OpenShift Container Platform 3.9

スタートガイド

OpenShift Container Platform 3.9 のスタートガイド

Last Updated: 2018-07-23

OpenShift Container Platform 3.9 スタートガイド

OpenShift Container Platform 3.9 のスタートガイド

法律上の通知

概要

開発者の場合でも、プラットフォームの管理者の場合でも、本書のトピックを使用して、Openshift の使用を開始することができます。管理者は、インストールユーティリティーや対話式の CLI ツールを使用して、複数のホストに新しい OpenShift インスタンスをすばやくインストールし、設定できます。また、開発者は OpenShift CLI または Web コンソールを使用して、既存の OpenShift インスタンスにログインしてアプリケーションの作成を開始できます。

目次

第1章 概要

1.1. はじめに

OpenShift Container Platform は、Red Hat 提供の PaaS (Platform as a Service) で、Docker と Kubernetes を統合し、これらのサービスを管理する API を提供します。OpenShift Container Platform では、コンテナーの作成や管理が可能になります。

1.1.1. OpenShift を使用する理由

コンテナーは、独自の環境内で実行されるスタンドアロンのプロセスで、オペレーティングシステムや下層のインフラストラクチャーからは独立しています。OpenShift を使用すると、コンテナーベースのアプリケーションを開発、デプロイ、管理しやすくなります。また、セルフサービスのプラットフォームが提供されるので、オンデマンドでアプリケーションを作成、修正、デプロイできるため、開発やリリースのライフサイクルを加速化することができます。

イメージはクッキーの抜き型で、またコンテナーは実際のクッキーであると **考えてみてください**。

また、OpenShift はオペレーティングシステムと、イメージはオペレーティングシステム上で実行するアプリケーションと、コンテナーはこれらのアプリケーションを実際に実行するインスタンスと **考えてみてください**。

すでに OpenShift Container Platform をインストールされている場合には、ロールにあった適切なトピックを探して、使用を開始してください。

担当内容	適切なトピックへのリンク
プラットフォームの管理者	「基本的な OpenShift Container Platform の環境」または「実稼働の OpenShift Container Platform 環境のインストール」
開発者	「Web コンソールを使用したイメージの作成およびビルド」や、初めてのプロジェクトよびアプリケーションの作成の基本的な手順

第2章 OPENSHIFT CONTAINER PLATFORM のインストール

2.1. 概要

本書では、OpenShift Container Platform に関する基本的な概念を紹介し、基本的なアプリケーションをインストールできるようになります。本書は、OpenShift Container Platform の実稼働環境のデプロイや、インストールには適していません。

重要

本書は、最初のクラスターをインストールする時に、クイックインストーラーを使用しますが、OpenShift Container Platform 3.9 から、クイックインストールの方法は非推奨になります。今後のリリースでは、完全になくなり、本書の手順は随時更新されます。また、クイックインストーラーを使用したバージョン 3.7 から 3.9 のアップグレードはサポートされていません。新規インストールやクラスターのアップグレード用の「Advanced Installation (通常インストール)」の方法は、今後もサポートされます。

2.1.1. 前提条件

OpenShift Container Platform のインストールには、以下が必要です。

- 最低でも RHEL 7 以降の物理または仮想マシン 2 台。このマシンには、完全修飾ドメイン名 (物理的またはネットワーク上) および、各マシン同士で「パスワードなしで SSH」経由でアクセスできるように設定しておく必要があります。本書では **`master.openshift.example.com`** および **`node.openshift.example.com`** を使用します。これらのマシンでは、ドメイン名を使用して相互に ping を送信できる必要があります。
- 有効な Red Hat サブスクリプション
- ワイルドカードを指定した DNS 解決。お使いのドメインがノードの IP に対して解決できるようにしておきます。つまり、DNS サーバーに以下のようなエントリーを設定します。

```
master.openshift.example.com. 300 IN A <master_ip>
node.openshift.example.com. 300 IN A <node_ip>
*.apps.openshift.example.com. 300 IN A <node_ip>
```

ドメイン名の APPS にワイルドカードのエントリーを指定する理由

OpenShift Container Platform を使用して、アプリケーションをデプロイする場合は、内部ルーターにより、受信要求を適切なアプリケーション pod にプロキシーする必要があります。アプリケーションドメインの一部として **apps** を使用して、アプリケーショントラフィックを適切な pod にマークします。

apps 以外、どれでも使用することができます。

```
*.cloudapps.openshift.example.com. 300 IN A <node_ip>
```

上記を設定したら、以下の手順を使用して 2 台のマシンに OpenShift Container Platform のインストール設定を行います。

2.1.2. OpenShift Container Platform サブスクリプションのアタッチ

1. ターゲットマシンで (マスターとノード両方)、root として **subscription-manager** を使用し、Red Hat にシステムを登録します。

```
$ subscription-manager register
```

2. RHSM から最新のサブスクリプションデータをプルします。

```
$ subscription-manager refresh
```

3. 利用可能なサブスクリプションを一覧表示します。

```
$ subscription-manager list --available
```

4. OpenShift Container Platform サブスクリプションを提供するプール ID を検索してアタッチします。

```
$ subscription-manager attach --pool=<pool_id>
```

5. **<pool_id>** の文字列は、OpenShift Container Platform を提供するプール ID に置き換えます。プール ID は、長い英数字の文字列となっています。

これらの RHEL システムには、OpenShift Container Platform をインストールできるようになりました。次に、これらのシステムに対して、OpenShift Container Platform をどこから入手するかを指示する必要があります。

2.1.3. リポジトリーの設定

マスターとノード上の両方で、**subscription-manager** を使用して、OpenShift Container Platform のインストールに必要なリポジトリーを有効化します。今回の例にある最初のリポジトリー 2 つはすでに有効化されている場合があります。

```
$ subscription-manager repos --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.9-rpms" \
    --enable="rhel-7-fast-datapath-rpms" \
    --enable="rhel-7-server-ansible-2.4-rpms"
```

このコマンドでは、RHEL システムに対して、OpenShift Container Platform をインストールするのに必要なツールが上記のリポジトリーから入手できることを指示しています。次に、Ansible をベースにする OpenShift Container Platform インストーラーが必要です。

2.1.4. OpenShift Container Platform パッケージのインストール

OpenShift Container Platform のインストーラーは、**atomic-openshift-utils** パッケージで提供されます。マスターとノードの両方で、**yum update** を実行してから **yum** を使用してインストールしてください。

```
$ yum -y install wget git net-tools bind-utils iptables-services bridge-
utils bash-completion kexec-tools sos psacct
$ yum -y update
$ yum -y install atomic-openshift-utils
$ yum -y install docker
```

2.1.5. パスワードなしの SSH アクセスの設定

マスターでインストーラーを実行する前に、パスワードなしの SSH アクセスを設定します。インストーラーがこれらのマシンにアクセスできるようにするために、この作業は必要です。マスターで以下のコマンドを実行します。

```
$ ssh-keygen
```

プロンプトの指示に従い、パスフレーズを求められたら Enter を押してください。

SSH 鍵を簡単に配信する方法として、**bash** ループを使用します。

```
$ for host in master.openshift.example.com \
    node.openshift.example.com; \
    do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
    done
```

2.1.6. インストーラーの実行

マスターでインストーラーを実行します。

```
$ atomic-openshift-installer install
```

これは、さまざまな手順を経て、対話式にインストールしていくプロセスです。大半は、デフォルトオプションを使用してください。インストールが開始されたら、OpenShift Container Platform のオプションを選択してください。マスター 1 つ、ノード 1 つをインストールし、ドメイン名は本セクションの最初で記載した **master.openshift.example.com** と **node.openshift.example.com** の FQDN に指定します。

重要

インストーラーにより、ルートの FQDN が求められた場合には、前述したように、**openshift.example.com** ではなく、**apps.openshift.example.com** または **cloudapps.openshift.example.com** を使用する **必要があります**。間違ってしまった場合には、インストールプロセスの最後で **/etc/origin/master/master-config.yaml** を編集します。このファイルで **subdomain** エントリーを探してご自身でこの変更を加えてください。

このインストールプロセスは、約 5-10 分かかります。

2.1.7. OpenShift Container Platform の起動

インストールに成功すると、以下のコマンドを使用して OpenShift Container Platform を起動します。

```
# systemctl restart atomic-openshift-master-api atomic-openshift-master-
controllers
```

インストールして起動が済んだら、基本認証、ユーザーアクセス、ルートを設定してから新規プロジェクトを追加してください。

2.2. OPENSHIFT CONTAINER PLATFORM との対話

OpenShift Container Platform では、対話用に以下の 2 つのコマンドラインユーティリティーがあります。

- **oc**: 通常のプロジェクトおよびアプリケーション管理用
- **oc adm**: 管理者タスク
  oc adm コマンドの実行時は、Ansible ホストのインベントリーファイルに記載されている最初のマスターからのみ実行する必要があります。デフォルトは、**/etc/ansible/hosts** です。

oc --help および **oc adm --help** を使用して、利用可能な全オプションを表示します。

また、Web コンソールを使用してプロジェクトとアプリケーションを管理することもできます。Web コンソールは、**https://<master_fqdn>:8443/console** から利用できます。次のセクションでは、コンソールにアクセスするためのユーザーアカウントを作成する方法を記載しています。

注記

リモートのシステムだけでなく、これらのコマンドラインユーティリティーを使用して、OpenShift Container Platform インスタンスと対話できます。OpenShift CLI としてバンドルされており、Windows、Mac または Linux 環境用のユーティリティーを ここ からダウンロードできます。

2.3. ロールおよび認証の理解

デフォルトでは、初回インストール時には OpenShift Container Platform にロールやユーザーアカウントが作成されないので、作成する必要があります。オプションとして、(最初に) 新規ロールを作成するか、誰でもログインできるようにポリシーを定義する方法があります。

最初に、デフォルトの **system:admin** ユーザーで最低でも 1 回ログインして、マスターで以下のコマンドを実行します。

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

注記

これ以降、別途記載がない限り、コマンドはマスターで実行してください。

このアカウントで最低でも 1 回ログインして、**system:admin** ユーザーの設定ファイルを作成します。このファイルがあれば、以降ログインができるようになります。

このシステムアカウントにはパスワードはありません。

以下のコマンドを実行して、OpenShift Container Platform が正常にインストール、起動したことを確認します。マスターとノードが **Ready** ステータスで表示されます。

```
$ oc get nodes
```

基本的な OpenShift Container Platform 環境の設定を続行するには、「OpenShift Container Platform の設定」に記載の手順に従います。

第3章 OPENSHIFT CONTAINER PLATFORM の設定

3.1. 概要

本書では、OpenShift Container Platform に関する基本的な概念を紹介し、基本的なアプリケーションを設定できるようになります。本書は、「OpenShift Container Platform の基本環境のインストール」に沿った設定手順であるため、OpenShift の実稼働環境のデプロイやインストールには適していません。

3.2. ログイン ID プロバイダーの変更

OpenShift Container Platform インスタンスを新規インストールした場合のデフォルトの動作は、全ユーザーのログインを拒否します。この認証方法を HTPasswd に変更するには、以下を実行します。

1. **/etc/origin/master/master-config.yaml** ファイルを編集モードで開きます。
2. **identityProviders** のセクションを検索します。
3. **DenyAllPasswordIdentityProvider** を **HTPasswdPasswordIdentityProvider** プロバイダーに変更します。
4. 名前ラベルの値を **htpasswd_auth** に変更し、プロバイダーセクションに **file: /etc/origin/openshift-passwd** の新しい行を追加します。**identityProviders** セクションに **HTPasswdPasswordIdentityProvider** を指定した例は、以下のようになります。

```
oauthConfig:
  ...
  identityProviders:
  - challenge: true
    login: true
    name: htpasswd_auth provider
    provider:
      apiVersion: v1
      kind: HTPasswdPasswordIdentityProvider
      file: /etc/origin/openshift-passwd
```

5. ファイルを保存します。

3.3. ユーザーアカウントの作成

HTPasswdPasswordIdentityProvider プロバイダーを使用できるようになったので、これらのユーザーアカウントを作成する必要があります。

1. **httpd-tools** パッケージを使用して、これらのアカウントを生成可能な **htpasswd** バイナリーを取得できます。

```
# yum -y install httpd-tools
```

2. ユーザーアカウントを作成します。

```
# touch /etc/origin/openshift-passwd
# htpasswd -b /etc/origin/openshift-passwd admin redhat
```

パスワードが **redhat** の **admin** というユーザーを作成しました。

3. 続行する前に OpenShift を再起動します。

```
# systemctl restart atomic-openshift-master-api atomic-openshift-
master-controllers
```

4. このユーザーアカウントに **cluster-admin** の特権を指定して、あらゆる操作ができるようにします。

```
$ oc adm policy add-cluster-role-to-user cluster-admin admin
```

oc adm コマンドの実行時は、Ansible ホストのインベントリーファイルに記載されている最初のマスターからのみ実行する必要があります。デフォルトは、**/etc/ansible/hosts** です。

5. Web コンソールやコマンドラインから、このユーザー名/パスワードの組み合わせを使用してログインします。これをテストするには、以下のコマンドを実行します。

```
$ oc login -u admin
```

default プロジェクトに変更してから、次に進みます。

```
$ oc project default
```

詳細は、「roles」および「authentication」を参照してください。

3.4. OPENSHIFT ルーターのデプロイ

OpenShift ルーターは、OpenShift サービスが宛先の外部ネットワークトラフィックのエントリーポイントとなります。SNI を使用する HTTP、HTTPS および TLS トラフィックをサポートし、ルーターが正しいサービスにトラフィックを送信できるようにします。

ルーターなしでは、OpenShift サービスと pod は、OpenShift インスタンスの外部にあるリソースと通信できません。

インストーラーにより、デフォルトのルーターが作成されます。

1. 以下のコマンドを使用してデフォルトのルーターを削除します。

```
$ oc delete all -l router=router
```

2. 新しいデフォルトのルーターを作成します。

```
$ oc adm router --replicas=1 --service-account=router
```

OpenShift ドキュメントには、「Router Overview」に関する詳細情報が含まれます。

3.5. 内部レジストリーのデプロイ

Openshift には、内部の「integrated Docker registry」がある、このレジストリーをデプロイして、イメージをローカルで管理できます。また、OpenShift は **docker-registry** を使用して、Docker イメージの保存、取得、ビルドだけでなく、ライフサイクル全体でイメージをデプロイ、管理します。

インストーラーは、デフォルトのレジストリーを作成します。

1. 以下のコマンドを使用して、デフォルトのレジストリーを削除します。

```
$ oc delete all -l docker-registry=default
```

2. **registry** サービスアカウントを使用して、**default** プロジェクトに **docker-registry** サービスを作成します。

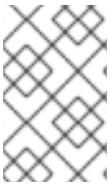
```
$ oc adm registry
```

3.6. レジストリー用の永続ストレージの作成

以前の手順で作成したレジストリーは、イメージとメタデータを保存し、永続ストレージが設定されていない場合には pod のデプロイメント用に一時ボリュームを使用します。この一時ボリュームは、pod が終了した時点で破棄され、レジストリーにビルドまたはプッシュされたイメージなど、すべてのデータが失われます。

レジストリー用に永続ストレージを設定するには、以下を実行します。

- お使いのネットワーク上にあるストレージサーバーを参照するボリュームをプロビジョニングします (ここではマスターにボリュームを作成します)。
- Volume Claim (ボリューム要求) を作成します。
- レジストリーサービスに要求を手動で追加します。



注記

レジストリー用の永続ストレージのを設定する以下の手順は、レジストリーだけでなく、永続データ必要とするイメージ用のストレージにも該当します。レジストリーは、OpenShift 環境ではイメージの 1 つにすぎません。

3.6.1. 永続ボリュームのプロビジョニング

1. ここで記載されているように、レジストリーのボリュームファイルを作成して、**registry-volume.yaml** という名前を指定します。

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: registry-volume
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  nfs:
    path: /root/storage
    server: master.openshift.example.com
```

/root/storage というフォルダーは、必要です。サーバーエントリーがマスターを参照するように変更してください。

2. OpenShift でレジストリーの永続ボリュームを作成します。

```
$ oc create -f registry-volume.yaml
```

3.6.2. Persistent Volume Claim (永続ボリューム要求) の作成

さきほど作成した永続ボリュームをバインドするための要求を作成します。この要求を使用して、レジストリーサービスと永続ボリュームをリンクします。

1. **registry-volume-claim.yaml** と呼ばれる別のファイルを作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: registry-volume-claim
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 3Gi
```

2. 要求を作成します。

```
$ oc create -f registry-volume-claim.yaml
```

永続ボリュームと Persistent Volume Claim (永続ボリューム要求) が作成されましたので、この要求をレジストリーに追加する必要があります。

3.6.3. Persistent Volume Claim (永続ボリューム要求) のレジストリーへの追加

```
$ oc volume dc/docker-registry \
    --add --overwrite -t persistentVolumeClaim \
    --claim-name=registry-volume-claim \
    --name=registry-storage
```

これで、**docker-registry** は、イメージとメタデータの保存用に作成した 3 GB の永続ボリュームを使用するようになります。

第4章 WEB コンソールを使用したイメージの作成およびビルド

4.1. 概要

このスタートガイドでは、最もシンプルな方法で、OpenShift Container Platform でサンプルプロジェクトを稼働させる手順を説明しています。プロジェクトでイメージを起動する方法は複数ありますが、このトピックでは、最もすばやく簡単に起動する方法に焦点を当てていきます。

本書から読み始めており、OpenShift Container Platform バージョン 3 (v3) のコアとなる概念に慣れていない場合には、「What's New」を先に読んでみてください。本バージョンの OpenShift Container Platform は、バージョン 2 (v2) とは大きく異なります。

OpenShift Container Platform 3 では、開発者向けに、すぐアプリケーション開発を開始していただけるように、適切な実装およびチュートリアルと、追加設定なしに使用できる一連の「言語」および「データベース」を提供しています。言語サポートは、「クイックスタートテンプレート」を軸として展開されており、このテンプレートは、「ビルダーイメージ」順にを活用します。

言語	実装およびチュートリアル
Ruby	Rails
Python	Django
Node.js	Node.js
PHP	CakePHP
Perl	Dancer
Java	

OpenShift Container Platform が提供する他のイメージには以下が含まれます。

- MySQL
- MongoDB
- PostgreSQL
- Jenkins

さらに、JBoss ミドルウェアでは、xPaaS サービスの一部として、幅広い種類の「OpenShift Container Platform テンプレート」だけでなく、「イメージ」がまとめられています。

特に xPaaS サービスで利用可能な技術は以下のとおりです。

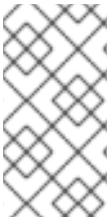
- JBoss EAP 6 が提供する Java EE 6 Application Server
- JBoss Fuse および JBoss A-MQ が提供する統合およびメッセージサービス
- JBoss Data Grid が提供する Data Grid サービス

- JBoss BRMS が提供する Real Time Decision Service
- Tomcat 7 および Tomcat 8 が提供する Java Web Server 3.0

上記のオファリングではそれぞれ、一連の組み合わせが提供されています。

- HTTP のみ vs HTTP および HTTPS
- データベースを必要としない場合や、MongoDB、PostgreSQL または MySQL のいずれかを使用する場合
- 希望に応じた、A-MQ との統合

このようなアプリケーションの構築を例示するために、以下のセクションにはプロジェクトが含まれています。このプロジェクトには、ウェルカムページや現在の訪問者数 (データベースに保存) を提供するサンプルの Node.js アプリケーションが含まれています。



注記

このトピックでは、「クイックスタート」と「インスタントアプリ」 のテンプレートとアプリケーションの両方について説明します。クイックスタートは、アプリケーション開発のスタート地点ではありますが、便利なアプリケーションを作成するには開発作業が必要です。反対に、Jenkins などのインスタントアプリは即座に利用できます。

4.1.1. ブラウザーの要件

Web コンソールへのアクセスに使用可能な、「ブラウザーのバージョンとオペレーティングシステム」を確認してください。

4.2. 作業を開始する前に

作業を開始する前に、以下を確認してください。

- 実行中の OpenShift Container Platform インスタントにアクセスできる必要があります。アクセスできない場合には、クラスター管理者にお問い合わせください。
- インスタンスは、クラスター管理者によって、「インスタントアプリのテンプレート」および「ビルダーイメージ」で事前設定されている必要があります。インスタンスが用意されていない場合には、クラスター管理者に「デフォルトのイメージストリームとテンプレートの読み込み」トピックを参照するように案内してください。
- OpenShift Container Platform CLI を 「ダウンロード、インストール」しておく必要があります。

4.3. サンプルリポジトリーのフォーク

1. GitHub にログインした状態で 「Ruby の例」 のページに移動します。

注記

以下のトピックは、Ruby の例に沿っていますが、「OpenShift Container Platform GitHub プロジェクトで提供される」言語であればどれでも使用できます。

2. 「リポジトリーをフォーク」します。
   新規のフォークにリダイレクトされます。

3. フォーク用のクローン URL をコピーします。

4. ローカルのマシンにリポジトリーをクローンします。

4.4. プロジェクトの作成

アプリケーションを作成するには、最初に新規プロジェクトを作成してから、InstantApp テンプレートを選択します。そこから、OpenShift Container Platform はビルドプロセスを開始し、新規デプロイメントを作成します。

1. ブラウザーから OpenShift Container Platform の Web コンソールに移動します。Web コンソールは、自己署名証明書を使用するので、プロンプトが表示されたら、ブラウザーの警告を OK で進みます。

2. 管理者が推奨するユーザー名とパスワードを使用してログインします。

3. 新規プロジェクトを作成するには、**New Project** をクリックします。

4. 新規プロジェクトの一意の名前、表示名および説明を入力します。

5. **Create** をクリックします。
   web コンソールの welcome 画面が読み込まれます。

4.5. アプリケーションの作成

Select Image または Template ページでは、公開されている git リポジトリーまたは、テンプレートのどちらをもとに作成するかのオプションが表示されます。

1. 新規プロジェクトを作成しても、自動的に、Select Image または Template ページに自動的にリダイレクトされない場合には、**Add to Project** をクリックする必要がある場合があります。

2. **Browse** をクリックしてから、ドロップダウンリストから **ruby** を選択します。

3. **ruby:latest** ビルダーイメージをクリックします。

4. アプリケーションの **名前** を入力して、**Git リポジトリーの URL** を入力します。Git リポジトリーの URL は `https://github.com/<your_github_username>/ruby-ex.git` です。

5. オプションで、**Show advanced routing, build, and deployment options** をクリックします。ただし、デフォルトでは、この例のアプリケーションでは自動的にルート、Webhook トリガー、ビルド変更トリガーが作成されます。

6. **Create** をクリックします。

注記

作成後に、Web コンソールからこれらの設定の一部を変更できます。手順は、**Browse**、**Builds** をクリックしてビルドを選択してから、**Actions** をクリックして **Edit** または **Edit YAML** を選択します。

アプリケーションの作成には時間がかかる可能性があります。Web コンソールの Overview ページでは、作成した新規リソースを表示し、ビルドやデプロイメントの進捗を確認できます。

Ruby pod が作成されても、pod のステータスは保留中と表示されます。次に、Ruby pod が起動し、新たに割り当てられた IP アドレスが表示されます。Ruby pod が実行されると、ビルドが完了します。

4.6. アプリケーションの実行の確認

DNS が正しく設定されている場合には、新規アプリケーションは Web ブラウザーからアクセスできます。アプリケーションにアクセスできない場合には、システム管理者にお問い合わせください。

新規アプリケーションを表示するには以下を実行します。

1. Web コンソールでは、Overview ページを表示して、アプリケーションの Web アドレスを判断します。たとえば、**SERVICE: RUBY-EX** で、**`ruby-ex-my-test.example.openshiftapps.com`** とよく似た内容が表示されます。

2. 新規アプリケーションの Web アドレスに移動します。

4.7. 自動化ビルドの設定

OpenShift Container Platform GitHub リポジトリー からこのアプリケーションのソースコードをフォークしたので、フォークしたリポジトリーにコードの変更がプッシュされるたびに、Webhook を使用して自動的にアプリケーションのリビルドをトリガーできます。

アプリケーションの Webhook を設定するには以下を実行します。

1. Web コンソールで、作成したアプリケーションが含まれるプロジェクトに移動します。

2. **Browse** タブをクリックしてから **Builds** をクリックします。

3. ビルド名をクリックしてから、**Configuration** タブをクリックします。

4. **GitHub webhook URL** の横にある  をクリックして、webhook payload ペイロード URL をコピーします。

5. GitHub のフォークされたリポジトリーに移動してから **Settings** をクリックします。

6. **Webhooks & Services** をクリックします。

7. **Add webhook** をクリックします。

8. webhook URL を **Payload URL** フィールドにコピーします。

9. **Add webhook** をクリックして保存します。

GitHub は、ping のペイロードを OpenShift Container Platform サーバーに送信して、通信が成功したことを確認します。Webhook URL の横に緑のチェックマークが表示された場合には、正しく設定されています。チェックマークの上にマウスをかざして、最終配信のステータスを表示します。

フォークされたリポジトリーにコード変更をプッシュする次回のタイミングで、アプリケーションが自動的に再ビルドされます。

4.8. コード変更の記述

ローカルで作業して、アプリケーションに変更をプッシュします。

1. ローカルマシンで、テキストエディターを使用して、**ruby-ex/config.ru** ファイルのサンプルアプリケーションのソースを変更します。

2. コードの変更をアプリケーション内から表示できるようにします。 たとえば、行 229 で、タイトルを **`Welcome to your Ruby application on OpenShift`** から **`This is my Awesome OpenShift Application`** に変更してから、変更を保存します。

3. Git に変更をコミットして、フォークに変更をプッシュします。
webhook が正しく設定されている場合には、変更をもとに、アプリケーションは即座にリビルドされます。リビルドに成功した場合には、以前に作成したルートを使用すると、更新したアプリケーションが表示されます。

次に、必要な作業はコードの更新をプッシュするだけで、OpenShift Container Platform が残りを行います。

4.8.1. イメージの手動リビルド

Webook が機能しない場合や、ビルドに失敗して、コードを変更せずにビルドを再起動する場合には、イメージを手動でリビルドすると便利です。直近にコミットされた変更をもとに、イメージを手動でリビルドするには、フォークしたリポジトリーに移動します。

1. **Browse** タブをクリックしてから **Builds** をクリックします。

2. ビルドを探し出して、**Start Build** をクリックします。

第5章 CLI を使用したイメージの作成およびビルド

5.1. 概要

このスタートガイドでは、最もシンプルな方法で、OpenShift Container Platform でサンプルプロジェクトを稼働させる手順を説明しています。プロジェクトでイメージを起動する方法は複数ありますが、このトピックでは、最もすばやく簡単に起動する方法に焦点を当てていきます。

本書から読み始めており、OpenShift Container Platform バージョン 3 (v3) のコアとなる概念に慣れていない場合には、「What's New」を先に読んでみてください。本バージョンの OpenShift Container Platform は、バージョン 2 (v2) とは大きく異なります。

OpenShift Container Platform 3 では、開発者向けに、すぐアプリケーション開発を開始していただけるように、適切な実装およびチュートリアルと、追加設定なしに使用できる一連の「言語」および「データベース」を提供しています。言語サポートは、「クイックスタートテンプレート」を軸として展開されており、このテンプレートは、「ビルダーイメージ」順にを活用します。

言語	実装およびチュートリアル
Ruby	Rails
Python	Django
Node.js	Node.js
PHP	CakePHP
Perl	Dancer
Java	

OpenShift Container Platform が提供する他のイメージには以下が含まれます。

- MySQL
- MongoDB
- PostgreSQL
- Jenkins

さらに、JBoss ミドルウェアでは、xPaaS サービスの一部として、幅広い種類の「OpenShift Container Platform テンプレート」だけでなく、「イメージ」がまとめられています。

特に xPaaS サービスで利用可能な技術は以下のとおりです。

- JBoss EAP 6 が提供する Java EE 6 Application Server
- JBoss Fuse および JBoss A-MQ が提供する統合およびメッセージサービス
- JBoss Data Grid が提供する Data Grid サービス

- JBoss BRMS が提供する Real Time Decision Service
- Tomcat 7 および Tomcat 8 が提供する Java Web Server 3.0

上記のオファリングではそれぞれ、一連の組み合わせが提供されています。

- HTTP のみ vs HTTP および HTTPS
- データベースを必要としない場合や、MongoDB、PostgreSQL または MySQL のいずれかを使用する場合
- 希望に応じた、A-MQ との統合

このようなアプリケーションの構築を例示するために、以下のセクションにはプロジェクトが含まれています。このプロジェクトには、ウェルカムページや現在の訪問者数 (データベースに保存) を提供するサンプルの Node.js アプリケーションが含まれています。

注記

このトピックでは、「クイックスタート」と「インスタントアプリ」 のテンプレートとアプリケーションの両方について説明します。クイックスタートは、アプリケーション開発のスタート地点ではありますが、便利なアプリケーションを作成するには開発作業が必要です。反対に、Jenkins などのインスタントアプリは即座に利用できます。

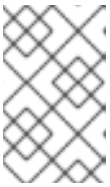
5.2. 作業を開始する前に

作業を開始する前に、以下を確認してください。

- 実行中の OpenShift Container Platform インスタントにアクセスできる必要があります。アクセスできない場合には、クラスター管理者にお問い合わせください。
- インスタンスは、クラスター管理者によって、「インスタントアプリのテンプレート」および「ビルダーイメージ」で事前設定されている必要があります。インスタンスが用意されていない場合には、クラスター管理者に「デフォルトのイメージストリームとテンプレートの読み込み」トピックを参照するように案内してください。
- OpenShift Container Platform CLI を 「ダウンロード、インストール」しておく必要があります。

5.3. サンプルリポジトリーのフォーク

1. GitHub にログインした状態で 「Ruby の例」 のページに移動します。



注記

以下のトピックは、Ruby の例に沿っていますが、「OpenShift Container Platform GitHub プロジェクトで提供される」言語であればどれでも使用できます。

2. 「リポジトリーをフォーク」します。
   新規のフォークにリダイレクトされます。
3. フォーク用のクローン URL をコピーします。
4. ローカルのマシンにリポジトリーをクローンします。

5.4. プロジェクトの作成

アプリケーションを作成するには、新規プロジェクトを作成して、ソースの場所を指定する必要があります。ここから、OpenShift Container Platform はビルドプロセスを開始して、新規デプロイメントを作成します。

1. CLI で OpenShift Container Platform にログインします。
   - ユーザー名とパスワードを使用してログインする場合:

     ```
     $ oc login -u=<username> -p=<password> --server=<your-openshift-server> --insecure-skip-tls-verify
     ```

   - oauth トークンを使用してログインする場合:

     ```
     $ oc login <https://api.your-openshift-server.com> --token=<tokenID>
     ```

2. 新規プロジェクトを作成するには、以下を実行します。

   ```
   $ oc new-project <projectname> --description="<description>" --display-name="<display_name>"
   ```

新規プロジェクトの作成後、新規プロジェクトの namespace に自動的に切り替えられます。

5.5. アプリケーションの作成

フォークしたリポジトリーにあるコードをもとに新規アプリケーションを作成します。

1. コードのソースを指定してアプリケーションを作成します。

   ```
   $ oc new-app openshift/ruby-20-centos7~https://github.com/<your_github_username>/ruby-ex
   ```

   OpenShift Container Platform は一致するビルダーイメージを検索して (今回は **ruby-20-centos7**)、アプリケーションのリソース (イメージストリーム、ビルド設定、デプロイメント設定、サービス) を作成します。また、ビルドのスケジューリングも行います。

2. ビルドの進捗を追跡します。

   ```
   $ oc logs -f bc/ruby-ex
   ```

3. ビルドが完了し、作成されたイメージがレジストリーに正常にプッシュされたら、アプリケーションのステータスを確認します。

   ```
   $ oc status
   ```

   または、Web コンソールからビルドを表示できます。

アプリケーションの作成には時間がかかる可能性があります。Web コンソールの Overview ページでは、作成した新規リソースを表示し、ビルドやデプロイメントの進捗を確認できます。**`oc get pods`** コマンドでは、pod がいつ稼働しているかを確認し、**`oc get builds`** では、ビルドの統計を表示します。

Ruby pod が作成されても、pod のステータスは保留中と表示されます。次に、Ruby pod が起動し、新たに割り当てられた IP アドレスが表示されます。Ruby pod が実行されると、ビルドが完了します。

oc status コマンドでは、サービスが実行されている IP アドレスを表示します。デプロイ先のデフォルトのポートは、8080 です。

5.6. ルートの作成

OpenShift Container Platform ルートがホスト名でサービスを公開するので、対象の外部クライアントが名前を使用して到達できます。新規アプリケーションへのルートを作成するには、以下を実行します。

1. **ruby-ex** のサービスを公開します。

```
$ oc expose service ruby-ex
```

2. 新規ルートを表示します。

```
$ oc get route
```

3. ルートの場所をコピーします。たとえば **ruby-ex-my-test.example.openshiftapps.com** などです。

5.7. アプリケーションの実行の確認

新規アプリケーションを表示するには、(前のセクションで) コピーしたルートの場所を Web ブラウザーのアドレスバーに貼り付けて、Enter を押します。

サンプルの **ruby-ex** アプリケーションは、単純なウェルカム画面となっており、コード変更のデプロイ、アプリケーションや他の開発リソースの管理方法に関する詳細が含まれます。

次に、GitHub Webhook トリガーを使用した自動化ビルドを設定し、フォークしたリポジトリーのコードが変更されると、アプリケーションがリビルドされます。

5.8. 自動化ビルドの設定

OpenShift Container Platform GitHub リポジトリー からこのアプリケーションのソースコードをフォークしたので、フォークしたリポジトリーにコードの変更がプッシュされるたびに、Webhook を使用して自動的にアプリケーションのリビルドをトリガーできます。

アプリケーションの Webhook を設定するには以下を実行します。

1. **BuildConfig** のトリガーセクションを表示して、GitHub webhook トリガーが存在することを確認します。

```
$ oc edit bc/ruby-ex
```

以下のような内容が表示されているはずです。

```
triggers
- github:
    secret: Q1tGY0i9f1ZFihQbX07S
    type: GitHub
```

シークレットは、ユーザー自身およびリポジトリーしか、ビルドをトリガーできないようにします。

2. 以下のコマンドを実行して、**BuildConfig** に連携されている Webhook URL を表示します。

```
$ oc describe bc ruby-ex
```

3. 上記のコマンドで表示された GitHub webhook ペイロード URL の出力をコピーします。
4. GitHub のフォークされたリポジトリーに移動してから **Settings** をクリックします。
5. **Webhooks & Services** をクリックします。
6. **Add webhook** をクリックします。
7. webhook URL を **Payload URL** フィールドにコピーします。
8. **Add webhook** をクリックして保存します。

GitHub は、ping のペイロードを OpenShift Container Platform サーバーに送信して、通信が成功したことを確認します。Webhook URL の横に緑のチェックマークが表示された場合には、正しく設定されています。チェックマークの上にマウスをかざして、最終配信のステータスを表示します。

フォークされたリポジトリーにコード変更をプッシュする次回のタイミングで、アプリケーションが自動的に再ビルドされます。

5.9. コード変更の記述

ローカルで作業して、アプリケーションに変更をプッシュします。

1. ローカルマシンで、テキストエディターを使用して、**ruby-ex/config.ru** ファイルのサンプルアプリケーションのソースを変更します。
2. コードの変更をアプリケーション内から表示できるようにします。 たとえば、行 229 で、タイトルを **`Welcome to your Ruby application on OpenShift`** から **`This is my Awesome OpenShift Application`** に変更してから、変更を保存します。
3. Git に変更をコミットして、フォークに変更をプッシュします。
webhook が正しく設定されている場合には、変更をもとに、アプリケーションは即座にリビルドされます。リビルドに成功した場合には、以前に作成したルートを使用すると、更新したアプリケーションが表示されます。

次に、必要な作業はコードの更新をプッシュするだけで、OpenShift Container Platform が残りを行います。

5.9.1. イメージの手動リビルド

Webook が機能しない場合や、ビルドに失敗して、コードを変更せずにビルドを再起動する場合には、イメージを手動でリビルドすると便利です。直近にコミットされた変更をもとに、イメージを手動でリビルドするには、フォークしたリポジトリーに移動します。

```
$ oc start-build ruby-ex
```

5.10. トラブルシューティング

プロジェクトの変更

oc new-project コマンドは自動的に現在プロジェクトをさきほど作成したプロジェクトに設定しますが、以下を実行することでいつでもプロジェクトを変更できます。

```
$ oc project <project-name>
```

プロジェクトの一覧を表示します。

```
$ oc get projects
```

ビルドの手動トリガー

ビルドが自動的に開始されない場合には、ビルドを開始して、ログをストリームします。

```
$ oc start-build ruby-ex --follow
```

または、上記のコマンドに **--follow** を追加せずに、ビルドのトリガー後に以下のコマンドを実行します。**n** は追跡するビルドの数に置き換えます。

```
$ oc logs -f build/ruby-ex-n
```