



Red Hat 3scale API Management 2.2

インフラストラクチャー

異なるプラットフォームへの Red Hat 3scale API Management のデプロイメントについて詳しく知る。

Red Hat 3scale API Management 2.2 インフラストラクチャー

異なるプラットフォームへの Red Hat 3scale API Management のデプロイメントについて詳しく知る。

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

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Infrastructure.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、Red Hat 3scale API Management 2.2 のデプロイメントおよびインフラストラクチャの管理について説明します。

目次

第1章 3SCALE 2.1から 2.2 へのアップグレード	5
1.1. 前提条件:	5
1.2. プロジェクトの選択	5
1.3. 必要値の収集	5
1.4. 新規変数値の設定	7
1.5. データベース POD のアップグレード	8
1.5.1. バックアップの作成	8
1.5.2. アップグレードを実行する	10
1.5.3. MySQL 文字セットの変更および照合	11
1.5.4. バックアップの削除	13
1.6. システムの新規ルートおよびサービスの作成	13
1.7. パッチシステムコンポーネント	14
1.8. パッチバックエンドコンポーネント	29
1.9. APICAST のパッチ適用	32
1.10. ZYNC コンポーネントにパッチを適用	35
1.11. アップグレードを確認	36
第2章 API DEPLOYMENT ON MICROSOFT AZURE	37
2.1. MICROSOFT AZURE 仮想マシンの作成および設定	37
2.2. INSTALL OPENRESTY	38
2.3. GITHUB リポジトリの設定	38
2.3.1. 警告	38
2.4. API の設定	38
2.4.1. 3scale の場合	38
2.4.2. Capistrano の設定	41
2.5. CAPISTRANO の設定	41
2.6. デプロイ	43
2.6.1. トラブルシューティング	43
第3章 AWS ROOKIES 向けの AMAZON EC2 での API のデプロイ	45
3.1. 前提条件	45
3.2. EC2 インスタンスの作成および設定	45
3.3. デプロイメント用のインスタンスの準備	45
3.4. アプリケーションのデプロイ	46
3.4.1. Optional	47
3.5. 3SCALE での API 管理の有効化	47
3.6. APICAST のインストールとデプロイ（ご自分の API ゲートウェイ）	49
第4章 ORACLE DATABASE のリレーショナルデータベース管理システムを使用した 3SCALE API MANAGEMENT システムイメージのビルド	50
4.1. 始める前に	50
4.1.1. Oracle ソフトウェアコンポーネントの取得	50
4.1.2. 会議の前提条件	50
4.2. ORACLE DATABASE の準備	50
4.3. システムイメージのビルド	51
第5章 オンプレミス型 3SCALE AMP インストールガイド	53
5.1. 前提条件	53
5.2. 3SCALE AMP OPENSIFT テンプレート	53
5.3. システム要件	53
5.3.1. 環境要件	53
5.3.2. ハードウェア要件	53

5.4. ノードとエンタイトルメントの設定	54
5.5. テンプレートを使用した OPENSIFT への 3SCALE AMP のデプロイ	54
5.5.1. 前提条件:	54
5.5.2. AMP テンプレートのインポート	55
5.5.3. SMTP 変数の設定 (任意)	56
5.6. 3SCALE AMP テンプレートパラメーター	57
5.7. OPENSIFT 上の AMP と共に APICAST を使用	60
5.7.1. AMP に関する既存の OpenShift クラスターへの APICAST テンプレートのデプロイ	60
5.7.2. AMP に関する OpenShift クラスターとは別の OpenShift クラスターからの APICAST への接続	61
5.7.3. 他のデプロイメントからの APICAST の接続	61
5.7.4. 組み込み APICAST のデフォルト動作の変更	62
5.7.5. 内部サービスルートを紹介した単一 OpenShift クラスターでの複数 APICAST デプロイメントの接続	62
5.8. 7.トラブルシューティング	63
5.8.1. 以前の Deployment Leaves Dirty Persistent Volume Claim (永続ボリューム要求、PVC)	63
5.8.2. Docker レジストリーから誤ってプルする	64
5.8.3. 永続ボリュームがローカルでマウントされる場合の MySQL の権限の問題	64
5.8.4. 永続ボリュームが OpenShift によって Writable ではないため、Logo または Images をアップロードできない	65
5.8.5. OpenShift でのセキュリティ保護されたルートの作成	65
5.8.6. Secret の問題が原因で AMP の別のプロジェクトでの APICAST デプロイに失敗する	65
第6章 RED HAT 3SCALE AMP 2.2 ON-PREMISES OPERATIONS AND SCALING GUIDE	67
6.1. はじめに	67
6.1.1. 前提条件	67
6.1.2. 詳細はこちら	67
6.2. APICAST の再デプロイ	67
6.3. APICAST BUILT-IN ワイルドカードルーティング (テクノロジープレビュー)	68
6.3.1. ワイルドカードの変更	68
6.4. オンプレミス型 AMP のスケールアップ	68
6.4.1. ストレージのスケールアップ	68
6.4.1.1. 方法 1、バックアップ、および Swap 永続ボリューム	69
6.4.1.2. 方法 2:AMP のバックアップおよび再デプロイ	69
6.4.2. パフォーマンスのスケールアップ	69
6.4.2.1. オンプレミス型 3scale デプロイメントの設定	70
6.4.2.2. 垂直および Horizontalハードウェアのスケールアップ	70
6.4.2.3. ルーターのスケールアップ	70
6.4.2.4. 詳細はこちら	71
6.5. 操作のトラブルシューティング	71
6.5.1. ログへのアクセス	71
6.5.2. ジョブキュー	71
第7章 HOW TO DEPLOYING A FULL-STACK API SOLUTION WITH FUSE, 3SCALE, AND OPENSIFT	72
7.1. パート 1: FUSE ON OPENSIFT の設定	73
7.1.1. ステップ 1	73
7.1.2. ステップ 2	74
7.1.3. ステップ 3	74
7.1.4. ステップ 4	75
7.1.5. ステップ 5	76
7.1.6. ステップ 6	77
7.1.7. ステップ 7	77
7.1.8. ステップ 8	78
7.1.9. ステップ 9	78
7.1.10. ステップ 10	79

7.1.11. ステップ 11	80
7.2. パート 2: 3SCALE API MANAGEMENT の設定	80
7.2.1. ステップ 1	80
7.2.2. ステップ 2	80
7.2.3. ステップ 3	81
7.2.4. ステップ 4	82
7.2.5. ステップ 5	83
7.3. パート 3: API サービスの統合	83
7.4. パート 4: API および API 管理のテスト	84
7.4.1. ステップ 1	84
7.4.2. ステップ 2	84
7.4.3. ステップ 3	85
7.4.4. ステップ 4	85
7.4.5. ステップ 5	85

第1章 3SCALE 2.1 から 2.2 へのアップグレード

本セクションの手順を実施して、オンプレミス型 Red Hat 3scale API Management のデプロイメントをバージョン 2.1 から 2.2 にアップグレードします。

1.1. 前提条件:

- オンプレミス型 3scale 2.1
- OpenShift CLI
- 3scale AMP 2.2 テンプレート
- OpenShift サーバーおよびプロジェクトへのアクセスおよびパーミッション
- MySQL データベースのバックアップを保持するのに十分な容量を持つ永続ボリューム。



警告

このプロセスにより、サービスが中断される可能性があります。メンテナンス期間があることを確認してください。

1.2. プロジェクトの選択

1. OpenShift [クラスターのバックアップを作成します](#)。
2. ターミナルセッションから、OpenShift クラスターにログインします。

```
oc login https://<YOUR_OPENSHIFT_CLUSTER>:8443
```

3. アップグレードするプロジェクトを選択します。

```
oc project <YOUR_AMP_21_PROJECT>
```

1.3. 必要値の収集

新しい 3scale API Management 2.2 マルチテナンシー機能には、以下のパラメーターが必要です。これらのパラメーターに新しい値を指定するか、デフォルトを残すか、いずれかを選択できます。

パラメーター	説明
MASTER_USER	マスター管理ポータルของผู้ใช้ชื่อเริ่มต้น : "master"
MASTER_PASSWORD	マスター管理ポータルのパスワード指定のない場合は自動生成

パラメーター	説明
MASTER_ACCESS_TOKEN	アップグレード時に自動生成されたマスターのアクセストークン。ただし、MASTER_ACCESS_TOKEN はデフォルトではシードされたシステムには追加されません。
APICAST_REGISTRY_URL	APICAST ポリシーレジストリー管理を参照する URL。デフォルト: http://apicast-staging:8090/policies

1. 現在の 2.1 デプロイメントのシステムコンポーネントから以下の値を収集します。

- DATABASE_URL
- THREESCALE_SUPERDOMAIN
- TENANT_NAME
- APICAST_ACCESS_TOKEN
- ADMIN_ACCESS_TOKEN
- USER_LOGIN
- USER_PASSWORD
- EVENTS_SHARED_SECRET
- APICAST_BACKEND_ROOT_ENDPOINT
- CONFIG_INTERNAL_API_USER
- CONFIG_INTERNAL_API_PASSWORD
- SECRET_KEY_BASE
- BACKEND_ROUTE

2. 現在のデプロイメントからアクティブなシェルにエクスポートします。

```
export `oc env dc/system-app --list | grep -E
'^((DATABASE_URL|THREESCALE_SUPERDOMAIN|TENANT_NAME|APICAST_ACCESS_
TOKEN|ADMIN_ACCESS_TOKEN|USER_LOGIN|USER_PASSWORD|EVENTS_SHARED_
SECRET|APICAST_BACKEND_ROOT_ENDPOINT|CONFIG_INTERNAL_API_USER|CON
FIG_INTERNAL_API_PASSWORD|SECRET_KEY_BASE|BACKEND_ROUTE)= ' | tr "\n" ' '`
```

3. オプションで、OpenShift CLI から個別の値をクエリーするには、以下の **oc get** コマンドを実行します。**<variable_name>** はクエリーする変数名に置き換えます。

```
oc get "-o=custom-columns=NAME:.spec.template.spec.containers[0].env[?(.name==\"
<variable_name>\")].value" dc/system-app
```

4. 現在の 2.1 デプロイメントの **system-mysql** コンポーネントから以下の値を収集します。

- MYSQL_USER
- MYSQL_PASSWORD
- MYSQL_DATABASE
- MYSQL_ROOT_PASSWORD

5. これらの値を現在のデプロイメントからアクティブなシェルにエクスポートします。

```
export `oc env dc/system-mysql --list | grep -E
'^ (MYSQL_USER|MYSQL_PASSWORD|MYSQL_DATABASE|MYSQL_ROOT_PASSWORD)
=' | tr "\n" ' '`
```

6. オプションで、OpenShift CLI から個別の値をクエリーするには、以下の **oc get** コマンドを実行します。**<variable_name>** はクエリーする変数名に置き換えます。

```
oc get "-o=custom-columns=NAMEs:.spec.template.spec.containers[0].env[?(.name==\"
<variable_name>\")].value" dc/system-mysql
```

7. 現在の 2.1 デプロイメントの **APICAST** コンポーネントから以下の値を収集します。

- APICAST_MANAGEMENT_API
- OPENSSL_VERIFY
- APICAST_RESPONSE_CODES

8. これらの値を現在のデプロイメントからアクティブなシェルにエクスポートします。

```
export `oc env dc/apicast-production --list | grep -E
'^ (APICAST_MANAGEMENT_API|OPENSSL_VERIFY|APICAST_RESPONSE_CODES)= ' |
tr "\n" ' '`
```

9. オプションで、OpenShift CLI から個別の値をクエリーするには、以下の **oc get** コマンドを実行します。**<variable_name>** はクエリーする変数名に置き換えます。

```
oc get "-o=custom-columns=NAMEs:.spec.template.spec.containers[0].env[?(.name==\"
<variable_name>\")].value" dc/apicast-production
```

1.4. 新規変数値の設定

1. AMP リリースの新しいバージョンの値を設定します。

```
export AMP_RELEASE=2.2.0
```

2. AMP 2.2 で導入された新しいオプションパラメーターの値を設定します。[これらのパラメーターは、収集が必要な値セクションの先頭に記述されます。](#) **export** コマンドを使用して、括弧の値を置き換えます。

```
export MASTER_ACCESS_TOKEN=<MASTER_ACCESS_TOKEN>
export APICAST_REGISTRY_URL=<APICAST_REGISTRY_URL>
```

MASTER_USER および MASTER_PASSWORD については、以下を考慮してください。

- デフォルト値を使用する場合は、アクションは必要ありません。
- これらの環境変数に値を指定した場合は、以下のコマンドでそれらをエクスポートします。

```
export MASTER_USER=<MASTER_USER>
export MASTER_PASSWORD=<MASTER_PASSWORD>
```

3. 収集する必要な値のセクションに収集された必要な値がアクティブなシェルにエクスポートされ、このセクションで新しい値が設定されることを確認します。

```
echo AMP_RELEASE=$AMP_RELEASE

echo DATABASE_URL=$DATABASE_URL
echo THREESCALE_SUPERDOMAIN=$THREESCALE_SUPERDOMAIN
echo TENANT_NAME=$TENANT_NAME
echo APICAST_ACCESS_TOKEN=$APICAST_ACCESS_TOKEN
echo ADMIN_ACCESS_TOKEN=$ADMIN_ACCESS_TOKEN
echo USER_LOGIN=$USER_LOGIN
echo USER_PASSWORD=$USER_PASSWORD
echo EVENTS_SHARED_SECRET=$EVENTS_SHARED_SECRET
echo
APICAST_BACKEND_ROOT_ENDPOINT=$APICAST_BACKEND_ROOT_ENDPOINT
echo CONFIG_INTERNAL_API_USER=$CONFIG_INTERNAL_API_USER
echo CONFIG_INTERNAL_API_PASSWORD=$CONFIG_INTERNAL_API_PASSWORD
echo SECRET_KEY_BASE=$SECRET_KEY_BASE
echo BACKEND_ROUTE=$BACKEND_ROUTE

echo MYSQL_USER=$MYSQL_USER
echo MYSQL_PASSWORD=$MYSQL_PASSWORD
echo MYSQL_DATABASE=$MYSQL_DATABASE
echo MYSQL_ROOT_PASSWORD=$MYSQL_ROOT_PASSWORD

echo APICAST_MANAGEMENT_API=$APICAST_MANAGEMENT_API
echo OPENSLL_VERIFY=$OPENSLL_VERIFY
echo APICAST_RESPONSE_CODES=$APICAST_RESPONSE_CODES

echo MASTER_USER=$MASTER_USER
echo MASTER_PASSWORD=$MASTER_PASSWORD
echo MASTER_ACCESS_TOKEN=$MASTER_ACCESS_TOKEN
echo APICAST_REGISTRY_URL=$APICAST_REGISTRY_URL
```

1.5. データベース POD のアップグレード

データベースをアップグレードするには、Pod のバックアップを作成し、新規 Pod をデプロイします。

1.5.1. バックアップの作成

1. MySQL データベースを保存するのに十分なストレージのある永続ボリュームを作成します。

- MySQL データベースを保持するのに十分なストレージで Persistent Volume Claim (永続ボリューム要求、PVC) を作成するには、以下のコマンドを実行します。**<size>** の値をデータベースに適したサイズに置き換えます。

```
echo "apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-backup
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>" | oc create -f -
```

このコマンドは、**mysql-backup** という名前の永続ボリューム要求(PVC)を調整します。

- Pod を作成してバックアップデータベースを格納するには、以下のコマンドを実行します。

```
echo "apiVersion: v1
kind: Pod
metadata:
  name: mysql-backup
  labels:
    name: mysql-backup
spec:
  containers:
    - name: mysql-backup
      image: registry.access.redhat.com/rhsccl/mysql-57-rhel7:5.7-5
      args:
        - sleep
        - infinity
      volumeMounts:
        - mountPath: /backup
          name: mysql-backup
  volumes:
    - name: mysql-backup
      persistentVolumeClaim:
        claimName: mysql-backup" | oc create -f -
```

- 以下の **oc rsh** コマンドを使用して Pod が作成され、バックアップを作成します。

```
oc rsh mysql-backup /opt/rh/rh-mysql57/root/usr/bin/mysqldump -h system-mysql -u
${MYSQL_USER} -p${MYSQL_PASSWORD} system -r /backup/backup.sql
```

- バックアップの内容を確認するには、以下の **oc rsh** コマンドを実行して、元のファイルサイズに対してバックアップファイルのサイズを確認します。

```
oc rsh mysql-backup ls -lha /backup/backup.sql
```



注記

障害が発生した場合には、以下の **oc rsh** コマンドを使用して、バックアップからデータベースを再デプロイできます。

```
oc rsh mysql-backup /bin/bash -c "/usr/bin/cat /backup/backup.sql | /opt/rh/rh-mysql57/root/usr/bin/mysql -h system-mysql -uroot -p${MYSQL_ROOT_PASSWORD} system"
```

1.5.2. アップグレードを実行する

1. 以下の **oc delete** コマンドを使用して、システムサービスを削除します。アプリケーションがデータベースとの接続が失われるため、この時点でアプリケーションが停止される点に注意してください。

```
oc delete service system-mysql
```

2. MySQL DeploymentConfig にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/system-mysql -p "spec:
template:
spec:
containers:
- name: system-mysql
image: registry.access.redhat.com/rhsc1/mysql-57-rhel7:5.7-5
args:
- /opt/rh/rh-mysql57/root/usr/libexec/mysqld
- '--datadir=/var/lib/mysql/data/"
```

3. 続行する前に、新規 Pod が正常にデプロイされていることを確認します。
4. 新しい Pod の詳細を取得するには、以下の **oc get** コマンドを実行します。
 - a. Pod 名を取得するには、以下を実行します。

```
oc get pods -l deploymentconfig=system-mysql
```

- b. Pod IP アドレスを取得するには、以下を実行します。

```
oc get pods -o=custom-columns=IP:.status.podIP -l deploymentconfig=system-mysql
```

5. 以下の **oc rsh** コマンドを使用して Pod にログインします。<pod_name> および <pod_ip> を前述のステップの名前および IP アドレスに置き換えます。

```
oc rsh <pod_name> /opt/rh/rh-mysql57/root/usr/bin/mysql_upgrade -h <pod_ip> -u root -p${MYSQL_ROOT_PASSWORD}
```

6. 2.2 バージョンで行われた変更で mysql Deployment Config にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/system-mysql -p "
```

```

metadata:
  labels:
    app: System
spec:
  template:
    spec:
      containers:
      - name: system-mysql
        args:
          imagePullPolicy: IfNotPresent
        resources:
          limits:
            memory: 2Gi
          requests:
            cpu: 250m
            memory: 512Mi
"

```

1.5.3. MySQL 文字セットの変更および照合

1. mysql-charset.cnf 設定ファイルを使用して **mysql-extra-conf** ConfigMap を作成するには、以下の **oc create** コマンドを実行します。

```

echo "kind: ConfigMap
apiVersion: v1
metadata:
  name: mysql-extra-conf
data:
  mysql-charset.cnf: |
    [client]
    default-character-set = utf8

    [mysql]
    default-character-set = utf8

    [mysqld]
    character-set-server = utf8
    collation-server = utf8_unicode_ci" | oc create -f -

```

2. **mysql-main-conf** ConfigMap を作成するには、以下の **oc create** コマンドを実行します。

```

echo 'kind: ConfigMap
apiVersion: v1
metadata:
  name: mysql-main-conf
data:
  my.cnf: |
    !include /etc/my.cnf
    !includedir /etc/my-extra.d' | oc create -f -

```

3. 最後の手順で作成された configmap で始まる system-mysql を設定するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/system-mysql -p "spec:

```

```

template:
spec:
  containers:
  - name: system-mysql
    env:
      - name: MYSQL_USER
        value: "${MYSQL_USER}"
      - name: MYSQL_PASSWORD
        value: "${MYSQL_PASSWORD}"
      - name: MYSQL_DATABASE
        value: "${MYSQL_DATABASE}"
      - name: MYSQL_ROOT_PASSWORD
        value: "${MYSQL_ROOT_PASSWORD}"
      - name: MYSQL_LOWER_CASE_TABLE_NAMES
        value: '1'
      - name: MYSQL_DEFAULTS_FILE
        value: "/etc/my-extra/my.cnf"
    volumeMounts:
      - name: 'mysql-storage'
        mountPath: /var/lib/mysql/data
      - name: 'mysql-extra-conf'
        mountPath: /etc/my-extra.d
      - name: 'mysql-main-conf'
        mountPath: /etc/my-extra
    volumes:
      - name: 'mysql-storage'
        persistentVolumeClaim:
          claimName: 'mysql-storage'
      - name: 'mysql-extra-conf'
        configMap:
          name: 'mysql-extra-conf'
      - name: 'mysql-main-conf'
        configMap:
          name: 'mysql-main-conf'

```

4. デプロイメントが system-mysql で完了し、以下の **oc get** コマンドを実行して新しい MySQL Pod 名を取得するまで待ちます。

```
oc get pods -l deploymentconfig=system-mysql
```

5. 新しい MySQL Pod IP アドレスを取得するには、以下の **oc get** コマンドを実行します。

```
oc get pods -o=custom-columns=IP:.status.podIP -l deploymentconfig=system-mysql
```

6. データベースとすべてのテーブルに設定された文字を変更するには、**oc rsh** コマンドを実行し、取得された **<pod_name>** および **<pod_ip>** を指定します。

```

oc rsh <pod_name> /bin/bash -c "echo ALTER DATABASE system CHARACTER SET utf8
COLLATE utf8_general_ci | mysql -h <pod_ip> -u root -p${MYSQL_ROOT_PASSWORD} --
default-character-set=utf8"
oc rsh <pod_name> /bin/bash -c "/opt/rh/rh-mysql57/root/usr/bin/mysql -h <pod_ip> -u root -
p${MYSQL_ROOT_PASSWORD} --default-character-set=utf8 -B -N -e 'SHOW TABLES'
system | awk '{print \"SET foreign_key_checks = 0; ALTER TABLE\", \$1, \"CONVERT TO

```



```
CHARACTER SET utf8 COLLATE utf8_general_ci; SET foreign_key_checks = 1; \"} |
/opt/rh/rh-mysql57/root/usr/bin/mysql -h <pod_ip> -u root -p${MYSQL_ROOT_PASSWORD}
--default-character-set=utf8 system"
```

7. **system-mysql** サービスを作成するには、以下の **oc create** コマンドを実行します。

```
echo "kind: Service
apiVersion: v1
metadata:
  name: 'system-mysql'
spec:
  ports:
    - name: system-mysql
      protocol: TCP
      port: 3306
      targetPort: 3306
      nodePort: 0
  selector:
    name: 'system-mysql'" | oc create -f -
```

1.5.4. バックアップの削除

1. 更新されたデータベースを確認し、Pod が実行されていることを確認します。
2. バックアップ Pod および永続ボリューム要求を削除するには、以下の **oc delete** コマンドを実行します。

```
oc delete pod/mysql-backup
oc delete pvc/mysql-backup
```

1.6. システムの新規ルートおよびサービスの作成

新しい変数の値を設定したら、以下の **oc create** コマンドを実行して新規ルートおよびサービスを作成します。

```
echo "
apiVersion: v1
kind: Service
metadata:
  name: system-master
annotations:
  service.alpha.openshift.io/dependencies: '[{"name": "system-developer", "kind": "Service"}]'
spec:
  ports:
    - port: 3000
      protocol: TCP
      targetPort: master
      name: http
  selector:
    name: system-app
" | oc create -f -
```

```
echo "
```

```

apiVersion: v1
kind: Route
metadata:
  name: system-master-admin-route
spec:
  host: master-account-admin.${THREESCALE_SUPERDOMAIN}
  to:
    kind: Service
    name: system-master
  port:
    targetPort: http
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Allow
" | oc create -f -

```

1.7. パッチシステムコンポーネント

oc patch コマンドを使用してインプレースアップグレードを続行します。**oc patch** コマンドを使用すると、デプロイメント設定、イメージストリームおよび ConfigMap にパッチを適用できます。

このアップグレードのセクションで、システム設定マップにパッチを適用する必要があります。以下の Pod のデプロイメント設定にパッチを適用する必要があります。

- system-app
- system-resque
- system-sidekiq
- system-sphinx

設定マップおよびデプロイメント設定にパッチを適用するには、以下の手順に従います。

1. **system** ConfigMap にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch cm/system -p "
data:
  zync.yml: |
    production:
      endpoint: 'http://zync:8080'
      authentication:
        token: \"<%= ENV.fetch('ZYNC_AUTHENTICATION_TOKEN') %>\"
      connect_timeout: 5
      send_timeout: 5
      receive_timeout: 10
      root_url:
    rolling_updates.yml: |
      production:
        old_charts: false
        new_provider_documentation: false
        proxy_pro: false
        instant_bill_plan_change: false
        service_permissions: true
        async_apicast_deploy: false
        duplicate_application_id: true

```

```

duplicate_user_key: true
plan_changes_wizard: false
require_cc_on_signup: false
apicast_per_service: true
new_notification_system: true
cms_api: false
apicast_v2: true
forum: false
published_service_plan_signup: true
apicast_oidc: true
policies: true"

```

2. **system-resque** デプロイメント設定にパッチを適用するには、以下を考慮してください。

- MASTER_USER および MASTER_PASSWORD 環境変数のデフォルト値を使用する場合は、以下の **oc patch** コマンドでそれらを記述しないでください。
- または、MASTER_USER および MASTER_PASSWORD の値を指定する場合は、以下の **oc patch** コマンドに追加します。

```

oc patch dc/system-resque -p "
metadata:
  labels:
    app: System
spec:
  template:
    spec:
      containers:
      - env:
        - name: RAILS_ENV
          value: \"production\"
        - name: DATABASE_URL
          value: \"${DATABASE_URL}\"
        - name: FORCE_SSL
          value: \"true\"
        - name: THREESCALE_SUPERDOMAIN
          value: \"${THREESCALE_SUPERDOMAIN}\"
        - name: MASTER_USER
          value: \"${MASTER_USER}\"
        - name: MASTER_PASSWORD
          value: \"${MASTER_PASSWORD}\"
        - name: TENANT_NAME
          value: \"${TENANT_NAME}\"
        - name: APICAST_ACCESS_TOKEN
          value: \"${APICAST_ACCESS_TOKEN}\"
        - name: ADMIN_ACCESS_TOKEN
          value: \"${ADMIN_ACCESS_TOKEN}\"
        - name: PROVIDER_PLAN
          value: 'enterprise'
        - name: USER_LOGIN
          value: \"${USER_LOGIN}\"
        - name: USER_PASSWORD
          value: \"${USER_PASSWORD}\"
        - name: RAILS_LOG_TO_STDOUT
          value: \"true\"
        - name: RAILS_LOG_LEVEL

```

```
value: \"info\"
- name: THINKING_SPHINX_ADDRESS
  value: \"system-sphinx\"
- name: THINKING_SPHINX_PORT
  value: \"9306\"
- name: THINKING_SPHINX_CONFIGURATION_FILE
  value: \"/tmp/sphinx.conf\"
- name: EVENTS_SHARED_SECRET
  value: \"${EVENTS_SHARED_SECRET}\"
- name: THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE
  value: \"VERIFY_NONE\"
- name: APICAST_BACKEND_ROOT_ENDPOINT
  value: \"${APICAST_BACKEND_ROOT_ENDPOINT}\"
- name: CONFIG_INTERNAL_API_USER
  value: \"${CONFIG_INTERNAL_API_USER}\"
- name: CONFIG_INTERNAL_API_PASSWORD
  value: \"${CONFIG_INTERNAL_API_PASSWORD}\"
- name: SECRET_KEY_BASE
  value: \"${SECRET_KEY_BASE}\"
- name: AMP_RELEASE
  value: \"${AMP_RELEASE}\"
- name: ZYNC_AUTHENTICATION_TOKEN
  valueFrom:
    secretKeyRef:
      name: zync
      key: ZYNC_AUTHENTICATION_TOKEN
- name: SMTP_ADDRESS
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: address
- name: SMTP_USER_NAME
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: username
- name: SMTP_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: password
- name: SMTP_DOMAIN
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: domain
- name: SMTP_PORT
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: port
- name: SMTP_AUTHENTICATION
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: authentication
```

```
- name: SMTP_OPENSSL_VERIFY_MODE
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: openssl.verify.mode
- name: BACKEND_ROUTE
  value: \"${BACKEND_ROUTE}\"
- name: SSL_CERT_DIR
  value: \"/etc/pki/tls/certs\"
- name: APICAST_REGISTRY_URL
  value: \"${APICAST_REGISTRY_URL}\"
image: registry.access.redhat.com/3scale-amp22/system:1.7
imagePullPolicy: IfNotPresent
name: system-resque
resources:
  limits:
    cpu: 150m
    memory: 450Mi
  requests:
    cpu: 100m
    memory: 300Mi
- env:
  - name: RAILS_ENV
    value: \"production\"
  - name: DATABASE_URL
    value: \"${DATABASE_URL}\"
  - name: FORCE_SSL
    value: \"true\"
  - name: THREESCALE_SUPERDOMAIN
    value: \"${THREESCALE_SUPERDOMAIN}\"
  - name: MASTER_USER
    value: \"${MASTER_USER}\"
  - name: MASTER_PASSWORD
    value: \"${MASTER_PASSWORD}\"
  - name: TENANT_NAME
    value: \"${TENANT_NAME}\"
  - name: APICAST_ACCESS_TOKEN
    value: \"${APICAST_ACCESS_TOKEN}\"
  - name: ADMIN_ACCESS_TOKEN
    value: \"${ADMIN_ACCESS_TOKEN}\"
  - name: PROVIDER_PLAN
    value: 'enterprise'
  - name: USER_LOGIN
    value: \"${USER_LOGIN}\"
  - name: USER_PASSWORD
    value: \"${USER_PASSWORD}\"
  - name: RAILS_LOG_TO_STDOUT
    value: \"true\"
  - name: RAILS_LOG_LEVEL
    value: \"info\"
  - name: THINKING_SPHINX_ADDRESS
    value: \"system-sphinx\"
  - name: THINKING_SPHINX_PORT
    value: \"9306\"
  - name: THINKING_SPHINX_CONFIGURATION_FILE
    value: \"/tmp/sphinx.conf\"
```

```
- name: EVENTS_SHARED_SECRET
  value: "${EVENTS_SHARED_SECRET}"
- name: THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE
  value: "VERIFY_NONE"
- name: APICAST_BACKEND_ROOT_ENDPOINT
  value: "${APICAST_BACKEND_ROOT_ENDPOINT}"
- name: CONFIG_INTERNAL_API_USER
  value: "${CONFIG_INTERNAL_API_USER}"
- name: CONFIG_INTERNAL_API_PASSWORD
  value: "${CONFIG_INTERNAL_API_PASSWORD}"
- name: SECRET_KEY_BASE
  value: "${SECRET_KEY_BASE}"
- name: AMP_RELEASE
  value: "${AMP_RELEASE}"
- name: ZYNC_AUTHENTICATION_TOKEN
  valueFrom:
    secretKeyRef:
      name: zync
      key: ZYNC_AUTHENTICATION_TOKEN
- name: SMTP_ADDRESS
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: address
- name: SMTP_USER_NAME
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: username
- name: SMTP_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: password
- name: SMTP_DOMAIN
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: domain
- name: SMTP_PORT
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: port
- name: SMTP_AUTHENTICATION
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: authentication
- name: SMTP_OPENSSL_VERIFY_MODE
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: openssl.verify.mode
- name: BACKEND_ROUTE
  value: "${BACKEND_ROUTE}"
```

```

- name: SSL_CERT_DIR
  value: "/etc/pki/tls/certs/"
- name: APICAST_REGISTRY_URL
  value: "${APICAST_REGISTRY_URL}"
image: registry.access.redhat.com/3scale-amp22/system:1.7
imagePullPolicy: IfNotPresent
name: system-scheduler
resources:
  limits:
    cpu: 150m
    memory: 250Mi
  requests:
    cpu: 50m
    memory: 200Mi
"

```

3. **system-sidekiq** デプロイメント設定にパッチを適用するには、以下を考慮してください。

- MASTER_USER および MASTER_PASSWORD 環境変数のデフォルト値を使用する場合は、以下の **oc patch** コマンドでそれらを記述しないでください。
- または、MASTER_USER および MASTER_PASSWORD の値を指定する場合は、以下の **oc patch** コマンドに追加します。

```

oc patch dc/system-sidekiq -p "
spec:
  template:
    spec:
      containers:
        - name: system-sidekiq
          volumeMounts:
"

```

```

oc patch dc/system-sidekiq -p "
metadata:
  labels:
    app: System
spec:
  template:
    spec:
      containers:
        - env:
            - name: RAILS_ENV
              value: "production"
            - name: DATABASE_URL
              value: "${DATABASE_URL}"
            - name: FORCE_SSL
              value: "true"
            - name: THREESCALE_SUPERDOMAIN
              value: "${THREESCALE_SUPERDOMAIN}"
            - name: MASTER_USER
              value: "${MASTER_USER}"
            - name: MASTER_PASSWORD
              value: "${MASTER_PASSWORD}"
            - name: TENANT_NAME
              value: "${TENANT_NAME}"
"

```

```
- name: APICAST_ACCESS_TOKEN
  value: \"${APICAST_ACCESS_TOKEN}\"
- name: ADMIN_ACCESS_TOKEN
  value: \"${ADMIN_ACCESS_TOKEN}\"
- name: PROVIDER_PLAN
  value: 'enterprise'
- name: USER_LOGIN
  value: \"${USER_LOGIN}\"
- name: USER_PASSWORD
  value: \"${USER_PASSWORD}\"
- name: RAILS_LOG_TO_STDOUT
  value: \"true\"
- name: RAILS_LOG_LEVEL
  value: \"info\"
- name: THINKING_SPHINX_ADDRESS
  value: \"system-sphinx\"
- name: THINKING_SPHINX_PORT
  value: \"9306\"
- name: THINKING_SPHINX_CONFIGURATION_FILE
  value: \"tmp/sphinx.conf\"
- name: EVENTS_SHARED_SECRET
  value: \"${EVENTS_SHARED_SECRET}\"
- name: THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE
  value: \"VERIFY_NONE\"
- name: APICAST_BACKEND_ROOT_ENDPOINT
  value: \"${APICAST_BACKEND_ROOT_ENDPOINT}\"
- name: CONFIG_INTERNAL_API_USER
  value: \"${CONFIG_INTERNAL_API_USER}\"
- name: CONFIG_INTERNAL_API_PASSWORD
  value: \"${CONFIG_INTERNAL_API_PASSWORD}\"
- name: SECRET_KEY_BASE
  value: \"${SECRET_KEY_BASE}\"
- name: AMP_RELEASE
  value: \"${AMP_RELEASE}\"
- name: ZYNC_AUTHENTICATION_TOKEN
  valueFrom:
    secretKeyRef:
      name: zync
      key: ZYNC_AUTHENTICATION_TOKEN
- name: SMTP_ADDRESS
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: address
- name: SMTP_USER_NAME
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: username
- name: SMTP_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: password
- name: SMTP_DOMAIN
  valueFrom:
```



```

    configMapKeyRef:
      name: smtp
      key: domain
- name: SMTP_PORT
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: port
- name: SMTP_AUTHENTICATION
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: authentication
- name: SMTP_OPENSSL_VERIFY_MODE
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: openssl.verify.mode
- name: BACKEND_ROUTE
  value: "${BACKEND_ROUTE}"
- name: SSL_CERT_DIR
  value: "/etc/pki/tls/certs"
- name: APICAST_REGISTRY_URL
  value: "${APICAST_REGISTRY_URL}"
image: registry.access.redhat.com/3scale-amp22/system:1.7
volumeMounts:
- name: system-storage
  mountPath: /opt/system/public/system
- name: system-config
  mountPath: /opt/system-extra-configs
- name: system-tmp
  mountPath: /tmp
image: registry.access.redhat.com/3scale-amp22/system:1.7
imagePullPolicy: IfNotPresent
name: system-sidekiq
resources:
  limits:
    cpu: 1000m
    memory: 2Gi
  requests:
    cpu: 100m
    memory: 500Mi
volumes:
- name: system-tmp
  emptyDir:
    medium: Memory
- name: system-storage
  persistentVolumeClaim:
    claimName: system-storage
- name: system-config
  configMap:
    name: system
    items:
      - key: zync.yml
        path: zync.yml

```

```

- key: rolling_updates.yml
  path: rolling_updates.yml
"

```

4. **system-app** デプロイメント設定にパッチを適用するには、以下を考慮してください。

- MASTER_USER および MASTER_PASSWORD 環境変数のデフォルト値を使用する場合は、以下の **oc patch** コマンドでそれらを記述しないでください。
- または、MASTER_USER および MASTER_PASSWORD の値を指定する場合は、以下の **oc patch** コマンドに追加します。

```

oc patch dc/system-app -p "
spec:
  template:
    spec:
      containers:
        - name: system-provider
          volumeMounts:
            - name: system-developer
              volumeMounts:
"

```

```

oc patch dc/system-app -p "
metadata:
  labels:
    app: System
spec:
  strategy:
    rollingParams:
      pre:
        execNewPod:
          containerName: system-master
          env:
            - name: RAILS_ENV
              value: \"production\"
            - name: DATABASE_URL
              value: \"${DATABASE_URL}\"
            - name: FORCE_SSL
              value: \"true\"
            - name: THREESCALE_SUPERDOMAIN
              value: \"${THREESCALE_SUPERDOMAIN}\"
            - name: MASTER_USER
              value: \"${MASTER_USER}\"
            - name: MASTER_PASSWORD
              value: \"${MASTER_PASSWORD}\"
            - name: TENANT_NAME
              value: \"${TENANT_NAME}\"
            - name: APICAST_ACCESS_TOKEN
              value: \"${APICAST_ACCESS_TOKEN}\"
            - name: ADMIN_ACCESS_TOKEN
              value: \"${ADMIN_ACCESS_TOKEN}\"
            - name: PROVIDER_PLAN
              value: 'enterprise'
            - name: USER_LOGIN
              value: \"${USER_LOGIN}\"
"

```

```
- name: USER_PASSWORD
  value: \"${USER_PASSWORD}\"
- name: RAILS_LOG_TO_STDOUT
  value: \"true\"
- name: RAILS_LOG_LEVEL
  value: \"info\"
- name: THINKING_SPHINX_ADDRESS
  value: \"system-sphinx\"
- name: THINKING_SPHINX_PORT
  value: \"9306\"
- name: THINKING_SPHINX_CONFIGURATION_FILE
  value: \"tmp/sphinx.conf\"
- name: EVENTS_SHARED_SECRET
  value: \"${EVENTS_SHARED_SECRET}\"
- name: THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE
  value: \"VERIFY_NONE\"
- name: APICAST_BACKEND_ROOT_ENDPOINT
  value: \"${APICAST_BACKEND_ROOT_ENDPOINT}\"
- name: CONFIG_INTERNAL_API_USER
  value: \"${CONFIG_INTERNAL_API_USER}\"
- name: CONFIG_INTERNAL_API_PASSWORD
  value: \"${CONFIG_INTERNAL_API_PASSWORD}\"
- name: SECRET_KEY_BASE
  value: \"${SECRET_KEY_BASE}\"
- name: AMP_RELEASE
  value: \"${AMP_RELEASE}\"
- name: ZYNC_AUTHENTICATION_TOKEN
  valueFrom:
    secretKeyRef:
      name: zync
      key: ZYNC_AUTHENTICATION_TOKEN
- name: SMTP_ADDRESS
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: address
- name: SMTP_USER_NAME
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: username
- name: SMTP_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: password
- name: SMTP_DOMAIN
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: domain
- name: SMTP_PORT
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: port
```

```

- name: SMTP_AUTHENTICATION
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: authentication
- name: SMTP_OPENSSL_VERIFY_MODE
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: openssl.verify.mode
- name: BACKEND_ROUTE
  value: "${BACKEND_ROUTE}"
- name: SSL_CERT_DIR
  value: "/etc/pki/tls/certs"
- name: APICAST_REGISTRY_URL
  value: "${APICAST_REGISTRY_URL}"
  command:
  - bash
  - -c
  - bundle exec rake boot openshift:deploy
MASTER_ACCESS_TOKEN="${MASTER_ACCESS_TOKEN}"
post:
  execNewPod:
    containerName: system-master
template:
  spec:
    containers:
    - args:
    env:
    - name: RAILS_ENV
      value: "production"
    - name: DATABASE_URL
      value: "${DATABASE_URL}"
    - name: FORCE_SSL
      value: "true"
    - name: THREESCALE_SUPERDOMAIN
      value: "${THREESCALE_SUPERDOMAIN}"
    - name: MASTER_USER
      value: "${MASTER_USER}"
    - name: MASTER_PASSWORD
      value: "${MASTER_PASSWORD}"
    - name: TENANT_NAME
      value: "${TENANT_NAME}"
    - name: APICAST_ACCESS_TOKEN
      value: "${APICAST_ACCESS_TOKEN}"
    - name: ADMIN_ACCESS_TOKEN
      value: "${ADMIN_ACCESS_TOKEN}"
    - name: PROVIDER_PLAN
      value: 'enterprise'
    - name: USER_LOGIN
      value: "${USER_LOGIN}"
    - name: USER_PASSWORD
      value: "${USER_PASSWORD}"
    - name: RAILS_LOG_TO_STDOUT
      value: "true"
    - name: RAILS_LOG_LEVEL

```

```
value: \"info\"
- name: THINKING_SPHINX_ADDRESS
  value: \"system-sphinx\"
- name: THINKING_SPHINX_PORT
  value: \"9306\"
- name: THINKING_SPHINX_CONFIGURATION_FILE
  value: \"/tmp/sphinx.conf\"
- name: EVENTS_SHARED_SECRET
  value: \"${EVENTS_SHARED_SECRET}\"
- name: THREESCALE_SANDBOX_PROXY_OPENSSL_VERIFY_MODE
  value: \"VERIFY_NONE\"
- name: APICAST_BACKEND_ROOT_ENDPOINT
  value: \"${APICAST_BACKEND_ROOT_ENDPOINT}\"
- name: CONFIG_INTERNAL_API_USER
  value: \"${CONFIG_INTERNAL_API_USER}\"
- name: CONFIG_INTERNAL_API_PASSWORD
  value: \"${CONFIG_INTERNAL_API_PASSWORD}\"
- name: SECRET_KEY_BASE
  value: \"${SECRET_KEY_BASE}\"
- name: AMP_RELEASE
  value: \"${AMP_RELEASE}\"
- name: ZYNC_AUTHENTICATION_TOKEN
  valueFrom:
    secretKeyRef:
      name: zync
      key: ZYNC_AUTHENTICATION_TOKEN
- name: SMTP_ADDRESS
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: address
- name: SMTP_USER_NAME
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: username
- name: SMTP_PASSWORD
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: password
- name: SMTP_DOMAIN
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: domain
- name: SMTP_PORT
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: port
- name: SMTP_AUTHENTICATION
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: authentication
```

```

- name: SMTP_OPENSSL_VERIFY_MODE
  valueFrom:
    configMapKeyRef:
      name: smtp
      key: openssl.verify.mode
- name: BACKEND_ROUTE
  value: "${BACKEND_ROUTE}"
- name: SSL_CERT_DIR
  value: "/etc/pki/tls/certs"
- name: APICAST_REGISTRY_URL
  value: "${APICAST_REGISTRY_URL}"
image: registry.access.redhat.com/3scale-amp22/system:1.7
imagePullPolicy: IfNotPresent
args: [ 'env', 'TENANT_MODE=master', 'PORT=3002', 'container-entrypoint',
'bundle', 'exec', 'unicorn', '-c', 'config/unicorn.rb' ]
command:
name: system-master
resources:
  limits:
    cpu: 1000m
    memory: 800Mi
  requests:
    cpu: 50m
    memory: 600Mi
livenessProbe:
  timeoutSeconds: 10
  initialDelaySeconds: 20
  tcpSocket:
    port: master
  periodSeconds: 10
readinessProbe:
  httpGet:
    path: /check.txt
    port: master
    scheme: HTTP
  httpHeaders:
    - name: X-Forwarded-Proto
      value: https
  initialDelaySeconds: 30
  timeoutSeconds: 10
  periodSeconds: 30
ports:
- containerPort: 3002
  protocol: TCP
  name: master
volumeMounts:
- name: system-storage
  mountPath: /opt/system/public/system
- name: system-config
  mountPath: /opt/system-extra-configs
- name: system-provider
  env:
    - name: MASTER_USER
      value: ${MASTER_USER}
    - name: MASTER_PASSWORD
      value: ${MASTER_PASSWORD}

```

```

- name: AMP_RELEASE
  value: ${AMP_RELEASE}
- name: APICAST_REGISTRY_URL
  value: ${APICAST_REGISTRY_URL}
image: registry.access.redhat.com/3scale-amp22/system:1.7
imagePullPolicy: IfNotPresent
resources:
  limits:
    cpu: 1000m
    memory: 800Mi
  requests:
    cpu: 50m
    memory: 600Mi
command:
  args: [ 'env', 'TENANT_MODE=provider', 'PORT=3000', 'container-entrpoint',
'bundle', 'exec', 'unicorn', '-c', 'config/unicorn.rb' ]
volumeMounts:
- name: system-storage
  mountPath: /opt/system/public/system
- name: system-config
  mountPath: /opt/system-extra-configs
- name: system-developer
env:
- name: MASTER_USER
  value: ${MASTER_USER}
- name: MASTER_PASSWORD
  value: ${MASTER_PASSWORD}
- name: AMP_RELEASE
  value: ${AMP_RELEASE}
- name: APICAST_REGISTRY_URL
  value: ${APICAST_REGISTRY_URL}
image: registry.access.redhat.com/3scale-amp22/system:1.7
imagePullPolicy: IfNotPresent
command:
  args: [ 'env', 'PORT=3001', 'container-entrpoint', 'bundle', 'exec', 'unicorn', '-c',
'config/unicorn.rb' ]
volumeMounts:
- name: system-storage
  readOnly: true
  mountPath: /opt/system/public/system
- name: system-config
  mountPath: /opt/system-extra-configs
triggers:
- type: ConfigChange
- type: ImageChange
imageChangeParams:
  automatic: true
  containerNames:
    - system-provider
    - system-developer
    - system-master
from:
  kind: ImageStreamTag
  name: amp-system:latest

```

"

5. **amp-system** イメージにパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch is/amp-system -p "
spec:
  tags:
    - name: 2.2.0
      annotations:
        openshift.io/display-name: AMP system 2.2.0
  from:
    kind: DockerImage
    name: 'registry.access.redhat.com/3scale-amp22/system:1.7'
  - name: latest
    from:
      kind: ImageStreamTag
      name: 2.2.0
"
```

6. **system-sphinx** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/system-sphinx -p "
metadata:
  labels:
    app: System
spec:
  template:
    spec:
      containers:
        - imagePullPolicy: IfNotPresent
          image: registry.access.redhat.com/3scale-amp22/system:1.7
          name: system-sphinx
      resources:
        limits:
          cpu: 1000m
          memory: 512Mi
        requests:
          cpu: 80m
          memory: 250Mi
"
```

7. **system-redis** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/system-redis -p '
metadata:
  labels:
    app: System
spec:
  template:
    spec:
      containers:
        - imagePullPolicy: IfNotPresent
          name: system-redis
          command:
            - "/opt/rh/rh-redis32/root/usr/bin/redis-server"
```



```

args:
- "/etc/redis.d/redis.conf"
- "--daemonize"
- "no"
resources:
  limits:
    memory: 32Gi
    cpu: 500m
  requests:
    cpu: 150m
    memory: 256Mi
  volumeMounts:
  - name: system-redis-storage
    mountPath: "/var/lib/redis/data"
  - name: redis-config
    mountPath: /etc/redis.d/

```

8. **system-memcache** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/system-memcache -p "
metadata:
  labels:
    app: System
spec:
  template:
    spec:
      containers:
      - imagePullPolicy: IfNotPresent
        name: memcache
        resources:
          limits:
            cpu: 250m
            memory: 96Mi
          requests:
            cpu: 50m
            memory: 64Mi
"

```

1.8. パッチバックエンドコンポーネント

1. **backend-cron** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/backend-cron -p "
metadata:
  labels:
    app: Backend
spec:
  template:
    spec:
      containers:
      - name: backend-cron
        env:

```

```

- name: CONFIG_REDIS_PROXY
  value: redis://backend-redis:6379/0
- name: CONFIG_REDIS_SENTINEL_HOSTS
  value: ""
- name: CONFIG_REDIS_SENTINEL_ROLE
  value: ""
- name: CONFIG_QUEUES_MASTER_NAME
  value: redis://backend-redis:6379/1
- name: CONFIG_QUEUES_SENTINEL_HOSTS
  value: ""
- name: CONFIG_QUEUES_SENTINEL_ROLE
  value: ""
- name: RACK_ENV
  value: "production"
image: registry.access.redhat.com/3scale-amp22/backend:1.6
imagePullPolicy: IfNotPresent
resources:
  limits:
    cpu: 150m
    memory: 80Mi
  requests:
    cpu: 50m
    memory: 40Mi
"

```

2. **backend-worker** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/backend-worker -p "
metadata:
  labels:
    app: Backend
spec:
  template:
    spec:
      containers:
        - name: backend-worker
          env:
            - name: CONFIG_REDIS_PROXY
              value: redis://backend-redis:6379/0
            - name: CONFIG_REDIS_SENTINEL_HOSTS
            - name: CONFIG_REDIS_SENTINEL_ROLE
            - name: CONFIG_QUEUES_MASTER_NAME
              value: redis://backend-redis:6379/1
            - name: CONFIG_QUEUES_SENTINEL_HOSTS
            - name: CONFIG_QUEUES_SENTINEL_ROLE
            - name: RACK_ENV
              value: \"production\"
            - name: PUMA_WORKERS
              value: \"16\"
            - name: CONFIG_EVENTS_HOOK
              value: http://system-master:3000/master/events/import
            - name: CONFIG_EVENTS_HOOK_SHARED_SECRET
              value: ${EVENTS_SHARED_SECRET}
          image: registry.access.redhat.com/3scale-amp22/backend:1.6
          imagePullPolicy: IfNotPresent

```

```

resources:
  limits:
    cpu: 1000m
    memory: 300Mi
  requests:
    cpu: 150m
    memory: 50Mi
"

```

3. **backend-listener** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/backend-listener -p "
metadata:
  labels:
    app: Backend
spec:
  template:
    spec:
      containers:
      - name: backend-listener
        env:
          - name: CONFIG_REDIS_PROXY
            value: redis://backend-redis:6379/0
          - name: CONFIG_REDIS_SENTINEL_HOSTS
          - name: CONFIG_REDIS_SENTINEL_ROLE
            value: ""
          - name: CONFIG_QUEUES_MASTER_NAME
            value: redis://backend-redis:6379/1
          - name: CONFIG_QUEUES_SENTINEL_HOSTS
          - name: CONFIG_QUEUES_SENTINEL_ROLE
            value: ""
          - name: RACK_ENV
            value: \"production\"
          - name: CONFIG_INTERNAL_API_USER
            value: \"${CONFIG_INTERNAL_API_USER}\"
          - name: CONFIG_INTERNAL_API_PASSWORD
            value: \"${CONFIG_INTERNAL_API_PASSWORD}\"
          - name: PUMA_WORKERS
            value: \"16\"
        image: registry.access.redhat.com/3scale-amp22/backend:1.6
        imagePullPolicy: IfNotPresent
      resources:
        limits:
          cpu: 1000m
          memory: 700Mi
        requests:
          cpu: 500m
          memory: 550Mi
"

```

4. **amp-backend** イメージストリームにパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch is/amp-backend -p "

```

```
spec:
  tags:
    - name: 2.2.0
  annotations:
    openshift.io/display-name: AMP backend
  from:
    kind: DockerImage
    name: 'registry.access.redhat.com/3scale-amp22/backend:1.6'
  - name: latest
  from:
    kind: ImageStreamTag
    name: 2.2.0
"
```

5. **backend-redis** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/backend-redis -p '
metadata:
  labels:
    app: Backend
spec:
  template:
    spec:
      containers:
        - name: backend-redis
          command:
            - "/opt/rh/rh-redis32/root/usr/bin/redis-server"
          args:
            - "/etc/redis.d/redis.conf"
            - "--daemonize"
            - "no"
          imagePullPolicy: IfNotPresent
      resources:
        limits:
          cpu: 2000m
          memory: 32Gi
        requests:
          cpu: 1000m
          memory: 1024Mi
      volumeMounts:
        - name: backend-redis-storage
          mountPath: "/var/lib/redis/data"
        - name: redis-config
          mountPath: /etc/redis.d/
'
```

1.9. APICAST のパッチ適用

1. **apicast-staging** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/apicast-staging -p "
metadata:
  labels:
```

```

    app: APICast
spec:
  template:
    spec:
      containers:
        - name: apicast-staging
          env:
            - name: THREESCALE_PORTAL_ENDPOINT
              value: \"http://${APICAST_ACCESS_TOKEN}@system-
master:3000/master/api/proxy/configs\"
            - name: APICAST_CONFIGURATION_LOADER
              value: \"lazy\"
            - name: APICAST_CONFIGURATION_CACHE
              value: \"0\"
            - name: THREESCALE_DEPLOYMENT_ENV
              value: \"sandbox\"
            - name: APICAST_MANAGEMENT_API
              value: \"${APICAST_MANAGEMENT_API}\"
            - name: BACKEND_ENDPOINT_OVERRIDE
              value: http://backend-listener:3000
            - name: OPENSLL_VERIFY
              value: \"${OPENSLL_VERIFY}\"
            - name: APICAST_RESPONSE_CODES
              value: \"${APICAST_RESPONSE_CODES}\"
            - name: REDIS_URL
              value: \"redis://system-redis:6379/2\"
          image: registry.access.redhat.com/3scale-amp22/apicast-gateway:1.8
          imagePullPolicy: IfNotPresent
      resources:
        limits:
          cpu: 100m
          memory: 128Mi
        requests:
          cpu: 50m
          memory: 64Mi
  \"

```

2. **apicast-production** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/apicast-production -p "
metadata:
  labels:
    app: APICast
spec:
  template:
    spec:
      containers:
        - name: apicast-production
          env:
            - name: THREESCALE_PORTAL_ENDPOINT
              value: \"http://${APICAST_ACCESS_TOKEN}@system-
master:3000/master/api/proxy/configs\"
            - name: APICAST_CONFIGURATION_LOADER
              value: \"boot\"
            - name: APICAST_CONFIGURATION_CACHE

```

```

    value: \"300\"
  - name: THREESCALE_DEPLOYMENT_ENV
    value: \"production\"
  - name: APICAST_MANAGEMENT_API
    value: \"${APICAST_MANAGEMENT_API}\"
  - name: BACKEND_ENDPOINT_OVERRIDE
    value: http://backend-listener:3000
  - name: OPENSSSL_VERIFY
    value: \"${APICAST_OPENSSSL_VERIFY}\"
  - name: APICAST_RESPONSE_CODES
    value: \"${APICAST_RESPONSE_CODES}\"
  - name: REDIS_URL
    value: \"redis://system-redis:6379/1\"
image: registry.access.redhat.com/3scale-amp22/apicast-gateway:1.8
imagePullPolicy: IfNotPresent
resources:
  limits:
    cpu: 1000m
    memory: 128Mi
  requests:
    cpu: 500m
    memory: 64Mi
"

```

3. **amp-apicast** イメージストリームにパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch is/amp-apicast -p "
spec:
  tags:
    - name: 2.2.0
  annotations:
    openshift.io/display-name: AMP apicast
  from:
    kind: DockerImage
    name: 'registry.access.redhat.com/3scale-amp22/apicast-gateway:1.8'
  - name: latest
  from:
    kind: ImageStreamTag
    name: 2.2.0
"

```

4. **apicast-wildcard-router** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/apicast-wildcard-router -p "
metadata:
  labels:
    app: APICast
spec:
  template:
    spec:
      containers:
        - name: apicast-wildcard-router
          env:

```

```

- name: API_HOST
  value: \"http://${APICAST_ACCESS_TOKEN}@system-master:3000\"
image: registry.access.redhat.com/3scale-amp22/wildcard-router:1.6
imagePullPolicy: IfNotPresent
resources:
  limits:
    cpu: 500m
    memory: 64Mi
  requests:
    cpu: 120m
    memory: 32Mi
"

```

5. **amp-wildcard-router** イメージストリームにパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch is/amp-wildcard-router -p "
spec:
  tags:
    - name: 2.2.0
  annotations:
    openshift.io/display-name: AMP wildcard router
  from:
    kind: DockerImage
    name: 'registry.access.redhat.com/3scale-amp22/wildcard-router:1.6'
  - name: latest
  from:
    kind: ImageStreamTag
    name: 2.2.0
"

```

1.10. ZYNC コンポーネントにパッチを適用

1. **zync-database** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```

oc patch dc/zync-database -p "
metadata:
  labels:
    app: Zync
spec:
  template:
    spec:
      containers:
        - name: postgresql
          imagePullPolicy: IfNotPresent
          resources:
            limits:
              memory: 2Gi
              cpu: 250m
            requests:
              cpu: 50m
              memory: 250Mi
"

```

2. **zync** デプロイメント設定にパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch dc/zync -p "
metadata:
  labels:
    app: Zync
spec:
  template:
    spec:
      containers:
      - name: zync
        image: 'registry.access.redhat.com/3scale-amp22/zync:1.6'
      resources:
        limits:
          cpu: 1
          memory: 512Mi
        requests:
          cpu: 150m
          memory: 250Mi
"
```

3. **zync** イメージストリームにパッチを適用するには、以下の **oc patch** コマンドを実行します。

```
oc patch is/amp-zync -p "
spec:
  tags:
  - name: 2.2.0
    annotations:
      openshift.io/display-name: AMP zync
    from:
      kind: DockerImage
      name: 'registry.access.redhat.com/3scale-amp22/zync:1.6'
  - name: latest
    from:
      kind: ImageStreamTag
      name: 2.2.0
"
```

1.11. アップグレードを確認

アップグレード手順を実行した後に、3scale 管理ポータル の右上隅にあるバージョン番号を確認して、アップグレードの操作が正常に行われたことを確認します。



注記

再デプロイメントの操作が OpenShift で完了するのに時間がかかる場合があります。

第2章 API DEPLOYMENT ON MICROSOFT AZURE

API はプラットフォームに依存しないため、任意のプラットフォームにデプロイできます。このチュートリアルでは、Microsoft Azure での高速 Web API デプロイメントです。Ruby Grape gem を使用して、API インターフェース、NGINX プロキシ、シンサーバー、および Capistrano を使用してデプロイします。

本チュートリアルでは、サードパーティーサーバーで実行中の Ruby ベースの API を使用するか、Echo-API のクローンを作成することができます。

2.1. MICROSOFT AZURE 仮想マシンの作成および設定

SSH を Azure 仮想マシンに 2048 ビットの RSA キーペアを使って X509 証明書を生成します。これは、仮想マシンを設定する場合に役立ちます。

このタイプのキーを生成するには、次のコマンドを実行します。

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout myPrivateKey.key -out myCert.pem
```

次に、Microsoft Azure アカウントを作成して開始します。このチュートリアルでは、無料トライアルオプションを使用できます。Azure アカウントの作成後に、Virtual Machines タブの Dashboard に移動します。次は、最初の仮想マシンを作成するガイドです。from gallery オプションを選択し、Ubuntu Server 12.04 LTS を選択します。

ステップ 2 では、前のステップで作成した pem をアップロードできます。

手順 3 および 4 で、ニーズに最も適したオプションを選択します。

仮想マシンの準備が整うまでに数分かかります。その場合には、そのダッシュボードにアクセスできます。このダッシュボードでは、仮想マシンのアクティビティ（CPU、ディスク、ネットワーク）を監視し、そのサイズをアップグレードできます。

仮想マシンにはいくつかのパッケージがインストールされるため、他のコンポーネントをインストールする必要があります。キーが作成されると、仮想マシンに ssh を実行できます。

```
ssh -i myPrivateKey.key -p 22 username@servicename.cloudapp.net
```

仮想マシンで以下のコマンドを実行し、必要なものをすべてインストールします。

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y install ruby1.9.3 build-essential libsqlite3-dev libpcre3 libpcre3-dev libssl-dev openssl
libreadline6 libreadline6-dev libxml2-dev libxslt1-dev
```

以下を実行して Ruby インストールが完了していることを確認できます。

```
ruby -v
```

ruby 1.9.3p194 (2012-04-20 リビジョン 35410) [x86_64-linux]などの出力が表示されるはずです。

bundler および thin をインストールする必要もあります。

```
sudo gem install bundler
sudo gem install thin
```

ここでは、仮想マシン上で必要なすべてのものが存在するはずですが、Dashboard に戻り、Endpoints タブをクリックします。ポート **80** に **HTTP** エンドポイントを追加し、フィールドは自動的に入力する必要があります。

2.2. INSTALL OPENRESTY

このステップを合理化するためには、Web アプリケーションを自由にインストールすることを推奨します。**OpenResty**これは、必要なサードパーティー NGINX モジュールのほとんどすべてと共にバンドルされた標準の NGINX コアです。

Azure VM Compile で NGINX をインストールします。

```
cd ~
sudo wget http://agentzh.org/misc/nginx/nginx_openresty-VERSION.tar.gz
sudo tar -zxvf nginx_openresty-VERSION.tar.gz
cd nginx_openresty-VERSION/
sudo ./configure --prefix=/opt/openresty --with-luajit --with-http_iconv_module -j2
sudo make
sudo make install
```

2.3. GITHUB リポジトリの設定

このチュートリアルでは、GitHub を使用してコードをホストします。API にリポジトリがない場合は、github.com でそのリポジトリを作成し、これをホストするようにしてください。Git と GitHub に精通していない場合は、[この簡単なチュートリアルを確認してください](#)。

仮想マシンで Git を使用し、GitHub リポジトリにアクセスできるようにするには、仮想マシンで SSH キーを生成し、[ここで説明するように](#) Github に追加する必要があります。

2.3.1. 警告

公開 GitHub リポジトリでコードをホストすると、脆弱になります。公開されているプッシュ前に、プロバイダーキーなどの機密情報が含まれていないことを確認します。

2.4. API の設定

システムが機能する仕組みを以下に示します。

1. シンサーバーは、ポート 8000 で起動します。
2. アップストリームの **YOURAPINAME** は、localhost:8000 でリッスンしています。
3. ポート 80 の今後の接続（**server** セクションで定義されているように）は、**YOURAPINAME** に「リダイレクト」です。

2.4.1. 3scale の場合

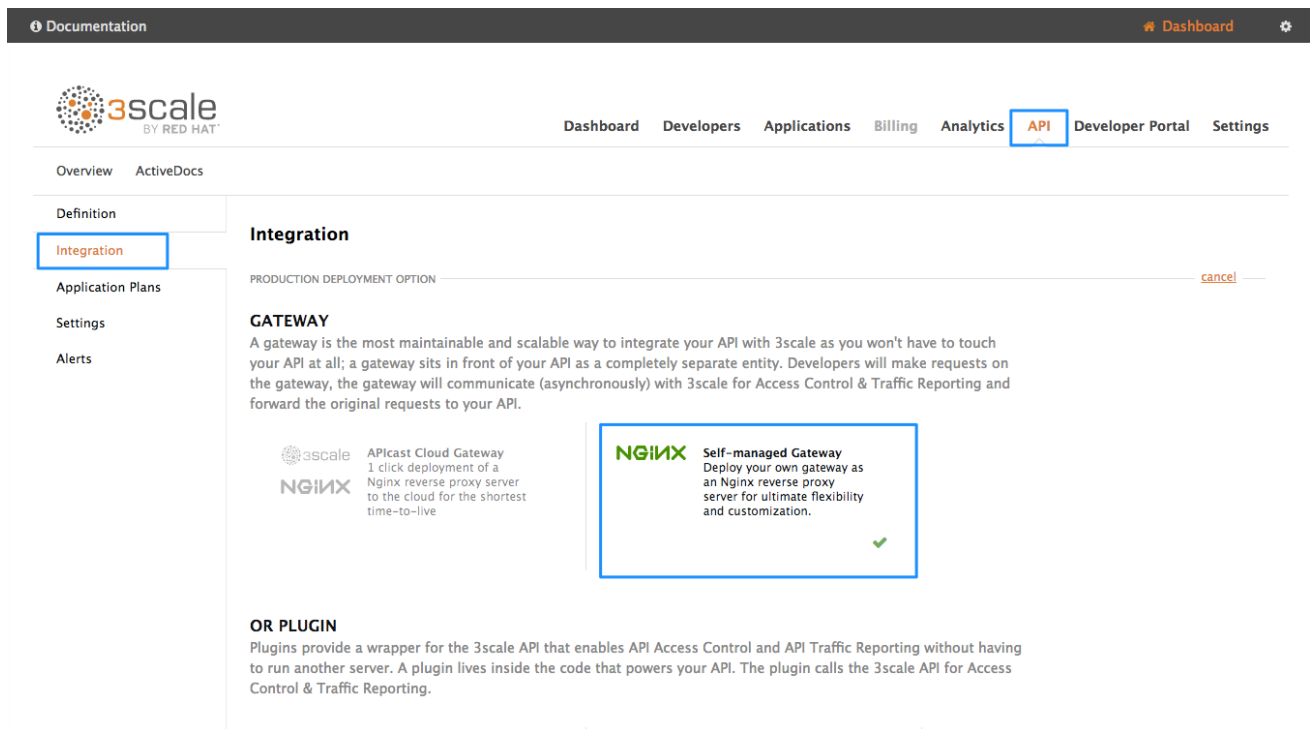
wheel を修正して流量制御、アクセス制御、および解析をゼロから実装するのではなく、3scale を使用します。[アカウントがまだアカウントをお持ちでない場合は、ここにサインアップしてアクティベートし](#)、表示されたリンクで新規インスタンスにログインします。初回ログイン時に、作成されるサンプルデータのオプションを選択してください。したがって、後で使用する [API キーがいくつかあります](#)。tour を確認して、システム機能（オプション）を把握してから実装に進んでください。

即時の結果を取得するには、ステージング環境で API ゲートウェイを開始します。このゲートウェイは、開発中に使用することができます。その後、完全な実稼働デプロイメント用にスケールアップできる NGINX プロキシを設定します。

ここでは、API プロキシの設定方法や、より高度な設定オプションについてのドキュメントを参照してください。

3scale アカウントにサインインしたら、メイン Dashboard 画面で API を起動するか、または API に Go to API→Select the service(API)→Integration in the sidebar→Proxy

<https://www.3scale.net/2015/06/how-to-deploy-an-api-amazon-ec2/>



API バックエンドのアドレスを設定します。

```
`http://YOURAPP.cloudapp.net:80`
```

1. 3scale で一部のアプリケーション認証情報を作成したら、ステージング環境用 API ゲートウェイエンドポイントに到達して API をテストすることができます。

```
`https://XXX.staging.apicast.io/v1/words/awesome.json?
app_id=APP_ID&app_key=APP_KEY`
```

ここで、**XXX** はステージング API ゲートウェイに固有の設定で、**APP_ID** は 3scale アカウントへの初回ログイン時に作成したサンプルアプリケーションの ID とキーです。**APP_KEY**（その手順がない場合は、開発者アカウントを作成し、そのアカウント内にアプリケーションを作成します）。

これは、次回認証情報が正しくない状態で、アプリケーション認証情報なしで試行します。認証後、定義した流量制御内、およびその流量制御で認証します。満足度に機能したら、NGINX の設定ファイルをダウンロードします。



注記

エラーがあれば、API に直接アクセスできるかどうかを確認します(your-public-dns:3000/v1/words/awesome.json)。利用できない場合は、AWS インスタンスが実行しているかどうかと、タインサーバーがインスタンスで実行されているかどうかを確認します。*

API バックエンドアドレスを <http://YOURAPP.cloudapp.net:80> に変更できます。

終了したら、**Download your nginx config** をクリックします。これにより、アプリケーションを設定するために使用する **.conf** および **.lua** ファイルが含まれるアーカイブがダウンロードされます。

.conf を適宜変更します。

API ゲートウェイと API が同じ仮想マシンにある場合は、ブロックを削除します。

```
upstream backend_YOURAPP.cloudapp.net{
  server ....
}
```

...and replace with...

```
upstream YOURAPINAME {
  server 127.0.0.1:8000;
}
```



警告

YOURAPINAME には、RFC 3986 で定義された URL の有効な文字のみを含めることができます。

.lua ファイルで、**ngx.var.proxy_pass = "http://backend_YOURAPP.cloudapp.net"** の行を変更します。

すべてのケースで **ngx.var.proxy_pass = "http://YOURAPINAME"** を使用します。

server_name api.2445580546262.proxy.3scale.net; を

server_name YOURSERVICENAME.cloudapp.net;

server ブロックで、上部に以下を追加します。

```
root /home/USERNAME/apps/YOURAPINAME/current;
access_log /home/USERNAME/apps/YOURAPINAME/current/log/thin.log;
error_log /home/USERNAME/apps/YOURAPINAME/current/log/error.log;
```

replace **access_by_lua_file lua_tmp.lua;**

...with... **access_by_lua_file /opt/openresty/nginx/conf/lua_tmp.lua;**

post_action /out_of_band_authrep_action; の前に以下を追加します。

```
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header Host $http_host;
```

最後に、これらのファイル **nginx.conf** および **tmp_lua.lua** の名前を変更します。

2.4.2. Capistrano の設定

Capistrano を使用して API をデプロイします。Capistrano は自動化ツールで、デプロイメントのタスクを設定して、コマンドラインインターフェースを使用して実行できます。Capistrano は、リモートマシンで使用して、リモート仮想マシンにデプロイします。

Capistrano をインストールするには、以下の行を gem ファイルに追加します。 **gem 'capistrano'**

以下のコマンドを実行して、新しい gems をインストールし、Capistrano を設定します。 **bundle capify.**

nginx.conf および **tmp_lua.lua** を **/config** にコピーします。

2.5. CAPISTRANO の設定

capify コマンドを実行すると、**Capfile** および **deploy.rb** の2つのファイルが作成されます。**deploy.rb** では、アプリケーションをデプロイするために必要なすべてのコマンドを記述します。

/config edit **deploy.rb** で、コンテンツを以下に置き換えます。

```
require "bundler/capistrano"
set :application, "YOURAPINAME"
set :user, "USERNAME"
set :scm, :git
set :repository, "git@github.com:GITHUBUSERNAME/REPO.git"
set :branch, "master"

set :use_sudo, false

server "VNDNSname", :web, :app, :db, primary: true

set :deploy_to, "/home/#{user}/apps/#{application}"
default_run_options[:pty] = true
ssh_options[:forward_agent] = false
ssh_options[:port] = 22
ssh_options[:keys] = ["/PATH/TO/myPrivateKey.key"]

namespace :deploy do
  task :start, :roles => [:web, :app] do
    run "cd #{deploy_to}/current && nohup bundle exec thin start -C config/production_config.yml -R config.ru"
    sudo "/opt/openresty/nginx/sbin/nginx -p /opt/openresty/nginx/ -c /opt/openresty/nginx/conf/nginx.conf"
  end

  task :stop, :roles => [:web, :app] do
    run "kill -QUIT cat /opt/openresty/nginx/logs/nginx.pid"
    run "cd #{deploy_to}/current && nohup bundle exec thin stop -C config/production_config.yml -R"
```

```

config.ru"
end

task :restart, :roles => [:web, :app] do
  deploy.stop
  deploy.start
end

task :setup_config, roles: :app do
  sudo "ln -nfs #{current_path}/config/nginx.conf /opt/openresty/nginx/conf/nginx.conf"
  sudo "ln -nfs #{current_path}/config/lua_tmp.lua /opt/openresty/nginx/conf/lua_tmp.lua"
  sudo "mkdir -p #{shared_path}/config"
end
after "deploy:setup", "deploy:setup_config"
end

```

これにより、Capistrano が **rake:migrate** の実行を試行しないようにします。（これは Rails プロジェクトではありません！）

```

task :cold do
  deploy.update
  deploy.start
end

```

上記のテキストで、以下を置き換えます。

- **VNDNSname** .cloudapp.net DNS に置き換えます。
- **YOURAPINAME** applicationname を使用して、
- **USERNAME** 仮想マシンへのログインに使用されるユーザー名を使用します。
- **GITHUBUSERNAME** Github のユーザー名と合わせて使用できます。
- **REPO** Github リポジトリ名を使用する。
- **/PATH/TO** 前のステップで作成した SSH キーにアクセスするためのパスを指定します。

API にデータベースがない場合は、上記が正常に機能します。データベースがある場合は、行をコメント化します。

```

task :cold do
  deploy.update
  deploy.start
end

```

また、**/config** に **production_config.yml** ファイルを追加して Thin サーバーを設定する必要があります。

```

environment: production
chdir: /home/USERNAME/apps/YOURAPINAME/current/
address: 127.0.0.1
user: USERNAME
port: 8000
pid: /home/USERNAME/apps/YOURAPINAME/current/tmp/thin.pid

```

```
rackup: /home/USERNAME/apps/YOURAPINAME/current/config.ru
log: /home/USERNAME/apps/YOURAPINAME/current/log/thin.log
max_conns: 1024
timeout: 30
max_persistent_conns: 512
daemonize: true
```

ここでも、ユーザー名とパスを変更します。

プロジェクトの変更をコミットし、それらを GitHub にアップロードします。

```
git add .
git commit -m "adding config files"
git push
```

ほとんど行われています。

2.6. デプロイ

ローカル開発マシンから以下のコマンドを実行し、リモート Azure 仮想マシンを設定します。

```
cap deploy:setup
```

ssh キーへのパスが正しい場合は、パスワードの入力を要求しないでください。

Capistrano は、仮想マシンに接続し、ユーザーアカウントの **home** ディレクトリーに **apps** ディレクトリーを作成します。

これで、API を仮想マシンにデプロイすることができ、以下のコマンドを使用して Thin サーバーを起動できます。 **cap deploy:cold**

このコマンドは、GitHub で最新のコミットを取得する必要があります。OpenResty および Thin サーバーを起動します。

API が URL で利用可能になりました。

MYAPI.cloudapp.net/path/to/resources

2.6.1. トラブルシューティング

API にアクセスできない場合は、仮想マシンに対して ssh を実行し、**curl** を使用して **localhost** で呼び出すことができることを確認します。次のようになります。

```
curl -X GET http://localhost:8000/v2/words/hello.json?app_id=APPID&app_key=APPKEY`
```

これが機能する場合は、nginx 設定に誤りがあります。

仮想マシンの nginx ログで、

```
cat /opt/openresty/nginx/logs/error.log
```

これで、API が Azure Linux インスタンスで実行されているはずです。

このチュートリアルを使用しました。ご質問やご不明な点がございましたら、質問またはコメントをお寄せください。お聞かせください。

第3章 AWS ROOKIES 向けの AMAZON EC2 での API のデプロイ

3scale では、アプリケーションスタックを完全に制御するため、Amazon は API の実行に最適なプラットフォームです。ただし、AWS を初めて使用する場合には、学習曲線はかなりサブできます。そのため、ベストプラクティスをこの簡単なチュートリアルにまとめることができます。Amazon EC2 のほかにも Ruby Grape gem を使用して、アクセス制御を処理する API インターフェースおよび NGINX ゲートウェイを作成します。このチュートリアル内のすべてのすべては、完全に無料です。

3.1. 前提条件

本チュートリアルの目的のために、Ruby および Thin サーバーをベースとした実行中の API が必要です。「Deploying the Application」セクションで以下で説明されているように、サンプルリポジトリのクローンを作成できます。

Amazon EC2 インスタンスの作成および設定を開始します。EC2 インスタンスがすでにある場合（マイクロかどうか）は、次のステップ「Preparing Instance for Deployment」に移動できます。

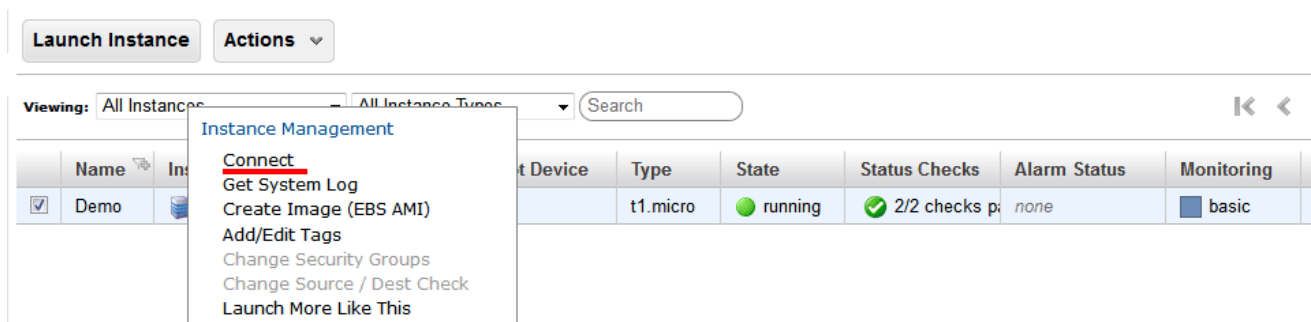
3.2. EC2 インスタンスの作成および設定

まず、Amazon Elastic Compute Cloud(Amazon EC2)の署名で開始します。[空き層は](#)、すべての基本的なニーズに対応するだけで十分です。アカウントの作成後に、AWS 管理コンソールの EC2 ダッシュボードに移動し、「launch instance」ボタンをクリックします。これにより、プロセスを続行するポップアップウィンドウに転送されます。

- 従来のウィザードを選択します。
- AMI (Ubuntu Server 12.04.1 LTS 32 ビット、T1micro インスタンス) を選択し、「インスタンス詳細」のその他の設定をデフォルトとして残します。
- キーペアを作成し、ダウンロードします。これは、サーバーへの ssh 接続を確立するために使用するキーです。VERY IMPORTANT!
- ソースは常に 0.0.0.0/0 (HTTP、HTTPS、ALL ICMP、Ruby Thin サーバーによって使用される TCP ポート 3000) でファイアウォールに受信ルールを追加します。

3.3. デプロイメント用のインスタンスの準備

インスタンスを作成して実行したら、コンソールから直接接続することができます (PuTTY の Windows ユーザー)。インスタンスをクリックして接続し、スタンドアロン SSH Client との接続を選択します。



手順に従って、指定の例でユーザー名を "ubuntu" (root ではなく) に変更します。

```
Terminal x Terminal x Terminal
[redacted]:~/.ssh$ ssh -i amazon_aws.pem ubuntu@ec2-184-73-23-174.compute-1.amazonaws.com
The authenticity of host 'ec2-184-73-23-174.compute-1.amazonaws.com (184.73.23.174)' can't be established.
ECDSA key fingerprint is e9:ff:d3:1c:3f:a9:64:a0:cc:89:da:f1:08:30:df:10.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-184-73-23-174.compute-1.amazonaws.com,184.73.23.174' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.1 LTS (GNU/Linux 3.2.0-31-virtual i686)

 * Documentation:  https://help.ubuntu.com/

System information as of Thu Jan 31 16:09:50 UTC 2013

System load:  0.0                Processes:    66
Usage of /:   18.7% of 7.87GB    Users logged in: 0
Memory usage: 6%                IP address for eth0: 10.212.101.187
Swap usage:  0%

Graph this data and manage this system at https://landscape.canonical.com/

21 packages can be updated.
8 updates are security updates.

Get cloud support with Ubuntu Advantage Cloud Guest
http://www.ubuntu.com/business/services/cloud
ubuntu@ip-10-212-101-187:~$
```

この手順を完了した後、インスタンスに接続している。新しいパッケージをインストールする必要があります。root 認証情報が必要なため、新たに root パスワード(**sudo passwd root**)を設定する必要があります。その後、root でログインします：**su root**

root 認証情報を使用して、以下を実行します。 **sudo apt-get update**

exit コマンドで通常のユーザーに戻り、必要なパッケージをすべてインストールします。

- **rvm**、Ruby および **Git** で必要なライブラリーをインストールします。

```
sudo apt-get install build-essential git zlib1g-dev libssl-dev libreadline-gplv2-dev imagemagick
libxml2-dev libxslt1-dev openssl zlib1g libyaml-dev libxslt-dev autoconf libc6-dev ncurses-dev
automake libtool bison libpq-dev libpq5 libeditline-dev
```

```
sudo apt-get install libreadline6 libreadline6-dev
```

- [Git](#) のインストール（ソースではなく Linux）のインストール
- [rvm](#) のインストール
- Ruby のインストール

```
rvm install 1.9.3
rvm use 1.9.3 --default
```

3.4. アプリケーションのデプロイ

この例では、Sentiment API は GitHub にあります。リポジトリのクローンを作成してみてください。

```
git clone git@github.com:jerzyn/api-demo.git
```

このアプリの作成およびデプロイについては、[コードおよびチュートリアルを確認してください](#)。認証情報はゲートウェイを通過するため、v1 のみを使用する変更細心の注意を払うことに注意してください。

これで、**bundle install** を実行してアプリケーションをデプロイすることができます。

これで、シンサーバー **thin start** を開始できます。

API に直接（セキュリティやアクセス制御なしで）アクセスにアクセスするには、**your-public-ip:3000/v1/words/awesome.json** インスタンスの詳細ウィンドウで AWS EC2 Dashboard > インスタンスからパブリック IP を検索できます。

Instance: **i-0fc94bdf**

Public IP: **52.5.23.192**

Description

Status Checks

Monitoring

Tags

Instance ID

i-0fc94bdf

Public DNS

-

Instance state

running

Public IP

52.5.23.192

3.4.1. Optional

カスタムドメインを Amazon インスタンスに割り当てる場合は、A レコードをドメインの DNS レコードに追加し、ドメインをパブリック IP アドレスにマップする必要があります。

ドメインプロバイダーは、A レコード（IPv4 アドレス）の設定方法を指定するか、ドメインのネームサーバーを編集することができます。A レコードを直接 DNS 管理サービスとして設定できない場合、ドメインをゾーンとして登録します。このサービスにより、ドメインプロバイダーの管理者パネルにネームサーバーが入ります。次に、ドメインに A レコードを追加できます。使用できる DNS 管理サービスには、ZoneEdit (basic、free) または Amazon route 53 が含まれます。


この時点で、API はグローバルに開きます。これは、共有するのが適切であり、適切ではありませんが、一部のアプリがサーバーのリソースを強制終了しなくても、サーバーのリソースを強制終了し、API を使用しているユーザーや、その使用方法に関する洞察はありません。このソリューションは、API 管理を追加することです。

3.5. 3SCALE での API 管理の有効化

wheel を再入力して流量制御、アクセス制御、および解析をゼロから実装するのではなく、3scale API Management Platform を利用できます。[3scale アカウントにサインアップしていない場合は](#)、アクティベーションリンクからログインします。初回ログイン時に、サンプルデータが作成されます。したがって、後で使用する API キーがあります。ウィザードを確認して、システムの機能について理解することができます（オプション）。その後、実装から開始します。

即時の結果を取得するには、開発環境の API ゲートウェイを使い始めます。このゲートウェイは、開発中に使用することができます。その後、完全な実稼働デプロイメントのためにスケールアップできる NGINX ゲートウェイを設定します。[以下は、API ゲートウェイの設定に関するいくつかのドキュメントと、より高度な設定オプションです。](#)

3scale アカウントにサインインしたら、Dashboard > API > サービス(API) > Integration > edit integration settings の順に選択し、Self-managed APIcast を選択します。



Overview

ActiveDocs

Dashboard

Developers

Applications

Billing

Analytics

API

Developer Portal

Settings

Integration

Settings

Naming

Alerts

Application plans

Integration



Dear 3scale,

Please take a moment to update your integration settings and tell us which deployment option you are using. If you're using multiple integration options, pick the one that's most important to your business. This setting has no functional consequences, it adjusts the user interface to better suit your use case.

DEPLOYMENT OPTION

GATEWAY

A gateway is the most maintainable and scalable way to integrate your API with 3scale as you won't have to touch your API at all; a gateway sits in front of your API as a completely separate entity. Developers will do requests on the gateway, the gateway will communicate (asynchronously) with 3scale for Access Control & Traffic Reporting and forward the original requests to your API.




APIcast Cloud Gateway

1 click deploy of an Nginx reverse proxy server to the cloud for the shortest time-to-live

NGINX

On-premise Gateway

Deploy your own gateway as an Nginx reverse proxy server for ultimate flexibility and customization.



Overview

ActiveDocs

Dashboard

Developers

Applications

Billing

Analytics

API

Developer Portal

Settings

Integration

Settings

Naming

Alerts

Application plans

Integration

Deployment option: On-premise Gateway

Authentication: API Key (user_key)

[edit integration settings](#)

Configure your API gateway in the staging environment. At any moment, you can download the nginx config files to deploy your on-premise API gateway to a suitable production environment.

Staging - configure & test your integration

API

Private Base URL *

Use Hello World API

Private address of your API that will be called by the API gateway. For end-to-end encryption your private base URL scheme should be https.

API GATEWAY

Public Base URL *

http://api.28816.proxy.3scale.net:80

Use 3scale Sandbox Proxy

Public address of your API gateway in the staging environment. You can use this address to call the API for testing purposes.

MAPPING RULES

AUTHENTICATION SETTINGS

CLIENT

API test GET request

Optional client GET request to a API gateway endpoint. This call has been left blank and therefore it will not be possible to test if the connection between client, API gateway & API is working correctly.

The API test GET request has been left blank. You should set it before checking the connections between client, gateway & API.

Save & Deploy

Production

On-premise API gateway

To deploy an on-premise API gateway, [Download the Nginx Config files](#) and [follow the documentation](#).

API バックエンドのアドレスを設定します。これは、カスタムドメインが設定されていない場合は、http プロトコルやポート 3000 を含むパブリック IP アドレスでなければなりません。これで、ステージング環境で API ゲートウェイへの変更を保存し、ステージング環境のエンドポイントにアクセスして API をテストすることができます。

```
http://api.XXX.proxy.3scale.net/v1/words/awesome.json?user_key=USER_KEY
```

XXX は 3scale アカウントに固有のもので、USER_KEY は、最初に 3scale アカウントにログインしたときに作成したサンプルアプリケーションの 1 つの認証キーです。（開発者アカウントの作成と、そのアカウント内にアプリケーションを作成するだけです。）

アプリケーションのクレデンシャルなしで試行します。次に誤ったクレデンシャルで、次に認証後に定義した流量制御で認証を行います。満足度に機能したら、NGINX の設定ファイルをダウンロードできます。



注記

エラーがあれば、API に直接アクセスできるかどうかを確認します(your-public-dns:3000/v1/words/awesome.json)。これが利用できない場合は、AWS インスタンスが実行中かどうか、および Thin サーバーがインスタンスで実行されているかどうかを確認する必要があります。

3.6. APICAST のインストールとデプロイ（ご自分の API ゲートウェイ）

最後に APIcast のインストールおよびデプロイを行うには、「ローカル」デプロイメントの [Self-managed APIcast 2.0 チュートリアルに記載の手順に従います](#)。

ほとんど対応しています！最後の手順では、NGINX ゲートウェイを起動し、それを介していくつかのトラフィックを配置することです。まだ実行していない場合は、EC2 インスタンスの端末（以前は ssh で接続していたもの）に移動して、これを起動します。

最後のステップは、トラフィックが適切に承認されることを検証します。これを実行するには、以下にアクセスします。

```
http://your-public-ip/v1/words/awesome.json?app_id=APP_ID&app_key=APP_KEY
```

APP_ID および APP_KEY は、API 呼び出しを通じてアクセスするアプリケーションの ID です。

すべてが正常に機能していることが確認できたら、ポート 3000 の API バックエンドへのパブリックアクセスをブロックし、アクセス制御を回避します。

第4章 ORACLE DATABASE のリレーショナルデータベース管理システムを使用した 3SCALE API MANAGEMENT システムイメージのビルド

デフォルトでは、3scale には設定データを MySQL データベースに保管する **system** というコンポーネントが含まれています。任意で、デフォルトのデータベースをオーバーライドし、情報を外部の Oracle Database に保管することができます。本章の手順に従って、独自の Oracle Database クライアントバイナリーでカスタムのシステムコンテナイメージをビルドし、3scale を OpenShift にデプロイします。

4.1. 始める前に

4.1.1. Oracle ソフトウェアコンポーネントの取得

カスタムの 3scale システムコンテナイメージをビルドする前に、以下の Oracle [ソフトウェアコンポーネントのサポート対象バージョン](#) を取得する必要があります。

- Oracle Instant Client Package Basic または Basic Light
- Oracle Instant Client パッケージ SDK
- Oracle Instant Client パッケージ ODBC

4.1.2. 会議の前提条件

以下の前提条件を満たす必要があります。

- OpenShift クラスターからアクセスできる Oracle Database の [サポート対象バージョン](#)
- インストール手順に必要な Oracle Database の **system** ユーザーへのアクセス
- Red Hat 3scale 2.2 amp.yml テンプレートの作成

4.2. ORACLE DATABASE の準備

1. 新規データベースの作成

Oracle Database が 3scale と動作するようにするには、以下の設定が必要です。

```
ALTER SYSTEM SET max_string_size=extended SCOPE=SPFILE;
```

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

2. データベースの詳細を収集します。

3scale の設定に必要な以下の情報を取得します。

- Oracle Database の URL
- Oracle Database の [サービス名](#)
- Oracle Database の **system** ユーザー名およびパスワード
- Oracle Database の サービス名

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

4.3. システムイメージのビルド

1. [3scale-amp-openshift-templates github](#) リポジトリのクローンを作成します。
2. Oracle Database の Instant Client パッケージファイルを **3scale-amp-openshift-templates/amp/system-oracle/oracle-client-files** ディレクトリーに置きます。
3. `-f` オプションで **build.yml** OpenShift テンプレートを指定して **oc new-app** コマンドを実行します。

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

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

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

5. 「[Oracle Database の準備](#)」セクションで収集した以下の情報を指定して、以下のシェル **for** ループコマンドを入力します。

- **{USER}**: Oracle Database で 3scale を表すユーザー名
- **{PASSWORD}**: **USER** のパスワード
- **{ORACLE_DB_URL}**: Oracle Database の URL
- **{DATABASE}**: Oracle Database で作成したデータベースのサービス名
- **{PORT}**: Oracle Database のポート番号

```
for dc in system-app system-resque system-sidekiq system-sphinx; do oc env dc/$dc --
  overwrite DATABASE_URL="oracle-enhanced://{USER}:
  {PASSWORD}@{ORACLE_DB_URL}:{PORT}/{DATABASE}"; done
```

6. 上記のステップで提供した **USER**、**PASSWORD**、**ORACLE_DB_URL**、**PORT**、および **DATABASE** の同じ値を指定し、以下の **oc patch** コマンドを入力します。

```
$ oc patch dc/system-app -p '{"op": "replace", "path":
  "/spec/strategy/rollingParams/pre/execNewPod/env/1/value", "value": "oracle-
  enhanced://{USER}:{PASSWORD}@{ORACLE_DB_URL}:{PORT}/{DATABASE}"}' --
  type=json
```

7. 以下の **oc patch** コマンドを入力し、**SYSTEM_PASSWORD** フィールドに独自の Oracle Database の **system** ユーザーのパスワードを指定します。

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

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

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


第5章 オンプレミス型 3SCALE AMP インストールガイド

本ガイドでは、OpenShift テンプレートを使用して 3scale 2.2（オンプレミス型）OpenShift をインストールする方法を説明します。

5.1. 前提条件

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

5.2. 3SCALE AMP OPENSIFT テンプレート

Red Hat 3scale API Management Platform(AMP)2.2 は OpenShift テンプレートを提供します。このテンプレートを使用して AMP を OpenShift Container Platform にデプロイすることができます。

3scale AMP テンプレートは、以下で構成されます。

- Embedded APIcast API ゲートウェイ 2 つ
- 1 つの AMP 管理ポータルおよびデベロッパーポータルが永続ストレージを持つ

5.3. システム要件

3scale Api Management OpenShift テンプレートには、以下が必要です。

5.3.1. 環境要件

3scale API Management には、[「Red Hat 3scale API Management Supported Configurations」](#) で規定される環境が必要です。

永続ボリューム：

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

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

5.3.2. ハードウェア要件

ハードウェア要件は、使用の必要性に応じて異なります。Red Hat では、テストを行い、特定の要件を満たすように環境を設定することを推奨します。OpenShift 上の 3scale の環境を設定する場合、以下が推奨されます。

- クラウド環境へのデプロイメントには、コンピュータタスクに最適化したノードを使用します (AWS c4.2xlarge または Azure Standard_F8)。
- メモリーの要件が現在のノードで使用できる RAM よりも大きい場合、非常に大きなインストールでは、Redis に別のノードが必要になることがあります (AWS M4 シリーズまたは Azure Av2 シリーズ)。
- ルーティングタスクとコンピュータタスクには別のノードを使用します。

- コンピュートノードを 3scale 固有のタスクに割り当てます。
- バックエンドリスナーの **PUMA_WORKERS** 変数をコンピュートノードのコア数に設定します。

5.4. ノードとエンタイトルメントの設定

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

以下の手順を実行して、エンタイトルメントを設定します。

1. 各ノードに [Red Hat Enterprise Linux \(RHEL\)](#) をインストールします。
2. Red Hat [Subscription Manager\(RHSM\)](#)を使用して、ノードを Red Hat に登録します。
3. RHSM を使用して ノードを 3scale サブスクリプションに割り当てます。
4. 以下の要件に準拠して、ノードに [OpenShift](#) をインストールします。
 - サポートされている OpenShift バージョンを使用する必要があります。
 - 複数の書き込みをサポートするファイルシステムで永続ストレージを設定する必要があります。
5. [OpenShift コマンドラインインターフェース](#)をインストールします。
6. サブスクリプションマネージャーを使用して、**rhel-7-server-3scale-amp-2.2-rpms** リポジトリへのアクセスを有効にします。

```
sudo subscription-manager repos --enable=rhel-7-server-3scale-amp-2.2-rpms
```

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

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

5.5. テンプレートを使用した OPENSIFT への 3SCALE AMP のデプロイ

5.5.1. 前提条件:

- [第3章の「ノードおよびエンタイトルメントの設定」セクション](#)で指定されたとおりに設定された OpenShift クラスター
- OpenShift クラスターに対して解決する [ドメイン](#)（できればワイルドカード）
- Red Hat [コンテナーカタログ](#) へのアクセス
- (オプション)稼働中の電子メール機能用 SMTP サーバー

以下の手順に従って、.yaml テンプレートを使用して AMP を OpenShift にインストールします。

- [AMP テンプレートのインポート](#)
- [SMTP 変数の設定（任意）](#)

5.5.2. AMP テンプレートのインポート

AMP テンプレートを OpenShift クラスターにインポートするには、以下の手順を行います。

1. ターミナルセッションから OpenShift にログインします。

```
oc login
```

2. プロジェクトを選択するか新しいプロジェクトを作成します。

```
oc project <project_name>
```

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

3. **oc new-app** コマンドを入力します。

- **--file** オプションを使用して、「ノードおよびエンタイトルメントの設定」セクションでダウンロードした **amp.yml** ファイルへのパスを指定します。
- **--param** オプションを使用して、**WILDCARD_DOMAIN** パラメーターに OpenShift クラスターのドメインを設定します。
- また、任意で **--param** オプションで **WILDCARD_POLICY** パラメーターに **subdomain** を設定すると、ワイルドカードドメインルーティングを有効にすることができます。

ワイルドカードルーティングを有効にしない場合:

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

ワイルドカードルーティングを有効にする場合:

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

4. ターミナルには、新たに作成された AMP 管理ポータルにマスターおよびテナント URL および認証情報が表示されます。この出力には以下の情報が含まれます。

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

```
Log in to https://user-admin.3scale-project.example.com as admin/xXxXyz123.
```

```
...
```

```
* With parameters:
```

```
* ADMIN_PASSWORD=xXxXyz123 # generated
```

```

* ADMIN_USERNAME=admin
* TENANT_NAME=user

...

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

...

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

...

```

後で確認できるようにするため、詳細を書き留めておきます。



注記

AMP が OpenShift に完全にデプロイされ、ログインとクレデンシャルが有効になるまで数分かかる場合があります。

詳細情報

OpenShift でのワイルドカードドメインに関する詳細は、「[\(サブドメインの\) ワイルドカードルートの使用](#)」を参照してください。

5.5.3. SMTP 変数の設定（任意）

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

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

1. OpenShift にログインしていない場合はログインします。

```
oc login
```

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

変数	説明
----	----

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

たとえば、以下ようになります。

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

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

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

5.6. 3SCALE AMP テンプレートパラメーター

テンプレートパラメーターは、デプロイメント時およびデプロイメント後の AMP yml テンプレートの環境変数を設定します。

名前	説明	デフォルト値	必須?
AMP_RELEASE	AMP リリースタグ。	2.2.0	はい
ADMIN_PASSWORD	無作為に生成される AMP 管理者アカウントのパスワード	該当なし	必須

ADMIN_USERNAME	AMP 管理者アカウントのユーザー名	admin	はい
APICAST_ACCESS_TOKEN	APICAST が設定のダウンロードに使用する読み取り専用アクセストークン	該当せず	必須
ADMIN_ACCESS_TOKEN	すべての API をスコープとし、書き込みアクセス権限が設定された管理者アクセストークン	該当せず	任意
WILDCARD_DOMAIN	ワイルドカードルートのルートドメイン。たとえば、ルートドメイン example.com は 3scale-admin.example.com を生成します。	該当せず	必須
WILDCARD_POLICY	この値を「Subdomain」に設定して、組み込まれた APICAST ゲートウェイへのワイルドカードルートを有効にします。	なし	はい
TENANT_NAME	ルート下のテナント名。 -admin 接尾辞を付けて Admin UI が利用できるようになります。	3scale	はい
MYSQL_USER	データベースのアクセスに使用される MySQL ユーザーのユーザー名	mysql	はい
MYSQL_PASSWORD	MySQL ユーザーのパスワード	該当せず	必須
MYSQL_DATABASE	アクセスされた MySQL データベースの名前	system	はい
MYSQL_ROOT_PASSWORD	Root ユーザーのパスワード	該当せず	必須
SYSTEM_BACKEND_USERNAME	内部 3scale api auth の内部 3scale API ユーザー名	3scale_api_user	はい

SYSTEM_BACKEND_PASSWORD	内部 3scale api auth の内部 3scale API パスワード	該当せず	必須
REDIS_IMAGE	使用する Redis イメージ	registry.access.redhat.com/rhscsl/redis-32-rhel7:3.2	はい
MYSQL_IMAGE	使用する Mysql イメージ	registry.access.redhat.com/rhscsl/mysql-57-rhel7:5.7-5	はい
SYSTEM_BACKEND_SHARED_SECRET	バックエンドからシステムにイベントをインポートするための共有シークレット	該当せず	必須
SYSTEM_APP_SECRET_KEY_BASE	システムアプリケーションの秘密鍵ベース	該当せず	必須
APICAST_MANAGEMENT_API	APICAST Management API のスコープ。 disable、status、または debug を設定できます。ヘルスチェックには最低でも status が必要です。	status	no
APICAST_OPENSSL_VERIFY	設定のダウンロード時に OpenSSL ピア検証を有効または無効にします。 true または false を設定できます。	false	no
APICAST_RESPONSE_CODES	APICAST のログインレスポンスコードを有効にします。	true	任意
APICAST_REGISTRY_URL	APICAST ポリシーの場所に解決する URL	http://apicast-staging:8090/policies	はい
MASTER_USER	マスター管理者アカウントのユーザー名	master	はい
MASTER_NAME	マスター管理ポータルサブドメイン値。 - master 接尾辞が付けられます。	master	はい
MASTER_PASSWORD	無作為に生成されるマスター管理者のパスワード	該当せず	必須

MASTER_ACCESS_TOKEN	API 呼び出しのマスターレベル権限が設定されたトークン	該当せず	はい
---------------------	------------------------------	------	----

5.7. OPENSIFT 上の AMP と共に APICAST を使用

OpenShift 上で AMP を使用する APIcast は、ホスト AMP を使用する APIcast とは異なり、固有の設定手順が必要です。

本セクションでは、AMP を OpenShift で APIcast をデプロイする方法を説明します。

5.7.1. AMP に関する既存の OpenShift クラスターへの APIcast テンプレートのデプロイ

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

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

1. [以下の設定でアクセストークンを作成します。](#)

- スコープ：Account Management API
- 読み取り専用アクセスがある

2. APIcast クラスターにログインします。

```
oc login
```

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

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



注記

TENANT_NAME 管理 UI が利用できるルート下の名前です。 **TENANT_NAME** 「3scale」のデフォルト値です。AMP デプロイメントでカスタム値を使用した場合は、ここでその値を使用する必要があります。

4. 3scale GitHub にある **apicast.yml** をダウンロードし、**oc new-app** コマンドを実行し、**apicast.yml** ファイルに **--file** オプションを指定し、APIcast テンプレートをインポートします。

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


5.7.2. AMP に関する OpenShift クラスターとは別の OpenShift クラスターからの APIcast への接続

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

1. 以下の設定でアクセストークンを作成します。

- スコープ：Account Management API
- 読み取り専用アクセスがある

2. APIcast クラスターにログインします。

```
oc login
```

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

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



注記

TENANT_NAME 管理 UI が利用できるルート下の名前です。 **TENANT_NAME** のデフォルト値は「3scale」です。AMP デプロイメントでカスタム値を使用した場合は、ここでその値を使用する必要があります。

4. `oc new-app` コマンドで、APIcast を OpenShift クラスター外部の OpenShift クラスターにデプロイします。 **--file** オプションで **apicast.yml** ファイルのファイルパスを指定します。

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

5. `apicast` **BACKEND_ENDPOINT_OVERRIDE** 環境変数を URL **backend.** に設定し、その後に AMP デプロイメントが含まれる OpenShift クラスターのワイルドカードドメインを設定します。

```
oc env dc/apicast --overwrite BACKEND_ENDPOINT_OVERRIDE=https://backend-
<TENANT_NAME>.<WILDCARD_DOMAIN>
```

5.7.3. 他のデプロイメントからの APIcast の接続

他のプラットフォームに APIcast をデプロイしたら、AMP OpenShift クラスターで **BACKEND_ENDPOINT_OVERRIDE** 環境変数を設定すると、OpenShift 上の AMP に接続できます。

1. AMP OpenShift クラスターにログインします。

```
oc login
```

2. `system-app` オブジェクト **BACKEND_ENDPOINT_OVERRIDE** 環境変数を設定します。

ネイティブインストールを使用している場合は、以下を実行します。

```
BACKEND_ENDPOINT_OVERRIDE=https://backend.<your_openshift_subdomain>
bin/apicast
```

Docker コンテナ環境を使用している場合には、以下を実行します。

```
docker run -e BACKEND_ENDPOINT_OVERRIDE=https://backend.
<your_openshift_subdomain>
```

5.7.4. 組み込み APIcast のデフォルト動作の変更

外部の APIcast デプロイメントでは、APIcast OpenShift [テンプレートのテンプレートパラメーターを変更することで](#)、デフォルトの動作を変更できます。

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

5.7.5. 内部サービスルートを経た単一 OpenShift クラスターでの複数 APIcast デプロイメントの接続

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

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

ovs-subnet

ovs-subnet OpenShift SDN プラグインを使用している場合は、以下の手順に従って内部ルート経由で接続します。

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

```
oc login
```

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

- **--param** オプションを使用して、**BACKEND_ENDPOINT_OVERRIDE** パラメーターに OpenShift クラスターの AMP プロジェクトのドメインを設定します。

```
oc new-app -f apicast.yml --param BACKEND_ENDPOINT_OVERRIDE=http://backend-listener.
<AMP_PROJECT>.svc.cluster.local:3000
```

ovs-multitenant

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

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

```
oc login
```

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

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

3. **oc new-app** と **apicast.yml** ファイルへのパスを入力します。

- **--param** オプションを使用して、**BACKEND_ENDPOINT_OVERRIDE** パラメーターに OpenShift クラスターの AMP プロジェクトのドメインを設定します。

```
oc new-app -f apicast.yml --param BACKEND_ENDPOINT_OVERRIDE=http://backend-listener.<AMP_PROJECT>.svc.cluster.local:3000
```

詳細情報

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

5.8.7.トラブルシューティング

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

- [以前の Deployment Leaves Dirty Persistent Volume Claim](#)（永続ボリューム要求、PVC）
- [Docker レジストリーから誤ってプルする](#)
- [永続ボリュームがローカルでマウントされる場合の MySQL の権限の問題](#)
- [永続ボリュームは OpenShift によって Writable ではないため、Logo または Images B をアップロードできない](#)
- [OpenShift でのセキュリティー保護されたルートの作成](#)
- [AMP 以外のプロジェクトでの APICAST とシークレットの問題のデプロイに失敗する](#)

5.8.1. 以前の Deployment Leaves Dirty Persistent Volume Claim（永続ボリューム要求、PVC）

問題

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

原因

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

解決策

1. **oc get pvc** を使用してエラーのある MySQL データが含まれる PVC を検索します。

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

```
mysql-storage      Bound  vol006  100Gi  RWO,RWX  4d
system-redis-storage Bound  vol008  100Gi  RWO,RWX  4d
system-storage     Bound  vol004  100Gi  RWO,RWX  4d
```

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

5.8.2. Docker レジストリーから誤ってプルする

問題

インストール中に以下のエラーが発生する。

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

原因

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

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

解決方法

適切なバージョンの Docker コンテナ環境を使用します。

5.8.3. 永続ボリュームがローカルでマウントされる場合の MySQL の権限の問題

問題

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

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

原因

MySQL プロセスが不適切なユーザー権限で起動されている。

解決方法

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

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

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

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

5.8.4. 永続ボリュームが OpenShift によって Writable ではないため、Logo または Images をアップロードできない

問題

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

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

原因

OpenShift が永続ボリュームに書き込みを行うことができない。

解決方法

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

5.8.5. OpenShift でのセキュリティー保護されたルートの作成

問題

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

原因

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

解決策

OpenShift ルーター設定で "secure route" チェックボックスが有効であることを確認します。

5.8.6. Secret の問題が原因で AMP の別のプロジェクトでの APIcast デプロイに失敗する

問題

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

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

以下のエラーが Pod に表示されます。

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

原因

シークレットが適切に設定されていない。

解決方法

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

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

第6章 RED HAT 3SCALE AMP 2.2 ON-PREMISES OPERATIONS AND SCALING GUIDE

6.1. はじめに

本書では、オンプレミス型 Red Hat 3scale AMP 2.2 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。

6.1.1. 前提条件

本書の手順を実施する前に、サポートされる OpenShift バージョンでオンプレミス型 AMP をインストールして初期設定している必要があります。

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。

6.1.2. 詳細はこちら

- [『開発者ガイド』の「アプリケーションの正常性」](#)
- [OpenShift のドキュメント](#)

6.2. APICAST の再デプロイ

オンプレミス型 AMP をデプロイし、希望の APIcast デプロイメント方法を確認したら、AMP ダッシュボード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. システムの変更
2. UI でステージング環境にデプロイしてテスト
3. UI で実稼働環境にプロモートします。
4. 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. システムの変更
3. UI でステージング環境にデプロイしてテスト

4. UI で実稼働環境にプロモートします。
5. APIcast は設定された頻度でプロモートされた更新を取得しパブリッシュします。

6.3. APICAST BUILT-IN ワイルドカードルーティング（テクノロジープレビュー）

AMP デプロイメントに付随する組み込み APIcast ゲートウェイは、サブドメインレベルで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。この機能により、実稼働およびステージング用の公開ベース 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、オンプレミスのインストール中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はテクノロジープレビューです。現在のテクノロジープレビューには、以下の制限があります。

- アンダーラインを含む HTTP ヘッダーを使用すると、サービスは 403 のエラーコードで失敗します。回避策として、すべてのヘッダー名からアンダースコアを削除します。
- テンプレートパラメーター **TENANT_NAME** を数字で起動しない値に設定する必要があります。

AMP は DNS 機能を提供しないため、指定した公開ベース 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。

6.3.1. ワイルドカードの変更

ワイルドカードを変更するには、以下の手順を実行します。

1. AMP にログインします。
2. API ゲートウェイの設定ページに移動します：**APIs → your API → Integration → edit APIcast configuration**
3. 任意の文字列のプレフィックスを指定してステージング環境および実稼働環境用の公開ベース URL を変更し、以下の要件に従います。
 - API エンドポイントは数値文字で開始できない

以下は、ドメイン **example.com** 上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
```

詳細情報

[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。](#)

6.4. オンプレミス型 AMP のスケールアップ

6.4.1. 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、以下のいずれかの方法で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。

6.4.1.1. 方法 1、バックアップ、および Swap 永続ボリューム

1. 既存の永続ボリューム上のデータのバックアップ
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、新しい PVC のサイズと永続ボリュームの名前を指定します。永続ボリューム名には **volumeName** フィールドを使用します。
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。

6.4.1.2. 方法 2:AMP のバックアップおよび再デプロイ

1. 既存の永続ボリューム上のデータのバックアップ
2. 3scale Pod のシャットダウン
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。
 - 新しい PVC のサイズ
 - 永続ボリューム名 (**volumeName** フィールドを使用)
6. AMP.yml のデプロイ
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。

6.4.2. パフォーマンスのスケールアップ

6.4.2.1. 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 クラスターへのログイン

```
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、必要な Pod ごとのプロセス数を設定します。

- **APICAST_WORKERS** APICast Pod の場合（Red Hat では、デプロイメントごとに2を超えないことを推奨します）
- バックエンド Pod: **PUMA_WORKERS**
- システム Pod: **UNICORN_WORKERS**

```
oc env dc/apicast --overwrite APICAST_WORKERS=<number_of_processes>
```

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

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

6.4.2.2. 垂直および Horizontalハードウェアのスケーリング

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）、または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）が可能です。

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限り、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。

備考

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。

6.4.2.3. 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。

6.4.2.4. 詳細はこちら

- タスクのスケールリング、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

6.5. 操作のトラブルシューティング

6.5.1. 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. ログを必要とする Pod の ID を確認します。

```
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 にはコンテナが 2 つあり、それぞれに別個のログがあります。コンテナのログにアクセスするには、**--container** パラメーターで **system-provider** および **system-developer** を指定します。

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

6.5.2. ジョブキュー

ジョブキューには、**system-resque** および **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>
```

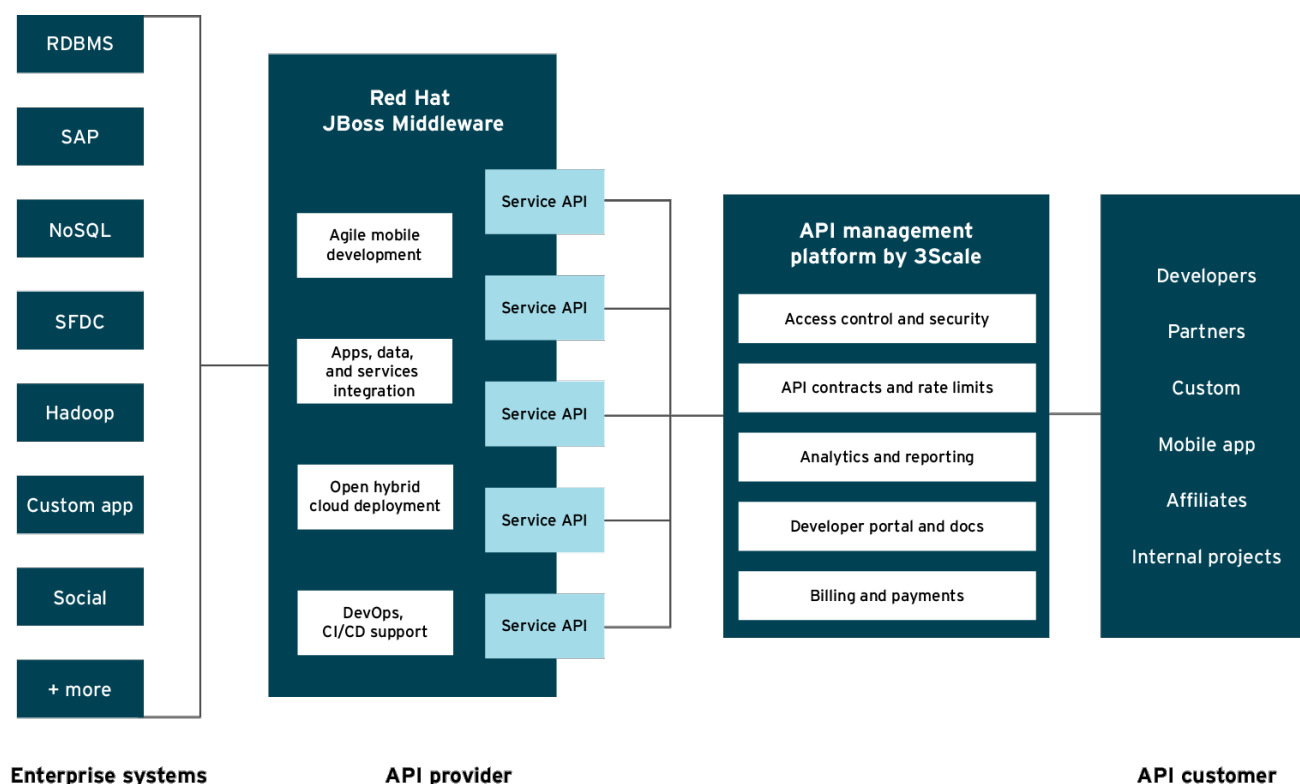
第7章 HOW TO DEPLOYING A FULL-STACK API SOLUTION WITH FUSE, 3SCALE, AND OPENSSHIFT

このチュートリアルでは、Red Hat JBoss xPaaS for OpenShift および 3scale API Management Platform - Cloud を使用して、フルスタック API ソリューション（API 設計、開発、ホスト、アクセス制御、モニタリングなど）を取得する方法を説明します。

チュートリアルは、フルスタック API ソリューションを提供するために Red Hat と 3scale 間の連携に基づいています。このソリューションには、Red Hat JBoss xPaaS for OpenShift での API の設計、開発、ホスト、完全な制御、可視性、およびモニタリング機能のための 3scale API Management Platform を組み合わせたものです。

API 自体は Red Hat JBoss xPaaS for OpenShift にデプロイすることができます。OpenShift では、クラウドおよびオンプレミス（Red Hat 部分）でホストできます。API 管理（3scale の部分）は、3scale APIcast または OpenShift を使用して Amazon Web Services(AWS)でホストすることができます。これにより、デプロイメント柔軟性を最大限にするためにさまざまな設定オプションが提供されます。

以下の図は、この結合ソリューションの主要要素の概要を示しています。エンタープライズバックエンドシステム、ミドルウェア、API 管理、および API ユーザーを含む統合チェーン全体を表示します。



JB0095

具体的なサポートに関する質問は、サポートにご連絡ください。

このチュートリアルでは、3つの異なるデプロイメントシナリオの手順をステップごとに説明します。

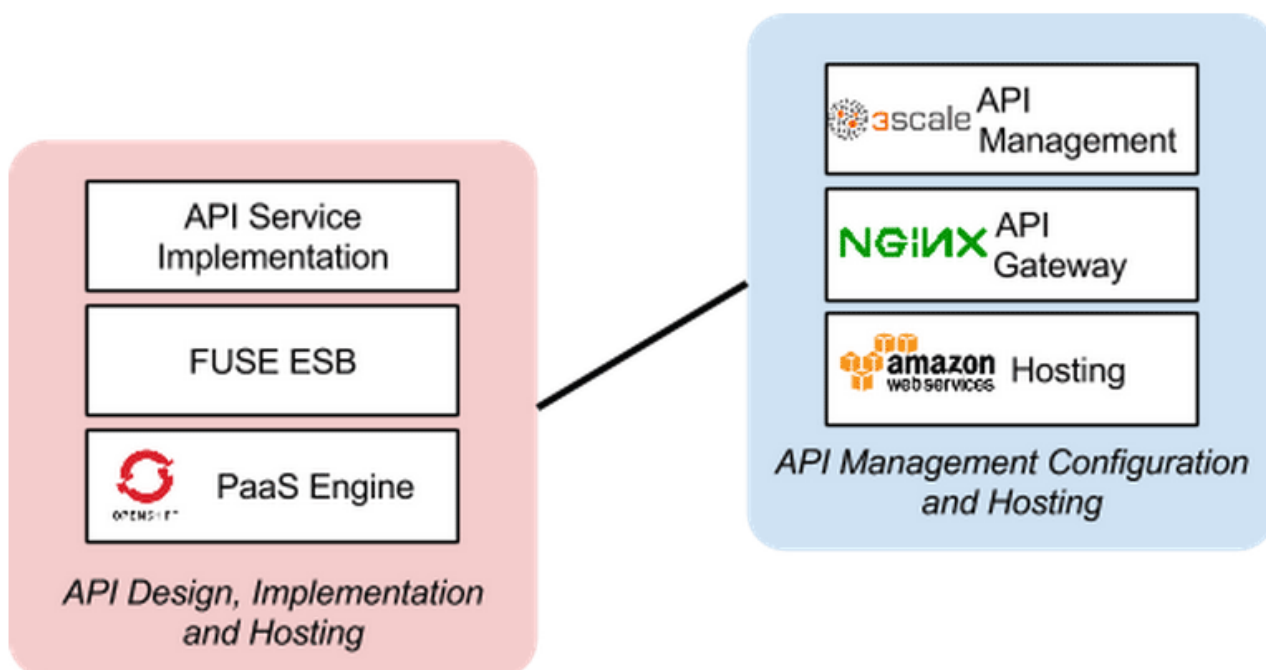
1. シナリオ 1: API が含まれる [Fuse on OpenShift](#) アプリケーションAPI は、3scale [AMI](#) を使用して [Amazon Web Services\(AWS\)](#)でホストされる [API ゲートウェイ](#)で 3scale によって管理されます。

2. シナリオ 2: API が含まれる Fuse on OpenShift アプリケーションAPI は、[APIcast](#)（3scale のクラウドホスト API ゲートウェイ）でホストされる API ゲートウェイで 3scale により管理されます。
3. シナリオ 3 - API が含まれる Fuse on OpenShift アプリケーションAPI は、[OpenShift](#)でホストされる API ゲートウェイを使用して 3scale により管理されます。

このチュートリアルは、以下の 4 つの部分に分割されます。

- [パート 1](#): API を設計および実装する [Fuse on OpenShift](#) 設定
- [パート 2](#): 3scale API Management の設定
- [パート 3](#): API サービスの統合
- [パート 4](#): API および API 管理のテスト

以下の図は、この設定におけるさまざまな部分のプレイのロールを示しています。

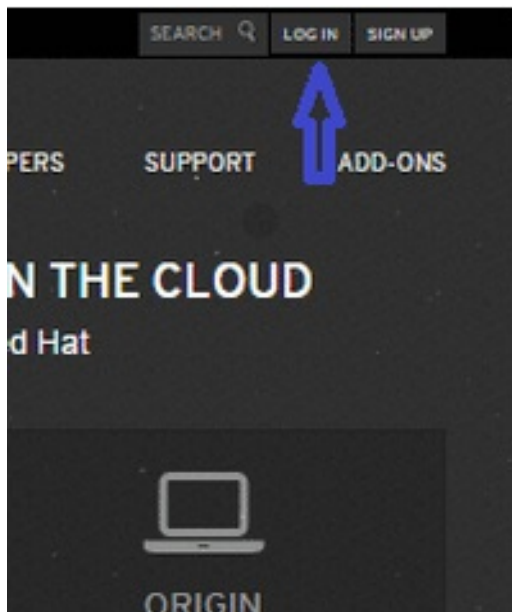


7.1. パート 1: FUSE ON OPENSIFT の設定

管理する API が含まれる [Fuse on OpenShift](#) アプリケーションを作成します。Fuse 6.1 に含まれる REST クイックスタートを使用します。小規模なギアを使用するので、中規模または大規模なギアが必要になると、メモリーエラーや破損のパフォーマンスが低下します。

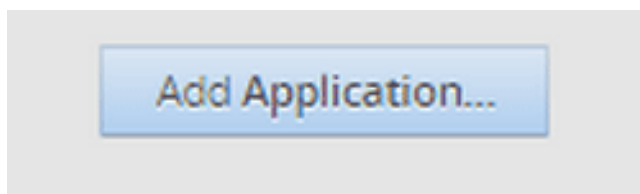
7.1.1. ステップ 1

OpenShift オンラインアカウントにサインインします。OpenShift オンラインアカウントにサインアップしていない場合はサインアップしてください。



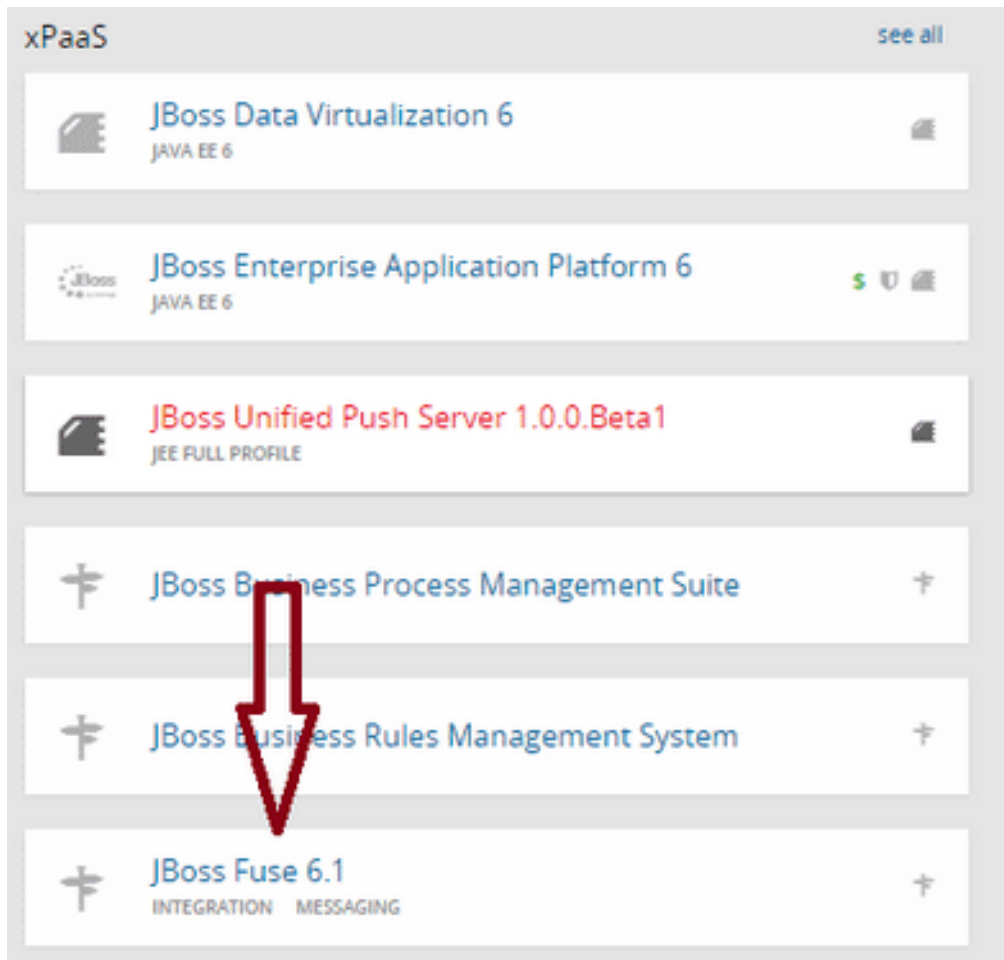
7.1.2. ステップ 2

署名後に「add application」 ボタンをクリックします。



7.1.3. ステップ 3

xPaaS で、アプリケーションの Fuse タイプを選択します。



7.1.4. ステップ 4

これでアプリケーションを設定します。アプリケーションのように、"restapitest" など、表示するサブドメインを入力します。これにより、「appname-domain.rhcloud.com」の形式の完全 URL が「restapitest-ossmentor.rhcloud.com」になります。ギアのサイズは、Fuse カートリッジに必要な medium または large に変更します。「Create application」をクリックします。

Applications
Settings
Support
Add-ons

1 Choose a type of application
2 **Configure the application**
3 Next steps

Based On

JBoss Fuse 6.1 Quickstart

The JBoss Fuse enterprise service bus is a technology for building and implementing communication between different applications, services and data. It's specifically designed for extensive connectivity. This cartridge is an alpha release of JBoss Fuse 6.1 for OpenShift.

Note: It is recommended that you use a medium sized gear to deploy JBoss Fuse due to memory requirements. Running in a small gear may result in slow interface responsiveness.

[Learn more](#)

☆ OpenShift maintained

Does not receive automatic security updates

Public URL

OpenShift will automatically register this domain name for your application. You can add your own domain name later.

Source Code

We'll create a Git code repository in the cloud, and populate it with a set of reasonable defaults. If you provide a Git URL, your application will start with an exact copy of the code and configuration provided in this Git repository.

Gears

Gears are the application containers running your code. For most applications, the small gear size provides plenty of resources. If you require more resources, select a different gear size here. You can also [upgrade your plan](#) to get access to more gear sizes.

Cartridges

manifest.yml

Applications are composed of cartridges - each of which exposes a service or capability to your code. All applications must have a web cartridge.

Downloaded cartridges do not receive updates automatically.

Scaling

OpenShift automatically routes web requests to your web gear. If you allow your application to scale, we'll set up a load balancer and allocate more gears to handle traffic as you need it.

Region

☐ No preference

☒ **aws-us-east-1**
All gear sizes can be deployed to the US Region.

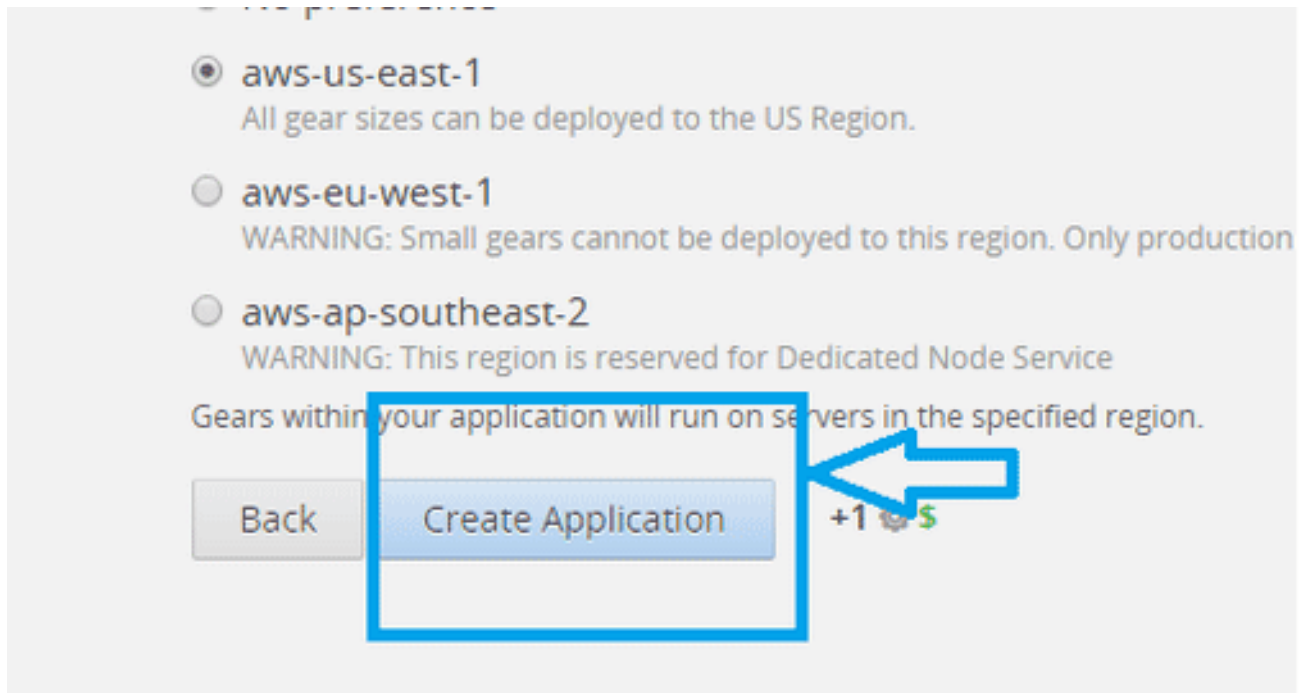
☐ aws-eu-west-1
WARNING: Small gears cannot be deployed to this region. Only production gears can be deployed to the EU Region (small,highcpu, medium, and large).

☐ aws-ap-southeast-2
WARNING: This region is reserved for Dedicated Node Service
Gears within your application will run on servers in the specified region.

+1 \$

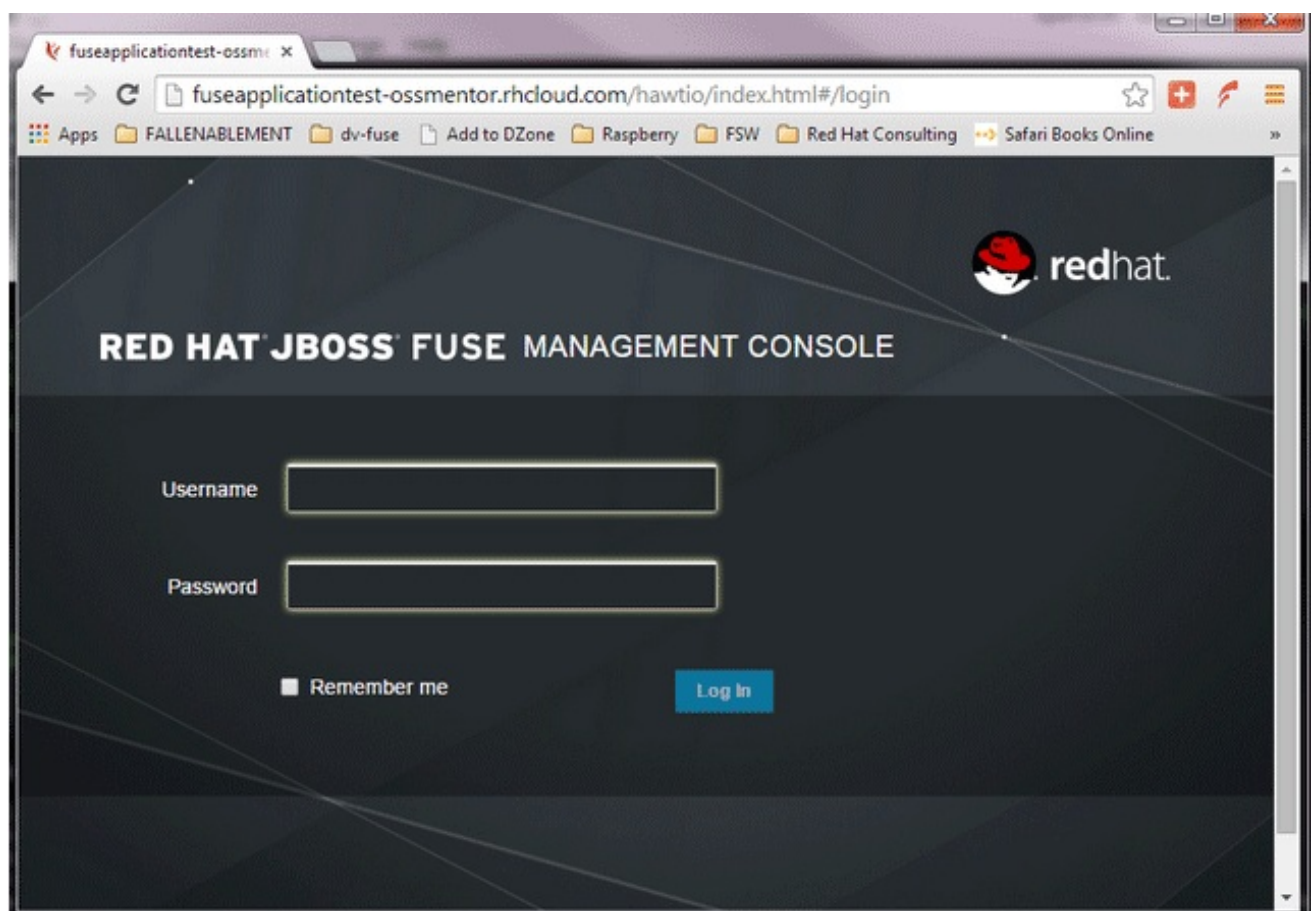
7.1.5. ステップ 5

「Create application」をクリックします。



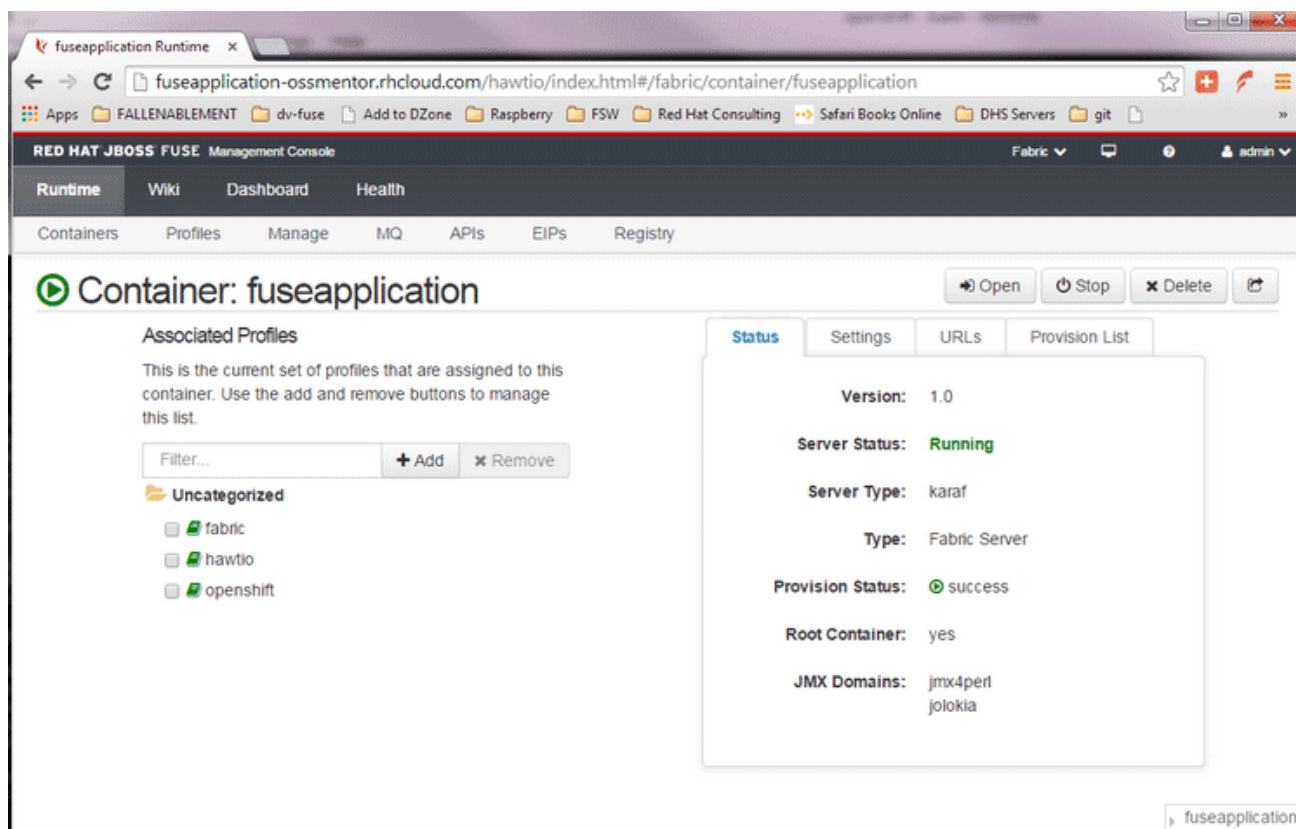
7.1.6. ステップ 6

アプリケーションの hawtio コンソールを参照し、サインインします。



7.1.7. ステップ 7

署名後、"runtime" タブおよびコンテナをクリックし、REST API の例を追加します。

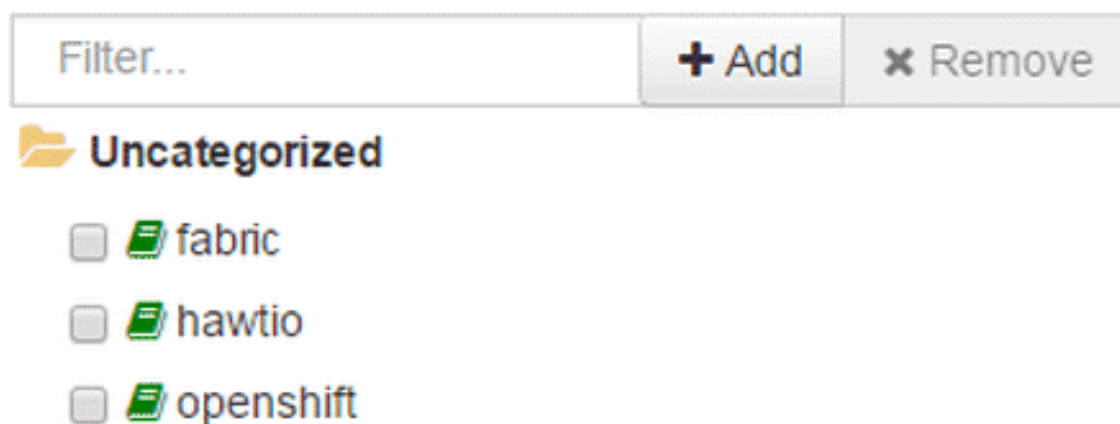


7.1.8. ステップ 8

「プロファイルの追加」ボタンをクリックします。

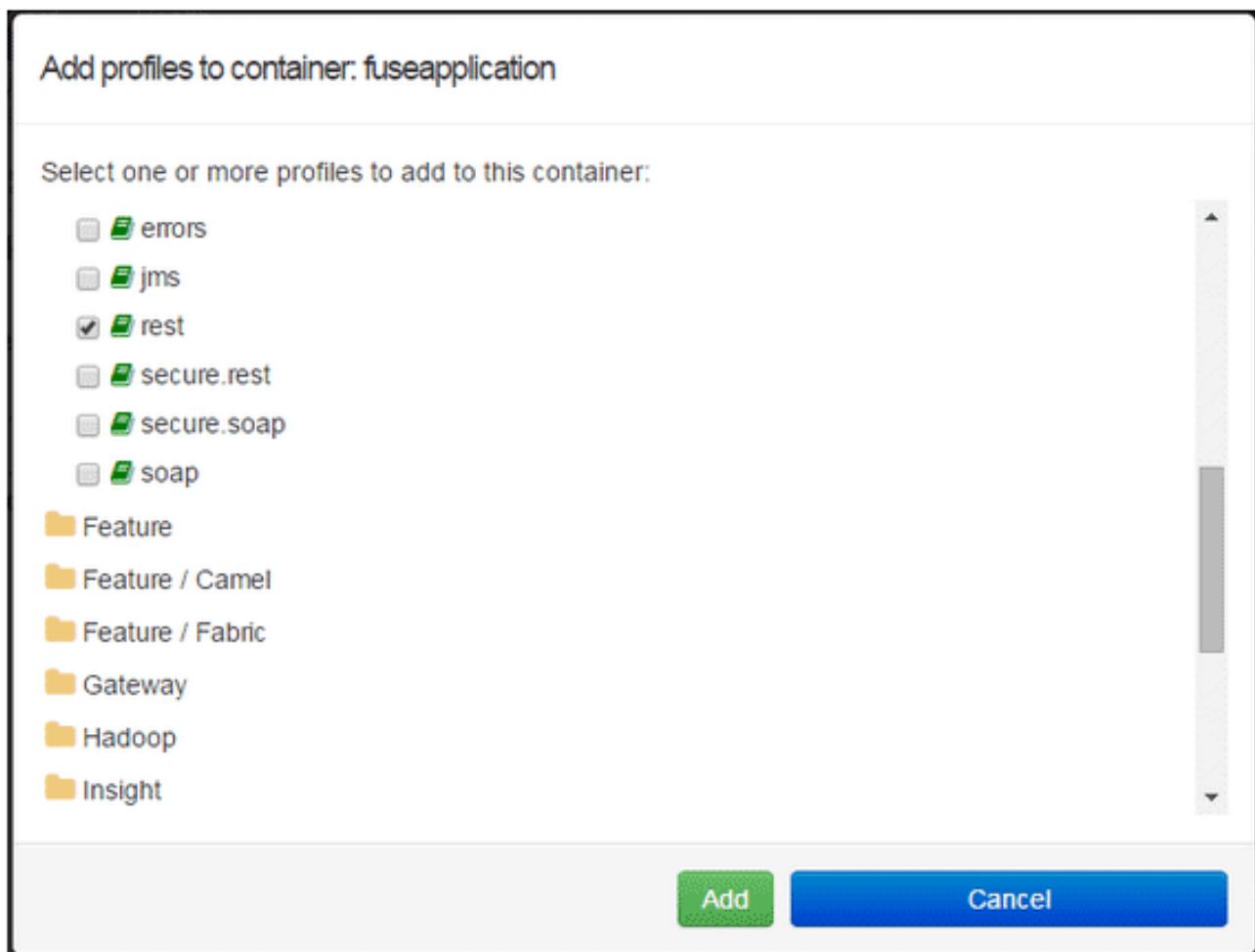
Associated Profiles

This is the current set of profiles that are assigned to this container. Use the add and remove buttons to manage this list.



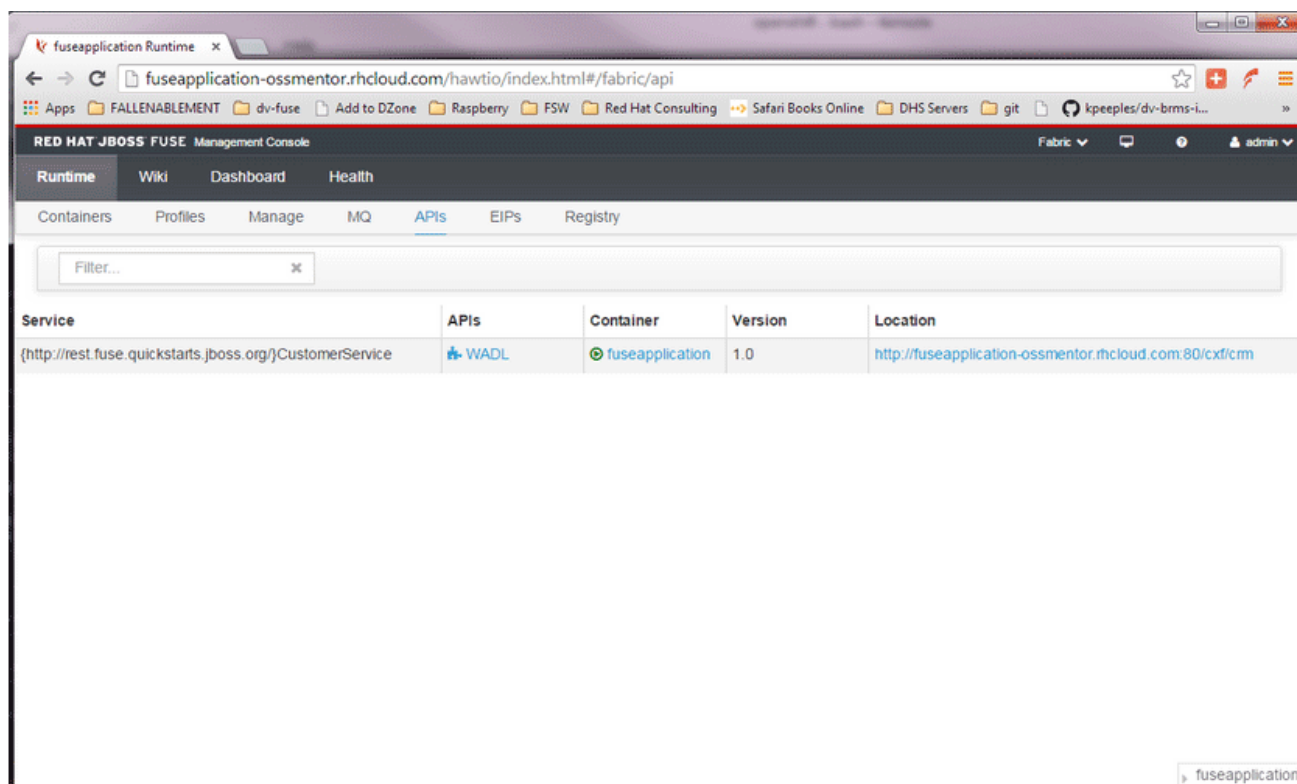
7.1.9. ステップ 9

examples/quickstarts にスクロールダウンし、「REST」チェックボックス、さらに「add」をクリックします。REST プロファイルは、コンテナに関連付けられたプロファイルページに表示されるはずです。



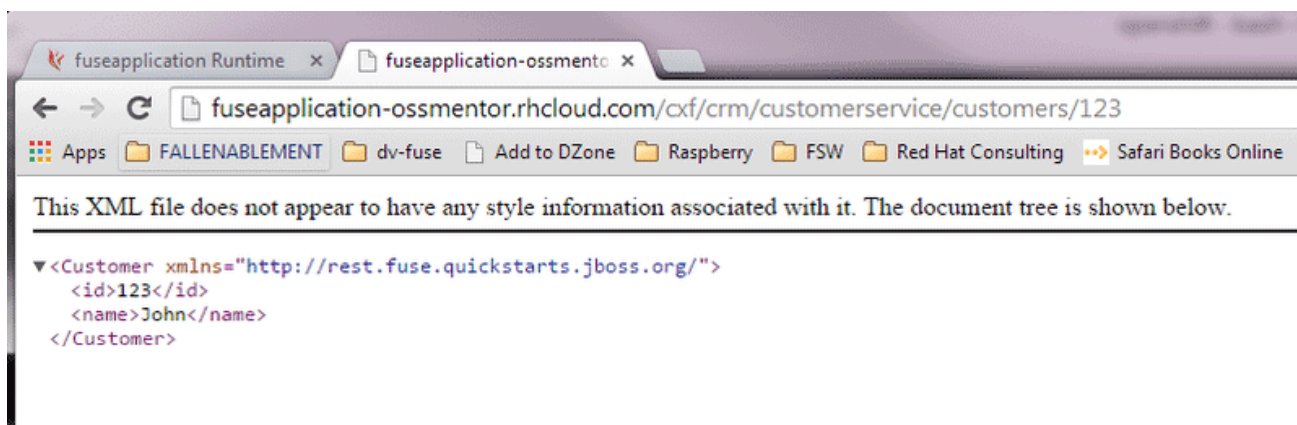
7.1.10. ステップ 10

runtime/APIs タブをクリックして REST API プロファイルを確認します。



7.1.11. ステップ 11

REST API が機能していることを確認します。XML 形式の ID と名前を返す顧客 123 を参照します。



7.2. パート 2: 3SCALE API MANAGEMENT の設定

3scale API Management を使用してパート 1 で作成した API を保護するには、まず、該当する 3 つのシナリオのいずれかに従って、後でデプロイした設定（設定）を実行する必要があります。

OpenShift で API を設定したら、これを 3scale で設定して、アクセス制御と使用状況のモニタリングの管理層を提供することができます。

7.2.1. ステップ 1

3scale アカウントにログインします。3scale アカウントをお持ちでなければ、www.3scale.net でサインアップすることができます。アカウントへの初回ログイン時には、ウィザードに従って API と 3scale の統合の基本について説明します。

7.2.2. ステップ 2

API > Integration で、先ほど作成した OpenShift アプリケーションの Fuse アプリケーションの公開 URL を入力します（例: "restapitest-ossmentor.rhcloud.com"）。これにより、ステージング環境で 3scale API ゲートウェイに対する設定をテストします。ステージング環境の API ゲートウェイを使用すると、プロキシ設定を AWS にデプロイする前に 3scale 設定をテストすることができます。

Staging: 3scale-hosted to configure & test your integration [documentation](#)

[deployed](#) | [deployment history](#)

The screenshot shows the 3scale Staging configuration interface. On the left, a vertical green line connects three icons: a puzzle piece for API, a server for API GATEWAY, and a smartphone for CLIENT. Each icon has a question mark in a blue circle to its right.

API

Private Base URL* [Use Echo API](#)

Private address of your API that will be called by the API gateway.

API GATEWAY

Public Base URL*

Public address of your API gateway in the staging environment. You can use this address to call the API for testing purposes.

▶ MAPPING RULES

▶ AUTHENTICATION SETTINGS

CLIENT

API test GET request

Optional GET request to a API gateway endpoint. We will use this call to validate your API gateway setup using credentials of the first live application. You can try it yourself by copying the following command into your shell:

```
curl "https://api-2445581450779.staging.apicast.io:443/v1/word/good.json?user_key=44e72dedd214c812990c1b3ab12f5ba3"
```

Connection between client, gateway & API is working correctly as [reflected in the analytics section](#).

[Update & Test Staging Configuration](#)

7.2.3. ステップ 3

次の手順では、監視および流量制御する API メソッドを設定します。そのためには、API > Definition に移動し、「New method」をクリックします。



Definition

Name: API
System Name: api

[edit](#)[Create new method](#)

Methods

Add the methods of this API to get data on their individual usage. Method calls trigger the built-in Hits-metric. Usage limits and pricing rules for individual methods are defined from within each [Application Plan](#). A method needs to be mapped to one or more URL patterns in the [Mapping Rules section](#) of the integration page so specific calls to your API up the count of specific methods.

Method	System Name	Unit	Description	Mapped	New method
transactions/create_single	transactions/create_single	hit		✓	
transactions/create_multiple	transactions/create_multiple	hit		✓	
transactions/confirm	transactions/confirm	hit		✓	
transactions/destroy	transactions/destroy	hit		Add a mapping rule	

Metrics

Hits are the built-in top-level metric and the parent metric of the methods. Other top level metrics can be added here if needed. A metric needs to be mapped to one or more URL patterns in the [Mapping Rules section](#) of the integration page so specific calls to your API up the count of specific metrics.

[Create new metric](#)

Metric	System Name	Unit	Description	Mapped	New metric
Hits	hits	hit	Number of API hits	✓	
Number of transactions	transactions	transaction		Add a mapping rule	

メソッド作成に関する詳細は、[API 定義チュートリアル](#) を参照してください。

7.2.4. ステップ 4

アプリケーションプランで監視および制御を行うすべてのメソッドが完了したら、それらを API のエンドポイントで実際の HTTP メソッドにマッピングする必要があります。統合ページに戻り、「マッピングルール」セクションを展開します。

▼ MAPPING RULES

?

Verb	Pattern		+	Metric or Method (Define)
GET	/	1		hits
				Add Mapping Rule

アプリケーションプランで作成した各メソッドにマッピングルールを作成します。

Rule	Pattern	+/-	Create Proxy Rule
POST	/setAB	1	✓ hits getHelloMethodSystemName

この作業を完了したら、マッピングルールは以下のようになります。

▼ MAPPING RULES



Verb	Pattern	+	Metric or Method (Define)
GET	/v1/words/{word}.json	1	get_word
GET	/v1/sentences/{sentence}.json	1	get_sente
POST	/v1/words/{word}.json	1	set_word
+ Add Mapping Rule			

マッピングルールの詳細は、「マッピングルールに関するチュートリアル」を参照してください。

7.2.5. ステップ 5

「更新およびテスト」をクリックして設定を保存し、テストしたら、AWS で API ゲートウェイを設定できるようにする設定ファイルをダウンロードすることが準備が整います。API [ゲートウェイ](#)では、[nginx](#)と呼ばれる高パフォーマンスのオープンソースプロキシを使用する必要があります。

「production」セクションにスクロールダウンして、同じ統合ページで、nginx に必要な設定ファイルを確認できます。

Production: On-premises Gateway

To deploy an on-premises API gateway, add the Public Base URL of your API, download the Nginx Config files and [follow the documentation](#) to install in your servers.



API

Private Base URL



API GATEWAY

Public Base URL

Public address of your API gateway in the production environment. This is used to customize the server_name directive in the Nginx Config file which will otherwise be set to the variable \$hostname.

[Update Production Configuration](#)



[Download the Nginx Config files](#)

次のセクションでは、さまざまなホストシナリオを通じて実施します。

7.3. パート 3: API サービスの統合

3scale で API サービスを統合する方法は、さまざまな方法があります。ニーズに最も適したものを選択します。

- [AWS でホストされる APIcast](#)
- [Hosted APIcast](#)
- [APIcast on OpenShift](#)

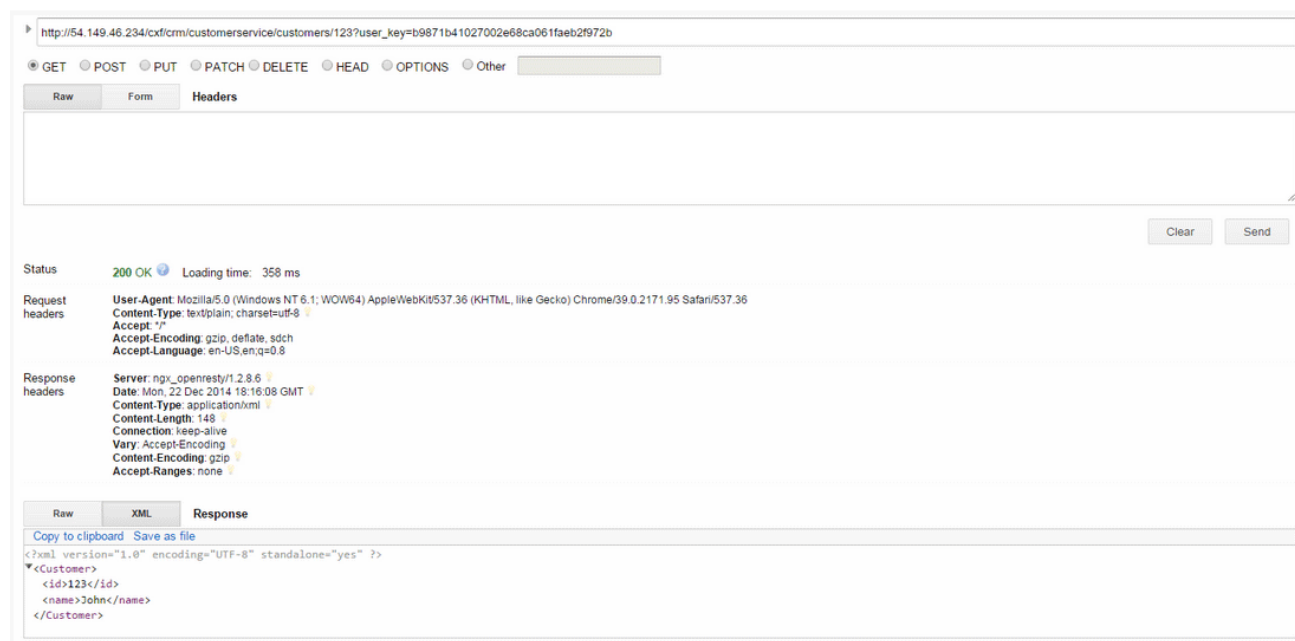
7.4. パート 4: API および API 管理のテスト

API の正しい機能をテストすることは、選択したシナリオから独立しています。favorite REST クライアントを使用して、以下のコマンドを実行します。

7.4.1. ステップ 1

ID 123 の顧客インスタンスを取得します。

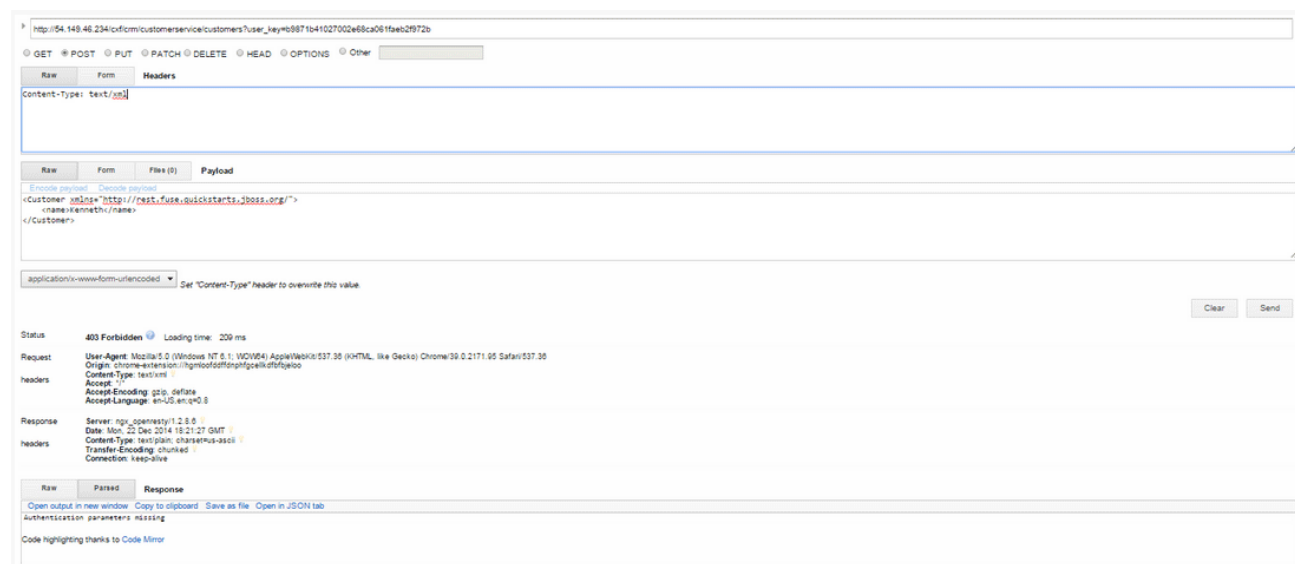
```
http://54.149.46.234/cxf/crm/customerservice/customers/123?
user_key=b9871b41027002e68ca061faeb2f972b
```



7.4.2. ステップ 2

顧客を作成します。

```
http://54.149.46.234/cxf/crm/customerservice/customers?
user_key=b9871b41027002e68ca061faeb2f972b
```



7.4.3. ステップ 3

ID 123 で顧客インスタンスを更新します。

`http://54.149.46.234/cxf/crm/customerservice/customers?user_key=b9871b41027002e68ca061faeb2f972b`

The screenshot shows a REST client interface with the following details:

- URL:** `http://54.149.46.234/cxf/crm/customerservice/customers?user_key=b9871b41027002e68ca061faeb2f972b`
- Method:** PUT
- Headers:** `Content-Type: text/xml`
- Payload:**

```
<?xml version='1.0' encoding='UTF-8'>
<Customer xmlns='http://rest.fuse.quickstarts.jboss.org/'>
  <name>Harry</name>
  <id>123</id>
</Customer>
```
- Status:** 403 Forbidden (Loading time: 200 ms)
- Request Headers:**

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36
Origin: chrome-extension://hgmofooffdofnpgfgeikofzofgeio
Content-Type: text/xml
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.9
```
- Response Headers:**

```
Server: nginx/1.2.3.4
Date: Mon, 22 Dec 2014 18:24:03 GMT
Content-Type: text/plain; charset=us-ascii
Transfer-Encoding: chunked
Connection: keep-alive
```

7.4.4. ステップ 4

ID 123 の顧客インスタンスを削除します。

`http://54.149.46.234/cxf/crm/customerservice/customers/123?user_key=b9871b41027002e68ca061faeb2f972b`

The screenshot shows a REST client interface with the following details:

- URL:** `http://54.149.46.234/cxf/crm/customerservice/customers/123?user_key=b9871b41027002e68ca061faeb2f972b`
- Method:** DELETE
- Headers:** (Empty)
- Payload:** (Empty)
- Status:** 403 Forbidden (Loading time: 211 ms)
- Request Headers:**

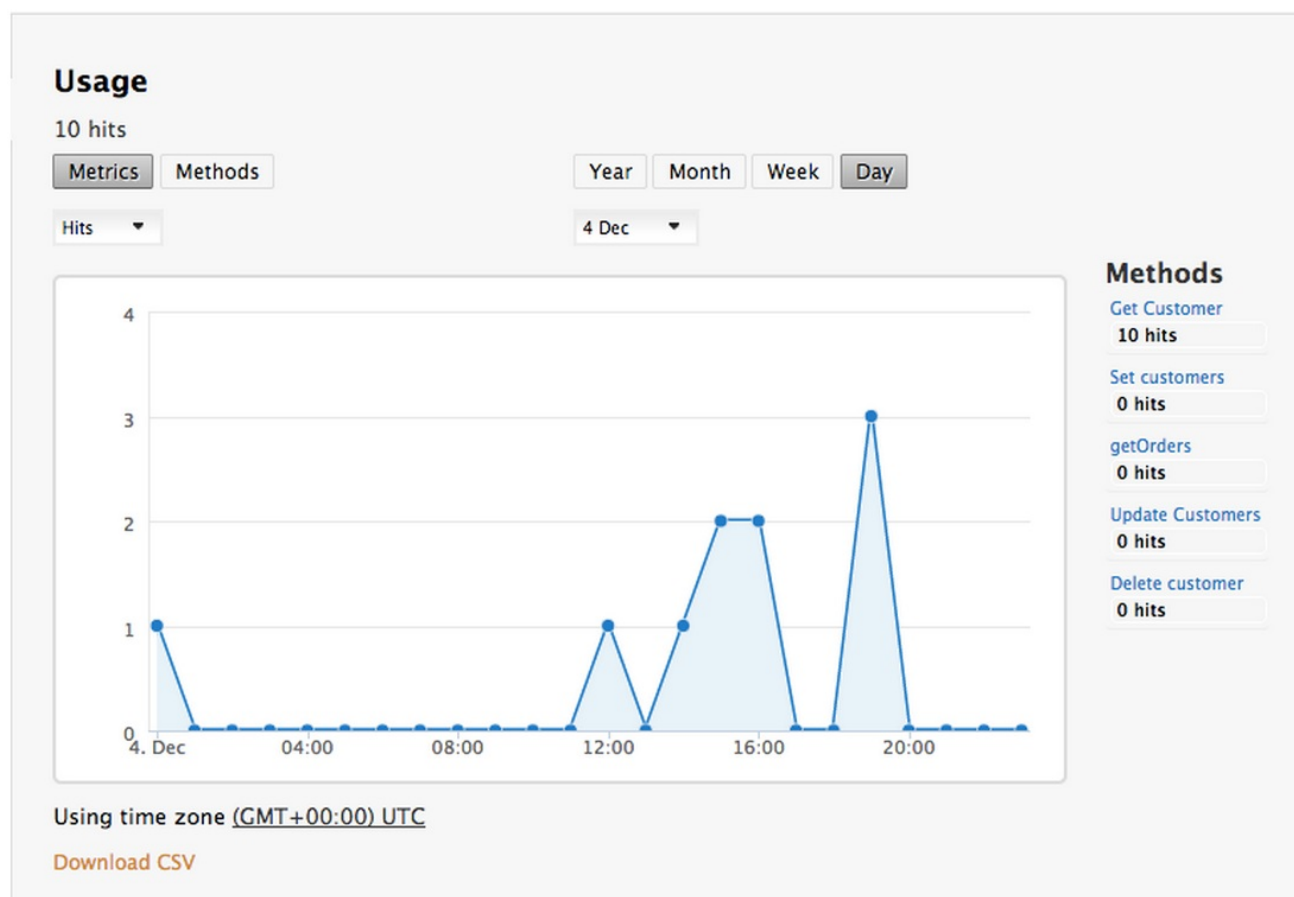
```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36
Origin: chrome-extension://hgmofooffdofnpgfgeikofzofgeio
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.9
```
- Response Headers:**

```
Server: nginx/1.2.3.4
Date: Mon, 22 Dec 2014 18:25:03 GMT
Content-Type: text/plain; charset=us-ascii
Transfer-Encoding: chunked
Connection: keep-alive
```

7.4.5. ステップ 5

API の API Management の分析を確認してください。

3scale アカウントにログインし、Monitoring > Usage に移動すると、グラフとして表される API エンドポイントのさまざまなヒットを確認できます。



これは API 管理の 1 つの要素であり、API を完全に制御することができます。その他の機能は以下のとおりです。

1. アクセス制御
2. 使用状況のポリシーおよび流量制御
3. レポート (Reporting)
4. API ドキュメントおよびデベロッパーポータル
5. Monetization and billing

特定の API Management の機能とその利点に関する詳細は、[3scale API Management Platform の製品ドキュメント](#)を参照してください。

特定の Red Hat Fuse 製品機能やその利点に関する詳細は、[JBoss Fuse Overview](#) を参照してください。

Red Hat Fuse on OpenShift の実行に関する詳細は、『[Getting Started with JBoss Fuse on OpenShift](#)』を参照してください。