
Red Hat
CodeReady
Containers
(OpenShift4)
dabble in...
Extra edition

Red Hat CodeReady Containers (OpenShift 4)

dobble in...

Extra edition

目次

■はじめに

この本は、様々な(主にオープンソースの)ソフトウエアを導入したり実際に使ってみる本です。
「dabble in…」ってのはいわゆる「いっちょかみ」的な意味の言葉らしいです。
閑話休題。

若干仕事がらみではありますが、OpenShiftに絡んだ業務を担当するチームがいるということで、勝手に環境を作ってみたので、勢いで本書を作成しました。
正直勢いだけなので、何かしら抜け漏れがありましたらガンガンご指摘下さい。

■OpenShiftとは

まず、「OpenShift」とはなんなのか。
みんな大好き「Wikipedia」から。

> OpenShift（オープンシフト）はレッドハットが開発するコンテナ化ソフトウェア・ファミリー。
> penShiftは、コンテナ機能であるDockerと、Dockerコンテナのオーケストレーション機能を提供するKubernetesをベースに、更にコンテナを使用した開発や運用に便利な機能を提供する。
> 1つはアクセスコントロール、もう1つはコンテナのビルドとデプロイ機能である。
> アクセスコントロールは、リソース分離と権限分掌によって実現する。
> ビルドとデプロイの機能は、Kubernetesの機能に追加して、コンテナ型のアプリケーションとして実行するためのDocker Imageの作成やデプロイに関わる機能を追加する

…これだけだとワケがわかりませんが、分解して理解できるようにしてみましょう。

・Docker :
「OSレベルの仮想化の実行環境・システム」を用いてアプリケーションを開発、配置、実行
・Kubernetes :
Dockerコンテナを自動で利用可能に　したり、リソース配分をしたり、管理する
・OpenShift :
リソース分離、権限分掌によるアクセスコントロール、Kubernetesの機能に追加して、Dockerコンテナとして動作するイメージの作成、利用可能にするための便利な機能を提供する

いわゆる「OpenPaaS」と呼ばれる部類になるようです。
PaaS : Platform as a Service
アプリケーションソフトが稼動するためのリソースやOSなどのプラットフォーム一式を、ネットワーク上のサービスとして提供する形態のこと
というわけで、Redhat OpenShiftを触る環境を作ってみましょう。

■構築環境

構築の際の各種環境について簡単に記載します。

■使用ディストリビューションについて

今回は「OpenShift」自体がRedhat社が提供するプラットフォームであることもあり、
RedHat Enterprise Linuxを使います。

■仮想マシンについて

・4CPU
・32GBメモリ
・250GB HDD
を割り当てた仮想マシンで行います。

■ハードウェア環境について

今回は普段使っている環境と異なり、VMWare vSphere6.5の環境で構築しております。
マシンも24コアOpteron/MEM144GB環境ですがまぁ上記くらいのリソースを使うのは何とでもなる感じです。

■使用するソフトウェアについて

前述の通り、「OpenShift」はRedhat社が提供する商用のプラットフォームです。このため、実際に業務に使う際には、有償のサブスクリプションが必要になります。
(30日間の体験版ライセンスはありますが、30日では何が何やら、でしょう)
開発やテストのためにOpenShiftを使う手段がいくつか用意されていますので、今回はそちらを使います。

■OpenShiftの学習環境

OpenShiftの学習環境としては以下があります。
1．Interactive Learning Portal
2．OpenShift Online
3．Red Hat Container Development Kit（CDK）
4．Red Hat CodeReady Containers (CRC)
1/2はRedhatが提供しているオンライン環境です。1は完全な学習用環境、2は有償無償両方のあるオンラインツールです。
3/4は、自身のマシンやクラウド環境で動作を確認するための、テスト用のミニ構成の仮想環境とテストシナリオが同梱されたものです。今回はこれを使って構築しようと思います。なお、
・3．CDKはOpenShift3の環境　(現在3.11、2024年までサポートされているようです)
・4．CRCはOpenShfit4の環境
　となっています。

■Red Hat CodeReady Containers

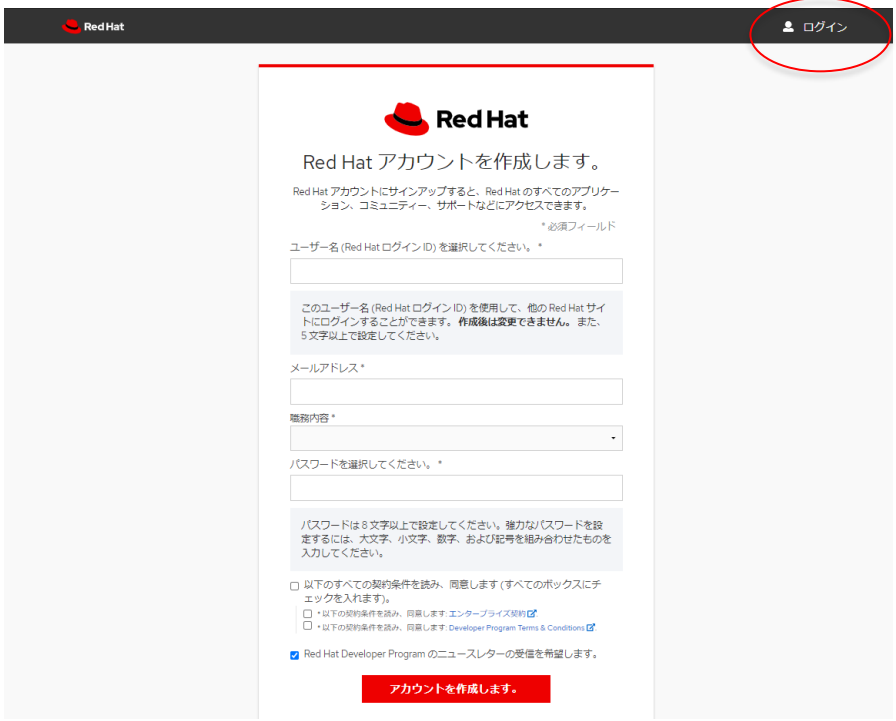
Red Hat CodeReady Containers（CRC）は「Red Hat Developer Program」に登録することで利用できます。またコンテナイメージのダウンロード、ビルドが可能になります。

なお、アカウント登録等の詳細については細かい説明は今回は省いています。ご了承下さい。

1. Red Hat Developer Programに登録

https://developers.redhat.com/

にアクセスし、画面右上の「Login」から登録画面に遷移し、アカウントを作成します。



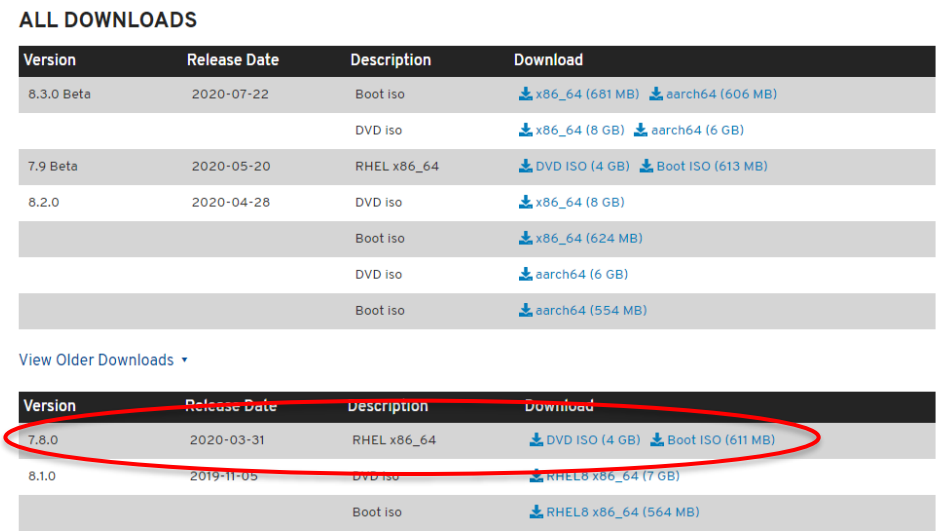
2. Red Hat Enterprise Linuxの準備

「Red Hat Developer Program」からダウンロードできる「RHEL 7.8」を利用します。

先ほど登録したアカウントで

https://developers.redhat.com/products/rhel/download

にアクセスし、「View Older Downloads」をクリックして、Version7の最新のものをダウンロードします



3. Red hat Enterprise Linuxのインストール

仮想マシンのスペックは前述の通りですが、さらにCPUのオプションとして仮想機能をONにします。

インストール自体は一般的なRedhat系のLinuxをインストールしたことがあればそれほど悩む点はないと思います。

インストール時の注意は以下です。

・ユーザーを管理者として作成する　　(su/sudoコマンドが利用可能なユーザーとして作成する)

・GUIありのサーバーとしてインストールする

インストール後にサブスクリプション設定でRHDPへの接続を設定することで、試用を継続できます。

コマンドラインで

```
# subscription-manager register
※Redhat Developers-programのIDとパスワードを
  聞かれるので入力
# subscription-manager attach


Installed Product Current Status:
Product Name: Red Hat Enterprise Linux Server
Status:       Subscribed
```

これで登録完了です。

なお、GUIの登録メニューもあったりするのですが、インストール後に上記のようにコマンドで登録するほうがうまくいきやすかったです。

4. Red Hat CodeReady Containersのインストール

では、CRCのインストールについて大雑把に説明します。

実は、導入自体は非常に楽です。

なお、前述のとおりVersion3に対応した「CDK」と、Version4に対応した「CRC」の2つがありますが、これから試験環境的に導入を試みる場合は、OpenSfhit3でなくてはならない理由がなければ、CRCを利用するほうが楽だと思います。

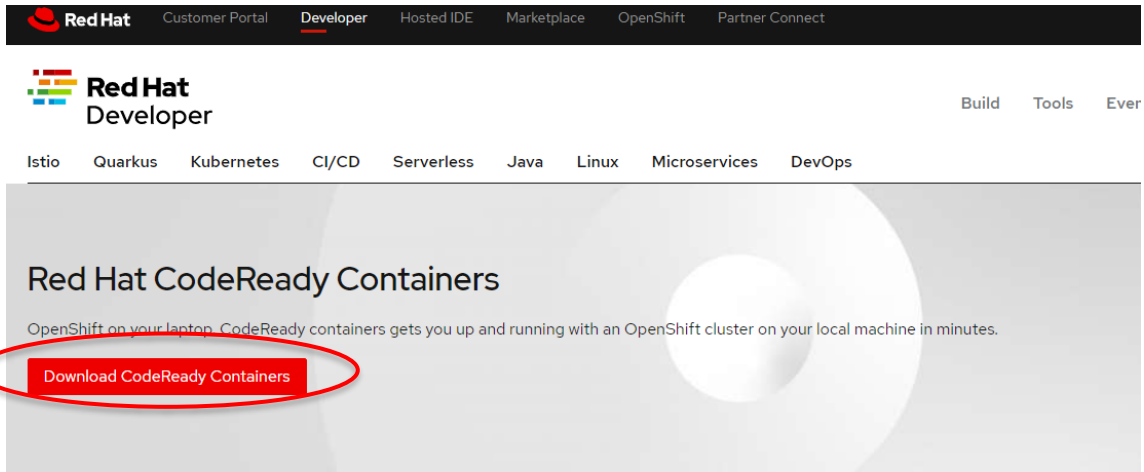
5. CRCバイナリをダウンロード

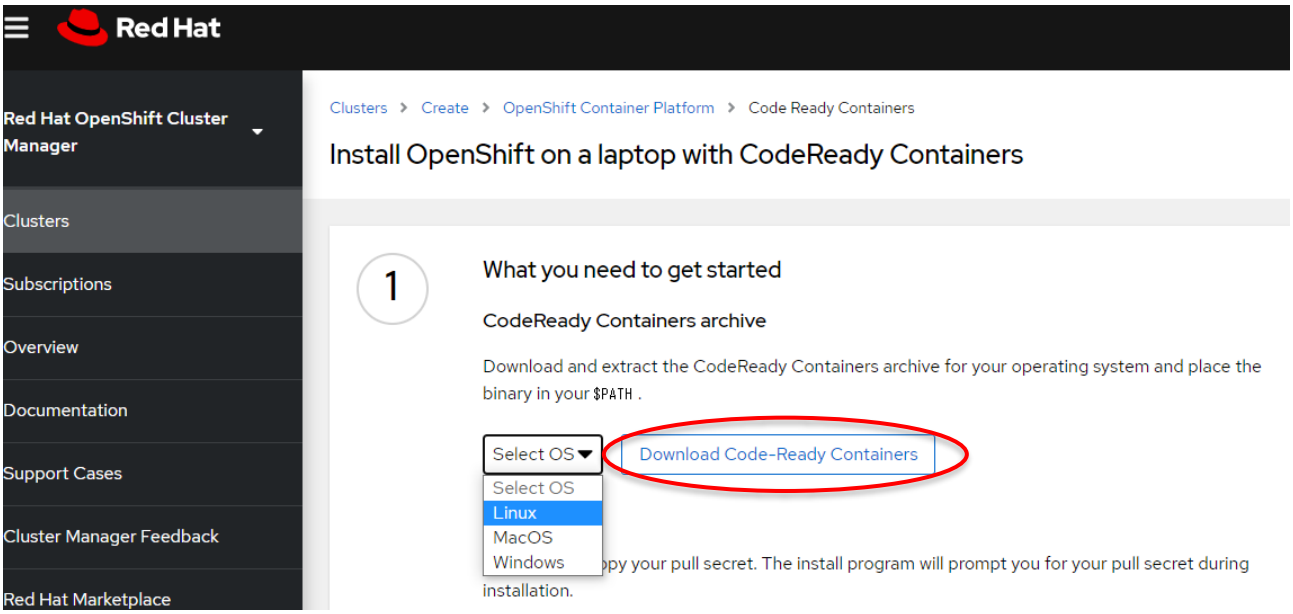
以下はGUIありの設定でインストールしたRedHatLinuxからブラウザ(FireFox)を立ち上げて実施します。

以下からダウンロードします。

https://developers.redhat.com/products/codeready-containers/overview

「Download CodeReady Containers」のリンクをクリックします。

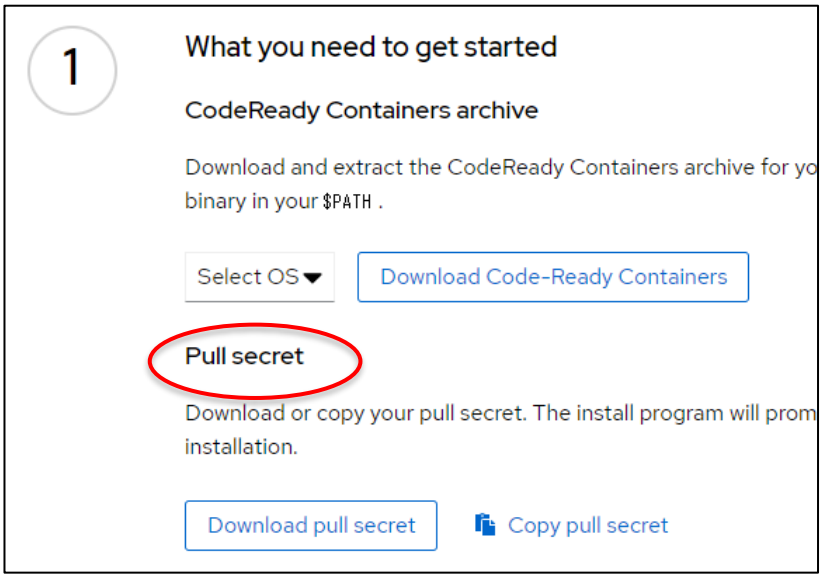




遷移したページの

①「What you need to get started」

にてOSを選択し「Download Code-Ready Containers」をクリックしてダウンロードします。



次にその下の「Download Pull Secret」をクリックして、「Pull Secret」をダウンロードしておきます。

6．CRCのインストール

リモート接続で実施すると失敗するケースもあるようなのでLinux上でターミナルを立ち上げて実施します。

また、CRCコマンドは必ず一般ユーザー(SU権限を持った)で実行する必要があります。(プロンプトが$になっているのはそのためです)

ダウンロードしたCRCを展開し、パスが通った場所にコピーします。

下記では/usr/local/binにコピーしていますが、パスが通って実行可能であればどこでも問題ないはずです。

```
$ tar xvf crc-linux-amd64.tar.xz
$ cp -p crc-linux-1.14.0-amd64/crc /usr/local/bin/
$ crc version  ★バージョン確認
CodeReady Containers version: 1.14.0+36ad776
OpenShift version: 4.5.4 (embedded in binary)
```

セットアップを実施します。

```
$ crc setup
```

しばらく標準出力に諸々表示されますが最終的に

```
Setup is complete, you can now run 'crc start' to start the OpenShift cluster
```

と表示されればセットアップ完了です。

7．CRCの起動

```
$ crc start
```

を実行すると起動します。初回は -pオプションでpull_secretの

ファイルを指定するか、上記を実行した際に

～出力中略～

? Image pull secret [? for help]

で一旦出力が停止するので、このタイミングでダウンロードした

pull_secretの中身をコピペすると出力が進みます。

8．CRCの操作

インストール実施時のプロンプトに以下のような行が表示されています。これを元に設定していきます。

```
INFO To access the cluster, first set up your environment by following 'crc oc-env' instructions
```

「OC」という操作コマンドを使えるように環境設定を実施するためのコマンド+オプションです。

```
INFO Then you can access it by running 'oc login -u developer -p developer https://api.crc.testing:6443'
INFO To login as an admin, run 'oc login -u kubeadmin -p DhjTx-8gIJC-2h2tK-eksGY https://api.crc.testing:6443'
```

コマンドライン、およびWeb GUIでそれぞれ開発者(Developer)、kubeadminアカウントでログインする際のパスワード情報が記載されています。ランダム生成のようです。

```
INFO You can now run 'crc console' and use these credentials to access the OpenShift web console
Started the OpenShift cluster
```

Web GUIを起動するためのコマンド「crc console」の説明です。

9．Web GUI

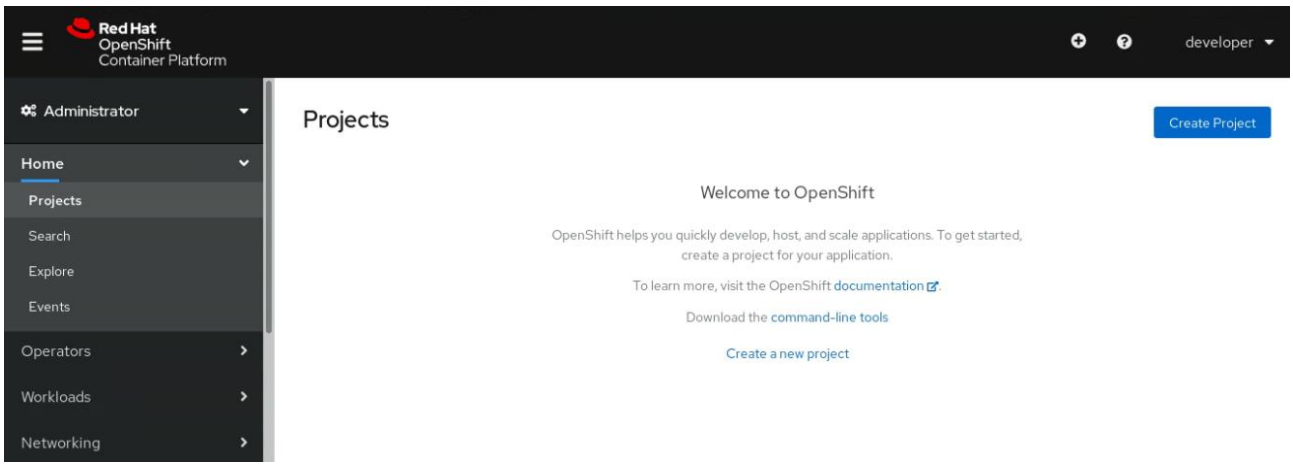
ここではわかりやすくWeb GUIにログインしてみます。

```
$ crc console
```

しばらく待つとWebブラウザーでOpenShfit環境が起動してきます。

「hspasswd_provider」にて「developer:developer」でログインしてみましょう

以下のようなWeb GUIにログインできるかと思います。



10．試用

というわけで、OpenShift試用の準備が整いました。

OpenShiftとしての挙動の確認のために、アプリケーションをテスト的にデプロイするなど、使い方は様々ですが、基本的に本環境は検証用のものであるということをご記憶の上、試用してみましょう。

ここでは非常に簡単なテストアプリケーションのデプロイ、試用、削除のサイクルを簡単に説明してまとめとします。

■主な実施コマンド

ocコマンドを利用できるようにします。

```
$ crc oc-env
$ eval $(crc oc-env)
```

ocコマンドでログインします。***は前述のpull-secretを指定します。

```
$ oc login -u kubeadmin -p *** https://api.crc.testing:6443
```

バージョン確認

```
$ oc version
Client Version: 4.5.4
Server Version: 4.5.4
Kubernetes Version: v1.18.3+012b3ec
```

マスターに認識されるすべてのノードを一覧表示する

インストールしたてなのでマスターノード以外何もありません。

```
$ oc get node
NAME                 STATUS   ROLES           AGE   VERSION
crc-fd5nx-master-0   Ready    master,worker   19d   v1.18.3+012b3ec
```

現在使用中のプロジェクトの概要を示します。今は何もありません。***は単に伏字にしてるだけです。

```
$ oc status
In project ***-first on server https://api.crc.testing:6443

You have no services, deployment configs, or build configs.
Run 'oc new-app' to create an application.
```

起動しているOperatorの状態を示すコマンドです。

```
$ oc get co
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
authentication                             4.5.4     True        False         False      19d
cloud-credential                           4.5.4     True        False         False      19d
cluster-autoscaler                         4.5.4     True        False         False      19d
～出力が長いので省略～
```

■コマンドラインでのサンプルプロジェクトの作成

以下はgitコマンドがインストールされていないとうまく動きません。今回デフォルトのGUIサーバとしてインストールしていたので、gitがインストールされていませんでした。下記コマンドで追加インストールしています。

```
$ sudo yum install git
```

Openshift ClusterにdeveloperアカウントでログインOpenshift Clusterにdeveloperアカウントでログイン

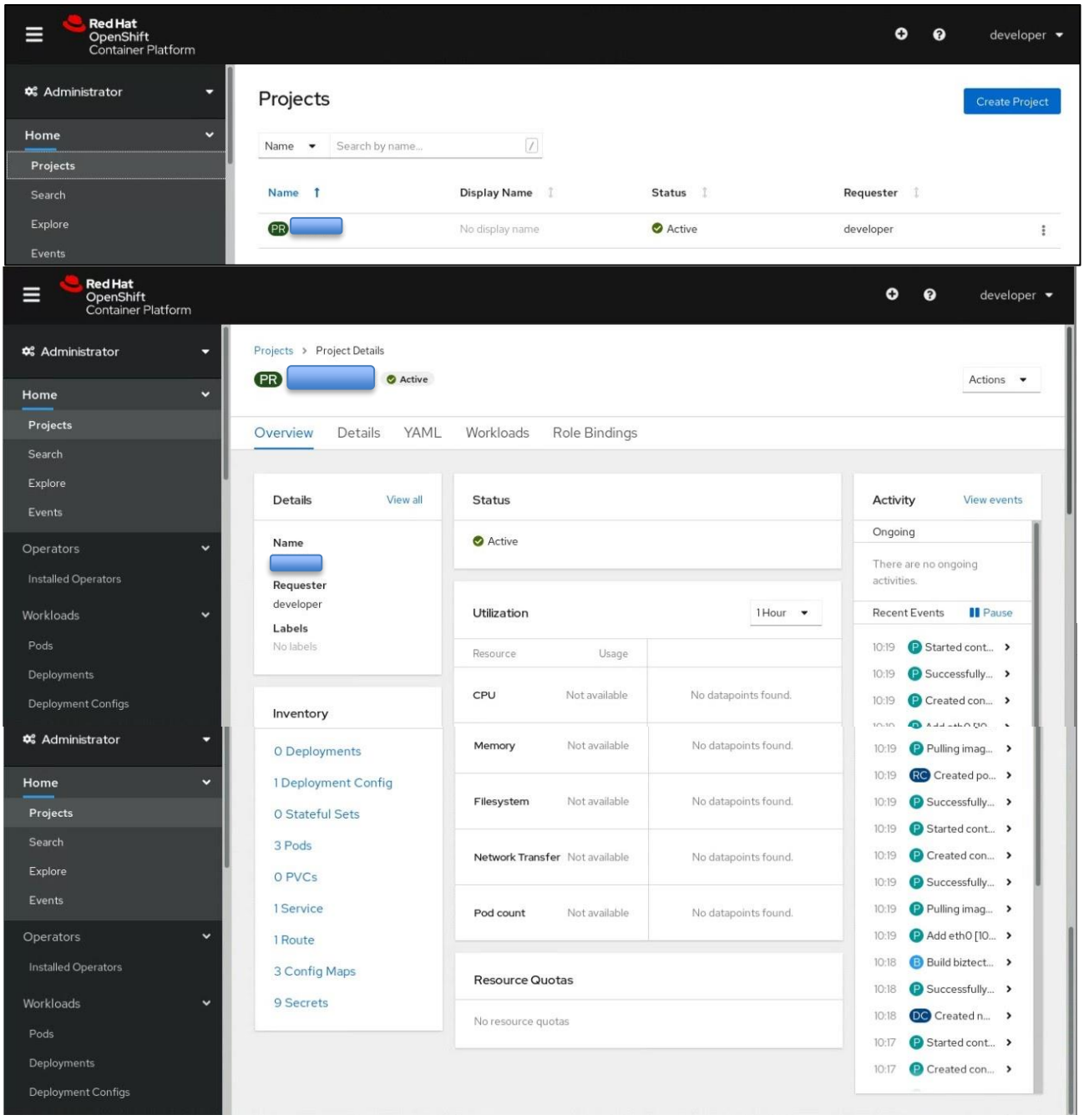
```
$ oc login -u developer -p developer
```

「crctest」という新規プロジェクトを作成

```
$ oc new-project crctest
```

新規プロジェクトを作成すると、WebGUIにも新規のプロジェクトが追加されます。

すみません実環境のスクショなので名前をマスクしてます。



■コマンドラインでサンプルアプリケーションのデプロイ

```
$ oc new-app httpd-example
--> Deploying template "openshift/httpd-example" to project crctest
～
```

■ビルドステータスの監視

```
$ oc logs -f bc/httpd-example
Cloning "https://github.com/sclorg/httpd-ex.git" ...
        Commit: 0ac6da93a1f65fe9175cb1b7838cfca7b23d5fbe (Merge pull request #1
5 from adambkaplan/sclorg-rename)
        Author: Honza Horak <hhorak@redhat.com>
        Date:   Fri Aug 3 13:08:12 2018 +0200
～中略～
Push successful
```

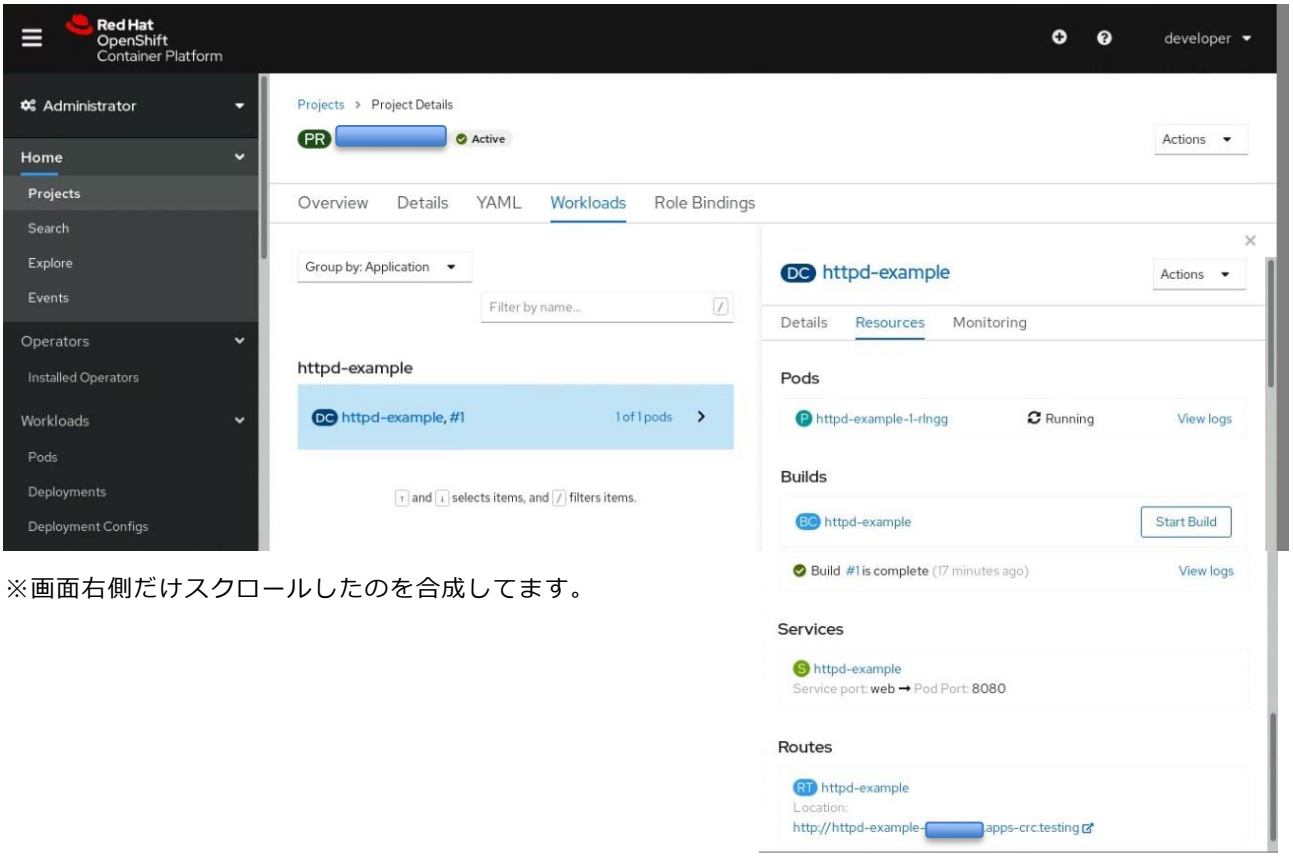
Push successful　が表示されたらビルド完了

```
$ oc get routes
NAME  HOST/PORT   PATH   SERVICES    PORT    TERMINATION   WILDCARD
httpd-example   httpd-example-crctest.apps-crc.testing          httpd-example
<all>                 None
```

■動作確認

```
$ curl -Ik httpd-example-crctest.apps-crc.testing
HTTP/1.1 200 OK
date: Tue, 18 Aug 2020 01:20:31 GMT
server: Apache/2.4.34 (Red Hat) OpenSSL/1.0.2k-fips
last-modified: Tue, 18 Aug 2020 01:18:42 GMT
etag: "924b-5ad1caaee8080"
accept-ranges: bytes
content-length: 37451
content-type: text/html; charset=UTF-8
set-cookie: b29deaf5e54eac06bb32ed4c55b3dfc6=d940b0cfe618b09acb2394f028a7d5af;
path=/; HttpOnly
cache-control: private
```

■アプリケーションのビルド、ルート取得結果の確認



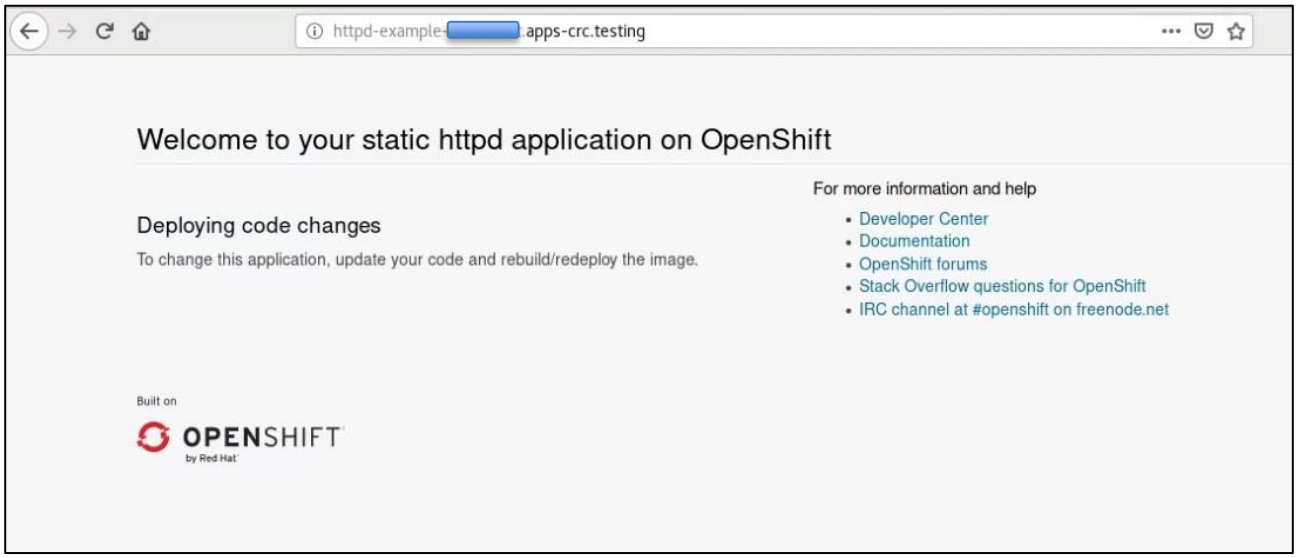
※画面右側だけスクロールしたのを合成してます。

■ビルドしたアプリケーションへのアクセス

ルートを取得した際に表示されたURL

httpd-example-crctest.apps-crc.testing

へブラウザアクセス

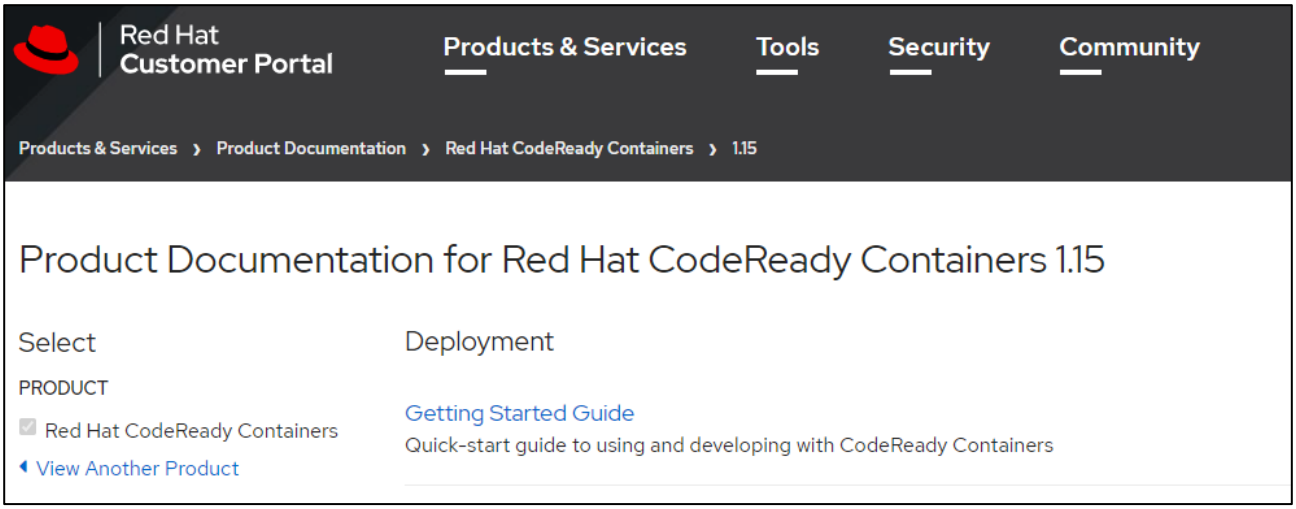


■おわりに

というわけで、ほんとに駆け足で「Red Hat CodeReady Containers」によるOpenShift試用環境の構築についてご紹介しました。

当方はまだまだDocker自体への理解が浅いこと、短時間で駆け足で作成したこともあって足りない部分も多いですが、Red Hat CodeReady Containers自体は商用のOpenShiftの検証用環境ということもありマニュアル類も非常に充実していますので、ご興味ある方は読んでみて下さい。

https://access.redhat.com/documentation/en-us/red_hat_codeready_containers/



一応この先の流れとしては、

・コマンドラインツール「odo」をインストールしてのデプロイ

```
# curl -L https://mirror.openshift.com/pub/openshift-v4/clients/odo/latest/odo-linux-amd64 -o /usr/local/bin/odo
```

```
# chmod +x /usr/local/bin/odo
```

・サンプルアプリケーションの使用。Gitリポジトリに以下のようなサンプルがあるようです。詳細は
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/cli_tools/index#using-sample-applications
他各種Redhatのマニュアルを参照されることを推奨します。

httpd

```
$ odo create httpd --git https://github.com/openshift/httpd-ex.git
```

java

```
$ odo create java --git https://github.com/spring-projects/spring-petclinic.git
```

nodejs

```
$ odo create nodejs --git https://github.com/openshift/nodejs-ex.git
```

perl

```
$ odo create perl --git https://github.com/openshift/dancer-ex.git
```

php

```
$ odo create php --git https://github.com/openshift/cakephp-ex.git
```

python

```
$ odo create python --git https://github.com/openshift/django-ex.git
```

ruby

```
$ odo create ruby --git https://github.com/openshift/ruby-ex.git
```

wildfly

```
$ odo create wildfly --git https://github.com/openshift/openshift-jee-
sample.git
```

バイナリサンプル

java

```
$ git clone https://github.com/spring-projects/spring-petclinic.git
$ cd spring-petclinic
$ mvn package
$ odo create java test3 --binary target/*.jar
$ odo push
```

wildfly

```
$ git clone https://github.com/openshiftdemos/os-sample-java-web.git
$ cd os-sample-java-web
$ mvn package
$ cd ..
$ mkdir example && cd example
$ mv ../os-sample-java-web/target/ROOT.war example.war
$ odo create wildfly --binary example.war
```

当方もまだまだ「導入しただけ」の段階ですが、無料でこういったリッチな環境が構築できてしまうのはとてもありがたいことですね。

■あとがき

というわけでかなりのやっつけ仕事ですが、直近でたまたま構築してみた際の資料が手元にあったので、他に出す予定もありませんし、ざっと作成してみました。
当方自身がまだまだDocker初心者ですが、必要になればやってみるもので、まぁリッチな検証用のソフトウェアがあること自体が非常にありがたいのですが、導入そのものはそこまで敷居は高くないので、是非皆さんもトライしてみてはいかがでしょうか。

お目通しいただき、ありがとうございました。

編集/発行/文責　辻瀬蒼伊
－－
「dabble in...」Extra Edition
－－
発行日：2020/09/12　技術書典9　発行
発　行：Far Northern Other World(Fnow)
　　〒107-0052
　　東京都港区赤坂１－４－４　冨士野ビル501
　　辻瀬蒼伊(立神梢一) aka 佐藤誠之
　URL：https://www.fnow.org
メール：makoyuki@fnwo.org
－－

DABBLE IN... EXTRA EDITION

FAR NORTHERN OTHER WORLD