Red Hat 3Scale 2-saas

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: 2022-12-11

Red Hat 3Scale 2-saas 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。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、CTO である Chris Wright のメッセージ をご覧ください。

第1章 3SCALE 用レジストリーサービスアカウント

3scale 2-saas 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。

レジストリーサービスアカウントを作成および変更するには、以下のセクションに概略を示す手順を実施します。

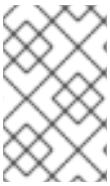
- レジストリーサービスアカウントの作成
- コンテナーレジストリー認証の設定
- レジストリーサービスアカウントの変更

1.1. レジストリーサービスアカウントの作成

レジストリーサービスアカウントを作成するには、以下の手順に従います。

手順

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。
   b. **Description** を入力します。
   c. **Create** をクリックします。
4. **Registry Service Accounts**のページに戻ります。
5. 作成した **サービスアカウント** をクリックします。
6. 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。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。

1.2. コンテナーレジストリー認証の設定

3scale 管理者は、3scale を OpenShift にデプロイする前に、**registry.redhat.io** との認証を設定します。

前提条件

- 管理者クレデンシャルのある Red Hat OpenShift Container Platform (OCP) アカウント。
- OpenShift **oc** クライアントツールがインストール済みである。詳細は、OpenShift CLI のドキュメント を参照してください。

手順

1. 管理者として OpenShift クラスターにログインします。

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

2. 3scale をデプロイするプロジェクトを開きます。

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

3. Red Hat カスタマーポータルアカウントを使用して **docker-registry** シークレットを作成します。**threescale-registry-auth** は作成するシークレットに置き換えます。

```
$ oc create secret docker-registry threescale-registry-auth \
  --docker-server=registry.redhat.io \
  --docker-username="customer_portal_username" \
  --docker-password="customer_portal_password" \
  --docker-email="email_address"
```

以下の出力が表示されるはずです。

```
secret/threescale-registry-auth created
```

4. シークレットをサービスアカウントにリンクして、シークレットをイメージをプルするために使用します。サービスアカウント名は、OpenShift Pod が使用する名前と一致する必要があります。以下は、**default** サービスアカウントを使用する例になります。

```
$ oc secrets link default threescale-registry-auth --for=pull
```

5. シークレットを **builder** サービスアカウントにリンクし、ビルドイメージをプッシュおよびプルするためにシークレットを使用します。

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

関連情報

コンテナーイメージに対する Red Hat の認証に関する詳細は、以下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
- Red Hat registry service accounts

1.3. レジストリーサービスアカウントの変更

Registry Service Account ページからサービスアカウントを編集または削除することができます。そのためには、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して、**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。

- **トークンの生成:** 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。
- **説明の更新:** 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。
- **アカウントの削除:** 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。

1.4. 関連情報

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

第2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 Red Hat 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。

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。

- ホスト型 3scale のアカウント

APIcast をインストールするには、以下のセクションに概略を示す手順を実施します。

- 「APIcast デプロイメントのオプション」
- 「APIcast の環境」
- 「インテグレーション設定」
- 「サービスの設定」
- 「APIcast operator のインストール」
- 「operator を使用した Self-managed APIcast ゲートウェイソリューションのデプロイ」

2.1. APICAST 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。

- Hosted APIcast:3scale がクラウド内に APIcast をホストします。この場合、APIcast はすでにデプロイ済みで、1 日あたり 50,000 の呼び出しに制限されます。
- **Self-managed APIcast**:APIcast を任意の場所にデプロイできます。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 コンテナー環境への APIcast のデプロイ :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。
  - Red Hat OpenShift 上での APIcast の実行 APIcast を サポート対象バージョン の OpenShift で実行します。Self-managed APIcast は、オンプレミス型 3scale インストール環境またはホスト型 3scale (SaaS) アカウントに接続できます。

2.2. APICAST の環境

デフォルトでは、3scale アカウントを作成する、または新規 API サービスを作成すると、2 つの異なる環境に **Hosted** APIcast が提供されます。

- **ステージング環境**: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。

- **実稼働**:1 日あたり呼び出しが 50,000 回に制限され、以下の追加設定なしの認証オプションがサポートされます。API キー、およびアプリケーション ID とアプリケーションキーペアの OpenID Connect

Self-managed のデプロイメントを使用する場合、同様に 2 つの環境があり、それぞれに APIcast インスタンスをデプロイする必要があります。環境変数 **THREESCALE_DEPLOYMENT_ENV** の値を **staging** or **production** に設定して、APIcast インスタンスが使用する設定 (ステージングまたは実稼働) を指定することができます。

2.3. インテグレーション設定

3scale 管理者は、3scale を実行する必要のある環境のインテグレーション設定を行います。

前提条件

管理者権限が設定された 3scale アカウント

手順

1. **[Your_API_name] > Integration > Settings**の順に移動します。
2. **Deployment** セクションでは、デフォルトのオプションは以下のとおりです。
   - デプロイメントオプション: Hosted APIcast
   - 認証モード:API キー
3. 希望するオプションに変更します。
4. 変更を保存するには、**Update Product** をクリックします。

2.4. サービスの設定

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。

本セクションでは、サービスの設定について各手順を説明します。

- API バックエンドの宣言
- 認証の設定
- API テストコールの設定

2.4.1. API バックエンドの宣言

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 がホストする **Echo API** を使用します。これは、任意のパスを受け入れ、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。

手順

- 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"
  }
```

2.4.2. 認証の設定

API の認証設定は **[Your_product_name] > Integration > Settings**の **AUTHENTICATION** セクションで行うことができます。

表2.1任意の認証フィールド

フィールド	説明
Auth user key	Credentials location に関連付けられたキーを設定します。
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。
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。

さらに [Your_product_name] > Integration > Settingsの順に移動し、GATEWAY RESPONSE エラーコードを設定できます。エラーの **レスポンスコード**、**コンテンツタイプ**、 および **レスポンスボディー** を定義します。Authentication failed、Authentication missing、および No match。

表2.2 レスポンスコードとデフォルトのレスポンスボディー

レスポンスコード	レスポンスのボディー
403	Authentication failed
403	Authentication parameters missing
404	No Mapping Rule matched
429	Usage limit exceeded

2.4.3. API テストコールの設定

API の設定では、リクエストコールを元にテストを行うために、プロダクトを含めたバックエンドのテストを行い、APIcast の設定をステージング環境および実稼働環境にプロモートする必要があります。

それぞれのプロダクトについて、リクエストはパスに従って対応するバックエンドにリダイレクトされます。このパスは、バックエンドをプロダクトに追加する際に設定されます。たとえば、プロダクトに 2 つのバックエンドを追加している場合、それぞれのバックエンドは固有のパスを持ちます。

前提条件

- プロダクトに追加された 1 つまたは複数の バックエンド
- プロダクトに追加された各バックエンドの マッピングルール
- アクセスポリシーを定義するための アプリケーションプラン
- アプリケーションプランを参照する アプリケーション

手順

1. [Your_product_name] > Integration > Configurationの順に移動して、APIcast 設定をステージング環境にプロモートします。
2. **APIcast Configuration** セクションに、プロダクトに追加された各バックエンドのマッピングルールが表示されます。**Promote v.[n] to Staging APIcast**をクリックします。
   - **v.[n]** は、プロモート先のバージョン番号を表します。
3. ステージング環境にプロモートしたら、実稼働環境にプロモートすることができます。**Staging APIcast** セクションで、**Promote v.[n] to Production APIcast**をクリックします。
   - **v.[n]** は、プロモート先のバージョン番号を表します。
4. コマンドラインで API へのリクエストをテストするには、**Example curl for testing**で提供されるコマンドを使用します。
   - curl コマンドの例は、プロダクトの最初のマッピングルールに基づいています。

API へのリクエストをテストする際に、メソッドおよびメトリクスを追加して マッピングルールを変更することができます。

設定を変更したら、API への呼び出しを行う前に、必ずステージング環境および実稼働環境にプロモートするようにしてください。ステージング環境にプロモートする保留中の変更がある場合には、管理ポータルの **Integration** メニュー項目の横に感嘆符が表示されます。

3scale Hosted APIcast ゲートウェイはクレデンシャルを検証し、API 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**Authentication failed**が表示されます。

2.5. APICAST OPERATOR のインストール

本セクションでは、OpenShift Container Platform (OCP) コンソールから APIcast operator をインストールする手順について説明します。

手順

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. **Projects** > **Create Project**の順に移動し、新規プロジェクト **operator-test** を作成します。
3. **Operators** > **Installed Operators**の順にクリックします。
4. **Filter by keyword** ボックスに **apicast** と入力し、APIcast operator を検索します。コミュニティーバージョンは使用しないでください。
5. APIcast operator をクリックします。APIcast operator に関する情報が表示されます。
6. **Install** をクリックします。**Create Operator Subscription**ページが表示されます。
7. **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。
    a. サブスクリプションのアップグレードステータスが **Up to date** と表示されます。
8. **Operators** > **Installed Operators**の順にクリックし、**operator-test** プロジェクトで APIcast operator の **ClusterServiceVersion** (CSV) ステータスが最終的に **InstallSucceeded** と表示されることを確認します。

2.6. OPERATOR を使用した SELF-MANAGED APICAST ゲートウェイソリューションのデプロイ

本セクションでは、Openshift Container Platform コンソールから APIcast operator を使用して Self-managed APIcast ゲートウェイソリューションをデプロイする手順について説明します。

前提条件

- OpenShift Container Platform (OCP) 4 以降およびその管理者権限
- APIcast operator のインストール の手順に従う。

手順

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. **Operators > Installed Operators**の順にクリックします。
3. **Installed Operators** のリストで **APIcast Operator** をクリックします。
4. **APIcast > Create APIcast**の順にクリックします。

2.6.1. APIcast のデプロイメントおよび設定オプション

Self-managed APIcast ゲートウェイソリューションは、以下に示す 2 とおりの方法を使用してデプロイおよび設定することができます。

- 3scale システムエンドポイントの指定
- 設定シークレットの指定

以下も参照してください。

- APIcast Operator を使用したカスタム環境の注入
- APIcast operator を使用したカスタムポリシーの注入
- APIcast operator を使用した OpenTracing の設定

2.6.1.1. 3scale システムエンドポイントの指定

手順

1. 3scale システム管理ポータルのエンドポイント情報が含まれる OpenShift シークレットを作成します。

```
oc create secret generic ${SOME_SECRET_NAME} --from-literal=AdminPortalURL=${MY_3SCALE_URL}
```

   - **${SOME_SECRET_NAME}** はシークレットの名前で、既存のシークレットと競合しない限り、任意の名前を付けることができます。
   - **${MY_3SCALE_URL}** は、3scale アクセストークンおよび 3scale システム管理ポータルのエンドポイントが含まれる URI です。詳細は、**THREESCALE_PORTAL_ENDPOINT** を参照してください。

     例

```
oc create secret generic 3scaleportal --from-literal=AdminPortalURL=https://access-token@account-admin.3scale.net
```

     シークレットの内容についての詳細は、APIcast Custom Resource reference のEmbeddedConfSecret を参照してください。

2. APIcast の OpenShift オブジェクトを作成します。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: example-apicast
```

```
spec:
  adminPortalCredentialsRef:
    name: SOME_SECRET_NAME
```

spec.adminPortalCredentialsRef.name は、3scale システム管理ポータルのエンドポイント情報が含まれる既存の OpenShift シークレットの名前でなければなりません。

3. APIcast オブジェクトに関連付けられた OpenShift デプロイメントの **readyReplicas** フィールドが **1** であることを確認し、APIcast Pod が動作状態にあり準備が整っていることを確認します。そうでなければ、以下のコマンドを使用してフィールドが設定されるまで待ちます。

```
$ echo $(oc get deployment apicast-example-apicast -o jsonpath='{.status.readyReplicas}')
1
```

2.6.1.1.1. APIcast ゲートウェイが動作中で利用可能であることの確認

手順

1. ローカルマシンから OpenShift Service APIcast にアクセス可能であることを確認し、テストリクエストを実行します。そのために、APIcast OpenShift Service を **localhost:8080** にポート転送します。

```
oc port-forward svc/apicast-example-apicast 8080
```

2. 設定した 3scale サービスに対してリクエストを行い、HTTP 応答が正常であることを確認します。サービスの **Staging Public Base URL** または **Production Public Base URL** 設定で指定したドメイン名を使用します。以下に例を示します。

```
$ curl 127.0.0.1:8080/test -H "Host: myhost.com"
```

2.6.1.1.2. Kubernetes Ingress 経由での APIcast の外部公開

Kubernetes Ingress 経由で APIcast を外部に公開するには、**exposedHost** セクションを設定します。**ExposedHost** セクションの **host** フィールドを設定すると、Kubernetes Ingress オブジェクトが作成されます。事前にインストールした既存の Kubernetes Ingress Controller はこの Kubernetes Ingress オブジェクトを使用し、APIcast を外部からアクセス可能にします。

APIcast を外部からアクセス可能にするのに使用できる Ingress Controllers およびその設定方法について詳しく知るには、Kubernetes Ingress Controllers のドキュメント を参照してください。

ホスト名 **myhostname.com** で APIcast を公開する例を以下に示します。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: example-apicast
spec:
  ...
  exposedHost:
    host: "myhostname.com"
  ...
```

この例では、HTTP 用ポート 80 に Kubernetes Ingress オブジェクトを作成します。APIcast デプロイメントが OpenShift 環境にある場合、OpenShift のデフォルト Ingress Controller は APIcast が作成す

る Ingress オブジェクト使用して Route オブジェクトを作成し、APIcast インストールへの外部アクセスが可能になります。

exposedHost セクションに TLS を設定することもできます。利用可能なフィールドの詳細を以下の表に示します。

表2.3 APIcastExposedHost の参照テーブル

json/yaml フィールド	タイプ	必須/任意	デフォルト値	説明
host	string	はい	該当なし	ゲートウェイにルーティングされているドメイン名
tls	[]extensions.IngressTLS	いいえ	該当なし	受信 TLS オブジェクトの配列。詳細は、TLS を参照してください。

2.6.1.2. 設定シークレットの指定

手順

1. 設定ファイルを使用してシークレットを作成します。

```
$ curl
https://raw.githubusercontent.com/3scale/APIcast/master/examples/configuration/echo.json -
o $PWD/config.json

oc create secret generic apicast-echo-api-conf-secret --from-file=$PWD/config.json
```

設定ファイルは **config.json** という名前にする必要があります。これは APIcast CRD の要件です。

シークレットの内容についての詳細は、APIcast Custom Resource reference のEmbeddedConfSecret を参照してください。

2. APIcast カスタムリソース を作成します。

```
$ cat my-echo-apicast.yaml
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: my-echo-apicast
spec:
  exposedHost:
    host: YOUR DOMAIN
  embeddedConfigurationSecretRef:
    name: apicast-echo-api-conf-secret

$ oc apply -f my-echo-apicast.yaml
```

a. 埋め込み設定シークレットの例を以下に示します。

```
apiVersion: v1
kind: Secret
metadata:
  name: SOME_SECRET_NAME
type: Opaque
stringData:
  config.json: |
    {
      "services": [
       {
         "proxy": {
          "policy_chain": [
           { "name": "apicast.policy.upstream",
             "configuration": {
               "rules": [{
                 "regex": "/",
                 "url": "http://echo-api.3scale.net"
               }]
             }
           }
          ]
         }
       }
      ]
    }
```

3. APIcast オブジェクトの作成時に以下の内容を設定します。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: example-apicast
spec:
  embeddedConfigurationSecretRef:
    name: SOME_SECRET_NAME
```

spec.embeddedConfigurationSecretRef.name は、ゲートウェイの設定が含まれる既存の OpenShift シークレットの名前でなければなりません。

4. APIcast オブジェクトに関連付けられた OpenShift デプロイメントの **readyReplicas** フィールドが **1** であることを確認し、APIcast Pod が動作状態にあり準備が整っていることを確認します。そうでなければ、以下のコマンドを使用してフィールドが設定されるまで待ちます。

```
$ echo $(oc get deployment apicast-example-apicast -o jsonpath='{.status.readyReplicas}')
1
```

2.6.1.2.1. APIcast ゲートウェイが動作中で利用可能であることの確認

手順

1. ローカルマシンから OpenShift Service APIcast にアクセス可能であることを確認し、テストリクエストを実行します。そのために、APIcast OpenShift Service を **localhost:8080** にポート転送します。

```
oc port-forward svc/apicast-example-apicast 8080
```

2. 設定した 3scale サービスに対してリクエストを行い、HTTP 応答が正常であることを確認します。サービスの **Staging Public Base URL** または **Production Public Base URL** 設定で指定したドメイン名を使用します。以下に例を示します。

```
$  curl 127.0.0.1:8080/test -H "Host: localhost"
{
  "method": "GET",
  "path": "/test",
  "args": "",
  "body": "",
  "headers": {
    "HTTP_VERSION": "HTTP/1.1",
    "HTTP_HOST": "echo-api.3scale.net",
    "HTTP_ACCEPT": "*/*",
    "HTTP_USER_AGENT": "curl/7.65.3",
    "HTTP_X_REAL_IP": "127.0.0.1",
    "HTTP_X_FORWARDED_FOR": ...
    "HTTP_X_FORWARDED_HOST": "echo-api.3scale.net",
    "HTTP_X_FORWARDED_PORT": "80",
    "HTTP_X_FORWARDED_PROTO": "http",
    "HTTP_FORWARDED": "for=10.0.101.216;host=echo-api.3scale.net;proto=http"
  },
  "uuid": "603ba118-8f2e-4991-98c0-a9edd061f0f0"
```

2.6.1.3. APIcast Operator を使用したカスタム環境の注入

Self-managed APIcast を使用する 3scale インストールでは、**3scale** Operator を使用してカスタム環境を注入できます。カスタム環境は、ゲートウェイが提供するすべてのアップストリーム API に APIcast が適用する動作を定義します。カスタム環境を作成するには、Lua コードでグローバル設定を定義します。

APIcast のインストールの一部として、またはインストール後にカスタム環境を注入できます。カスタム環境を注入した後、その環境を削除すると、**APIcast** Operator が変更を調整します。

前提条件

- APIcast Operator がインストールされている。

手順

1. 注入するカスタム環境を定義する Lua コードを記述します。たとえば、次の **env1.lua** ファイルは、**APIcast** Operator がすべてのサービスに対してロードするカスタムログポリシーを示しています。

```
local cjson = require('cjson')
local PolicyChain = require('apicast.policy_chain')
local policy_chain = context.policy_chain

local logging_policy_config = cjson.decode([[
```

```
{
  "enable_access_logs": false,
  "custom_logging": "\"{{request}}\" to service {{service.id}} and {{service.name}}"
}
]])

policy_chain:insert( PolicyChain.load_policy('logging', 'builtin', logging_policy_config), 1)

return {
  policy_chain = policy_chain,
  port = { metrics = 9421 },
}
```

2. カスタム環境を定義する Lua ファイルからシークレットを作成します。以下に例を示します。

```
oc create secret generic custom-env-1 --from-file=./env1.lua
```

シークレットには複数のカスタム環境を含めることができます。カスタム環境を定義する各ファイルの **-from-file** オプションを指定します。Operator は各カスタム環境をロードします。

3. 作成したシークレットを参照する **APIcast** カスタムリソースを定義します。以下の例は、カスタム環境を定義するシークレットの参照に関連するコンテンツのみを示しています。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: apicast1
spec:
  customEnvironments:
    - secretRef:
        name: custom-env-1
```

APIcast カスタムリソースは、カスタム環境を定義する複数のシークレットを参照できます。Operator は各カスタム環境をロードします。

4. カスタム環境を追加する **APIcast** カスタムリソースを作成します。たとえば、**APIcast** カスタムリソースを **apicast.yaml** ファイルに保存した場合は、以下のコマンドを実行します。

```
oc apply -f apicast.yaml
```

次のステップ

カスタム環境を更新する場合は、シークレットに更新が含まれるように、必ずそのシークレットを再作成します。**APIcast** operator は更新の有無を監視し、更新を発見すると自動的に再デプロイします。

2.6.1.4. APIcast operator を使用したカスタムポリシーの注入

Self-managed APIcast を使用する 3scale インストールでは、**APIcast** operator を使用してカスタムポリシーを注入できます。カスタムポリシーを注入すると、ポリシーコードが APIcast に追加されます。次に、以下のいずれかを使用して、カスタムポリシーを API 製品のポリシーチェーンに追加できます。

- 3scale API
- **Product** カスタムリソース

3scale 管理ポータルを使用してカスタムポリシーを製品のポリシーチェーンに追加するには、カスタムポリシーのスキーマを **CustomPolicyDefinition** カスタムリソースに登録する必要もあります。カスタムポリシー登録は、管理ポータルを使用して製品のポリシーチェーンを設定する場合にのみ必要です。

APIcast のインストールの一部として、またはインストール後にカスタムポリシーを注入できます。カスタムポリシーを注入した後、それを削除すると、**APIcast** operator が変更を調整します。

前提条件

- APIcast operator がインストールされているか、インストール中である。
- Write your own policy で説明されているように、カスタムポリシーを定義している。つまり、たとえば、カスタムポリシーを定義する**my-first-custom-policy.lua**、**apicast-policy.json**、および**init.lua**ファイルをすでに作成している。

手順

1. 1 つのカスタムポリシーを定義するファイルからシークレットを作成します。以下に例を示します。

```
oc create secret generic my-first-custom-policy-secret \
 --from-file=./apicast-policy.json \
 --from-file=./init.lua \
 --from-file=./my-first-custom-policy.lua
```

   複数のカスタムポリシーがある場合は、カスタムポリシーごとにシークレットを作成します。シークレットには、カスタムポリシーを 1 つだけ含めることができます。

2. 作成したシークレットを参照する **APIcast** カスタムリソースを定義します。以下の例は、カスタムポリシーを定義するシークレットの参照に関連するコンテンツのみを示しています。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: apicast1
spec:
  customPolicies:
    - name: my-first-custom-policy
      version: "0.1"
      secretRef:
        name: my-first-custom-policy-secret
```

   APIcast カスタムリソースは、カスタムポリシーを定義する複数のシークレットを参照できます。Operator は各カスタムポリシーをロードします。

3. カスタムポリシーを追加する**APIcast**カスタムリソースを作成します。たとえば、**APIcast** カスタムリソースを **apicast.yaml** ファイルに保存した場合は、以下のコマンドを実行します。

```
oc apply -f apicast.yaml
```

次のステップ

カスタムポリシーを更新する場合は、シークレットに更新が含まれるように、必ずそのシークレットを再作成します。**APIcast** operator は更新の有無を監視し、更新を発見すると自動的に再デプロイします。

関連情報

- APIcast custom resource definition

2.6.1.5. APIcast operator を使用した OpenTracing の設定

Self-managed APIcast を使用する 3scale のインストールでは、**APIcast** operator を使用して OpenTracing を設定できます。OpenTracing を有効にすると、APIcast インスタンスに関してより多くの洞察を得て、可観測性を向上できます。

前提条件

- **APIcast** operator がインストールされているか、インストール中である。
- OpenTracing を使用するための APIcast の設定 に記載の前提条件。
- Jaeger がインストールされている。

手順

1. **stringData.config** に OpenTracing 設定の詳細を含めて、シークレットを定義します。これは、OpenTracing 設定の詳細が含まれる属性の唯一有効な値です。その他の仕様では、APIcast が OpenTracing 設定の詳細を受け取れないようにします。以下の例は、有効なシークレット定義を示しています。

```
apiVersion: v1
kind: Secret
metadata:
  name: myjaeger
stringData:
  config: |-
     {
     "service_name": "apicast",
     "disabled": false,
     "sampler": {
       "type": "const",
       "param": 1
     },
     "reporter": {
       "queueSize": 100,
       "bufferFlushInterval": 10,
       "logSpans": false,
       "localAgentHostPort": "jaeger-all-in-one-inmemory-agent:6831"
     },
     "headers": {
       "jaegerDebugHeader": "debug-id",
       "jaegerBaggageHeader": "baggage",
       "TraceContextHeaderName": "uber-trace-id",
       "traceBaggageHeaderPrefix": "testctx-"
     },
     "baggage_restrictions": {
        "denyBaggageOnInitializationFailure": false,
        "hostPort": "127.0.0.1:5778",
        "refreshInterval": 60
```

```
    }
    }
type: Opaque
```

2. シークレットを作成します。たとえば、以前のシークレット定義を **myjaeger.yaml** ファイルに保存した場合は、以下のコマンドを実行します。

```
oc create secret generic myjaeger --from-file myjaeger.yaml
```

3. **OpenTracing** 属性を指定する **APIcast** カスタムリソースを定義します。CR 定義で、**spec.tracingConfigSecretRef.name** 属性を OpenTracing 設定の詳細が含まれるシークレットの名前に設定します。以下の例は、OpenTracing の設定に関するコンテンツのみを示しています。

```
apiVersion: apps.3scale.net/v1alpha1
kind: APIcast
metadata:
  name: apicast1
spec:
  ...
  openTracing:
    enabled: true
    tracingConfigSecretRef:
      name: myjaeger
    tracingLibrary: jaeger
...
```

4. OpenTracing を設定する **APIcast** カスタムリソースを作成します。たとえば、**APIcast** カスタムリソースを **apicast1.yaml** ファイルに保存した場合は、以下のコマンドを実行するはずです。

```
oc apply -f apicast1.yaml
```

次のステップ

OpenTracing のインストール方法に応じて、Jaeger サービスユーザーインターフェイスでトレースが表示されるはずです。

関連情報

- APIcast custom resource definition

2.7. 関連情報

APIcast の最新リリースとサポート対象バージョンについては、以下のアーティクルを参照してください。

- Red Hat 3scale API Management のサポート対象設定
- Red Hat 3scale API Management - Component Details
- Hosted APIcast バージョンに関する更新については、SaaS 版 Red Hat 3scale API Management リリースノート を参照してください。

第3章 HOSTED APICAST

本チュートリアルでは、ご自分の API をクラウド内のセキュアなゲートウェイで完全に保護する方法について説明します。

API をできるだけ早く公開したい場合や、お客様側のインフラストラクチャー変更を最小限に留めたい場合には、Hosted APIcast が最適なデプロイメントオプションです。

前提条件

- 3章*Hosted APIcast*でデプロイメントの選択肢を確認し、API を 3scale と統合するのに Hosted APIcast を使用する判断を下している。
- 一般のインターネットを通じて API バックエンドサービスにアクセス可能である。ユーザーがアクセス制御ゲートウェイをバイパスするのを防ぐために、セキュアな通信が確立される。
- API の需要が 1 日あたり 50,000 ヒットの制限を超えないと予想される。これを超える場合には、自己管理型のゲートウェイにアップグレードすることを推奨します。
- 「API と Hosted APIcast の組み合わせのステージング環境へのデプロイ」
- 「API と Hosted APIcast の組み合わせの実稼働環境へのデプロイ」

3.1. API と HOSTED APICAST の組み合わせのステージング環境へのデプロイ

最初のステップは、API を設定してステージング環境でテストすることです。

手順

1. プライベートベース URL およびそのエンドポイントを定義します。
2. クレデンシャルの配置場所およびその他の設定詳細を選択します。詳細は、Hosted APIcast に関するドキュメント を参照してください。
3. **Update Product** をクリックして設定を保存します。これらの設定で、APIcast ステージングインスタンスを通じて API にテストコールが実行されます。

設定が完了すると緑色の確認メッセージが表示されます。

次のステップに進む前に、必ずバックエンドサービスが検証するシークレットトークンを設定してください。**Authentication Settings** セクションでシークレットトークンの値を定義することができます。これにより、誰も APIcast によるアクセス制御をバイパスできなくなります。

3.2. API と HOSTED APICAST の組み合わせの実稼働環境へのデプロイ

この時点で、API 設定を実稼働環境に移行する準備が整っています。3scale Hosted APIcast インスタンスをデプロイするには、以下の手順を実施します。

手順

1. **Integration and Configuration** ページに戻り、**Promote v.x to Production**ボタンをクリックします。

2. 今後ステージング環境に加えた変更を実稼働環境にプロモートするには、このステップを繰り返します。

設定がすべてのクラウド APIcast インスタンスにデプロイおよび反映されるまでに、5 分から 7 分かかります。再デプロイメント中、API にダウンタイムは発生しません。呼び出しに対応するインスタンスによって、API コールは異なるレスポンスを返す場合があります。実稼働環境の周りのボックスが緑色に変わっていれば、デプロイメントに成功したことを意味します。

ステージング環境および実稼働環境用 APIcast インスタンスのベース URL はいずれも、**apicast.io** ドメインにあります。ステージング環境の URL にはステージングのサブドメインが付くため、これらを区別することができます。以下に例を示します。

- ステージング環境: https://api-2445581448324.staging.apicast.io:443
- 実稼働環境: https://api-2445581448324.apicast.io:443

3.3. 補足情報

- 実稼働環境の APIcast クラウドインスタンスを通じた API で最大限許容されるヒット数は、1 日当たり 50,000 です。管理ポータルの Analytics セクションで、API の使用状況を確認することができます。
- API トラフィックのあらゆるスパイクに対して、1 秒あたり 20 ヒットのハードなスロットル制限が適用されます。
- スロットル制限を超えると、APIcast は **403** のレスポンスコードを返します。これは、流量制御を超えたアプリケーションのデフォルトと同じです。エラーを区別するには、レスポンスのボディーを確認してください。

第4章 DOCKER コンテナー環境への APICAST のデプロイ

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

注記

Docker コンテナー環境に APIcast をデプロイする場合、サポートされる Red Hat Enterprise Linux (RHEL) および Docker のバージョンは以下のとおりです。

- RHEL 7.7
- Docker 1.13.1

前提条件

- *2章APIcast のインストール* に従って、3scale 管理ポータルで APIcast を設定している。
- Red Hat Ecosystem Catalog へのアクセス
  - レジストリーサービスアカウントを作成するには、レジストリーサービスアカウントの作成 を参照してください。

Docker コンテナー環境に APIcast をデプロイするには、以下のセクションに概略を示す手順を実施します。

- 「Docker コンテナー環境のインストール」
- 「Docker コンテナー環境ゲートウェイの実行」

4.1. DOCKER コンテナー環境のインストール

本セクションでは、RHEL 7 に Docker コンテナー環境を設定する手順を説明します。

Red Hat が提供する Docker コンテナーエンジンは、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** オプションを使用できます。詳細は、RHEL の製品ドキュメント を参照してください。

Amazon Web Services (AWS)、Amazon Elastic Compute Cloud (Amazon EC2) インスタンスに RHEL 7 をデプロイするには、以下の手順を実施します。

手順

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。

関連情報

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

4.2. DOCKER コンテナー環境ゲートウェイの実行

重要

3scale 2.11 では、RHEL7 および Docker のコンテナーとして実行されている APIcast デプロイメントのサポートは非推奨になりました。今後のリリースでは、3scale は RHEL8 および Podman のみをサポートします。Self-managed APIcast をコンテナーとして実行している場合は、インストールをサポート対象の設定にアップグレードしてください。

Docker コンテナー環境ゲートウェイを実行するには、以下の手順を実施します。

手順

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、そのまま使用できる Docker コンテナーエンジンのイメージをダウンロードできます。

+

```
sudo docker pull registry.redhat.io/3scale-amp2/apicast-gateway-rhel7:3scale2.13
```

1. Docker コンテナーエンジンで 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.redhat.io/3scale-amp2/apicast-gateway-rhel7:3scale2.13
```

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 コンテナーエンジンをポート **8080** で実行し、3scale 管理ポータルから 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。

4.2.1. docker コマンドのオプション

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>** は任意の属性で、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。

4.2.2. APIcast のテスト

以下の手順は、Docker コンテナーエンジンが独自の設定ファイルと、3scale レジストリーからの Docker コンテナー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。

関連情報

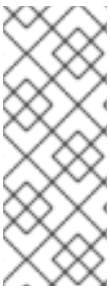
- 使用できるオプションの詳細については、Docker run reference を参照してください。

4.3. その他のリソース

- テスト済みのサポート対象設定の情報については、Red Hat 3scale API Management Supported Configurations を参照してください。

第5章 PODMAN への APICAST のデプロイ

ここでは、Pod Manager (Podman) コンテナー環境に Red Hat 3scale API ゲートウェイとして使用される APIcast をデプロイする方法を、ステップごとに説明します。



注記

Podman コンテナー環境に APIcast をデプロイする場合、サポートされる Red Hat Enterprise Linux (RHEL) および Podman のバージョンは以下のとおりです。

- RHEL 8
- Podman 1.4.2

前提条件

- *2章APIcast のインストール* に従って、3scale 管理ポータルで APIcast を設定している。
- Red Hat Ecosystem Catalog へのアクセス
  - レジストリーサービスアカウントを作成するには、レジストリーサービスアカウントの作成 を参照してください。

Podman コンテナー環境に APIcast をデプロイするには、以下のセクションに概略を示す手順を実施します。

- 「Podman コンテナー環境のインストール」
- 「Podman 環境の実行」

5.1. PODMAN コンテナー環境のインストール

本セクションでは、RHEL 8 に Podman コンテナー環境を設定する手順を説明します。Docker は RHEL 8 に含まれていないため、コンテナーの操作には Podman を使用します。

Podman と RHEL 8 の使用については、コンテナーのコマンドに関するリファレンスドキュメント を参照してください。

手順

- Podman コンテナー環境パッケージをインストールします。
  sudo dnf install podman

関連情報

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 Podman のドキュメントを参照してください。

- Podman Installation Instructions

5.2. PODMAN 環境の実行

Podman コンテナー環境を実行するには、以下の手順に従います。

手順

1. Red Hat レジストリーから、そのまま使用できる Podman コンテナーのイメージをダウンロードします。

```
podman pull registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.13
```

2. Podman で APIcast を実行します。

```
podman run --name apicast --rm -p 8080:8080 -e
THREESCALE_PORTAL_ENDPOINT=https://<access_token>@<domain>-admin.3scale.net
registry.redhat.io/3scale-amp2/apicast-gateway-rhel8:3scale2.13
```

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 Podman コンテナーエンジンをポート **8080** で実行し、3scale 管理ポータルから 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。

5.2.1. Podman による APIcast のテスト

以下の手順は、Podman コンテナーエンジンが独自の設定ファイルと、3scale レジストリーからの Podman コンテナー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。

5.3. PODMAN コマンドのオプション

podman コマンドでは、以下に例を示すオプションを使用することができます。

- **-d**:**デタッチモード** でコンテナーを実行し、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、**podman attach** コマンド (例: **podman 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。

- **ps** および **-a**:podman **ps** を使用して、作成中および実行中のコンテナーを一覧表示します。**ps** コマンドに **-a** を追加すると (例: **podman ps -a**)、すべてのコンテナー (実行中および停止中の両方) が表示されます。

- **inspect** および **-l**:実行中のコンテナーを検査します。たとえば、**inspect** を使用して、コンテナーに割り当てられた ID を表示します。**-l** を使用すると、最新のコンテナーの詳細を取得できます (たとえば **podman inspect -l | grep Id\":**)。

5.4. 関連情報

- テスト済みのサポート対象設定の情報については、Red Hat 3scale API Management Supported Configurations を参照してください。

- Podman を初めて使用する場合は、Basic Setup and Use of Podman を参照してください。