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: 2022-01-14

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

前書き

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」
- 「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。
   備考フォームのフィールドの前に、固定長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. 今作成した **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** にログインするのに使用されます。

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. 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。

- **トークンを再生成します。**許可されたユーザーが、**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できるようにします。
  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 を更新します。**許可されたユーザーが、**サービスアカウント** の説明を更新できるようにします。
- **アカウントの削除：**認定されたユーザーは、**サービスアカウント** を削除することができます。

1.3. 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 はスタンドアロンの API ゲートウェイではありません。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」
- 「APIcast の WebSocket プロトコルサポート」
- 「APIcast ゲートウェイの HTTP/2」

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 がクラウド内の APIcast をホストする。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 コンテナー環境に APIcast をデプロイします。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 を実行するためのすべての依存関係が含まれます。
  - Red Hat OpenShift 上で APIcast を実行します。サポートされるバージョンの OpenShift 上で APIcast を実行します。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とテスト中にのみ使用することが意図されています。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 50,000 の呼び出しに対応し、以下の追加設定なしの認証オプションをサポートします。API キー、アプリケーション ID とアプリケーションキーペア、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** を設定することで、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。**
   - **Deployment Option: 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

レスポンスコード	レスポンスのボディー
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、プロモートされる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、プロモートされる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
- まず「APIcast operator のインストール」に記載の手順に従う必要があります。

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. **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を指定する
- 設定シークレットを指定する

2.6.1.1. 3scale システムエンドポイントを指定する

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「AdminPortalSecret」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。以下は例になります。

```
$ 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	必須	該当せず	ゲートウェイにルーティングされているドメイン名

json/yaml フィールド	タイプ	必須/任意	デフォルト値	説明
tls	[]extensions.IngressTLS	任意	該当せず	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. 設定シークレットを指定する

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 の要件に規定されています。

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。以下は例になります。

```
$  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"
```

2.7. APICAST の WEBSOCKET プロトコルサポート

Red Hat 3scale は、バックエンド **API** への **Websocket** プロトコル接続に対する **APIcast** ゲートウェイをサポートします。

WebSocket プロトコルの実装を計画している場合は、以下のリストを考慮してください。

- **WebSocket** プロトコルは **JSON Web Token (JWT)** をサポートしない。
- **WebSocket** 標準は追加のヘッダーを許可しない。

- WebSocket プロトコルは HTTP/2 標準に含まれない。

2.7.1. WebSocket プロトコルのサポート

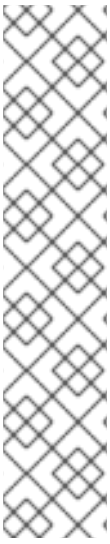
APIcast 設定ポリシーチェーンは以下のとおりです。

```
"policy_chain": [
  { "name": "apicast.policy.websocket" },
  { "name": "apicast.policy.apicast" }
],
```

API バックエンドはhttp[s]` または ws[s] として定義できます。

2.8. APICAST ゲートウェイの HTTP/2

Red Hat 3scale は、HTTP/2 および Remote Procedure Call (gRPC) 接続の APIcast ゲートウェイをサポートします。HTTP/2 プロトコル制御を使用すると、APIcast と API バックエンド間のデータ通信が可能になります。



注記

- api_key 承認は使用できません。代わりに JSON Web Token (JWT) またはヘッダーを使用してください。
- gRPC エンドポイントは Transport Layer Security (TLS) を終了します。
- GRPC ポリシー (HTTP/2) は、ポリシーチェーンで APIcast ポリシーよりも上位に設定する必要があります。

2.8.1. HTTP/2 プロトコルサポート

HTTP/2 終端では、APICast が有効な HTTP/2 およびバックエンドは HTTP/1.1 プレーンテキストまたは TLS にすることができます。

このポリシーを使用する HTTP/2 エンドポイントでは、以下の制約があります。

- このポリシーが想定通りに動作しない場合には、エンドポイントは TLS をリッスンする必要があります。

- gRPC 全フローは TLS ポリシーが有効な場合にのみ機能します。

APIcast 設定ポリシーチェーンは以下のとおりです。

```
"policy_chain": [
  { "name": "apicast.policy.grpc" },
  { "name": "apicast.policy.apicast" }
],
```

2.9. その他のリソース

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 のサポート対象構成

- Red Hat 3scale API Management - Component Details

- Hosted APIcast バージョンに関するアップデートについては、『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。
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 インスタンスには apicast.io ドメイン上にベース URL があります。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 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。

前提条件

- 「2章*APIcast のインストール*」に従って、Red Hat 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。

Red Hat OpenShift 上で APIcast を実行するには、以下のセクションに概略を示す手順を実施します。

- 「Red Hat OpenShift の設定」
- 「OpenShift テンプレートを使用した APIcast のデプロイ」
- 「OpenShift コンソール経由でのルートの作成」

4.1. RED HAT OPENSHIFT の設定

OpenShift クラスターが稼働中である場合は、本セクションを省略できます。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ース (CLI) v1.3.1

以下のセクションを使用して、Red Hat OpenShift を設定します。

- Docker コンテナー環境のインストール
- OpenShift クラスターの起動
- リモートサーバーでの OpenShift クラスターの設定 (任意)

4.1.1. 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。

手順

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
```

6. **以下のコマンドを使用して、コンテナーサービスが動作中であることを確認します。**

```
sudo systemctl status docker
```

4.1.2. OpenShift クラスターの起動

OpenShift クラスターを起動するには、以下の手順を実施します。

手順

1. **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

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。

2. **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
```

3. 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。

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

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。

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

OpenShift テンプレートを使用して APIcast をデプロイするには、以下の手順を使用します。

手順

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 アカウントの アクセストークン で、<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/2.11.0.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.
```

4.3. OPENSHIFT コンソール経由でのルートの作成

OpenShift コンソール経由でルートを作成するには、以下の手順を実施します。

手順

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。

   OpenShift のログイン画面が表示されます。

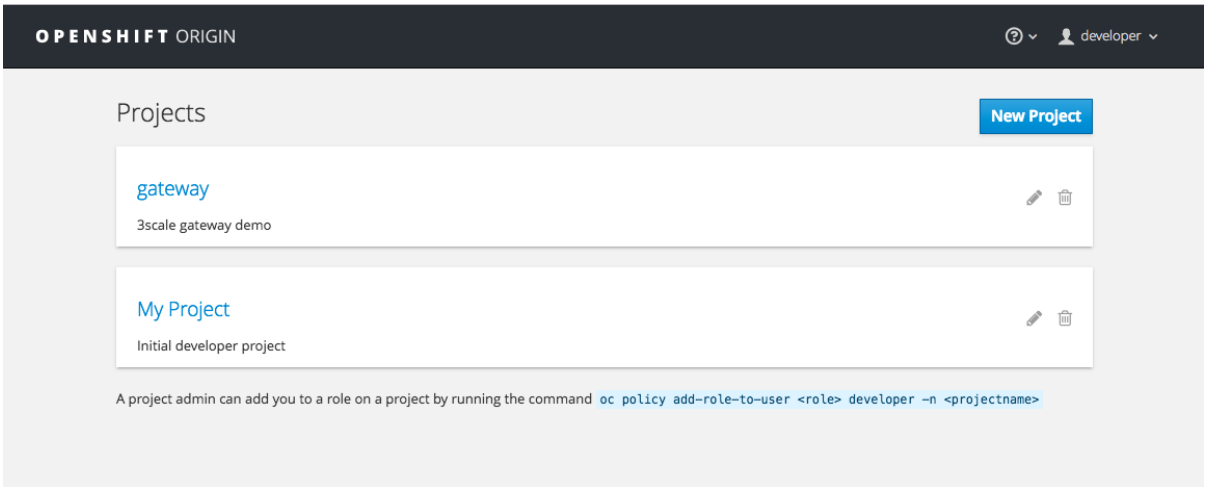
   注記

   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。これは本番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. 「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。

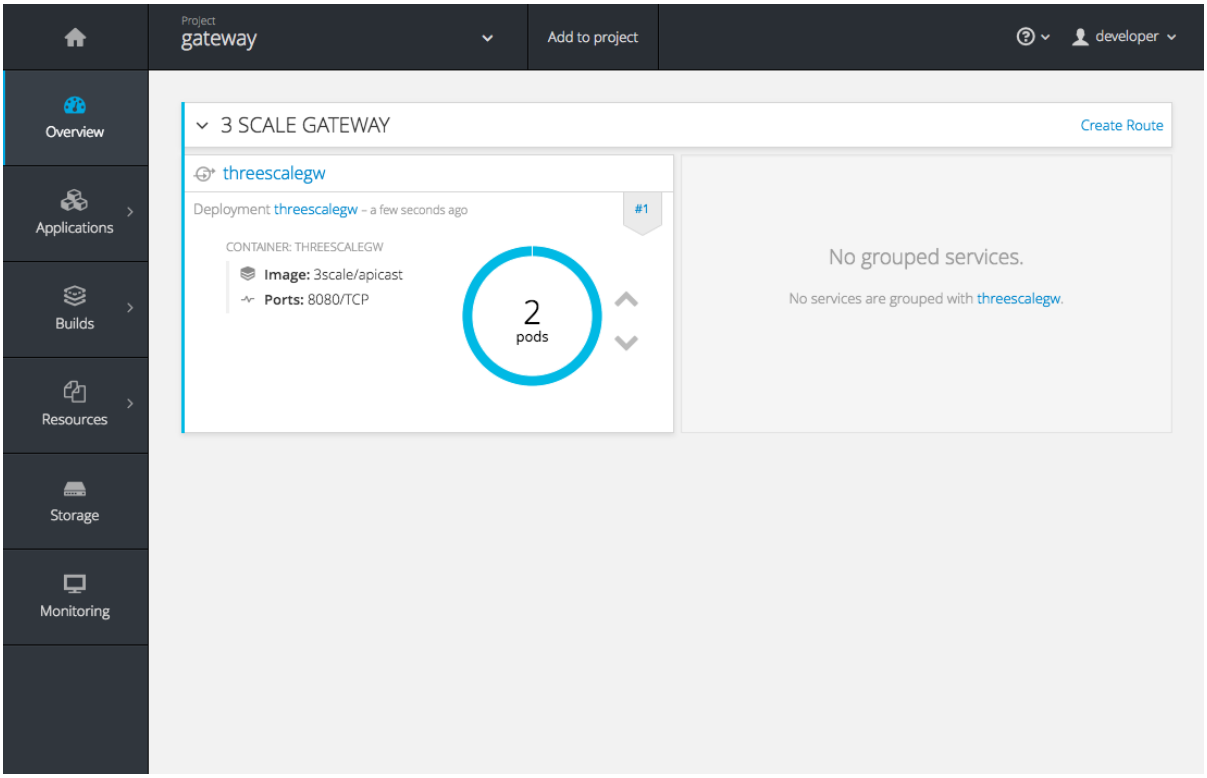


gateway プロジェクトが表示されない場合は、別のユーザーで作成した可能性があり、ポリシーロールをこのユーザーに割り当てる必要があります。

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、ユーザーインターフェース (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。

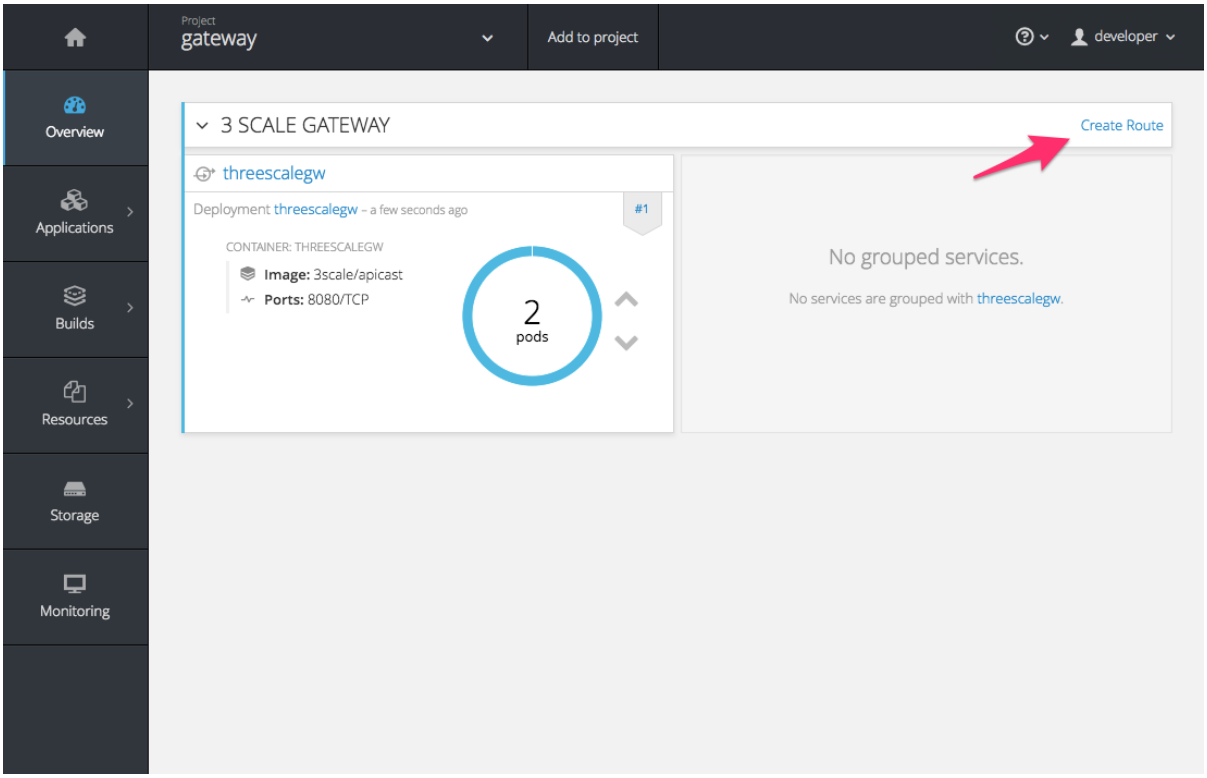


各 APIcast インスタンスが起動すると、3scale 管理ポータルの Integration ページの設定を使用して、3scale から必要な設定をダウンロードします。

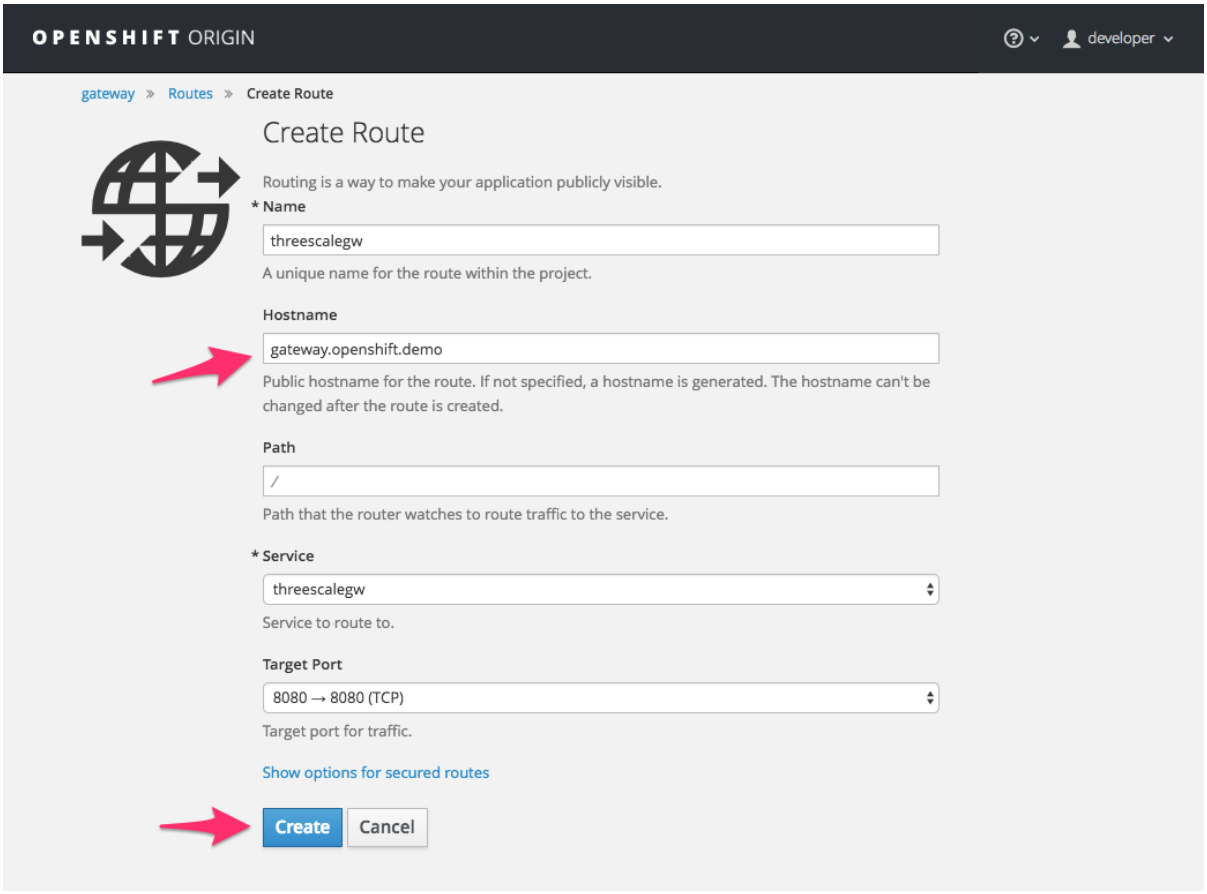
OpenShift は 2 つの APIcast インスタンスを維持し、両方のインスタンスの状態を監視します。状態の悪い APIcast インスタンスは自動的に新しいインスタンスに置き換えられます。

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** プロダクトについて、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 を設定する必要があります。
- 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」

5.1. 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。

その他のリソース

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

5.2. DOCKER コンテナー環境ゲートウェイの実行

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-{RHELVersionAPIcast}:3scale2.11
```

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-{RHELVersionAPIcast}:3scale2.11
```

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。

5.2.1. 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。

5.2.2. 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。

その他のリソース

- 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

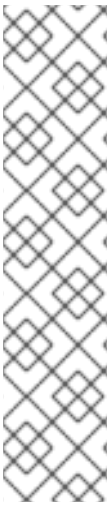
い。

5.3. その他のリソース

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

第6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 を設定する必要があります。
- 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」

6.1. 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
```

その他のリソース

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

6.2. 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-
{RHELVersionAPIcast}:3scale2.11
```

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-{RHELVersionAPIcast}:3scale2.11
```

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。

6.2.1. 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。

6.3. 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\":)。

6.4. 関連情報

- テスト済みのサポート対象構成の情報については、「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。