Red Hat 3scale API Management 2.5

3scale の操作

デプロイメントの自動化、環境のスケーリング、および問題のトラブルシューティングを行う方法

Last Updated: 2023-03-26

Red Hat 3scale API Management 2.5 3scale の操作

デプロイメントの自動化、環境のスケーリング、および問題のトラブルシューティングを行う方法

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

本書では、Red Hat 3scale API Management 2.5 のデプロイメント操作について説明します。

目次

第1章 3SCALE の操作とスケーリングのガイド

1.1. はじめに

本書では、オンプレミス型 Red Hat 3scale AMP 2.5 インストール環境での操作とスケーリングタスクについて説明します。

1.1.1. 前提条件

サポート対象バージョンの OpenShift にインストールされた初期設定済みのオンプレミス型 AMP インスタンス

本書は、ノートパソコンやこれに類するエンドユーザー機器上のローカルインストールを対象としていません。

1.1.2. 関連資料

- OpenShift のドキュメント
- 開発者ガイドの アプリケーションの正常性

1.2. APICAST の再デプロイ

オンプレミス型 AMP をデプロイし、希望の手法で APIcast をデプロイした後、3scale 管理ポータルでシステムの変更をテストしプロモートすることができます。デフォルトでは、OpenShift 上の APIcast デプロイメントでは (組み込み型のデプロイおよび他の OpenShift クラスター上のデプロイの両方で)、AMP UI を介して変更をステージング環境用と実稼働環境用のゲートウェイにパブリッシュできるように設定されています。

APIcast を OpenShift に再デプロイします。

1. システムに変更を加えます。
2. UI でステージング環境にデプロイしてテストします。
3. UI で実稼働環境にプロモートします。

デフォルトでは、APIcast はプロモートされた更新を 5 分ごとに取得し、パブリッシュします。

Docker コンテナー環境またはネイティブインストールで APIcast を使用している場合は、ステージング環境用と実稼働環境用のゲートウェイを設定し、パブリッシュした変更をゲートウェイが取得する頻度を設定する必要があります。APIcast ゲートウェイを設定したら、AMP UI で APIcast を再デプロイできます。

Docker コンテナー環境またはネイティブインストールに APIcast を再デプロイするには、以下を実行します。

1. APIcast ゲートウェイを設定し、オンプレミス型 AMP に接続します。
2. システムに変更を加えます。
3. UI でステージング環境にデプロイしてテストします。
4. UI で実稼働環境にプロモートします。

APIcast は、設定された頻度でプロモートされた更新を取得してパブリッシュします。

1.3. APICAST ビルトインワイルドカードルーティング

オンプレミスの 3scale デプロイメントに付随する組み込みの APIcast ゲートウェイは、サブドメインレベルでのワイルドカードドメインルーティングをサポートします。この機能を使用すると、実稼働およびステージングゲートウェイのパブリックベース URL のサブドメインの一部に名前を付けることができます。この機能を使用するには、3scale のインストール時に有効にする必要があります。

注記

ワイルドカードルーティングをサポートする OpenShift Container Platform バージョンを使用していることを確認してください。サポートされているバージョンについては、サポートされている設定 を参照してください。

AMP は DNS 機能を提供しないため、指定したパブリックベース URL は、デプロイされた OpenShift クラスターの **WILDCARD_DOMAIN** パラメーターで指定された DNS 設定と一致する必要があります。

1.3.1. ワイルドカードの変更

ワイルドカードを変更するには、次の手順を実行します。

1. AMP にログインします。
2. **API** → your API → **Integration** → **edit APIcast configuration** の順に、API ゲートウェイ設定ページに移動します。
3. 選択した文字列の接頭辞を使用してステージングおよび実稼働のパブリックベース URL を変更し、次の要件に従います。
    - API エンドポイントは数字で始めてはいけません

以下は、ドメイン **example.com** のステージングゲートウェイの有効なワイルドカードの例です。

```
apiname-staging.example.com
```

補足情報

ルーティングの詳細は、OpenShift ドキュメント を参照してください。

1.4. オンプレミス型 3SCALE のスケールアップ

1.4.1. ストレージのスケールアップ

APIcast デプロイメントの規模が大きくなると、利用可能なストレージの量を増やす必要が生じる可能性があります。ストレージをスケールアップする方法は、永続ストレージに使用しているファイルシステムのタイプによって異なります。

ネットワークファイルシステム (NFS) を使用している場合は、**oc edit pv** コマンドを使用して永続ボリュームをスケールアップできます。

```
oc edit pv <pv_name>
```

他のストレージ手段を使用している場合は、以降のセクションに挙げる方法のいずれかを使用して、永続ボリュームを手動でスケールアップする必要があります。

1.4.1.1. 方法 1: 永続ボリュームをバックアップしてスワップする

1. 既存の永続ボリューム上のデータをバックアップします。
2. 新しいサイズ要件に合わせて、ターゲット永続ボリュームを作成し、アタッチします。
3. 事前バインド型の永続ボリューム要求を作成します。新しい PVC のサイズと、**volumeName** フィールドを使用して永続ボリューム名を指定します。
4. 新しく作成した PV に、バックアップからデータを復元します。
5. 新しい PV の名前でデプロイメント設定を変更します。

```
oc edit dc/system-app
```

6. 新しい PV が設定され正常に機能していることを確認します。
7. 以前の PVC を削除して、それが要求していたリソースを解放します。

1.4.1.2. 方法 2: 3scale をバックアップして再デプロイする

1. 既存の永続ボリューム上のデータをバックアップします。
2. 3scale Pod をシャットダウンします。
3. 新しいサイズ要件に合わせて、ターゲット永続ボリュームを作成し、アタッチします。
4. 新しく作成した PV に、バックアップからデータを復元します。
5. 事前バインド型の永続ボリューム要求を作成します。以下の項目を指定します。
    a. 新しい PVC のサイズ
    b. 永続ボリューム名 (**volumeName** フィールドを使用)
6. AMP.yml をデプロイします。
7. 新しい PV が設定され正常に機能していることを確認します。
8. 以前の PVC を削除して、それが要求していたリソースを解放します。

1.4.2. パフォーマンスのスケールアップ

1.4.2.1. オンプレミス型 3scale デプロイメントの設定

デフォルトでは、3scale デプロイメントは Pod ごとに 1 つのプロセスを実行します。Pod ごとに実行するプロセスを増やすことで、パフォーマンスを向上させることができます。Red Hat は、各ノードのコアごとに 1 つまたは 2 つのプロセスを実行することを推奨します。

Pod にプロセスを追加するには、以下の手順を実行します。

1. OpenShift クラスターにログインします。

```
oc login
```

2. 3scale プロジェクトに切り替えます。

```
oc project <project_name>
```

3. 適切な環境変数に、希望する Pod ごとのプロセス数を設定します。

   a. APIcast Pod: **APICAST_WORKERS** (Red Hat は、APIcast が APIcast Pod で利用できる CPU の数によってワーカーの数を判断できるよう、この環境変数を未設定のままにすることを推奨します)

   b. バックエンド Pod: **PUMA_WORKERS**

   c. システム Pod: **UNICORN_WORKERS**

```
oc set env dc/apicast-{production/staging} --overwrite APICAST_WORKERS=
<number_of_processes>
```

```
oc set env dc/backend-listener --overwrite PUMA_WORKERS=<number_of_processes>
```

```
oc set env dc/system-app --overwrite UNICORN_WORKERS=<number_of_processes>
```

1.4.2.2. ハードウェアの垂直スケーリングと水平スケーリング

リソースを追加することで、OpenShift 上の AMP デプロイメントのパフォーマンスを高めることができます。コンピュートノードを Pod として OpenShift クラスターに追加すること (水平スケーリング) や、既存のコンピュートノードに割り当てるリソースを増やすこと (垂直スケーリング) ができます。

水平スケーリング

コンピュートノードを Pod として OpenShift に追加することができます。追加のコンピュートノードがクラスター内の既存ノードと一致する場合には、環境変数を再設定する必要はありません。

垂直スケーリング

既存のコンピュートノードに割り当てるリソースを増やすことができます。割り当てるリソースを増やす場合は、追加のプロセスを Pod に追加してパフォーマンスを高める必要があります。

注記

Red Hat は、3scale デプロイメントにおいて、仕様や設定の異なるコンピュートノードを混在させることは推奨していません。

1.4.2.3. ルーターのスケールアップ

トラフィックの増加に応じて、OCP ルーターがリクエストを適切に処理できるようにする必要があります。ルーターがリクエストのスループットを制限している場合には、ルーターノードをスケールアップする必要があります。

1.4.2.4. 関連資料

- タスクのスケーリング、ハードウェアコンピュートノードの OpenShift への追加

- コンピュートノードの追加
- ルーター

1.5. 操作のトラブルシューティング

1.5.1. ログへのアクセス

各コンポーネントのデプロイメント設定には、アクセスと例外のログが含まれます。デプロイメントで問題が発生した場合には、これらのログで詳細を確認してください。

3scale のログにアクセスするには、以下の手順に従います。

1. ログが必要な Pod の識別子 (ID) を見つけます。

```
oc get pods
```

2. **oc logs** と選択した Pod の ID を入力します。

```
oc logs <pod>
```

システム Pod にはコンテナーが 2 つあり、それぞれに別個のログがあります。コンテナーのログにアクセスするには、**--container** パラメーターで **system-provider** と **system-developer** を指定します。

```
oc logs <pod> --container=system-provider
oc logs <pod> --container=system-developer
```

1.5.2. ジョブキューの確認

ジョブキューには、**system-sidekiq** Pod から送られる情報のログが含まれます。これらのログを使用して、クラスターがデータを処理しているかどうかを確認します。OpenShift CLI を使用してログを照会することができます。

```
oc get jobs
```

```
oc logs <job>
```

1.5.3. 単調増加の防止

単調増加を防止するために、3scale はデフォルトで以下のテーブルの自動パージをスケジュールします。

- **user_sessions**: クリーンアップは週 1 回トリガーされ、2 週間より前のレコードを削除します。
- **audits**: クリーンアップは 1 日 1 回トリガーされ、3 カ月より前のレコードを削除します。
- **log_entries**: クリーンアップは 1 日 1 回トリガーされ、6 カ月より前のレコードを削除します。
- **event_store_events**: クリーンアップは週 1 回トリガーされ、1 週間より前のレコードを削除します。

alerts テーブルは例外で、データベース管理者が手動でパージする必要があります。

データベースタイプ	SQL コマンド
MySQL	`DELETE FROM alerts WHERE timestamp < NOW() - INTERVAL 14 DAY;`
PostgreSQL	`DELETE FROM alerts WHERE timestamp < NOW() - INTERVAL '14 day';`
Oracle	`DELETE FROM alerts WHERE timestamp <= TRUNC(SYSDATE) - 14;`

本セクションで説明しない他のテーブルについては、データベース管理者は、システムで自動的にパージされないテーブルを手動でクリーンアップする必要があります。

第2章 3SCALE TOOLBOX の使用

3scale toolbox は、コマンドラインから 3scale サービスを管理できる Ruby クライアントです。

2.1. TOOLBOX のインストール

ツールボックスをインストールするために公式にサポートされている唯一の方法は、**yum** または **rpm** を使用して Red Hat Enterprise Linux にインストールする方法です。

前提条件

次のリポジトリーへのアクセスを有効にする必要があります。

- **rhel-7-server-3scale-amp-2.5-rpms**
- **rhel-server-rhscl-7-rpms**

以下に例を示します。

```
$ sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2.5-rpms --enable rhel-server-rhscl-7-rpms
```

手順

1. ツールボックスをインストールします。

```
$ yum install 3scale_toolbox
```

2. インストールを確認します。

```
$ 3scale help
```

関連情報

ただし、Fedora Linux、Ubuntu Linux、Windows、または macOS に、サポートされないバージョンを使用できます。そのためには、**.rpm**、**.deb**、**.msi**、または **.pkg** ファイル を GitHub から直接ダウンロードしてインストールします。

2.2. サービスのインポート

以下のフィールドをこの順序で指定して、CSV ファイルからサービスをインポートします (CSV ファイルにこれらのヘッダーを含める必要もあります)。

```
service_name,endpoint_name,endpoint_http_method,endpoint_path,auth_mode,endpoint_system_name,type
```

以下の情報が必要になります。

- 3scale 管理アカウント: **{3SCALE_ADMIN}**
- 3scale インスタンスが動作中のドメイン: **{DOMAIN_NAME}** (Hosted APICast を使用している場合は 3scale.net)
- アカウントのアクセスキー: **{ACCESS_KEY}**

- サービスの CSV ファイル (**example/import_example.csv**)

以下のコマンドを実行してサービスをインポートします。

```
$ 3scale import csv --destination=https://{ACCESS_KEY}@{3SCALE_ADMIN}-admin.
{DOMAIN_NAME} --file=examples/import_example.csv
```

2.3. サービスのコピー

同じアカウントまたは別のアカウントから、既存のサービスをベースにして新しいサービスを作成します。サービスをコピーすると、関連する ActiveDocs もコピーされます。

以下の情報が必要です。

- コピーするサービスの ID: **{SERVICE_ID}**
- 3scale 管理アカウント: **{3SCALE_ADMIN}**
- 3scale インスタンスが動作中のドメイン: **{DOMAIN_NAME}** (Hosted APICast を使用している場合は 3scale.net)
- アカウントのアクセスキー: **{ACCESS_KEY}**
- 別のアカウントにコピーする場合は、コピー先アカウントのアクセスキー: **{DEST_KEY}**
- 新しいサービスの名前: **{NEW_NAME}**

```
$ 3scale copy service {SERVICE_ID} --source=https://{ACCESS_KEY}@{3SCALE_ADMIN}-admin.
{DOMAIN_NAME} --destination=https://{DEST_KEY}@{3SCALE_ADMIN}-admin.{DOMAIN_NAME} -
-target_system_name={NEW_NAME}
```

2.4. サービス設定のみのコピー

あるサービスから別の既存サービスに、サービス設定、プロキシー設定、メトリクス、メソッド、アプリケーションプラン、アプリケーションプランの制限、およびマッピングルールを一括コピー (更新とも呼ばれる) します。

以下の情報が必要になります。

- コピーするサービスの ID: **{SERVICE_ID}**
- コピー先のサービスの ID: **{DEST_ID}**
- 3scale 管理アカウント: **{3SCALE_ADMIN}**
- 3scale インスタンスが動作中のドメイン: **{DOMAIN_NAME}** (Hosted APICast を使用している場合は 3scale.net)
- アカウントのアクセスキー: **{ACCESS_KEY}**
- コピー先アカウントのアクセスキー: **{DEST_KEY}**

また、以下のオプションのフラグを使用できます。

- **-f**: コピーする前に既存の対象サービスのマッピングルールを削除します。

- **-r**: 対象サービスにマッピングルールのみをコピーします。

```
$ 3scale update [opts] service --source=https://{ACCESS_KEY}@{3SCALE_ADMIN}-admin.
{DOMAIN_NAME} --destination=https://{DEST_KEY}@{3SCALE_ADMIN}-admin.{DOMAIN_NAME}
{SERVICE_ID} {DEST_ID}
```

2.5. OPENAPI 定義のインポート

json および **yaml** の形式の OpenAPI 仕様 (OAS) に基づく定義をインポートできます。新しいサービスを作成する場合、または既存のサービスを更新する場合は、ローカルファイルまたは Web アドレスからの定義を使用します。

インポートのデフォルトのサービス名は、OpenAPI 定義の **info.title** から取得され、**--target_system_name=<NEW NAME>** を使用してオーバーライドできます。そのサービス名がすでに存在する場合は、それが更新されます。また、そのサービス名が存在しない場合は、作成されます。

以下のルールはすべてのインポートに適用されます。

- 定義は OpenAPI 2.0 として検証される。
- 3scale サービスのマッピングルールは、すべて削除される。
- 置き換えるためには、パターンの完全一致の使用により、すべてのメソッドの名前が OpenAPI 定義 (**operation.operationId**) で定義されるメソッドと同一でなければならない。
- OpenAPI 定義に含まれるメソッドのみが変更される。
- OpenAPI 定義にしか存在していなかったすべてのメソッドが、**Hits** メトリクスにアタッチされる。
- OpenAPI 定義のすべてのマッピングルールがインポートされる。これらについては **API** > **Integration** で確認できます。

```
$ 3scale import openapi [opts] --destination=https://{DEST_KEY}@{3SCALE_ADMIN}-admin.
{DOMAIN_NAME}
```

2.5.1. オプションのフラグ

-d --destination=<value>

3scale ターゲットインスタンス。形式: **http[s]://<authentication>@3scale_domain**

-t --target_system_name=<value>

ターゲットシステム名

2.6. リモート WEB アドレスの管理

3scale インスタンスの操作を容易にするために、設定ファイルでリモート Web アドレス (URL) を認証とともに定義し、3scale ツールボックスコマンドで短縮名を使って参照します。

設定ファイルのデフォルトの場所は **$HOME/.3scalerc.yaml** ですが、**THREESCALE_CLI_CONFIG** 環境変数、または **--config-file <config_file>** オプションを使用して、別の場所を指定することができます。

access_token または **provider_key** のいずれかを使用して、リモートを指定することができます。

- **http[s]://<access_token>@<3scale-instance-domain>**
- **http[s]://<provider_key>@<3scale-instance-domain>**

2.6.1. リモート Web アドレスの一覧表示

```
3scale remote list [--config-file <config_file>]
```

既存のリモートのリスト (名前、URL、および認証キー) を表示します。

例

```
$ 3scale remote list
instance_a https://example_a.net 123456789
instance_b https://example_b.net 987654321
```

2.6.2. リモート Web アドレスの追加

```
3scale remote add [--config-file <config_file>] <name> <url>
```

短縮名 **<name>** のリモートを **<url>** に追加します。

例

```
3scale remote add instance_a https://123456789@example_a.net
```

2.6.3. リモート Web アドレスの削除

```
3scale remote remove [--config-file <config_file>] <name>
```

短縮名 **<name>** のリモートを削除します。

例

```
3scale remote remove instance_a
```

2.6.4. リモート Web アドレスの名前変更

```
3scale remote rename [--config-file <config_file>] <old_name> <new_name>
```

リモートの短縮名を **<old_name>** から **<new_name>** に変更します。

例

```
3scale remote rename instance_a instance_b
```

2.7. SSL の問題のトラブルシューティング

Red Hat のドキュメントに 証明書 の詳細が記載されていますが、自己署名 SSL 証明書、**SSL 証明書の問題: 自己署名証明書** などに関連する問題が発生している場合は、リモート証明書をダウンロードして使用できます。

1. **openssl** を使用してリモート証明書をダウンロードします。

```
$ echo | openssl s_client -showcerts -servername self-signed.badssl.com -connect self-signed.badssl.com:443 2>/dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > self-signed-cert.pem
```

2. **curl** で使用して動作することを確認してください。

```
$ SSL_CERT_FILE=self-signed-cert.pem curl -v https://self-signed.badssl.com
```

   証明書が正常に機能している場合、SSL エラーは発生しません。

3. **3scale** コマンドの前に **SSL_CERT_FILE=self-signed-cert.pem** を付けます。

```
$ SSL_CERT_FILE=self-signed-cert.pem 3scale import csv
```

第3章 OPERATOR を使用した 3SCALE のサービスおよび設定のプロビジョニング (CAPABILITIES)

3.1. はじめに

このドキュメントでは、3scale Operator を介した 3scale サービスおよび設定のプロビジョニングに関する情報を提供します。

重要

3scale operator 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。

3.1.1. 前提条件

- オンプレミス型 3scale 2.5 インスタンス
- 3scale operator がインストールされている。
- OpenShift Container Platform 3.11
  - OpenShift クラスターの管理者権限を持つユーザーアカウント

警告

operator を使用して 3scale で API 設定を更新する場合は、カスタムリソース定義 (CRD) が信頼できるソースです。管理 UI に変更が加えられると、それらは永続化されず、最終的に CRD の定義で上書きされます。

3.2. CAPABILITIES に関連するカスタムリソースのデプロイ

Openshiftオブジェクト のみを使用して、新しく作成したテナントで API、**メトリクス**、およびマッピングルールの設定を開始します。

1. 最初の API の作成

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: API
metadata:
  creationTimestamp: 2019-01-25T13:28:41Z
  generation: 1
  labels:
    environment: testing
```

```
  name: api01
spec:
  planSelector:
    matchLabels:
      api: api01
  description: api01
  integrationMethod:
    apicastHosted:
      apiTestGetRequest: /
      authenticationSettings:
        credentials:
          apiKey:
            authParameterName: user-key
            credentialsLocation: headers
        errors:
          authenticationFailed:
            contentType: text/plain; charset=us-ascii
            responseBody: Authentication failed
            responseCode: 403
          authenticationMissing:
            contentType: text/plain; charset=us-ascii
            responseBody: Authentication Missing
            responseCode: 403
        hostHeader: ""
        secretToken: Shared_secret_sent_from_proxy_to_API_backend_9603f637ca51ccfe
      mappingRulesSelector:
        matchLabels:
          api: api01
      privateBaseURL: https://echo-api.3scale.net:443
  metricSelector:
    matchLabels:
      api: api01
```

すべてのセレクター (**metric**、**plan**、**mappingrules**) で、特定のラベル **api: api01** を使用します。これを変更して、ラベルをいくつでも追加し、セレクターを操作して、非常に複雑なシナリオにも対応できます。

2. **プラン** を追加します。

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: Plan
metadata:
  labels:
    api: api01
  name: plan01
spec:
  approvalRequired: false
  default: true
  costs:
    costMonth: 0
    setupFee: 0
  limitSelector:
    matchLabels:
      api: api01
  trialPeriod: 0
```

3. **metric01** というメトリックを追加します。

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: Metric
metadata:
  labels:
    api: api01
  name: metric01
spec:
  description: metric01
  unit: hit
  incrementHits: false
```

4. メトリックに 1 日あたり 10 ヒットの上限を設定します。

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: Limit
metadata:
  labels:
    api: api01
  name: plan01-metric01-day-10
spec:
  description: Limit for metric01 in plan01
  maxValue: 10
  metricRef:
    name: metric01
  period: day
```

5. **metric01** を 1 つず増加するように **MappingRule** を追加します。

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: MappingRule
metadata:
  labels:
    api: api01
  name: metric01-get-path01
spec:
  increment: 1
  method: GET
  metricRef:
    name: metric01
  path: /path01
```

6. バインディングオブジェクトを使用してバインドします。
テナントコントローラー で作成されたクレデンシャルを使用します。

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: Binding
metadata:
  name: mytestingbinding
spec:
  credentialsRef:
    name: ecorp-tenant-secret
```

```
  APISelector:
    matchLabels:
      environment: testing
```

バインディングオブジェクトは、**ecorp-tenant-secret** を参照して、**environment: staging** のラベルが設定された API オブジェクトを作成します。

7. 新しい 3scale テナントに移動し、すべてが作成されていることを確認します。

注記

詳細は、参考のドキュメント Capabilities CRD を確認してください。

3.3. オプションテナントカスタムリソースのデプロイ

オプションで、**Tenant** カスタムリソースオブジェクトをデプロイするその他のテナントを作成できます。

1. 管理者パスワードを保存するシークレットを作成して、3scale インスタンスに新しいテナントをデプロイします。

```
$ cat ecorp-admin-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: ecorp-admin-secret
type: Opaque
stringData:
  admin_password: <admin password value>

$ oc create -f ecorp-admin-secret.yaml
secret/ecorp-admin-secret created
```

2. 次の内容で新しいテナント **CR YAML** ファイルを作成します。

```
apiVersion: capabilities.3scale.net/v1alpha1
kind: Tenant
metadata:
  name: ecorp-tenant
spec:
  username: admin
  systemMasterUrl: https://master.<wildcardDomain>
  email: admin@ecorp.com
  organizationName: ECorp
  masterCredentialsRef:
    name: system-seed
  passwordCredentialsRef:
    name: ecorp-admin-secret
  tenantSecretRef:
    name: ecorp-tenant-secret
    namespace: operator-test
```

注記

Tenant Custom Resource フィールドと設定可能な値については、テナント CRD のリファレンス を参照してください。

```
export NAMESPACE="operator-test"
oc project ${NAMESPACE}
oc create -f <yaml-name>
```

a. これにより、**operator-test** プロジェクトの 3scale ソリューションで新しいテナントの作成がトリガーされます。

テナントの **provider_key** と **admin domain URL** はシークレットに格納されます。**tenantSecretRef** テナント仕様キーを使用して、シークレットの場所を指定できます。

3.4. 作成したカスタムリソースの削除

警告

APIManager を削除すると、3scale インストールが削除されます。

- **APIManager** カスタムリソースとそこからデプロイされた 3scale ソリューション要素を削除します。
  - **APIManager** を削除すると、デプロイされた場所にあるすべての 3scale 関連オブジェクトが削除されます。

    ```
    oc delete -f <yaml-name-of-the-apimanager-custom-resource>
    ```

- 3scale Operator、関連するロール、およびサービスアカウントを削除します。

  ```
  oc delete -f deploy/operator.yaml
  oc delete -f deploy/role_binding.yaml
  oc delete -f deploy/service_account.yaml
  oc delete -f deploy/role.yaml
  ```

- **APIManager** と **Capabilities** 関連の CRD を削除します。

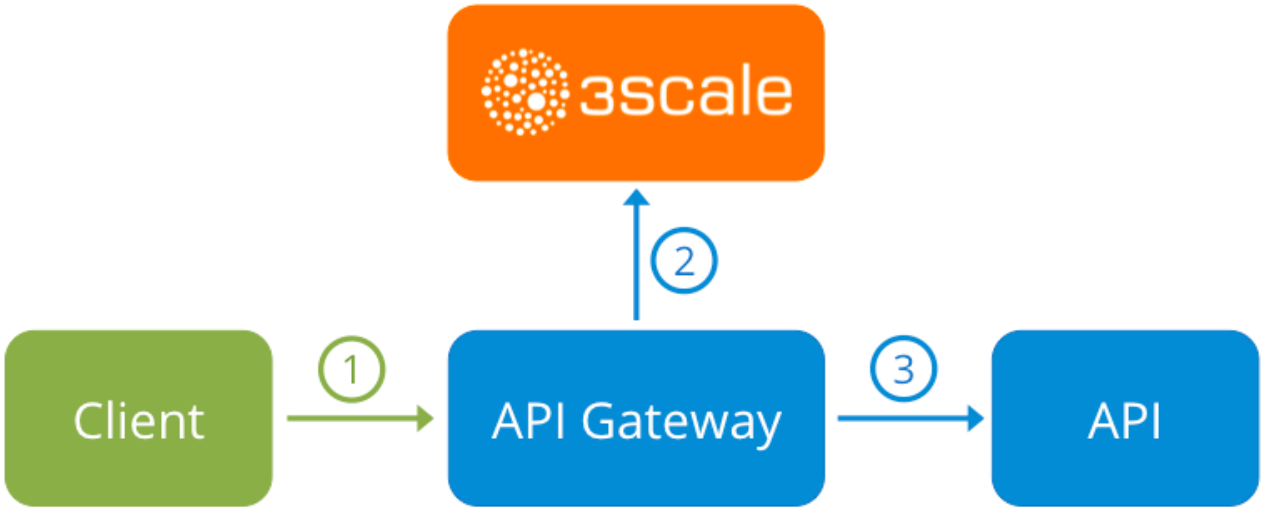
  ```
  oc delete -f deploy/crds/
  ```

第4章 トラブルシューティング

本章の目的は、ユーザーが API インフラストラクチャーに関連する問題の原因を特定して修正できるように支援することです。

API インフラストラクチャーは、長く複雑なトピックです。ただし、少なくとも、インフラストラクチャーには 3 つの可動部分があります。

1. API ゲートウェイ
2. 3scale
3. API



これらの 3 つの要素のいずれかでエラーが起こると、顧客は API にアクセスできなくなります。ただし、障害の原因となったコンポーネントを特定することは困難です。本章では、インフラストラクチャーのトラブルシューティングを行って問題を特定するためのヒントを紹介します。

4.1. 典型的な問題

3scale とのインテグレーションに関する非常に典型的な問題を示す症状がいくつかあります。これらは、API プロジェクトの最初の段階であるのか、インフラストラクチャーをセットアップしているのか、またはすでに実稼働環境に移行しているのかによって異なります。

4.1.1. インテグレーションの問題

以降のセクションで、3scale とのインテグレーションにおける初期段階 (Hosted APIcast 使用の初期段階および実稼働環境への移行前、ならびに Self-managed APIcast の稼働中) で、APIcast エラーログでよく見られる問題のいくつかについて概要を説明します。

4.1.1.1. Hosted APIcast

Service Integration 画面で API と Hosted APIcast を初めて統合する場合、以下のエラーのいずれかがページに表示されたり、インテグレーションの成功を確認するためのテストコールでエラーが返されたりする可能性があります。

- **Test request failed: execution expired**
  API が一般のインターネットからアクセス可能であることを確認します。Hosted APIcast は、

プライベート API を扱うことができません。Hosted APIcast と統合する際に API を一般に公開したくない場合は、Hosted APIcast と API との間にプライベートシークレットを設定し、API ゲートウェイ以外からの呼び出しを拒否することができます。

- The accepted format is protocol://address(:port)
  API のプライベートベース URL の最後にあるパスを削除します。これらのパスは、マッピングルールのパターン、または API テスト GET リクエストの最初に追加できます。

- **Test request failed with HTTP code XXX**

  - **405**: エンドポイントが GET リクエストを受け入れることを確認します。APIcast は、インテグレーションをテストするための GET リクエストのみをサポートしています。

  - **403: Authentication parameters missing**: API にすでに何らかの認証が設定されている場合は、APIcast はテストリクエストを送信することができません。

  - **403: Authentication failed**: 3scale でこれ以前にサービスを作成したことがある場合は、テストリクエストを行うためのクレデンシャルが設定されたサービスでアプリケーションを作成していることを確認します。これが統合する最初のサービスである場合は、サインアップ時に作成したテストアカウントまたはアプリケーションを削除していないことを確認します。

4.1.1.2. Self-managed APIcast

Self-managed APIcast とのインテグレーションのテストが正常に終了したら、API ゲートウェイを独自にホストすることが望ましい場合があります。以下は、自己管理型ゲートウェイを初めてインストールし、これを介して API を呼び出す際に生じる可能性のあるエラーです。

- **upstream timed out (110: Connection timed out) while connecting to upstream**
  API ゲートウェイと一般のインターネットの間に、Self-managed APIcast ゲートウェイが 3scale に到達するのを妨げるファイアウォールまたはプロキシーがないことを確認します。

- **failed to get list of services: invalid status: 403 (Forbidden)**

  ```
  2018/06/04 08:04:49 [emerg] 14#14: [lua] configuration_loader.lua:134: init(): failed to load configuration,   exiting (code 1)
  2018/06/04 08:04:49 [warn] 22#22: *2 [lua] remote_v2.lua:163: call(): failed to get list of services:   invalid status: 403 (Forbidden) url: https://example-admin.3scale.net/admin/api/services.json , context:   ngx.timer
  ERROR: /opt/app-root/src/src/apicast/configuration_loader.lua:57: missing configuration
  ```

  THREESCALE_PORTAL_ENDPOINT の値に使用するアクセストークンが正しいこと、またスコープに Account Management API が含まれていることを確認します。そのためには、**curl** コマンドを使用して確認します (**curl -v "https://example-admin.3scale.net/admin/api/services.json?access_token=<YOUR_ACCESS_TOKEN>"**)。

  JSON ボディーでレスポンス 200 が返されるはずです。エラーステータスコードを返す場合は、レスポンスのボディーで詳細を確認します。

- **service not found for host apicast.example.com**

  ```
  2018/06/04 11:06:15 [warn] 23#23: *495 [lua] find_service.lua:24: find_service(): service not found for host apicast.example.com, client: 172.17.0.1, server: _, request: "GET / HTTP/1.1", host: "apicast.example.com"
  ```

このエラーは、公開ベース URL が正しく設定されていないことを示しています。設定された公開ベース URL は、Self-managed APIcast へのリクエストに使用するものと同じにする必要があります。正しい公開ベース URL を設定した後、以下を実行します。

- APIcast が実稼働用に設定されていることを確認します (**THREESCALE_DEPLOYMENT_ENV** 変数で上書きされていない場合のスタンドアロン APIcast のデフォルト設定)。必ず設定を実稼働環境にプロモートしてください。
- 環境変数 **APICAST_CONFIGURATION_CACHE** と **APICAST_CONFIGURATION_LOADER** を使用して自動的に設定を再読み込みするように設定していなかった場合は、APIcast を再起動します。

以下は、Self-managed APIcast の誤ったインテグレーションを示すその他の症状の例です。

- **マッピングルールの不一致/API コールの二重カウント**: メソッドと API の実際の URL エンドポイント間のマッピングをどのように定義したかによって、場合により、メソッドが一致しない、またはリクエストごとに複数回カウントが増加することがあります。この問題のトラブルシューティングを行うには、3scale デバッグヘッダー を使用して API にテストコールを行います。これにより、API コールで一致したすべてのメソッドのリストが返されます。
- **認証パラメーターが見つからない**: パラメーターを Service Integration 画面で指定した正しい場所に送信していることを確認します。クレデンシャルをヘッダーとして送信しない場合、GET リクエストについてはクエリーパラメーターとして、その他の HTTP メソッドについてはボディーパラメーターとして送信する必要があります。3scale デバッグヘッダーを使用して、API ゲートウェイによりリクエストから読み取られるクレデンシャルを再確認します。

4.1.2. 実稼働環境の問題

セットアップを完全にテストし、しばらくの間実際に API を運用した後に、API ゲートウェイに関連して問題が発生することはほとんどありません。ただし、実際の実稼働環境で発生しうる問題の一部をここに挙げます。

4.1.2.1. 可用性の問題

可用性の問題は、通常、nginx error.log に **upstream timed out** エラーが表示されることが特徴です。以下に例を示します。

```
upstream timed out (110: Connection timed out) while connecting to upstream, client: X.X.X.X,
server: api.example.com, request: "GET /RESOURCE?CREDENTIALS HTTP/1.1", upstream:
"http://Y.Y.Y.Y:80/RESOURCE?CREDENTIALS", host: "api.example.com"
```

断続的に 3scale の可用性の問題が発生する場合、以下が原因の可能性があります。

- 使用されていない古い 3scale IP に解決しようとしている。
  最新バージョンの API ゲートウェイ設定ファイルは、毎回強制的に IP を解決するために、変数として 3scale を定義します。応急処置として、NGINX インスタンスを再読み込みします。長期的な修正としては、アップストリームブロックで 3scale バックエンドを定義するのではなく、たとえば以下のように、各サーバーブロック内の変数として定義します。

```
server {
  # Enabling the Lua code cache is strongly encouraged for production use. Here it is enabled
  .
  .
  .
  set $threescale_backend "https://su1.3scale.net:443";
```

これを参照する場合は、以下のとおりです。

```
location = /threescale_authrep {
  internal;
  set $provider_key "YOUR_PROVIDER_KEY";

  proxy_pass $threescale_backend/transactions/authrep.xml?
provider_key=$provider_key&service_id=$service_id&$usage&$credentials&log%5Bcode%5
D=$arg_code&log%5Brequest%5D=$arg_req&log%5Bresponse%5D=$arg_resp;
}
```

- すべての 3scale IP がホワイトリスト上に記載されていない。3scale が解決する IP の現在のリストを以下に示します。
  - 75.101.142.93
  - 174.129.235.69
  - 184.73.197.122
  - 50.16.225.117
  - 54.83.62.94
  - 54.83.62.186
  - 54.83.63.187
  - 54.235.143.255
    上記の問題は、3scale の可用性の問題と考えられます。ただし、API が AWS ELB の背後に置かれている場合、API ゲートウェイからの API 可用性に同様の問題が発生する可能性があります。これは、デフォルトでは NGINX が起動時に DNS 解決を行ってから IP アドレスをキャッシュするためです。ただし、ELB は静的 IP アドレスを確保せず、頻繁に変わる可能性があります。ELB が別の IP に変わると、NGINX はその IP に到達できません。

    この問題の解決方法は、強制的にランタイム DNS 解決を行う上述の修正と類似しています。

    1. **http** セクションの最上部に **resolver 8.8.8.8 8.8.4.4;** という行を追加して、Google DNS などの特定の DNS リゾルバーを設定します。
    2. **server** セクションの最上部近くの任意の場所に、API のベース URL を変数として設定します**set $api_base "http://api.example.com:80";**
    3. **location** / セクション内で **proxy_pass** の行を探し、それを **proxy_pass $api_base;** に置き換えます。

4.1.3. デプロイ後の問題

新しいエンドポイントを追加するなど、API に変更を加える場合、API ゲートウェイの新しい設定ファイルのセットをダウンロードする前に、必ず新しいメソッドおよび URL マッピングを追加する必要があります。

3scale からダウンロードした設定を変更した場合の最も典型的な問題は、Lua のコードエラーです。これにより、以下のような **500 - Internal server error** が発生します。

```
curl -v -X GET "http://localhost/"
* About to connect() to localhost port 80 (#0)
*   Trying 127.0.0.1... connected
> GET / HTTP/1.1
> User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23
librtmp/2.3
> Host: localhost
> Accept: */*
>
< HTTP/1.1 500 Internal Server Error
< Server: openresty/1.5.12.1
< Date: Thu, 04 Feb 2016 10:22:25 GMT
< Content-Type: text/html
< Content-Length: 199
< Connection: close
<

<head><title>500 Internal Server Error</title></head>

<center><h1>500 Internal Server Error</h1></center>
<hr><center>openresty/1.5.12.1</center>


* Closing connection #0
```

nginx error.log を見て、原因を確認することができます。以下に例を示します。

```
2016/02/04 11:22:25 [error] 8980#0: *1 lua entry thread aborted: runtime error:
/home/pili/NGINX/troubleshooting/nginx.lua:66: bad argument #3 to '_newindex' (number expected,
got nil)
stack traceback:
coroutine 0:
  [C]: in function '_newindex'
  /home/pili/NGINX/troubleshooting/nginx.lua:66: in function 'error_authorization_failed'
  /home/pili/NGINX/troubleshooting/nginx.lua:330: in function 'authrep'
  /home/pili/NGINX/troubleshooting/nginx.lua:283: in function 'authorize'
  /home/pili/NGINX/troubleshooting/nginx.lua:392: in function  while sending to client, client:
127.0.0.1, server: api-2445581381726.staging.apicast.io, request: "GET / HTTP/1.1", host: "localhost"
```

access.log では、以下のようになります。

```
127.0.0.1 - - [04/Feb/2016:11:22:25 +0100] "GET / HTTP/1.1" 500 199 "-" "curl/7.22.0 (x86_64-pc-
linux-gnu) libcurl/7.22.0 OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3"
```

上記のセクションでは、3scale 運用のいずれかのステージで発生する可能性のある最も典型的でよく知られた問題の概要を示します。

これらをすべて確認してもなお問題の原因と解決策が見つからない場合は、より詳細な操作上のトラブルシューティングに進む必要があります (以下のトラブルシューティングのチェックリストセクションを参照)。問題のある箇所の特定を試みるため、API から始めてクライアントまで遡って作業します。

4.2. トラブルシューティングの基本

サーバーへの接続時にエラーが発生する場合、それが API ゲートウェイでも、3scale でも、またはご自分の API でも、まずは以下のトラブルシューティング手順から作業を始めてください。

4.2.1. 1. 接続の可否を確かめる

telnet を使用して、基本的な TCP/IP 接続 (**telnet api.example.com 443**) を確認します。

- 正常に接続できる場合

```
telnet echo-api.3scale.net 80
Trying 52.21.167.109...
Connected to tf-lb-i2t5pgt2cfdnbdfh2c6qqoartm-829217110.us-east-1.elb.amazonaws.com.
Escape character is '^]'.
Connection closed by foreign host.
```

- 失敗

```
telnet su1.3scale.net 443
Trying 174.129.235.69...
telnet: Unable to connect to remote host: Connection timed out
```

4.2.2. 2.問題の発生場所を特定する

さまざまなネットワークの場所、デバイス、および宛先から、同じサーバーへの接続を試みます。たとえば、クライアントが API に到達できない場合は、API ゲートウェイなど、アクセスできるはずのマシンから API への接続を試みます。

接続試行のいずれかが成功した場合、実際のサーバーに関する問題を除外して、両者間のネットワークのトラブルシューティングに集中できます。問題がここにある可能性が最も高いからです。

4.2.3. 3. DNS に問題がないか調べる

ホスト名の代わりに IP アドレスを使ってサーバーへの接続を試みます。たとえば、**telnet apis.io 80** の代わりに **telnet 94.125.104.17 80** を使用します。

これにより、DNS に関する問題はすべて排除されます。

3scale の例では、**dig su1.3scale.net** または **dig any su1.3scale.net** (ホストが解決する IP が複数あると思われる場合) のように、**dig** を使用してサーバーの IP アドレスを取得することができます。

注記: 一部のホストは dig any をブロックします

4.2.4. 4. SSL に問題がないか調べる

OpenSSL を使用して、以下の項目をテストすることができます。

- ホストまたは IP へのセキュアな接続 (たとえば、シェルプロンプトから **openssl s_client -connect su1.3scale.net:443** を実行)
  出力:

```
CONNECTED(00000003)
depth=1 C = US, O = GeoTrust Inc., CN = GeoTrust SSL CA - G3
verify error:num=20:unable to get local issuer certificate
---
```

```
Certificate chain
 0 s:/C=ES/ST=Barcelona/L=Barcelona/O=3scale Networks, S.L./OU=IT/CN=*.3scale.net
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G3
 1 s:/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G3
   i:/C=US/O=GeoTrust Inc./CN=GeoTrust Global CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIE8zCCA9ugAwIBAgIQcz2Y9JNxH7f2zpOT0DajUjANBgkqhkiG9w0BAQsFADBE
...
TRUNCATED
...
3FZigX+OpWLVRjYsr0kZzX+HCerYMwc=
-----END CERTIFICATE-----
subject=/C=ES/ST=Barcelona/L=Barcelona/O=3scale Networks,
S.L./OU=IT/CN=*.3scale.net
issuer=/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G3
---
Acceptable client certificate CA names
/C=ES/ST=Barcelona/L=Barcelona/O=3scale Networks, S.L./OU=IT/CN=*.3scale.net
/C=US/O=GeoTrust Inc./CN=GeoTrust SSL CA - G3
Client Certificate Types: RSA sign, DSA sign, ECDSA sign
Requested Signature Algorithms:
RSA+SHA512:DSA+SHA512:ECDSA+SHA512:RSA+SHA384:DSA+SHA384:ECDSA+SHA384
:RSA+SHA256:DSA+SHA256:ECDSA+SHA256:RSA+SHA224:DSA+SHA224:ECDSA+SHA22
4:RSA+SHA1:DSA+SHA1:ECDSA+SHA1:RSA+MD5
Shared Requested Signature Algorithms:
RSA+SHA512:DSA+SHA512:ECDSA+SHA512:RSA+SHA384:DSA+SHA384:ECDSA+SHA384
:RSA+SHA256:DSA+SHA256:ECDSA+SHA256:RSA+SHA224:DSA+SHA224:ECDSA+SHA22
4:RSA+SHA1:DSA+SHA1:ECDSA+SHA1
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 3281 bytes and written 499 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID:
A85EFD61D3BFD6C27A979E95E66DA3EC8F2E7B3007C0166A9BCBDA5DCA5477B8
    Session-ID-ctx:
    Master-Key:
F7E898F1D996B91D13090AE9D5624FF19DFE645D5DEEE2D595D1B6F79B1875CF935B3
A4F6ECCA7A6D5EF852AE3D4108B
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
```

```
    0000 - a8 8b 6c ac 9c 3c 60 78-2c 5c 8a de 22 88 06 15   ..l..<`x,\.."...
    0010 - eb be 26 6c e6 7b 43 cc-ae 9b c0 27 6c b7 d9 13   ..&l.{C....'l...
    0020 - 84 e4 0d d5 f1 ff 4c 08-7a 09 10 17 f3 00 45 2c   ......L.z.....E,
    0030 - 1b e7 47 0c de dc 32 eb-ca d7 e9 26 33 26 8b 8e   ..G...2....&3&..
    0040 - 0a 86 ee f0 a9 f7 ad 8a-f7 b8 7b bc 8c c2 77 7b   ..........{...w{
    0050 - ae b7 57 a8 40 1b 75 c8-25 4f eb df b0 2b f6 b7   ..W.@.u.%O...+..
    0060 - 8b 8e fc 93 e4 be d6 60-0f 0f 20 f1 0a f2 cf 46   .......`.. ....F
    0070 - b0 e6 a1 e5 31 73 c2 f5-d4 2f 57 d1 b0 8e 51 cc   ....1s.../W...Q.
    0080 - ff dd 6e 4f 35 e4 2c 12-6c a2 34 26 84 b3 0c 19   ..nO5.,.l.4&....
    0090 - 8a eb 80 e0 4d 45 f8 4a-75 8e a2 06 70 84 de 10   ....ME.Ju...p...

    Start Time: 1454932598
    Timeout   : 300 (sec)
    Verify return code: 20 (unable to get local issuer certificate)
---
```

- SSLv3 のサポート (3scale ではサポートされません)
  openssl s_client -ssl3 -connect su.3scale.net:443

  出力

```
CONNECTED(00000003)
140735196860496:error:14094410:SSL routines:ssl3_read_bytes:sslv3 alert handshake
failure:s3_pkt.c:1456:SSL alert number 40
140735196860496:error:1409E0E5:SSL routines:ssl3_write_bytes:ssl handshake
failure:s3_pkt.c:644:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 7 bytes and written 0 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : SSLv3
    Cipher    : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1454932872
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
---
```

詳細は、OpenSSL の man ページ を参照してください。

4.3. トラブルシューティングのチェックリスト

API に対するリクエストのどこに問題があるのかを特定するには、以下のリストに従って確認を行います。

4.3.1. API

API が動作状態にあり、リクエストに応答していることを確認するため、同じリクエストを API に対し直接、API ゲートウェイを経由せずに実行します。API ゲートウェイを経由する場合のリクエストと同じパラメーターおよびヘッダーを送信していることを確認する必要があります。失敗したリクエストが正確にわからない場合は、API ゲートウェイと API 間のトラフィックを取得します。

呼び出しに成功する場合、API に関する問題を除外できますが、失敗した場合には、さらに API のトラブルシューティングを行う必要があります。

4.3.2. API ゲートウェイ > API

API ゲートウェイと API 間のネットワークの問題を除外するには、前と同じ呼び出しを、API に直接、ゲートウェイサーバーから実行します。

呼び出しに成功する場合、API ゲートウェイ自体のトラブルシューティングに進むことができます。

4.3.3. API ゲートウェイ

API ゲートウェイが正常に機能していることを確認するためには、多くのステップを順に実施します。

4.3.3.1. 1. API ゲートウェイが起動して稼働しているか調べる

ゲートウェイが稼働しているマシンにログインします。これに失敗する場合、ゲートウェイサーバーがダウンしている可能性があります。

ログインしたら、NGINX プロセスが実行中であることを確認します。そのためには、**ps ax | grep nginx** または **htop** を実行します。

リストに **nginx master process** と **nginx worker process** が表示されている場合、NGINX は稼働中です。

4.3.3.2. 2. ゲートウェイログでエラーの有無を確認する

以下は、error.log のゲートウェイログで表示される可能性のある典型的なエラーの例です。

- API ゲートウェイが API に接続できない

> upstream timed out (110: Connection timed out) while connecting to upstream, client: X.X.X.X, server: api.example.com, request: "GET /RESOURCE?CREDENTIALS HTTP/1.1", upstream: "http://Y.Y.Y.Y:80/RESOURCE?CREDENTIALS", host: "api.example.com"

- API ゲートウェイが 3scale に接続できない

> 2015/11/20 11:33:51 [error] 3578#0: *1 upstream timed out (110: Connection timed out) while connecting to upstream, client: 127.0.0.1, server: , request: "GET /api/activities.json?user_key=USER_KEY HTTP/1.1", subrequest: "/threescale_authrep", upstream:

> "https://54.83.62.186:443/transactions/authrep.xml?provider_key=YOUR_PROVIDER_KEY&service_id=SERVICE_ID&usage[hits]=1&user_key=USER_KEY&log%5Bcode%5D=", host: "localhost"

4.3.4. API ゲートウェイ > 3scale

API ゲートウェイが正常に動作していることを確認したら、次のステップは API ゲートウェイと 3scale 間の接続についてのトラブルシューティングです。

4.3.4.1. 1. API ゲートウェイが 3scale にアクセスできるか調べる

API ゲートウェイに NGINX を使用している場合、ゲートウェイが 3scale と通信できないときは、以下のメッセージが NGINX エラーログに表示されます。

> 2015/11/20 11:33:51 [error] 3578#0: *1 upstream timed out (110: Connection timed out) while connecting to upstream, client: 127.0.0.1, server: , request: "GET /api/activities.json?user_key=USER_KEY HTTP/1.1", subrequest: "/threescale_authrep", upstream: "https://54.83.62.186:443/transactions/authrep.xml?provider_key=YOUR_PROVIDER_KEY&service_id=SERVICE_ID&usage[hits]=1&user_key=USER_KEY&log%5Bcode%5D=", host: "localhost"

ここでは、upstream の値に注意してください。この IP は、3scale サービスが解決する IP の 1 つに対応します。これは、3scale へのアクセスに問題があることを意味します。逆引き DNS ルックアップを実行して、**nslookup** を呼び出すことで、IP のドメインを確認することができます。

たとえば、API ゲートウェイが 3scale にアクセスできないからといって、3scale がダウンしているとは限りません。この問題の最も典型的な理由の 1 つは、API ゲートウェイが 3scale に接続することを妨げるファイアウォールルールです。

ゲートウェイと 3scale の間に、接続のタイムアウトを引き起こすネットワークの問題が存在する可能性があります。この場合、一般的な接続の問題のトラブルシューティング に関する手順を実施して、問題がどこにあるのかを特定する必要があります。

ネットワークの問題を除外するため、traceroute または MTR を使用して、ルーティングおよびパケット送信を確認します。3scale と API ゲートウェイに接続できるマシンから同じコマンドを実行し、出力を比較することもできます。

さらに、API ゲートウェイと 3scale の間で送信されているトラフィックを確認するには、一時的に 3scale サービスの HTTP エンドポイント (**su1.3scale.net**) を使用するように切り替えている限りは、tcpdump を使用できます。

4.3.4.2. 2.API ゲートウェイが 3scale のアドレスを正しく解決しているか調べる

nginx.conf にリゾルバーディレクティブが追加されていることを確認します。

nginx.conf の設定例を以下に示します。

```
http {
  lua_shared_dict api_keys 10m;
  server_names_hash_bucket_size 128;
  lua_package_path ";;$prefix/?.lua;";
  init_by_lua 'math.randomseed(ngx.time()) ; cjson = require("cjson")';

  resolver 8.8.8.8 8.8.4.4;
```

Google DNS (8.8.8.8 および 1377 8.8.4.4) は希望する DNS と置き換え可能です。

API ゲートウェイから DNS 解決を確認するには、以下のように指定したリゾルバー IP で nslookup を呼び出します。

```
nslookup su1.3scale.net 8.8.8.8
;; connection timed out; no servers could be reached
```

上記の例は、Google DNS に到達できない場合に返されるレスポンスを示しています。この場合、リゾルバー IP を更新する必要があります。nginx の error.log に、以下のアラートが表示される場合もあります。

```
2016/05/09 14:15:15 [alert] 9391#0: send() failed (1: Operation not permitted) while resolving,
resolver: 8.8.8.8:53
```

最後に、**dig any su1.3scale.net** を実行して、現在 3scale Service Management API について動作中の IP アドレスを確認します。これは、3scale によって使用される可能性のある IP アドレスの範囲全体ではないことに注意してください。容量の理由から、一部の IP アドレスがスワップインまたはスワップアウトされる場合があります。さらに、3scale サービスのドメイン名を今後追加することもできます。このため、該当する場合は、インテグレーション中に指定された特定のアドレスに対して必ずテストを行う必要があります。

4.3.4.3. 3. API ゲートウェイが 3scale を正しく呼び出していることの確認

API ゲートウェイが 3scale に送信しているリクエストを確認する場合は、トラブルシューティング用途に限り、**nginx.conf** の 3scale authrep の場所 (API キーおよび App_id 認証モードの場合は **/threescale_authrep**) に、以下のスニペットを追加することができます。

```
body_filter_by_lua_block{
  if ngx.req.get_headers()["X-3scale-debug"] == ngx.var.provider_key then
    local resp = ""
    ngx.ctx.buffered = (ngx.ctx.buffered or "") .. string.sub(ngx.arg[1], 1, 1000)
    if ngx.arg[2] then
      resp = ngx.ctx.buffered
    end

    ngx.log(0, ngx.req.raw_header())
    ngx.log(0, resp)
  end
}
```

X-3scale-debug header が送信されると (例: **curl -v -H 'X-3scale-debug: YOUR_PROVIDER_KEY' -X GET "https://726e3b99.ngrok.com/api/contacts.json?access_token=7c6f24f5"**)、このスニペットにより以下の追加ロギングが nginx error.log に追加されます。

これにより、以下のログエントリーが生成されます。

```
2016/05/05 14:24:33 [] 7238#0: *57 [lua] body_filter_by_lua:7: GET /api/contacts.json?
access_token=7c6f24f5 HTTP/1.1
Host: 726e3b99.ngrok.io
User-Agent: curl/7.43.0
Accept: */*
X-Forwarded-Proto: https
X-Forwarded-For: 2.139.235.79
```

> *while sending to client, client: 127.0.0.1, server: pili-virtualbox, request: "GET /api/contacts.json?access_token=7c6f24f5 HTTP/1.1", subrequest: "/threescale_authrep", upstream: "https://54.83.62.94:443/transactions/oauth_authrep.xml?provider_key=REDACTED&service_id=REDACTED&usage[hits]=1&access_token=7c6f24f5", host: "726e3b99.ngrok.io"*
> *2016/05/05 14:24:33 [] 7238#0: *57 [lua] body_filter_by_lua:8: <?xml version="1.0" encoding="UTF-8"?><error code="access_token_invalid">access_token "7c6f24f5" is invalid: expired or never defined</error> while sending to client, client: 127.0.0.1, server: pili-virtualbox, request: "GET /api/contacts.json?access_token=7c6f24f5 HTTP/1.1", subrequest: "/threescale_authrep", upstream: "https://54.83.62.94:443/transactions/oauth_authrep.xml?provider_key=REDACTED&service_id=REDACTED&usage[hits]=1&access_token=7c6f24f5", host: "726e3b99.ngrok.io"*

最初のエントリー (**2016/05/05 14:24:33 [] 7238#0: *57 [lua] body_filter_by_lua:7:**) は、3scale に送信されたリクエストヘッダーを出力します。この例では、Host、User-Agent、Accept、X-Forwarded-Proto、および X-Forwarded-For です。

2 番目のエントリー (**2016/05/05 14:24:33 [] 7238#0: *57 [lua] body_filter_by_lua:8:**) は、3scale からのレスポンスを出力します。この例では、**<error code="access_token_invalid">access_token "7c6f24f5" is invalid: expired or never defined</error>** となります。

どちらの要求も、元の要求 (**GET /api/contacts.json?access_token=7c6f24f5**) とサブ要求の場所 (**/threescale_authrep**) およびアップストリーム要求 (**upstream: "hits=1&access_token=7c6f24f5"**) を出力します。 この最後の値で、どの 3scale IP が解決されているかと、3scale に行った実際のリクエストも確認できます。

4.3.5. 3scale

4.3.5.1. 1. 3scale が利用可能か確かめる

3scale のステータスページ または Twitter で @3scalestatus を確認してください。

4.3.5.2. 2. 3scale がエラーを返しているか調べる

3scale は利用可能だが、呼び出しが API に通ることを妨げるエラーを API ゲートウェイに返している可能性もあります。承認呼び出しを 3scale で直接実行して、レスポンスを確認します。エラーが発生した場合は、3scale のエラーコードセクションで何が問題かを確認します。

4.3.5.3. 3. 3scale デバッグヘッダーを使用する

たとえば、以下のように **X-3scale-debug** ヘッダーを設定して API を呼び出すことで、3scale デバッグヘッダーを有効にすることもできます。

curl -v -X GET "https://api.example.com/endpoint?user_key" X-3scale-debug: YOUR_PROVIDER_KEY

これにより、API レスポンスで以下のヘッダーが返されます。

```
X-3scale-matched-rules: /, /api/contacts.json
< X-3scale-credentials: access_token=TOKEN_VALUE
< X-3scale-usage: usage[hits]=2
< X-3scale-hostname: HOSTNAME_VALUE
```

4.3.5.4. 4.インテグレーションエラーを確認する

管理ポータルでインテグレーションエラーを確認し、3scale へのトラフィックに関する問題がないか確認することもできます。https://YOUR_DOMAIN-admin.3scale.net/apiconfig/errors を参照してください。

インテグレーションエラーの理由の 1 つは、サーバーブロックでは無効な **underscores_in_headers** ディレクティブによりヘッダーでクレデンシャルを送信していることです。

4.3.6. クライアント API ゲートウェイ

4.3.6.1. 1. 一般のインターネットから API ゲートウェイにアクセスできるか調べる

ブラウザーをゲートウェイサーバーの IP アドレス (またはドメイン名) に転送するよう試みます。これに失敗する場合、該当するポートのファイアウォールが開いていることを確認してください。

4.3.6.2. 2. クライアントから API ゲートウェイにアクセスできるか調べる

可能な場合は、前述の方法 (telnet、curl など) のいずれかを使用して、クライアントから API ゲートウェイへの接続を試みます。 接続に失敗する場合、クライアントと API ゲートウェイ間のネットワークに問題が発生しています。

そうでない場合は、API への呼び出しを行うクライアントのトラブルシューティングに進む必要があります。

4.3.7. クライアント

4.3.7.1. 1. 別のクライアントを使用して同じ呼び出しをテストする

リクエストが想定される結果を返さない場合は、別の HTTP クライアントでテストします。たとえば、Java HTTP クライアントで API を呼び出している時に何らかの問題が生じる場合、cURL を使用して結果を照合します。

クライアントとゲートウェイ間のプロキシー経由で API を呼び出し、クライアントが送信している正確なパラメーターとヘッダーを取得することもできます。

4.3.7.2. 2. クライアントから送信されたトラフィックを確認する

Wireshark などのツールを使用して、クライアントが送信しているリクエストを調べます。これにより、クライアントが API を呼び出しているかどうか、およびリクエストの詳細を確認することができます。

4.4. その他の問題

4.4.1. ActiveDocs の問題

コマンドラインから API を呼び出す場合には機能するが、ActiveDocs を経由する場合には失敗することがあります。

ActiveDocs 呼び出しを機能させるために、これらの呼び出しを 3scale 側のプロキシー経由で送信します。このプロキシーが追加する特定のヘッダーが API にとって想定外だった場合に、問題を引き起こす可能性があります。これを確認するには、以下の手順を試みます。

4.4.1.1. 1. petstore.swagger.io を使用する

Swagger では、petstore.swagger.io にホスト型の swagger-ui が用意されています。これを使用して、最新バージョンの swagger-ui により Swagger 仕様と API をテストすることができます。swagger-ui と ActiveDocs の両方が同じように失敗する場合、ActiveDocs や ActiveDocs プロキシーの問題は除外して、ご自分の仕様のトラブルシューティングに集中できます。あるいは、swagger-ui GitHub リポジトリーで、現在の swagger-ui のバージョンに関する既知の問題を確認できます。

4.4.1.2. 2. ファイアウォールが ActiveDocs プロキシーからの接続を許可していることを確認する

API を使用するクライアントの IP アドレスをホワイトリスト化しないよう推奨しています。ActiveDocs プロキシーは、高可用性を実現するためにフローティング IP アドレスを使用していますが、現在これらの IP の変更を通知する仕組みはありません。

4.4.1.3. 3. 無効なクレデンシャルを使用して API を呼び出す

ActiveDocs プロキシーが正しく機能しているかどうかを確認する方法の 1 つは、無効なクレデンシャルを使用して API を呼び出すことです。これにより、ActiveDocs プロキシーと API ゲートウェイの両方について、問題の有無を確認することができます。

API 呼び出しから 403 コード (または不正なクレデンシャルに対してゲートウェイで設定しているコード) が返される場合、呼び出しはゲートウェイに到達しているので、問題は API にあります。

4.4.1.4. 4. 呼び出しを比較する

ActiveDocs から行った呼び出しと ActiveDocs 外からの呼び出し間でヘッダーおよびパラメーターの相違点を特定するには、API に送信する前に HTTP 呼び出しを検証および比較できる特定のサービス (オンプレミスの APItools、Runscope など) を介して呼び出しを実行することが有益な場合もあります。これにより、API プロバイダー側で問題を引き起こす可能性のあるリクエスト内のヘッダーやパラメーターを特定することができます。

4.5. 付録

4.5.1. NGINX でのロギング

これについての包括的なガイドは、NGINX のロギングとモニターリング に関するドキュメントを参照してください。

4.5.1.1. デバッグログの有効化

デバッグログの有効化の詳細については、NGINX のデバッグログに関するドキュメント を参照してください。

4.5.2. 3scale のエラーコード

3scale Service Management API エンドポイントによって返されるエラーコードを確認するには、以下の手順に従って **3scale API Documentation** のページを参照します。

1. 管理ポータルの右上隅にある疑問符 (?) アイコンをクリックします。
2. **3scale API Docs** を選択します。

以下は、3scale によって返される HTTP レスポンスコードと、そのコードが返される条件の一覧です。

- 400: 不正なリクエスト。原因は以下のとおりです。
  - エンコーディングが無効である。
  - 負荷が大きすぎる。
  - コンテンツタイプが無効 (POST 呼び出しの場合) である。**Content-Type** ヘッダーの有効な値は、**application/x-www-form-urlencoded**、**multipart/form-data**、または空のヘッダーです。
- 403:
  - クレデンシャルが有効ではない。
  - 3scale に GET リクエスト用のボディーデータを送信している
- 404: アプリケーションやメトリクスなど、存在しないエンティティーが参照されている
- 409:
  - 使用制限の超過。
  - アプリケーションがアクティブではない。
  - アプリケーションキーが無効、または提供されない (**app_id/app_key** 認証メソッドの場合)。
  - 参照元が許可されていない、または提供されない (参照元フィルターが有効で必要な場合)
- 422: 必要なパラメーターが提供されない

これらのエラーレスポンスのほとんどには、マシンリーダブルなエラーカテゴリーと人が判読できる説明が含まれる XML ボディーも含まれています。

標準の API ゲートウェイ設定を使用する場合、3scale から 200 以外のコードが返されると、クライアントには以下のどちらかのコードと共にレスポンスが返されます。

- 403
- 404