指定できる値は **none**、**peer**、**client_once**、または **fail_if_no_peer_cert** です。

例

```
oc patch configmap smtp -p '{"data":{"address":"<your_address>"}}'
oc patch configmap smtp -p '{"data":{"username":"<your_username>"}}'
oc patch configmap smtp -p '{"data":{"password":"<your_password>"}}'
```

2. configmap 変数を設定した後、**system-app** および **system-sidekiq** Pod を再デプロイします。

```
oc rollout latest dc/system-app
oc rollout latest dc/system-sidekiq
```

2.5. 3SCALE テンプレートパラメーター

テンプレートパラメーターにより、デプロイメント中およびデプロイメント後の 3scale **amp.yml** テンプレートの環境変数を設定します。

名前	説明	デフォルト値	必須/任意
APP_LABEL	オブジェクトアプリのラベルに使用されます。	**3scale-api-management**	必須
ZYNC_DATABASE_PASSWORD	PostgreSQL 接続ユーザーのパスワード。指定のない場合は無作為に生成されます。	該当なし	必須
ZYNC_SECRET_KEY_BASE	Zync の秘密鍵ベース。指定のない場合は無作為に生成されます。	該当なし	必須
ZYNC_AUTHENTICATION_TOKEN	Zync の認証トークン。指定のない場合は無作為に生成されます。	該当なし	必須
AMP_RELEASE	3scale リリースタグ	**2.6.0**	はい
ADMIN_PASSWORD	無作為に生成される 3scale 管理者アカウントのパスワード	該当なし	必須
ADMIN_USERNAME	3scale 管理者アカウントのユーザー名	**admin**	必須
APICAST_ACCESS_TOKEN	APIcast が設定のダウンロードに使用する読み取り専用アクセストークン	該当なし	必須
ADMIN_ACCESS_TOKEN	すべての API をスコープとし、書き込みアクセス権限が設定された管理者アクセストークン	該当なし	任意

名前	説明	デフォルト値	必須/任意
WILDCARD_DOMAIN	ワイルドカードルートのルートドメイン。たとえば、ルートドメイン **example.com** は **3scale-admin.example.com** を生成します。	該当なし	必須
TENANT_NAME	ルート下のテナント名。-admin 接尾辞を付けて管理ポータルにアクセスすることができます。	**3scale**	必須
MYSQL_USER	データベースのアクセスに使用される MySQL ユーザーのユーザー名	**mysql**	必須
MYSQL_PASSWORD	MySQL ユーザーのパスワード	該当なし	必須
MYSQL_DATABASE	アクセスされた MySQL データベースの名前	**system**	必須
MYSQL_ROOT_PASSWORD	Root ユーザーのパスワード	該当なし	必須
SYSTEM_BACKEND_USERNAME	内部 3scale api auth の内部 3scale API ユーザー名	**3scale_api_user**	必須
SYSTEM_BACKEND_PASSWORD	内部 3scale api auth の内部 3scale API パスワード	該当なし	必須
REDIS_IMAGE	使用する Redis イメージ	**registry.redhat.io/rhscl/redis-32-rhel7:3.2**	必須
MYSQL_IMAGE	使用する Mysql イメージ	**registry.redhat.io/rhscl/mysql-57-rhel7:5.7**	必須
MEMCACHED_IMAGE	使用する Memcached イメージ	**registry.redhat.io/3scale-amp20/memcached:1.4.15**	はい
POSTGRESQL_IMAGE	使用する Postgresql イメージ	**registry.redhat.io/rhscl/postgresql-10-rhel7**	必須

名前	説明	デフォルト値	必須/任意
AMP_SYSTEM_IMAGE	使用する 3scale システムイメージ	**registry.redhat.io/3scale-amp26/system**	はい
AMP_BACKEND_IMAGE	使用する 3scale バックエンドイメージ	**registry.redhat.io/3scale-amp26/backend**	はい
AMP_APICAST_IMAGE	使用する 3scale APIcast イメージ	**registry.redhat.io/3scale-amp26/apicast-gateway**	はい
AMP_ZYNC_IMAGE	使用する 3scale Zync イメージ	**registry.redhat.io/3scale-amp26/zync**	はい
SYSTEM_BACKEND_SHARED_SECRET	バックエンドからシステムにイベントをインポートするための共有シークレット	該当なし	必須
SYSTEM_APP_SECRET_KEY_BASE	システムアプリケーションの秘密鍵ベース	該当なし	必須
APICAST_MANAGEMENT_API	APIcast Management API のスコープ。disable、status、または debug を設定できます。ヘルスチェックには最低でも status が必要です。	**status**	任意
APICAST_OPENSSL_VERIFY	設定のダウンロード時に OpenSSL ピア検証を有効または無効にします。true または false を設定できます。	**false**	任意
APICAST_RESPONSE_CODES	APIcast のロギングレスポンスコードを有効にします。	true	任意
APICAST_REGISTRY_URL	APIcast ポリシーの場所に解決する URL	**http://apicast-staging:8090/policies**	必須
MASTER_USER	マスター管理者アカウントのユーザー名	**master**	必須

名前	説明	デフォルト値	必須/任意
MASTER_NAME	マスター管理ポータルのサブドメイン値。**-master** 接尾辞が付けられます。	**master**	必須
MASTER_PASSWORD	無作為に生成されるマスター管理者のパスワード	該当なし	必須
MASTER_ACCESS_TOKEN	API 呼び出しのマスターレベル権限が設定されたトークン	該当なし	必須
IMAGESTREAM_TAG_IMPORT_INSECURE	イメージのインポート中にサーバーが証明書の検証を回避できる、または HTTP 経由で直接接続できる場合は、true を設定します。	**false**	必須

2.6. OPENSHIFT 上での 3SCALE と APICAST の使用

ホスト型 3scale および OpenShift Container Platform におけるオンプレミスインストールでは、API Manager との組み合わせで APIcast を利用することができます。両設定で、設定手順は異なります。本セクションでは、OpenShift 上で API Manager と共に APIcast をデプロイする方法を説明します。

2.6.1. 3scale が含まれる既存 OpenShift クラスターでの APIcast テンプレートのデプロイ

3scale OpenShift テンプレートには Embedded APIcast が 2 つデフォルトで含まれています。より多くの API ゲートウェイが必要な場合や、別の APIcast デプロイメントが必要な場合は、追加の APIcast テンプレートを OpenShift クラスターにデプロイすることができます。

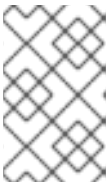
追加の API ゲートウェイを OpenShift クラスターにデプロイするには、以下の手順を実施します。

1. 以下の設定で アクセストークン を作成します。
   - スコープ: Account Management API
   - アクセス権限: 読み取り専用
2. APIcast クラスターにログインします。

```
oc login
```

3. APIcast が 3scale と通信できるようにするシークレットを作成します。**new-basicauth** および **apicast-configuration-url-secret** を指定し、**--password** パラメーターで 3scale デプロイメントのアクセストークン、テナント名、およびワイルドカードドメインを設定します。

```
oc secret new-basicauth apicast-configuration-url-secret --
password=https://<APICAST_ACCESS_TOKEN>@<TENANT_NAME>-admin.
<WILDCARD_DOMAIN>
```

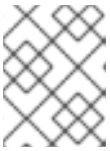


注記

TENANT_NAME は、管理ポータルにアクセスすることのできるルート下の名前です。**TENANT_NAME** のデフォルト値は **3scale** です。3scale デプロイメントでカスタム値を使用した場合は、ここでもその値を使用する必要があります。

4. **--file** オプションで **apicast.yml** ファイルを指定して、**oc new-app** コマンドを使用して APIcast テンプレートをインポートします。

```
oc new-app --file /opt/amp/templates/apicast.yml
```



注記

「ノードおよびエンタイトルメントの設定」 の説明に従って、最初に APIcast テンプレートをインストールします。

2.6.2. 異なる OpenShift クラスターからの APIcast への接続

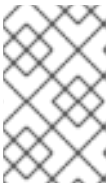
3scale クラスター外部の別の OpenShift クラスターに APIcast をデプロイする場合は、パブリックルート経由で接続する必要があります。

1. 以下の設定で アクセストークン を作成します。
   - スコープ: Account Management API
   - アクセス権限: 読み取り専用
2. APIcast クラスターにログインします。

```
oc login
```

3. APIcast が 3scale と通信できるようにするシークレットを作成します。**new-basicauth** および **apicast-configuration-url-secret** を指定し、**--password** パラメーターで 3scale デプロイメントのアクセストークン、テナント名、およびワイルドカードドメインを設定します。

```
oc secret new-basicauth apicast-configuration-url-secret --
password=https://<APICAST_ACCESS_TOKEN>@<TENANT_NAME>-admin.
<WILDCARD_DOMAIN>
```



注記

TENANT_NAME は、管理ポータルにアクセスすることのできるルート下の名前です。**TENANT_NAME** のデフォルト値は **3scale** です。3scale デプロイメントでカスタム値を使用した場合は、その値を使用する必要があります。

4. **oc new-app** コマンドで、別の OpenShift クラスターに APIcast をデプロイします。 **--file** オプションで **apicast.yml** ファイルへのパスを指定します。

```
oc new-app --file /path/to/file/apicast.yml
```

2.6.3. Embedded APIcast のデフォルト動作の変更

外部の APIcast デプロイメントでは、APIcast OpenShift テンプレートの テンプレートパラメーターを変更する ことにより、デフォルトの動作を変更できます。

Embedded APIcast デプロイメントでは、3scale および APIcast は単一のテンプレートからデプロイされます。Embedded APIcast デプロイメントのデフォルト動作を変更する場合は、デプロイメントの後に環境変数を編集する必要があります。

2.6.4. 内部サービスルートを介した、単一 OpenShift クラスター上の複数 APIcast デプロイメントの接続

同じ OpenShift クラスターに複数の APIcast ゲートウェイをデプロイする場合、デフォルトの外部ルート設定ではなく、バックエンドリスナーサービスを介して内部ルートを使用して接続するよう設定できます。

内部サービスルート経由で接続するには、OpenShift SDN プラグインがインストールされている必要があります。接続方法は、インストールされた SDN によって異なります。

ovs-subnet

ovs-subnet OpenShift Software-Defined Networking (SDN) プラグインを使用している場合は、次の手順を実行して内部ルート経由で接続します。

1. OpenShift クラスターにログインしていない場合はログインします。

```
oc login
```

2. 以下のコマンドを入力し、**backend-listener** ルートの URL を表示します。

```
oc get route backend
```

3. **apicast.yml** へのパスを指定して **oc new-app** コマンドを入力します。

```
oc new-app -f apicast.yml
```

ovs-multitenant

ovs-multitenant OpenShift SDN プラグインを使用している場合は、以下の手順を実施して内部ルート経由で接続します。

1. OpenShift クラスターにログインしていない場合はログインします。

```
oc login
```

2. 管理者として、**oadm** コマンドに **pod-network** および **join-projects** オプションを指定し、両方のプロジェクト間の通信を設定します。

```
oadm pod-network join-projects --to=<3SCALE_PROJECT> <APICAST_PROJECT>
```

3. 以下のコマンドを入力し、**backend-listener** ルートの URL を表示します。

```
oc get route backend
```

4. **apicast.yml** へのパスを指定して **oc new-app** コマンドを入力します。

```
oc new-app -f apicast.yml
```

補足情報

OpenShift SDN およびプロジェクトネットワークの分離についての情報は、OpenShift Container Platform ネットワークの OpenShift SDN を参照してください。

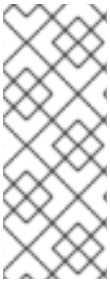
2.6.5. 他のデプロイメント上の APIcast の接続

APIcast を Docker にデプロイする場合には、**THREESCALE_PORTAL_ENDPOINT** パラメーターを OpenShift 上にデプロイした 3scale 管理ポータル の URL およびアクセストークンに設定することで、APIcast を OpenShift 上にデプロイした 3scale に接続することができます。この場合、**BACKEND_ENDPOINT_OVERRIDE** パラメーターを設定する必要はありません。

詳細は、*5章Docker コンテナー環境における APIcast* を参照してください。

2.7. OPERATOR を使用した 3SCALE のデプロイ

ここでは、**APIManager** カスタムリソースを使用して、3scale operator 経由で 3scale ソリューションをインストールおよびデプロイする方法を説明します。



注記

- 3scale 2.6 では、ワイルドカードルートは 廃止されています。
  - この機能は、バックグラウンドで Zync により処理されます。
- API プロバイダーが作成、更新、または削除されると、これらの変更が自動的にルートに反映されます。

2.7.1. 前提条件

- 「OpenShift でのレジストリー認証の設定」
- operator を使用して 3scale をデプロイするには、まず*6章OpenShift 4.x への 3scale operator のインストール*に記載の手順に従う必要があります。
- OpenShift Container Platform (OCP) 4.x
  - OpenShift クラスターの管理者権限を持つユーザーアカウント
  - **注記:** OCP 4.x は、operator を使用した 3scale のデプロイメントのみをサポートしています。

2.7.2. APIManager カスタムリソースのデプロイ

APIManager カスタムリソースをデプロイすると、operator がプロセスを開始し、そこから 3scale ソリューションがデプロイされます。

手順

1. **Catalog > Installed Operators** の順にクリックします。
   a. **Installed Operators** のリストで、**3scale Operator** をクリックします。
2. **API Manager** タブをクリックします。
3. **Create APIManager** をクリックします。
4. サンプルのコンテンツを消去して以下の **YAML** 定義をエディターに追加し、続いて **Create** をクリックします。

注記

wildcardDomain パラメーターには、有効な DNS ドメインである、IP アドレスに対して解決する任意の名前を指定できます。パラメーターのプレースホルダーを表す記号 < >を必ず削除するようにしてください。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIManager
metadata:
  name: example-apimanager
spec:
  wildcardDomain: <wildcardDomain>
  resourceRequirementsEnabled: true
```

注記

APIManager フィールドに関する詳細は、参考のドキュメント を参照してください。

2.7.3. APIManager 管理者のクレデンシャルの取得

operator ベースのデプロイメント後に 3scale にログインするには、管理者のクレデンシャルが必要です。これらのクレデンシャルを取得するには、次の手順を実施します。

1. 以下のコマンドを実行します。

```
oc get secret system-seed -o json | jq -r .data.ADMIN_USERNAME | base64 -d
oc get secret system-seed -o json | jq -r .data.ADMIN_PASSWORD | base64 -d
```

2. **APIManager** 管理者としてログインして、これらのクレデンシャルが機能していることを確認します。

2.7.4. 管理ポータルの URL の取得

operator を使用して 3scale をデプロイすると、固定 URL (**3scale-admin.${wildcardDomain}**) のデフォルトテナントが作成されます。

3scale の Dashboard には、テナントの新しいポータル URL が表示されます。たとえば、**<wildCardDomain>** が **3scale-project.example.com** の場合、管理ポータル URL は **https://3scale-admin.3scale-project.example.com** となります。

wildcardDomain は、インストール中に指定した **<wildCardDomain>** パラメーターです。以下のコマンドを使用し、ブラウザーでこの一意の URL を開きます。

```
xdg-open https://3scale-admin.3scale-project.example.com
```

オプションとして、**マスターポータル URL** (**master.${wildcardDomain}**) に新しいテナントを作成できます。

2.8. トラブルシューティング

本セクションでは、典型的なインストールの問題と、その問題を解決するためのアドバイスについて説明します。

- 「以前のデプロイメントがダーティーな永続ボリューム要求を残す」
- 「誤って Docker レジストリーからプルされる」
- 「MySQL の権限の問題」
- 「ログまたはイメージをアップロードできない」
- 「OpenShift でテストコールが動作しない」
- 「3scale 以外のプロジェクト上の APIcast」

2.8.1. 以前のデプロイメントがダーティーな永続ボリューム要求を残す

問題

以前のデプロイメントがダーティーな永続ボリューム要求 (PVC) を残そうとするため、MySQL コンテナーの起動に失敗する。

原因

OpenShift のプロジェクトを削除しても、それに関連する PVC は消去されない。

解決方法

1. **oc get pvc** コマンドを使用してエラーのある MySQL データが含まれる PVC を探します。

```
# oc get pvc
NAME                     STATUS    VOLUME    CAPACITY   ACCESSMODES   AGE
backend-redis-storage    Bound     vol003    100Gi      RWO,RWX       4d
mysql-storage            Bound     vol006    100Gi      RWO,RWX       4d
system-redis-storage     Bound     vol008    100Gi      RWO,RWX       4d
system-storage           Bound     vol004    100Gi      RWO,RWX       4d
```

2. OpenShift UI の **cancel deployment** をクリックして、system-mysql Pod のデプロイメントを停止します。
3. MySQL パス以下にあるものすべてを削除し、ボリュームをクリーンアップします。
4. 新たに **system-mysql** のデプロイメントを開始します。

2.8.2. 誤って Docker レジストリーからプルされる

問題

インストール中に以下のエラーが発生する。

```
svc/system-redis - 1EX.AMP.LE.IP:6379
  dc/system-redis deploys docker.io/rhscl/redis-32-rhel7:3.2-5.3
    deployment #1 failed 13 minutes ago: config change
```

原因

OpenShift は **docker** コマンドを実行し、コンテナーイメージを検索およびプルします。このコマンドは、**registry.redhat.io** Red Hat コンテナーレジストリーではなく、 **docker.io** Docker レジストリーを参照します。

これは、システムに予期せぬバージョンの Docker コンテナー環境が含まれる場合に発生します。

解決方法

適切なバージョン の Docker コンテナー環境を使用します。

2.8.3. MySQL の権限の問題

問題

system-msql Pod がクラッシュし、デプロイされないため、それに依存する他のシステムのデプロイメントに失敗する。Pod ログに以下のエラーが記録される。

```
[ERROR] Cannot start server : on unix socket: Permission denied
[ERROR] Do you already have another mysqld server running on socket: /var/lib/mysql/mysql.sock ?
[ERROR] Aborting
```

原因

MySQL プロセスが不適切なユーザー権限で起動されている。

解決方法

1. 永続ボリュームに使用されるディレクトリーには、root グループの書き込み権限が必要です。MySQL サービスは root グループの別のユーザーとして実行されるため、root ユーザーの読み取り/書き込み権限では不十分です。root ユーザーとして以下のコマンドを実行します。

   ```
   chmod -R g+w /path/for/pvs
   ```

2. 以下のコマンドを実行して、SELinux がアクセスをブロックしないようにします。

   ```
   chcon -Rt svirt_sandbox_file_t /path/for/pvs
   ```

2.8.4. ロゴまたはイメージをアップロードできない

問題

ロゴをアップロードできず、**system-app** ログに以下のエラーが表示される。

```
Errno::EACCES (Permission denied @ dir_s_mkdir - /opt/system/public//system/provider-name/2
```

原因

OpenShift が永続ボリュームに書き込みを行うことができない。

解決方法

OpenShift が永続ボリュームに書き込みを行えるようにします。永続ボリュームのグループ所有者を root グループにし、またグループによる書き込みを可能にする必要があります。

2.8.5. OpenShift でテストコールが動作しない

問題

OpenShift で新しいサービスとルートを作成した後に、テストコールが動作しない。curl 経由のダイレクトコールも失敗し、**service not available** が出力される。

原因

3scale はデフォルトで HTTPS ルートが必要で、OpenShift ルートはセキュアではない。

解決方法

OpenShift のルーター設定で **secure route** チェックボックスが選択されていることを確認します。

2.8.6. 3scale 以外のプロジェクト上の APIcast

問題

APIcast のデプロイに失敗する (Pod が青にならない)。以下のエラーがログに記録されます。

```
update acceptor rejected apicast-3: pods for deployment "apicast-3" took longer than 600 seconds to become ready
```

以下のエラーが Pod に表示されます。

```
Error synching pod, skipping: failed to "StartContainer" for "apicast" with RunContainerError: "GenerateRunContainerOptions: secrets \"apicast-configuration-url-secret\" not found"
```

原因

シークレットが適切に設定されていない。

解決方法

APIcast v3 でシークレットを作成する時に **apicast-configuration-url-secret** を指定します。

```
oc secret new-basicauth apicast-configuration-url-secret  --password=https://<ACCESS_TOKEN>@<TENANT_NAME>-admin.<WILDCARD_DOMAIN>
```

第3章 APICAST のインストール

APIcast は、内部および外部の API サービスを 3scale Platform と統合するのに使用される NGINX ベースの API ゲートウェイですAPIcast は、ラウンドロビン形式で負荷分散を行います。

本章では、デプロイメントオプション、提供される環境、および使用を開始する方法について説明します。

APIcast の最新リリースやサポートされるバージョンに関する情報は、アーティクル Red Hat 3scale API Management Supported Configurations および Red Hat 3scale API Management - Component Details を参照してください。

3.1. 前提条件

APIcast がスタンドアロンの API ゲートウェイではない。3scale API Manager への接続が必要です。

- 稼働中の オンプレミス型 3scale インスタンスが必要です。

3.2. デプロイメントのオプション

Hosted APIcast (ホスト型 APIcast) または Self-managed APIcast (自己管理型 APIcast) を使用できます。どちらの場合にも、APIcast は残りの 3scale API Management プラットフォームに接続している必要があります。

- **Embedded APIcast** (組み込み型 APIcast): 3scale API Management インストールにはデフォルトで 2 つの APIcast ゲートウェイ (ステージングと実稼働) が含まれています。これらのゲートウェイは事前設定されているため、そのまま使用することができます。

- **Self-managed APIcast** (自己管理型 APIcast): APIcast をどこにでもデプロイすることができます。Self-managed (自己管理) モードは、実稼働環境での操作を目的としたモードです。APIcast のデプロイメントにおける推奨オプションの一部は以下のとおりです。

  - Docker コンテナー環境: そのまま使用できる Docker 形式のコンテナーイメージをダウンロードします。これには、Docker 形式のコンテナーで APIcast を実行するための依存関係がすべて含まれています。

  - OpenShift: APIcast を サポート対象バージョン の OpenShift で実行します。Self-managed APIcast (自己管理の APIcast) は 3scale インストールまたは 3scale オンラインアカウントに接続できます。

3.3. 環境

デフォルトでは、3scale アカウントを作成すると、2 つの異なる環境の Embedded APIcast が提供されます。

- **ステージング**: API インテグレーションの設定中またはテスト中にのみ使用することが目的です。設定が想定どおりに動作していることが確認されたら、実稼働環境にデプロイすることができます。OpenShift テンプレートは、設定が各 API 呼び出し (**APICAST_CONFIGURATION_LOADER: lazy**、**APICAST_CONFIGURATION_CACHE: 0**) で再読み込みされるよう、ステージング APIcast のパラメーターを設定します。APIcast の設定変更を即座にテストするのに便利です。

- **実稼働**: 実稼働向けの環境です。**APICAST_CONFIGURATION_LOADER: boot** および **APICAST_CONFIGURATION_CACHE: 300** パラメーターは、OpenShift テンプレートの実稼働 APIcast のために設定されています。そのため、APIcast の開始時に設定が完全に読み込ま

れ、300 秒 (5 分) 間キャッシュに保存されます。設定は 5 分後に再読み込みされます。これにより、設定を実稼働環境にプロモートすると、APIcast の新しいデプロイメントを実行しない限り、設定の適用に 5 分程度かかる場合があります。

3.4. インテグレーション設定

[your_API_name] > Integration > Configurationの順に移動します。

Integration ページの上部に、インテグレーションのオプションが表示されます。デフォルトのデプロイメントオプションは Hosted APIcast で、デフォルトの認証モードは API キーです。右上隅の **edit integration settings** をクリックすると、設定を変更することができます。

Integration ページの上部に、インテグレーションのオプションが表示されます。デフォルトでは、以下の値が表示されます。

- デプロイメントオプション: Embedded APIcast
- 認証モード: API キー

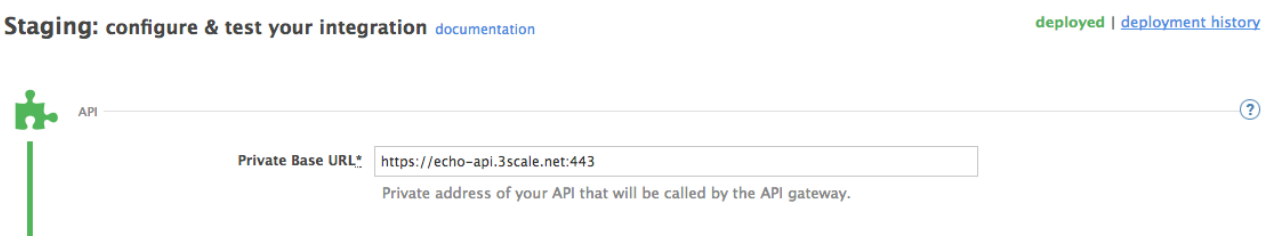
右上隅の **edit integration settings** をクリックすると、設定を変更することができます。

3.5. サービスの設定

3.5.1. API バックエンドの宣言

Private Base URL フィールドに API バックエンドのエンドポイントホストを指定して、API バックエンドを宣言する必要があります。すべての認証、承認、流量制御、および統計が処理された後、APIcast はすべてのトラフィックを API バックエンドにリダイレクトします。

通常、API のプライベートベース URL は、管理するドメイン (**yourdomain.com**) 上で **https://api-backend.yourdomain.com:443** のようになります。たとえば、Twitter API と統合する場合、プライベートベース URL は **https://api.twitter.com/** になります。この例では、3scale によってホストされる Echo API を使用します。この API は、すべてのパスを許可し、リクエストに関する情報 (パス、リクエストパラメーター、ヘッダーなど) を返すシンプルな API です。このプライベートベース URL は **https://echo-api.3scale.net:443** になります。



プライベート (アンマネージド) API が動作することをテストします。たとえば、Echo API の場合には **curl** コマンドを使用して以下の呼び出しを行うことができます。

```
curl "https://echo-api.3scale.net:443"
```

以下のレスポンスが返されます。

```
{
    "method": "GET",
    "path": "/",
```

```
  "args": "",
  "body": "",
  "headers": {
    "HTTP_VERSION": "HTTP/1.1",
    "HTTP_HOST": "echo-api.3scale.net",
    "HTTP_ACCEPT": "*/*",
    "HTTP_USER_AGENT": "curl/7.51.0",
    "HTTP_X_FORWARDED_FOR": "2.139.235.79, 10.0.103.58",
    "HTTP_X_FORWARDED_HOST": "echo-api.3scale.net",
    "HTTP_X_FORWARDED_PORT": "443",
    "HTTP_X_FORWARDED_PROTO": "https",
    "HTTP_FORWARDED": "for=10.0.103.58;host=echo-api.3scale.net;proto=https"
  },
  "uuid": "ee626b70-e928-4cb1-a1a4-348b8e361733"
}
```


3.5.2. 認証の設定

AUTHENTICATION SETTINGS セクションで API の認証設定を行うことができます。以下のフィールドはすべて任意です。

フィールド	説明
Host Header	カスタムの Host リクエストヘッダーを定義します。これは、API バックエンドが特定のホストからのトラフィックのみを許可する場合に必要です。
Secret Token	API バックエンドに直接送られる開発者リクエストをブロックするために使用します。ここにヘッダーの値を設定し、バックエンドがこのシークレットヘッダーを持つ呼び出しのみを許可するようにします。
Credentials location	クレデンシャルが HTTP ヘッダー、クエリーパラメーター、または HTTP Basic 認証として渡されるかどうかを定義します。
Auth user key	Credentials location に関連付けられたキーを設定します。
Errors	エラー発生時 (認証失敗、認証パラメーターがない、および一致するルールがない) のレスポンスコード、コンテンツタイプ、およびレスポンス本文を定義します。

3.5.3. API テストコールの設定

Hosted ステージング環境のテストコールを設定する必要があります。API に存在するパスを **API test GET request** フィールドに入力します (例: **/v1/word/good.json**)。

3.5.4. 設定の保存

ページ右下の **Update & test in Staging Environment**ボタンをクリックして、設定を保存します。これにより、APIcast の設定が 3scale Hosted ステージング環境にデプロイされます。すべてが正しく設定されていれば、左側の縦線が緑色に変わります。

Self-managed デプロイメントオプションの 1 つを使用してる場合 は、GUI から設定を保存し、Staging Public Base URL フィールドまたは Production Public Base URL フィールドに正しいホストを追加して、デプロイされた API ゲートウェイをポイントするようにします。実稼働ゲートウェイに呼び出しを行う前に、必ず **Promote v.x to Production APIcast**ボタンを押してください。

ステージングセクションの最後にある **curl** コマンドの例を確認し、コンソールからコマンドを実行します。

```
curl "https://XXX.staging.apicast.io:443/v1/word/good.json?user_key=YOUR_USER_KEY"
```

注記

上記と同じレスポンスを取得するはずですが、今回はリクエストが 3scale Hosted APIcast インスタンスを通ります。注記: アプリケーションのクレデンシャルがサービスに対して有効なクレデンシャルであることを確認してください。3scale の登録時に作成されたデフォルトの API サービスを使用している場合はアプリケーションがすでにあるはずです。テストの curl に **USER_KEY** または **APP_ID** と **APP_KEY** の値が出力された場合は、最初にこのサービスのアプリケーションを作成する必要があります。

これで API が 3scale と統合されました。

3scale Hosted APIcast ゲートウェイはクレデンシャルを検証し、アプリケーションのアプリケーションプランで定義した流量制御を適用します。クレデンシャルがない、あるいは無効なクレデンシャルで呼び出しを行うと、エラーメッセージが表示されます。

第4章 RED HAT OPENSHIFT 上での APICAST の実行

本チュートリアルでは、Red Hat OpenShift に APIcast API ゲートウェイをデプロイする方法について説明します。

4.1. 前提条件

以下のチュートリアルの手順を行うには、APIcast のインストール に従って、まず 3scale 管理ポータルで APIcast を設定する必要があります。インテグレーション設定でデプロイメントオプションに **Self-managed Gateway** が選択されていることを確認してください。手順を進めるには、ステージング環境と実稼働環境の両方を設定している必要があります。

4.2. RED HAT OPENSHIFT の設定

OpenShift クラスターが稼働中である場合は、本手順を省略できます。稼働中でなければ、以下の手順に従ってください。

実稼働デプロイメントの場合は、OpenShift のインストール手順 に従います。

本チュートリアルでは、OpenShift クラスターは以下を使用してインストールされます。

- Red Hat Enterprise Linux (RHEL) 7
- Docker コンテナー環境 (v1.10.3)
- OpenShift Origin コマンドラインインターフェイス (CLI) v1.3.1

4.2.1. Docker コンテナー環境のインストール

Red Hat が提供する Docker 形式のコンテナーイメージは、RHEL の Extras チャンネルの一部としてリリースされています。追加のリポジトリーを有効にするには、Subscription Manager または yum-config-manager を使用できます。詳細は、RHEL の製品ドキュメント を参照してください。

AWS EC2 インスタンスにデプロイされた RHEL 7 では、以下の手順を使用します。

1. すべてのリポジトリーを一覧表示します。

```
sudo yum repolist all
```

2. ***-extras** リポジトリーを探し、有効にします。

```
sudo yum-config-manager --enable rhui-REGION-rhel-server-extras
```

3. Docker 形式のコンテナーイメージをインストールします。

```
sudo yum install docker docker-registry
```

4. **/etc/sysconfig/docker** ファイルに以下の行を追加するか、アンコメントして、**172.30.0.0/16** のセキュアでないレジストリーを追加します。

```
INSECURE_REGISTRY='--insecure-registry 172.30.0.0/16'
```

5. Docker サービスを開始します。

```
sudo systemctl start docker
```

以下のコマンドを使用すると、コンテナーサービスが実行していることを確認できます。

```
sudo systemctl status docker
```

4.2.2. OpenShift クラスターの起動

OpenShift リリースページ から、クライアントツールの最新の安定版リリース (**openshift-origin-client-tools-VERSION-linux-64bit.tar.gz**) をダウンロードし、アーカイブから抽出した Linux **oc** バイナリーを **PATH** に置きます。

注記

- コマンドの **oc cluster** セットは、1.3 以降のリリースでのみ使用できることに注意してください。
- docker コマンドは **root** ユーザーとして実行されるため、**oc** または docker コマンドはすべて root 権限で実行する必要があります。

docker コマンドを実行する権限のあるユーザーでターミナルを開き、以下のコマンドを実行します。

```
oc cluster up
```

出力の最後に、デプロイされたクラスターの情報が表示されます。

```
-- Server Information ...
  OpenShift server started.
  The server is accessible via web console at:
  https://172.30.0.112:8443

  You are logged in as:
    User:     developer
    Password: developer

  To login as administrator:
    oc login -u system:admin
```

OpenShift サーバーに割り当てられた IP アドレスを書き留めておきます。本チュートリアルでは **OPENSHIFT-SERVER-IP** をその IP アドレスに置き換えてください。

4.2.3. リモートサーバーでの OpneShift クラスターの設定 (任意設定)

OpenShift クラスターをリモートサーバーにデプロイする場合、クラスターの起動時にパブリックホスト名とルーティング接尾辞を明示的に指定し、OpenShift Web コンソールへリモートアクセスできるようにする必要があります。

たとえば、AWS EC2 インスタンスでデプロイする場合は、以下のオプションを指定する必要があります。

```
oc cluster up --public-hostname=ec2-54-321-67-89.compute-1.amazonaws.com --routing-suffix=54.321.67.89.xip.io
```

ec2-54-321-67-89.compute-1.amazonaws.com はパブリックドメイン、**54.321.67.89** はインスタンスの IP アドレスに置き換えます。これにより、https://ec2-54-321-67-89.compute-1.amazonaws.com:8443 で OpenShift Web コンソールにアクセスできるようになります。

4.3. OPENSHIFT テンプレートを使用した APICAST のデプロイ

1. デフォルトでは、**developer** としてログインしていて、次のステップに進むことができます。そうでなければ、前の手順でダウンロードおよびインストールした OpenShift クライアントツールから **oc login** コマンドを使用して OpenShift にログインします。デフォルトのログインクレデンシャルは **username = "developer"** と **password = "developer"** です。

```
oc login https://OPENSHIFT-SERVER-IP:8443
```

出力に **Login successful.** が表示されるはずです。

2. プロジェクトを作成します。この例では表示名を **gateway** と設定します。

```
oc new-project "3scalegateway" --display-name="gateway" --description="3scale gateway demo"
```

応答は以下のようになります。

```
Now using project "3scalegateway" on server "https://172.30.0.112:8443".
```

コマンドプロンプトのテキスト出力で提案される次のステップを無視し、以下に示す次のステップに進みます。

3. プロジェクトを参照する新しいシークレットを作成します。**<access_token>** および **<domain>** はご自分のクレデンシャルに置き換えます。**<access_token>** および **<domain>** の詳細は、以下を参照してください。

```
oc create secret generic apicast-configuration-url-secret --from-literal=password=https://<access_token>@<admin_portal_domain>  --type=kubernetes.io/basic-auth
```

ここでは、**<access_token>** は 3scale Account Management API の アクセストークン で (サービストークンではありません)、**<domain>-admin.3scale.net** は 3scale 管理ポータルの URL になります。

応答は以下のようになります。

```
secret/apicast-configuration-url-secret
```

4. テンプレートから APIcast ゲートウェイのアプリケーションを作成し、デプロイメントを開始します。

```
oc new-app -f https://raw.githubusercontent.com/3scale/3scale-amp-openshift-templates/2.6.0.GA/apicast-gateway/apicast.yml
```

出力の最後に以下のメッセージが表示されるはずです。

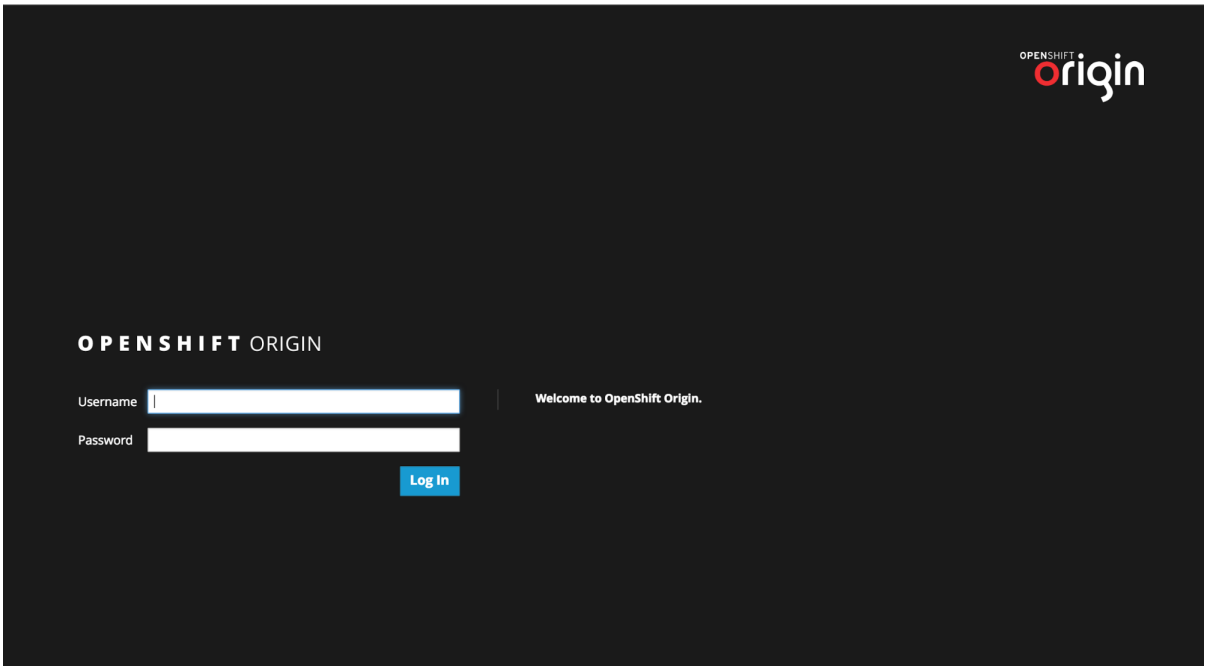
```
--> Creating resources with label app=3scale-gateway ...
    deploymentconfig "apicast" created
```

```
    service "apicast" created
--> Success
    Run 'oc status' to view your app.
```

4.4. OPENSHIFT コンソール経由でのルートの作成

1. ブラウザーで OpenShift クラスターの Web コンソール https://OPENSHIFT-SERVER-IP:8443/console/ を開きます。
   OpenShift クラスターをリモートサーバーで起動した場合は、**OPENSHIFT-SERVER-IP** ではなく、**--public-hostname** で指定した値を使用します。

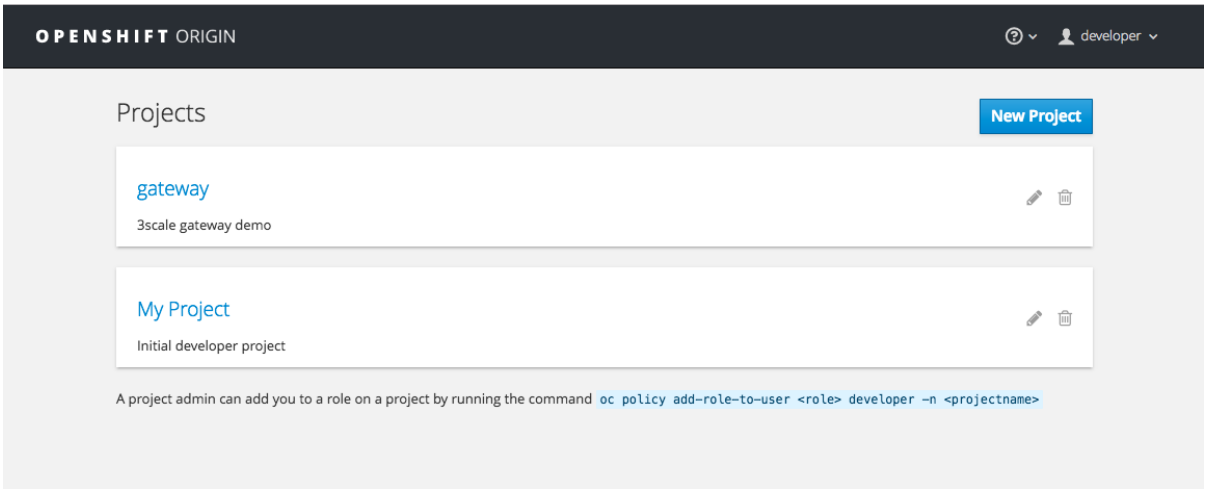
   ログイン画面が表示されます。



注記

信頼できない Web サイトに関する警告が表示される可能性があります。有効な証明書を設定せずに、セキュアなプロトコルで Web コンソールにアクセスしようとしているため、この警告は想定内です。これは実稼働環境で行うべきではありませんが、このテスト設定では続行してこのアドレスの例外を作成することができます。

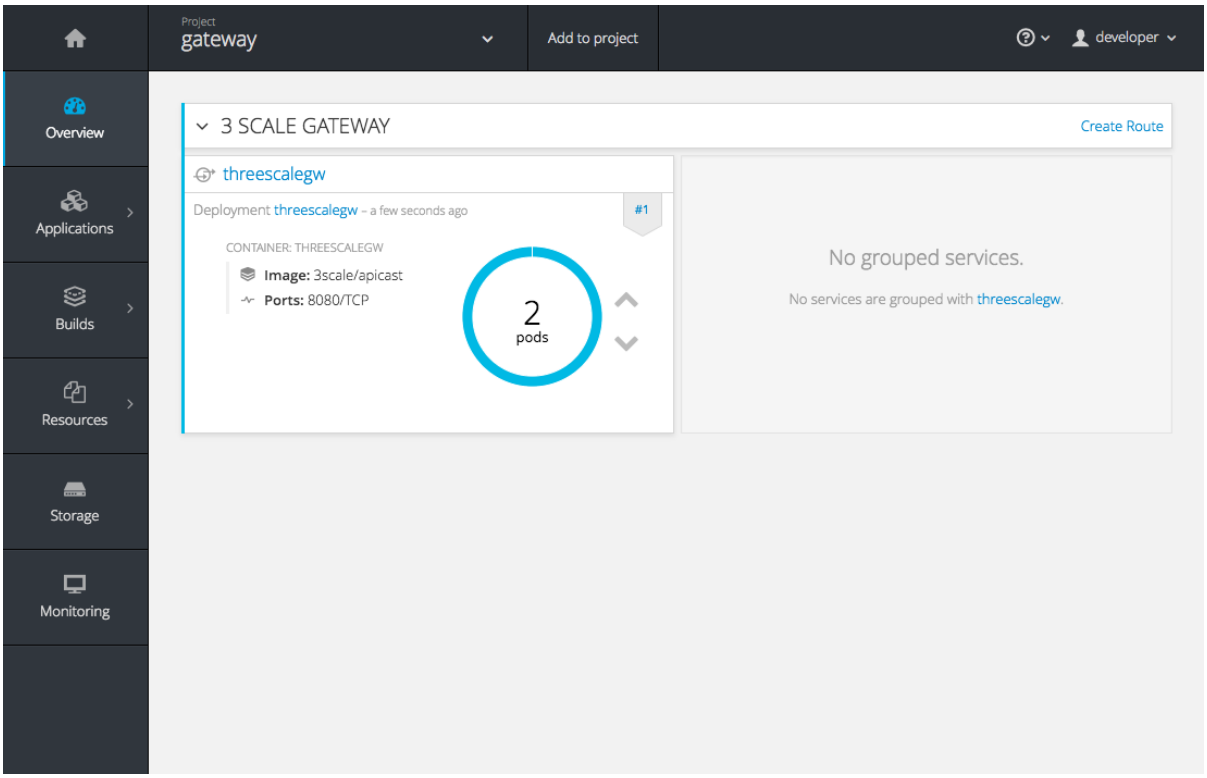
2. 上記の **Red Hat OpenShift の設定** セクションで作成または取得した **developer** のクレデンシャルを使用してログインします。
   上記のコマンドラインから作成した **gateway** プロジェクトが含まれる、プロジェクトのリストが表示されます。



gateway プロジェクトが表示されない場合は、別のユーザーで作成した可能性があり、ポリシーロールをこのユーザーに割り当てる必要があります。

3. **gateway** リンクをクリックすると、**Overview** タブが表示されます。
OpenShift は APIcast のコードをダウンロードし、デプロイメントを開始しました。デプロイメントの進行中に **Deployment #1 running** というメッセージが表示されることがあります。

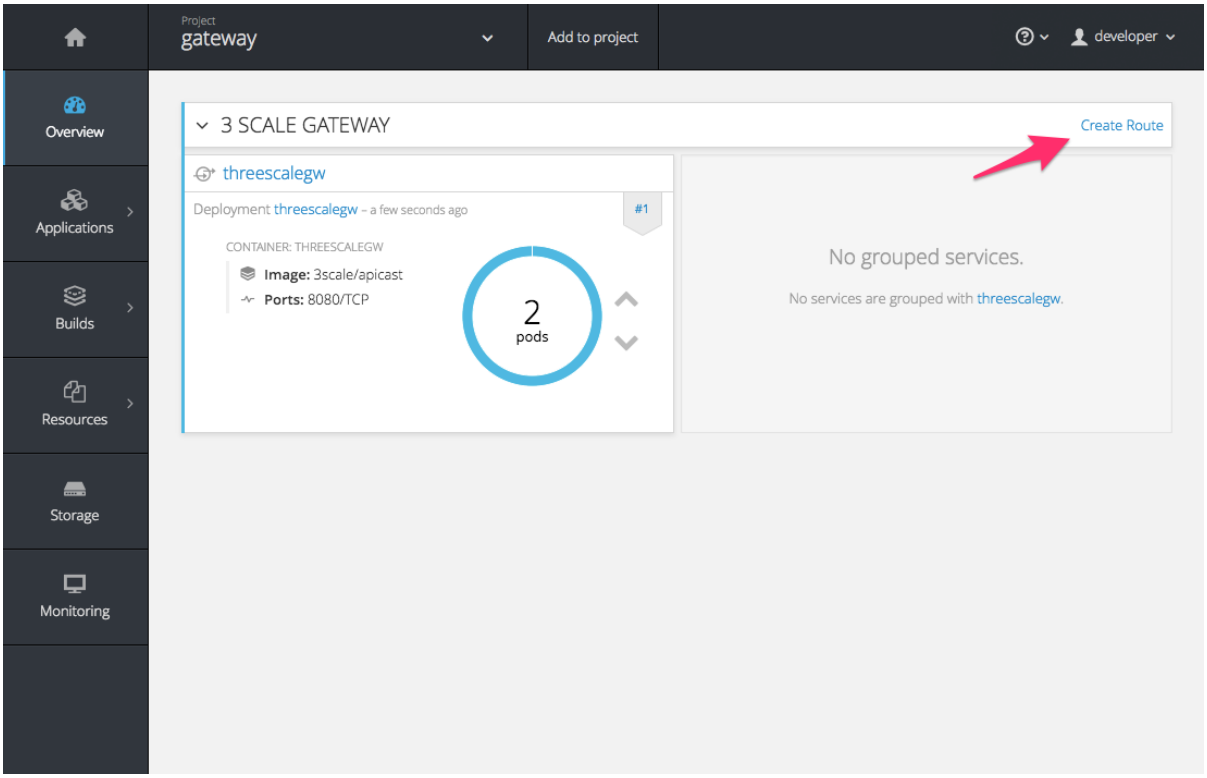
   ビルドが完了すると、UI が更新され、テンプレートで定義されたとおりに OpenShift によって起動された APIcast の 2 つのインスタンス (**2 つの Pod**) が表示されます。



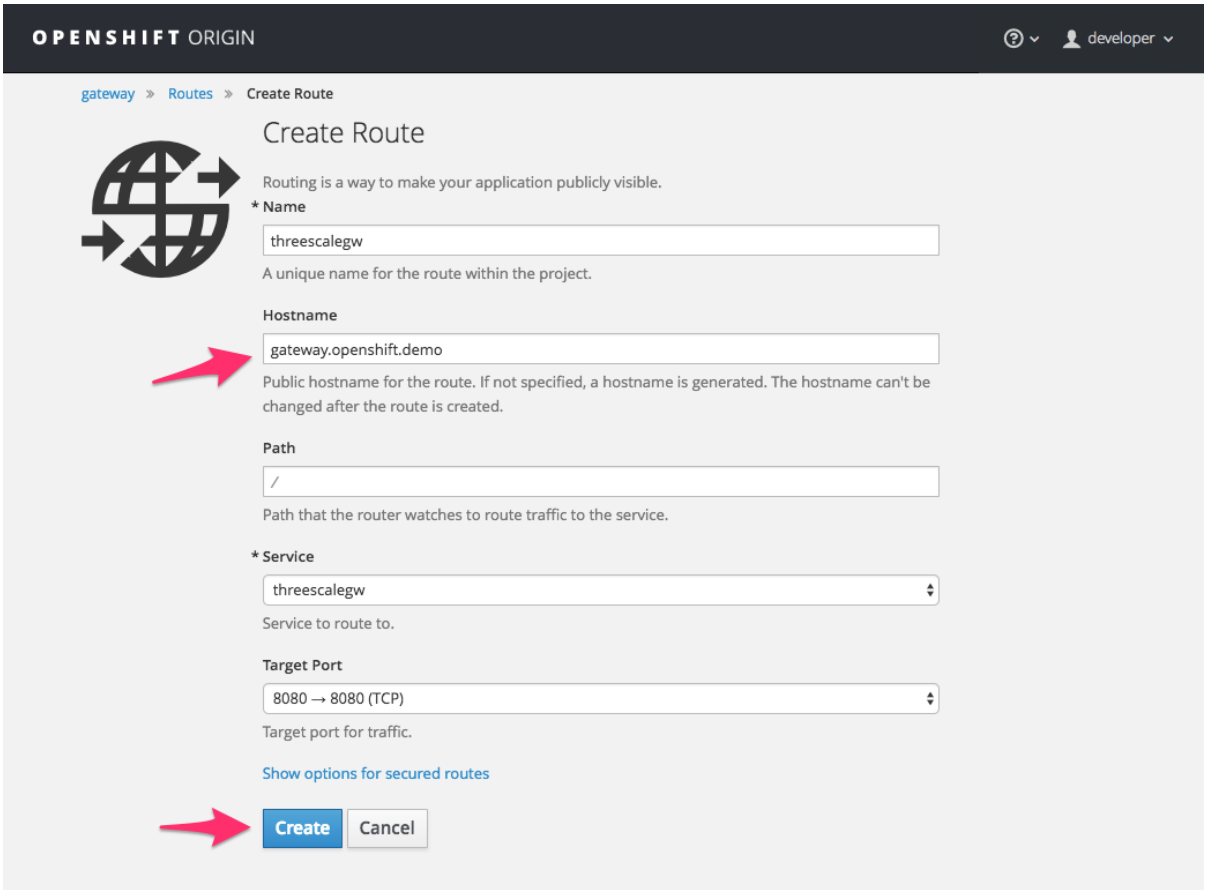
各 APIcast インスタンスが起動すると、3scale 管理ポータルの **Integration** ページの設定を使用して、3scale から必要な設定をダウンロードします。

OpenShift は 2 つの APIcast インスタンスを維持し、両方のインスタンスの状態を監視します。状態の悪い APIcast インスタンスは自動的に新しいインスタンスに置き換えられます。

4. APIcast インスタンスでトラフィックを受信できるようにするには、ルートを作成する必要があります。**Create Route** をクリックして操作を開始します。



上記の **Public Base URL** フィールドで 3scale に設定したホストを、**http://** とポートを削除して入力し (例: **gateway.openshift.demo**)、**Create** ボタンをクリックします。



定義するすべての 3scale サービスについて、新規ルートを作成する必要があります。

第5章 DOCKER コンテナー環境における APICAST

ここでは、3scale API ゲートウェイとして使用する準備が整っている Docker 形式のコンテナー内部に APIcast をデプロイする方法をステップごとに説明します。

5.1. 前提条件

APIcast のインストール に従って、3scale 管理ポータルで APIcast を設定する必要があります。

5.2. DOCKER コンテナー環境のインストール

ここでは、Red Hat Enterprise Linux (RHEL) 7 に Docker コンテナー環境を設定する方法をステップごとに説明します。

Red Hat が提供する Docker 形式のコンテナーは、RHEL の Extras チャンネルの一部としてリリースされています。追加のリポジトリーを有効にするには、Subscription Manager または yum-config-manager を使用できます。詳細は、RHEL の製品ドキュメント を参照してください。

AWS EC2 インスタンスで RHEL 7 をデプロイするには、以下の手順を実行します。

1. **sudo yum repolist all** ですべてのリポジトリーを一覧表示します。
2. ***-extras** リポジトリーを探します。
3. **sudo yum-config-manager --enable rhui-REGION-rhel-server-extras** を実行し、**extras** リポジトリーを有効にします。
4. **sudo yum install docker** を実行し、Docker コンテナー環境のパッケージをインストールします。

他のオペレーティングシステムをお使いの場合は、以下の Docker ドキュメントを参照してください。

- Installing the Docker containerized environment on Linux distributions
- Installing the Docker containerized environment on Mac
- Installing the Docker containerized environment on Windows

5.3. DOCKER コンテナー環境ゲートウェイの実行

1. Docker デーモンを開始します。
   sudo systemctl start docker.service
2. Docker デーモンが実行されているか確認します。
   sudo systemctl status docker.service

   Red Hat レジストリーから、使用する準備ができている Docker 形式のコンテナーイメージをダウンロードします。

   sudo docker pull registry.access.redhat.com/3scale-amp26/apicast-gateway.
3. Docker 形式のコンテナーで APIcast を実行します。
   sudo docker run --name apicast --rm -p 8080:8080 -e THREESCALE_PORTAL_ENDPOINT=https://<access_token>@<domain>-admin.3scale.net registry.access.redhat.com/3scale-amp26/apicast-gateway.

ここで、**<access_token>** は 3scale Account Management API のアクセストークンに置き換えます。アクセストークンの代わりにプロバイダーキーを使用することもできます。**<domain>-admin.3scale.net** は 3scale 管理ポータルの URL です。

このコマンドは、**apicast**という Docker 形式のコンテナーをポート **8080** で実行し、3scale ポータルから JSON 設定ファイルを取得します。その他の設定オプションについては、APIcast のインストール を参照してください。

5.3.1. Docker コマンドオプション

docker run コマンドでは、以下のオプションを使用できます。

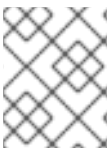
- **--rm**: 終了時にコンテナーを自動的に削除します。
- **-d** または **--detach**: コンテナーをバックグラウンドで実行し、コンテナー ID を出力します。このオプションを指定しないと、コンテナーはフォアグラウンドモードで実行され、**CTRL+c** を使用して停止することができます。デタッチモードで起動された場合、**docker attach** コマンド (例: **docker attach apicast**) を使用するとコンテナーに再アタッチすることができます。
- **-p** または **--publish**: コンテナーのポートをホストに公開します。値の書式は **<host port="">:<container port="">** とする必要があります。したがって、**-p 80:8080** の場合は、コンテナーのポート **8080** をホストマシンのポート **80** にバインドします。たとえば、Management API はポート **8090** を使用するため、**-p 8090:8090** を **docker run** コマンドに追加してこのポートを公開します。
- **-e** または **--env**: 環境変数を設定します。
- **-v** または **--volume**: ボリュームをマウントします。値は通常 **<host path="">:<container path="">[:<options>]** で表されます。**<options>** はオプションの属性で、ボリュームを読み取り専用に指定するには、**:ro** に設定します (デフォルトでは読み取り/書き込みモードでマウントされます)。たとえば、**-v /host/path:/container/path:ro** と設定します。

使用できるオプションの詳細については、Docker run reference を参照してください。

5.4. APICAST のテスト

次の手順は、Docker 形式のコンテナーが独自の設定ファイルと、3scale レジストリーからの Docker 形式のイメージで実行されるようにします。呼び出しは APIcast を介してポート **8080** でテストでき、3scale アカウントから取得できる正しい認証クレデンシャルを提供できます。

テストコールは、APIcast が適切に実行されていることを確認するだけでなく、認証とレポートが正常に処理されたことも確認します。



注記

呼び出しに使用するホストが **Integration** ページの **Public Base URL** フィールドに設定されたホストと同じであるようにしてください。

第6章 OPENSHIFT 4.X への 3SCALE OPERATOR のインストール

本セクションでは、以下の項目の実施方法について説明します。

- 新しいプロジェクトを作成する。
- 3scale インスタンスのデプロイ
- プロジェクトで **threescale-registry-auth** シークレットを作成する。
- Operator Lifecycle Manager (OLM) を使用して 3scale operator をインストールする。
- operator をデプロイした後にカスタムリソースをデプロイする。

前提条件

- 管理者権限を持つアカウントで OpenShift Container Platform (OCP) 4.1 クラスターにアクセスできること。
  - **注記:** OCP 4.x は、operator を使用した 3scale のデプロイメントのみをサポートしています。

警告

3scale operator とカスタムリソース定義 (CRD) は、新たに作成した空の **プロジェクト** にデプロイしてください。インフラストラクチャーが含まれる既存の **プロジェクト** にデプロイすると、既存の要素が変更または削除されることがあります。

6.1. 新しい OPENSHIFT プロジェクトの作成

以下のコマンドを実行して、新しい OpenShift プロジェクトを作成します。新規プロジェクトを作成し、operator およびカスタムリソースをインストールします。operator は、そのプロジェクトの OLM を通じてカスタムリソースを管理します。

注記

このコマンドでは、プロジェクト名の例として **operator-test** が使われています。このプロジェクト名を実際のプロジェクト名に置き換えてください。

```
oc new-project operator-test
```

これにより、operator、**APIManager** カスタムリソース (CR)、および **Capabilities** カスタムリソースがインストールされる新しい **OpenShift プロジェクト** が作成されます。

6.2. OLM を使用した 3SCALE OPERATOR のインストールと設定

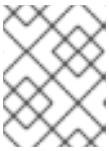
OLM を使用して、OpenShift Container Platform 4.1 クラスターに 3scale operator をインストールします。この際に、OpenShift Container Platform コンソールの OperatorHub を使用します。

注記

operator-test プロジェクトに 3scale operator をインストールおよびデプロイする必要があります。

手順

1. OpenShift Container Platform コンソールにおいて、管理者権限を持つアカウントを使用してログインします。
2. **Catalog > OperatorHub** の順にクリックします。
3. **Filter by keyword** ボックスに **3scale operator** と入力し、3scale operator を検索します。
4. 3scale operator をクリックします。operator に関する情報が表示されます。
5. operator に関する情報を確認し、**Install** をクリックします。Create Operator Subscription のページが表示されます。
6. **Create Operator Subscription** ページで、すべてのデフォルト設定を受け入れ **Subscribe** をクリックします。

   注記

   operator は、選択したクラスター上の特定の単一 namespace でしか使用することができません。

   3scale-operator の詳細ページが表示されるので、**Subscription Overview** を確認します。
7. サブスクリプションの **upgrade status** が **Up to date** と表示されたら **Catalog > Installed Operators** の順にクリックし、3scale operator の ClusterServiceVersion (CSV) が表示されたら、その Status が **operator-test** プロジェクトで最終的に InstallSucceeded となるのを確認します。
   a. インストールに成功すると、**APIManager** CRD および operator の **Capabilities** 機能に関連する CDR が **OpenShift API サーバー** に登録されます。
8. インストールが正常に完了したら、**oc get** を使用して CRD によって定義されたリソースタイプのクエリーを行います。
   a. たとえば、**APIManager** CRD が適切に登録され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 apimanagers
   ```
9. 以下の出力が表示されるはずです。

   ```
   No resources found.
   ```

   トラブルシューティングに関する情報は、OpenShift Container Platform のドキュメント を参照してください。

第7章 3SCALE 高可用性テンプレートおよび評価用テンプレート

7.1. はじめに

ここでは、Red Hat 3scale API Management 2.6 インストール環境で使用される 高可用性 テンプレートおよび 評価用 テンプレートについて説明します。

重要

3scale の高可用性テンプレートおよび評価用テンプレートは、テクノロジープレビューの機能としてのみ提供されます。テクノロジープレビューの機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではないことがあります。Red Hat は実稼働環境でこれらを使用することを推奨していません。テクノロジープレビューの機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

7.2. 前提条件

- 高可用性テンプレートおよび評価用テンプレートの要素をデプロイできる OpenShift クラスターが必要です。

7.3. 高可用性テンプレート

高可用性 (HA) テンプレートを使用すると、重要なデータベースの HA を設定できます。

7.3.1. 前提条件

- HA テンプレートをデプロイする前に、外部データベースをデプロイおよび設定し、負荷分散されたエンドポイントで HA を設定しておく。

7.3.2. HA テンプレートの使用

高可用性のために **amp-ha-tech-preview.yml** という名前のテンプレートを使用すると、OpenShift 外部に重要なデータベースをデプロイできます。ただし、以下は除外されます。

- Memcached
- Sphinx
- Zync

標準の **amp.yml** テンプレートと **amp-ha-tech-preview.yml** には、以下の違いがあります。

- 以下の要素が削除されています。
  - backend-redis およびその関連コンポーネント
  - system-redis およびその関連コンポーネント
  - system-mysql およびその関連コンポーネント
  - Redis および MySQL 関連の ConfigMaps

    - MYSQL_IMAGE、REDIS_IMAGE、MYSQL_USER、MYSQL_ROOT_PASSWORD パラメーター

- デフォルトで、データベースではない **DeploymentConfig** オブジェクトタイプのレプリカの数が 1 から 2 に増加されます。
- 以下の必須パラメーターが追加されているため、外部データベースの場所を制御できます。
    - BACKEND_REDIS_STORAGE_ENDPOINT
    - BACKEND_REDIS_QUEUES_ENDPOINT
    - SYSTEM_REDIS_URL
    - APICAST_STAGING_REDIS_URL
    - APICAST_PRODUCTION_REDIS_URL
    - SYSTEM_DATABASE_URL

amp-ha-tech-preview.yml を使用する場合、新たに追加された必須パラメーターによりクラスター外のデータベース接続を設定する必要があります (ただし、永続的なデータが含まれない **system-memcache**、**zync-database**、および **system-sphinx** は除外)。エンドポイントには、クレデンシャルを含む、データベースの負荷分散用接続文字列が必要です。また、データベースではないデプロイメントについては、アプリケーションレベルでの冗長性を確保するためにデフォルトで Pod レプリカの数が 2 に増えています。

7.4. 評価用テンプレート

評価の目的で、リソースのリクエストや制限のない 3scale 環境をデプロイする **amp-eval-tech-preview.yml** という名前のテンプレートが提供されています。

標準の **amp.yml** テンプレートとの唯一の機能的な違いは、リソースの制限とリクエストが削除されたことです。そのため、このバージョンでは CPU およびメモリーレベルでハードウェアの最低要件が Pod で削除されました。このテンプレートは、指定のハードウェアリソースを使用して、可能な限りコンポーネントをデプロイしようとするため、評価、テスト、および開発のみの使用を目的としています。

第8章 3SCALE の REDIS 高可用性 (HA) サポート

注記

operator と **backend-redis** には既知の問題があります。詳細は、Red Hat 3scale API Management 2.6 リリースノートの第 6 章 既知の問題 を参照してください。

高可用性 (HA) は、OpenShift Container Platform (OCP) によりほとんどのコンポーネントで提供されます。詳細は、OpenShift Container Platform 3.11 30.章、高可用性 を参照してください。

3scale では HA のデータベースコンポーネントに以下が含まれます。

- **backend-redis**: 統計ストレージおよび一時ジョブストレージに使用されます。
- **system-redis**: 3scale のバックグラウンドジョブの一時ストレージを提供し、**system-app** Pod の **Ruby** プロセスのメッセージバスとしても使用されます。

注記

backend-redis と **system-redis** には、どちらもサポートされる Redis Sentinel および Redis Enterprise 用 Redis 高可用性バリアントが含まれます。

8.1. ゼロダウンタイムのための REDIS 設定

ゼロダウンタイムを実現する必要がある場合は、Redis を OCP 外部に設定する必要があります。3scale Pod の設定オプションを使用して設定する方法は複数あります。

- 独自の自己管理型 Redis を設定する
- Redis Sentinel を使用する (Redis Sentinel Documentation を参照)
- サービスとして提供される Redis:
  例:
  - Amazon ElastiCache
  - Redis Labs

注記

Red Hat は上記のサービスにサポートを提供しません。このようなサービスの言及は、Red Hat による製品やサービスの推奨を意味するものではありません。Red Hat は、Red Hat 外部のコンテンツを使用 (または依存) して発生した損失や費用の責任を負いません。

8.2. 3SCALE 用バックエンドコンポーネントの設定

3scale 2.6 では、**back-end** コンポーネントの Redis HA (フェイルオーバー) を設定できます。これらの設定は、**backend-cron**、**backend-listener**、および **backend-worker** のデプロイメント設定で、環境変数として定義することができます。

注記

Redis ではなく Sentinel を使用するには、3scale をデプロイする前に、ポイントする Redis を設定するためにすべてのフィールドを指定して **system-redis** シークレットを作成する必要があります。3scale 2.6 の時点では、フィールドはバックエンドのパラメーターとしては提供されません。

8.2.1. backend-redis と system-redis シークレットの作成

以下の手順に従い、適宜 **backend-redis** および **system-redis** シークレットを作成します。

- 「HA 用 3scale の新規インストールのデプロイ」
- 「3scale の非 HA デプロイメントの HA への移行」

8.2.1.1. HA 用 3scale の新規インストールのデプロイ

1. 以下のフィールドをすべて指定して、**backend-redis** および **system-redis** シークレットを作成します。

   backend-redis

   ```
   REDIS_QUEUES_SENTINEL_HOSTS
   REDIS_QUEUES_SENTINEL_ROLE
   REDIS_QUEUES_URL
   REDIS_STORAGE_SENTINEL_HOSTS
   REDIS_STORAGE_SENTINEL_ROLE
   REDIS_STORAGE_URL
   ```

   system-redis

   ```
   MESSAGE_BUS_NAMESPACE
   MESSAGE_BUS_SENTINEL_HOSTS
   MESSAGE_BUS_SENTINEL_ROLE
   MESSAGE_BUS_URL
   NAMESPACE
   SENTINEL_HOSTS
   SENTINEL_ROLE
   URL
   ```

   - Redis と Sentinel を設定する場合、**backend-redis** および **system-redis** の該当する **URL** フィールドには、**redis://[:redis-password@]redis-group[/db]** のフォーマットで Redis グループを指定します。ここで、[x] はオプションの要素 **x** を意味し、**redis-password**、**redis-group**、および **db** 変数は適切な値に置き換えてください。

     例

     ```
     redis://:redispwd@mymaster/5
     ```

   - **SENTINEL_HOSTS** フィールドは、以下のフォーマットの Sentinel 接続文字列のコンマ区切りリストです。

     ```
     [redis://][:sentinel-password@]sentinel-hostname-or-ip[:port]
     ```

- リスト内の各要素に関して、[x] はオプションの要素 **x** を意味し、**sentinel-password**、**sentinel-hostname-or-ip**、および **port** 変数は適切な値に置き換えてください。

  例

  ```
  :sentinelpwd@123.45.67.009:2711,:sentinelpwd@other-sentinel:2722
  ```

- **SENTINEL_ROLE** フィールドの値は、**master** または **slave** のどちらかです。

2. 最新バージョンのテンプレートを使用して、テンプレートを使用した OpenShift への 3scale 2.6 のデプロイ に記載のとおり 3scale をデプロイします。

   a. **backend-redis** および **system-redis** がすでに存在するために表示されるエラーは無視します。

8.2.1.2. 3scale の非 HA デプロイメントの HA への移行

1. 「HA 用 3scale の新規インストールのデプロイ」 に示すように、すべてのフィールドを使用して **backend-redis** および **system-redis** シークレットを編集します。

2. 以下の **backend-redis** 環境変数がバックエンド Pod に対して定義されていることを確認してください。

```
name: BACKEND_REDIS_SENTINEL_HOSTS
  valueFrom:
    secretKeyRef:
      key: REDIS_STORAGE_SENTINEL_HOSTS
      name: backend-redis
name: BACKEND_REDIS_SENTINEL_ROLE
  valueFrom:
    secretKeyRef:
      key: REDIS_STORAGE_SENTINEL_ROLE
      name: backend-redis
```

3. 以下の **system-redis** の環境変数が **system-(app|sidekiq|sphinx)** Pod に対して定義されていることを確認してください。

```
name: REDIS_SENTINEL_HOSTS
  valueFrom:
    secretKeyRef:
      key: SENTINEL_HOSTS
      name: system-redis
name: REDIS_SENTINEL_ROLE
  valueFrom:
    secretKeyRef:
      key: SENTINEL_ROLE
      name: system-redis
name: MESSAGE_BUS_REDIS_SENTINEL_HOSTS
  valueFrom:
    secretKeyRef:
      key: MESSAGE_BUS_SENTINEL_HOSTS
      name: system-redis
name: MESSAGE_BUS_REDIS_SENTINEL_ROLE
  valueFrom:
```

```
secretKeyRef:
  key: MESSAGE_BUS_SENTINEL_ROLE
  name: system-redis
```

4. 指示に従って、3scale のアップグレード を続行します。

8.2.2. Redis Enterprise の使用

1. 3 つの異なる **redis-enterprise** インスタンスで、OpenShift にデプロイされた Redis Enterprise を使用します。
   a. **system-redis** シークレットを編集します。
      i. 個別の値を **MESSAGE_BUS_NAMESPACE** および **NAMESPACE** に設定します。
      ii. **URL** と **MESSAGE_BUS_URL** を同じデータベースに設定します。
   b. **backend-redis** のバックエンドデータベースを **REDIS_QUEUES_URL** に設定します。
   c. **backend-redis** の 3 番目のデータベースを **REDIS_STORAGE_URL** に設定します。

8.2.3. Redis Sentinel の使用

1. 3 つまたは 4 つの異なる Redis データベースで、Redis Sentinel を使用します。
   a. **system-redis** シークレットを編集します。
      i. 個別の値を **MESSAGE_BUS_NAMESPACE** および **NAMESPACE** に設定します。
      ii. **URL** および **MESSAGE_BUS_URL** を適切な Redis グループに設定します (たとえば、**redis://:redispwd@mymaster/5**)。
      iii. **SENTINEL_HOSTS** および **MESSAGE_BUS_SENTINEL_HOSTS** を、Sentinel ホストとポートのコンマ区切りリストに設定します (たとえば、**:sentinelpwd@123.45.67.009:2711,:sentinelpwd@other-sentinel:2722**)。
      iv. **SENTINEL_ROLE** および **MESSAGE_BUS_SENTINEL_ROLE** を master に設定します。
2. バックエンドの **backend-redis** シークレットを、以下の値に設定します。
   - **REDIS_QUEUES_URL**
   - **REDIS_QUEUES_SENTINEL_ROLE**
   - **REDIS_QUEUES_SENTINEL_HOSTS**
3. 3 番目のデータベースの変数を以下のように設定します。
   - **REDIS_STORAGE_URL**
   - **REDIS_STORAGE_SENTINEL_ROLE**
   - **REDIS_STORAGE_SENTINEL_HOSTS**

注記

- **system-app** および **system-sidekiq** コンポーネントは、統計を取得するために **back-end** Redis に直接接続します。
    - 3scale 2.7 の時点で、これらのシステムコンポーネントは Sentinel を使用する際にも **back-end** Redis (ストレージ) に接続することができます。
- **system-app** および **system-sidekiq** コンポーネントは、**backend-redis** ストレージ **しか使用しません** (**backend-redis** キューは使用しません)。
    - システムコンポーネントに加えた変更は、**backend-redis** ストレージと Sentinel の組み合わせをサポートします。

第9章 外部 MYSQL データベースの設定

本章では、*7章3scale 高可用性テンプレートおよび評価用テンプレート* の MySQL データベースを外部化する方法について説明します。そのためには、デフォルトの **amp.yml** ファイルを使用します。これは、デフォルトの **system-mysql** Pod を使用するとネットワークやファイルシステムなど複数のインフラストラクチャーの問題が生じる場合に役立ちます。

本章のアプローチと *7章3scale 高可用性テンプレートおよび評価用テンプレート* のアプローチの違いは、本アプローチでは、Red Hat 3scale API Management が最初にデフォルトの **amp.yml** テンプレートを使用していた場合に、MySQL データベースを外部化することができる点です。

9.1. 外部 MYSQL データベースに関する制約

MySQL データベースを外部化するプロセスの制約は以下のとおりです。

オンプレミス型 3scale のバージョン

オンプレミス型 3scale のバージョン 2.5 および 2.6 のみテストおよび検証済みです。

MySQL データベースユーザー

mysql2:// 形式の URL で、**'root'@'%'** を使用する必要があります。そうでないと、データベースへの接続に失敗します。3scale では **'root'@'%'** が使用されるため、**ユーザー名** と **パスワード** の組み合わせは、一切サポートされません。

MySQL ホスト

ホスト名 ではなく外部 MySQL データベースの **IP アドレス** を使用します。そうでない場合は、解決されません。たとえば、**mysql.mydomain.com** ではなく **1.1.1.1** を使用します。

システムデータベース

リモート MySQL サーバーに、現在存在する **system** という名前のデータベースを含めることはできません。

9.2. MYSQL データベースの外部化

MySQL データベースを完全に外部化するには、以下の手順を使用します。

前提条件

- 管理者権限を持つアカウントを使用して OpenShift Container Platform 3.11 クラスターにアクセスできること。
- OpenShift クラスター上にインストールされた 3scale インスタンス。*2章3scale を OpenShift にインストールするためのガイド*を参照してください。

警告

この操作により、プロセスの進行中に環境でダウンタイムが発生します。

手順

1. オンプレミス型 3scale インスタンスをホストする OpenShift ノードにログインし、そのプロジェクトに切り替えます。

```
oc login -u <user> <url>
oc project <3scale-project>
```

<user>、**<url>**、および **<3scale-project>** を、実際のクレデンシャルとプロジェクト名に置き換えます。

2. 以下に示す順序で手順実施し、すべての Pod をスケールダウンします。これにより、データの喪失が回避されます。

オンプレミス型 3scale の停止

OpenShift Web コンソールまたはコマンドラインインターフェイス (CLI) から、すべてのデプロイメント設定を以下の順序でゼロレプリカにスケールダウンします。

- 3scale 2.6 より前のバージョンの場合は **apicast-wildcard-router** と **zync**、3scale 2.6 以降の場合は **zync-que** と **zync**
- **apicast-staging** と **apicast-production**
- **system-sidekiq**、**backend-cron**、および **system-sphinx**
  - 3scale 2.3 の場合には **system-resque** を対象に含めます。
- **system-app**
- **backend-listener** と **backend-worker**
- **backend-redis**、**system-memcache**、**system-mysql**、**system-redis**、および **zync-database**
  以下の例は、**apicast-wildcard-router** と **zync** について、CLI でこの操作を実施する方法を示しています。

```
oc scale dc/apicast-wildcard-router --replicas=0
oc scale dc/zync --replicas=0
```

注記

各ステップのデプロイメント設定は同時にスケールダウンできます。たとえば、**apicast-wildcard-router** と **zync** を一緒にスケールダウンできます。ただし、各ステップの Pod が終了するのを待ってから、次の Pod をスケールダウンすることをお勧めします。3scale インスタンスは、完全に再起動されるまで一切アクセスできなくなります。

3. 3scale プロジェクトで実行中の Pod がないことを確認するには、以下のコマンドを使用します。

```
oc get pod
```

このコマンドは、**No resources found** を返すはずです。

4. 以下のコマンドを使用して、データベースレベルの Pod を再度スケールアップします。

```
oc scale dc/{backend-redis,system-memcache,system-mysql,system-redis,zync-database} --replicas=1
```

5. 次のステップに進む前に、**system-mysql** Pod を通じて外部 MySQL データベースにログインできることを確認してください。

```
oc rsh system-mysql-<system_mysql_pod_id>
mysql -u root -p -h <host>
```

- **<system_mysql_pod_id>**: system-mysql Pod の識別子
- ユーザーには必ず root を使用する。詳細は、「外部 MySQL データベースに関する制約」を参照してください。

    a. CLI に **mysql>** が表示されるようになります。**exit** と入力してから **enter** キーを押します。次のプロンプトで再度 **exit** と入力して、OpenShift ノードのコンソールに戻ります。

6. 以下のコマンドを使用して、MySQL のフルダンプを実行します。

```
oc rsh system-mysql-<system_mysql_pod_id> /bin/bash -c "mysqldump -u root --single-transaction --routines --triggers --all-databases" > system-mysql-dump.sql
```

- **<system_mysql_pod_id>** を一意の **system-mysql** Pod **ID** に置き換えます。
- 次の例のように、ファイル **system-mysql-dump.sql** に有効な MySQL レベルのダンプが含まれていることを確認します。

```
$ head -n 10 system-mysql-dump.sql
-- MySQL dump 10.13  Distrib 5.7.24, for Linux (x86_64)
--
-- Host: localhost    Database:
-- ------------------------------------------------------
-- Server version   5.7.24

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET
@OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION
*/;
/*!40101 SET NAMES utf8 */;
```

7. **system-mysql** Pod をスケールダウンし、レプリカが 0 (ゼロ) のままにします。

```
oc scale dc/system-mysql --replicas=0
```

8. <password> と <host> を適宜置き換え、URL **mysql2://root:<password>@<host>/system** の **base64** 変換値を取得します。

```
echo "mysql2://root:<password>@<host>/system" | base64
```

9. リモート MySQL データベースのデフォルトの **'user'@'%'** を作成します。これには SELECT 権限しか付与する必要はありません。また、その **base64** 変換値を取得します。

```
echo "user" | base64
echo "<password>" | base64
```

- <password> を **'user'@'%'** のパスワードに置き換えます。

10. バックアップを実行し、OpenShift シークレット **system-database** を編集します。

```
oc get secret system-database -o yaml > system-database-orig.bkp.yml
oc edit secret system-database
```

- **URL**: これを [step-8] で取得した値に置き換えます。
- **DB_USER** および **DB_PASSWORD**: 共に前のステップの値を使用します。

11. 以下のコマンドを使用して **system-mysql-dump.sql** をリモートデータベースサーバーに送信し、ダンプをサーバーにインポートします。

```
mysql -u root -p < system-mysql-dump.sql
```

12. **system** という新しいデータベースが作成されたことを確認します。

```
mysql -u root -p -se "SHOW DATABASES"
```

13. 以下の手順を使用して、**オンプレミス型 3scale を起動します**。これにより、すべての Pod が正しい順序でスケールアップされます。

オンプレミス型 3scale の起動

- **backend-redis**、**system-memcache**、**system-mysql**、**system-redis**、および **zync-database**
- **backend-listener** と **backend-worker**
- **system-app**
- **system-sidekiq**、**backend-cron**、および **system-sphinx**
  - 3scale 2.3 の場合には **system-resque** を対象に含めます。
- **apicast-staging** と **apicast-production**
- 3scale 2.6 より前のバージョンの場合は **apicast-wildcard-router** と **zync**、3scale 2.6 以降の場合は **zync-que** と **zync**
  以下の例は、**backend-redis**、**system-memcache**、**system-mysql**、**system-redis**、および **zync-database** について、CLI でこの操作を実行する方法を示しています。

```
oc scale dc/backend-redis --replicas=1
oc scale dc/system-memcache --replicas=1
oc scale dc/system-mysql --replicas=1
oc scale dc/system-redis --replicas=1
oc scale dc/zync-database --replicas=1
```

system-app Pod が問題なく起動し、実行されるはずです。

14. 確認後、上記の順序 で他の Pod をスケールアップして元の状態に戻します。

15. **system-mysql** DeploymentConfig オブジェクトのバックアップを作成します。数日後、すべて正常に動作していることが確認できたら、削除してかまいません。**system-mysql** DeploymentConfig を削除することで、この手順を今後再び実行する場合の混乱を防ぐことができます。

9.3. ロールバック

ステップ 13 を実施した後、**system-app** Pod が完全には動作状態に戻らず、その根本的な原因が判断できない、または対処できない場合、ロールバックの手順を実施します。

1. **system-database-orig.bkp.yml** の元の値を使用して、シークレット **system-database** を編集します。[step-10] を参照してください。

```
oc edit secret system-database
```

URL、**DB_USER**、および **DB_PASSWORD** を元の値に置き換えます。

2. **system-mysql** を含め、すべての Pod をスケールダウンしてから、再度スケールアップして元の状態に戻します。**system-app** Pod およびその後に起動されるその他の Pod が、再び起動して実行されるはずです。以下のコマンドを実行して、すべての Pod が元どおりに起動、実行されていることを確認します。

```
oc get pods -n <3scale-project>
```