EAP Quickstarts and OpenShift

前提条件 1/2

- Java 17
  - できればOpenJDK、もしくはOracle JDKかEclipse Adaptiumでもよい
- Git
- Maven
- VS Code もしくは IntelliJ IDEA
- GitHubアカウント

JavaとGitは自分が使うOSのパッケージマネージャを使うのがいい。
Mavenに関してはZIP版をダウンロードしPATHを通すことが多い。

Windows上では WSL2 の使用を想定している。

前提条件 2/2

Red Hatの製品版であることを前提としている。

- EAP 7.4.z
  - カスタマーポータルよりダウンロード (ログインアカウントが必要)
  - 最新の累積パッチ (7.4.z の z) もあわせてダウンロード
  - 具体的には jboss-eap-7.4.0.zip と jboss-eap-7.4.14-patch.zip
- OCP (OpenShift) 4 の開発者向けアカウントと oc コマンド
  - OCPインストール時にPull Secretを設定したはずで、Red Hatのイメージレジストリにアクセスできていること

Javaのメジャーバージョンの他は厳密に同じバージョンにする必要はない。

```
$ java --version
openjdk version "17.0.9" 2023-10-17
OpenJDK Runtime Environment (Red_Hat-17.0.9.0.9-2) (build 17.0.9+9)
OpenJDK 64-Bit Server VM (Red_Hat-17.0.9.0.9-2) (build 17.0.9+9, mixed mode, sharing)

$ javac --version
javac 17.0.9

$ git --version
git version 2.43.0

$ mvn --version
Apache Maven 3.9.1 (Red Hat 3.9.1-3)
Maven home: /usr/share/maven
Java version: 17.0.9, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-17-openjdk-17.0.9.0.9-1.fc39.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "6.6.7-200.fc39.x86_64", arch: "amd64", family: "unix"

$ oc version
Client Version: 4.14.1
Kustomize Version: v5.0.1
Kubernetes Version: v1.27.6+f67aeb3
```

PowerPC上の制限ついて

コンテナイメージはバイナリの実行可能ファイルの塊なので、CPUアーキテクチャごとにビルドされている必要がある。EAPに関してはサポートを謳っているので問題ないが、周辺ツールにはIntel系 (amd64) のイメージしか提供していないものが多い。

- EAP 7.4 with OpenJDK 17
  - ビルダーイメージ
  - ランタイムイメージ
- Nexus Repository Certified Operator
  - ビルドの高速化に必須だがPowerPC版はない
- GitLab Operator
  - Webhookを使いたいがインターネット (GitHub) からOpenShiftにアクセスできない場合の選択肢だがPowerPC版はない

EAP 7.4の累積パッチ摘要とJava 17対応

EAPのローカルマシンへのインストールは本体 (jboss-eap-7.4.0.zip) を解凍するだけだが、追加で累積パッチの摘要とJava 17対応のパッチを当てる必要がある。

EAP本体のインストール

```
$ unzip -q jboss-eap-7.4.0.zip
$ cd jboss-eap-7.4
$ ls -F
appclient/  domain/             LICENSE.txt  standalone/
bin/        JBossEULA.txt       migration/   version.txt
docs/       jboss-modules.jar   modules/     welcome-content/
```

このディレクトリを以降 JBOSS_HOME (もしくはEAP_HOME) と呼ぶ。

各種パッチの適用

```
# 累積パッチの適用
$ ./bin/jboss-cli.sh
grep: warning: stray \ before -
grep: warning: stray \ before -
You are disconnected at the moment. Type 'connect' to connect to the server or 'help' for the list of supported commands.
[disconnected /] patch apply /path/to/jboss-eap-7.4.14-patch.zip
{
    "outcome" : "success",
    "result" : {}
}

[disconnected /] patch info
Version:              7.4.14.GA
Cumulative patch ID: jboss-eap-7.4.14.CP
One-off patches:     none

[disconnected /] exit

# Java 17対応パッチの適用
$ ./bin/jboss-cli.sh --file=docs/examples/enable-elytron-se17.cli
```

EAPの起動

```
# インストール直後の初回のみ、管理コンソールを使うためのユーザを "admin/password" で登録する
$ ./bin/add-user.sh --user admin --password password

$ bin/standalone.sh
...
14:23:24,165 INFO ... WFLYSRV0025: JBoss EAP 7.4.14.GA (WildFly Core 15.0.32.Final-redhat-00001) started in 3381ms ...
...
```

- 管理コンソールには http://localhost:9990/ でアクセスできる
  - しかし (OpenShift上の場合は特に) 実際にはあまり使わずJBoss CLIで済ませることがほとんど
- ユーザのアプリは http://localhost:8080/ 以下のパスで提供される
- 停止は Ctrl-C でよい

サンプルプロジェクト集 Quickstarts のクローン

JBOSS_HOMEとは別のディレクトリにGitHubからクローンしておく。

```
$ mkdir /some/where
$ cd /some/where

$ git clone https://github.com/jboss-developer/jboss-eap-quickstarts.git

# このディレクトリを QUICKSTART_HOME と呼ぶこともある
$ cd jboss-eap-quickstarts

# README.htmlをブラウザで開く
$ xdg-open README.html
```

GitHubからクローンするのではなく、EAP本体と同じくカスタマーポータルからZIP形式でダウンロードすることもできる。

Quickstarts

EAPだけでなくRed Hatのミドルウェアにはその使い方を端的に示すためのサンプルプロジェクト集が提供されている。EAPを起動しておき、興味あるプロジェクトの中で `mvn package wildfly:deploy` とするだけで十分なものがほとんどである。

- トップページの表に難しさ (Beginner/Intermediate/Advanced) や使用している要素技術が併記されている
- 最初に学ぶべき特に重要な要素技術は以下のとおり。いずれもJava EEにおける仕様の名前である
  - CDI: アノテーションを使った依存性注入
  - JPA: DBアクセス (中ではHibernateを使用している)
  - JAX-RS: REST APIを作るためのフレームワーク

EAPはデフォルトでH2と呼ばれるテスト用のDBを内臓しているので、外部にDBを用意しなくともDBを使うサンプルプロジェクトを動かすことができる。

自分用にコピーを作成する

必要なプロジェクトだけを自分用にコピーして、自由に編集・使い捨てできるようにする。ここでは helloworld-html5 を例にするが、基本的にどのプロジェクトでもよい。

```
# どこか適当なディレクトリに移動しておき、helloworld-html5のみをコピーする
$ cp -r $QUICKSTART_HOME/helloworld-html5 .

# 親ディレクトリにあるpom.xmlも必要になるので別名にしてコピーしておく
$ cp $QUICKSTART_HOME/pom.xml helloworld-html5/pom-parent.xml

$ cd helloworld-html5
$ vi pom.xml
# 28行目の<relativePath>を親からコピーしたpom.xmlを指すように編集する
-:  <relativePath>../pom.xml</relativePath>
+:  <relativePath>pom-parent.xml</relativePath>
```

helloworld-html5のビルドとデプロイ

```
# ビルド (`mvn install`でもよい)
$ mvn package

# デプロイ (すでにEAPを起動済みであること)
$ mvn wildfly:deploy
```

EAPのログにデプロイが行われたことを表すログが出力され、
http://localhost:8080/helloworld-html5 でこのアプリにアクセス可能になる。

参考: mvnの便利なコマンド集

- `mvn clean package wildfly:deploy`
  - ビルド結果を削除して再ビルドおよびデプロイを続けて行う
- `mvn help:describe`
  - ヘルプの使い方を表示
  - 例えば `mvn help:describe -Dcmd=wildfly:deploy -Ddetail=true` のように使う
- `mvn package -DskipTests`
  - テストの依存性をダウンロードするがテストは実行せずにビルド
- `mvn package -Dmaven.test.skip`
  - テストの依存性のダウンロードもせずにビルドのみを実行

helloworld-html5用のGitリポジトリの作成

先ほどコピーしたhelloworld-html5のみを含むリポジトリを自分のアカウントのGitHubに作成する(以下ではSSHの設定が済んでいると仮定している)。自由に改変しコミットしてよい。

```
# GitHub画面右上の"+"で"New repository"を選び、"helloworld-html5"の名前で空のパブリックリポジトリを作成

# (pom-parent.xmlが存在する)自分のhelloworld-html5内で
$ echo target >> .gitignore
$ echo .idea >> .gitignore
$ echo .vscode >> .gitignore
$ echo "# My helloworld-html5 quickstart" > README.md
$ git init
$ git add .
$ git commit -m 'first commit'
$ git branch -M main
$ git remote add origin git@github.com:<自分のアカウント名>/helloworld-html5.git
$ git push -u origin main
```

後でOpenShiftはこのリポジトリからコードを取得しビルドおよびデプロイすることになる。

OpenShiftでの実行

開発者向けのアカウントですでに `oc login` コマンドでOpenShiftにログイン済みとする。ここでの作業用にプロジェクト(名前空間)を一つ作成する。

```
# シェルの補完を有効にする
$ source <(oc completion bash)

# ログインしているかどうかの確認
$ oc whoami

# プロジェクトの作成(既存のプロジェクトに移動する場合は`oc project <プロジェクト名>`)
$ oc new-project myproj

# 現在の状態の表示
$ oc status
```

複数のユーザで同時に作業している場合はプロジェクト名(ここではmyproj)が被らないようにする。

現在のプロジェクトにImageStreamとTemplateを導入

GitHubのjboss-container-images/jboss-eap-openshift-templatesにEAPの公式ImageStreamとTemplateがあるのでそれを現在のプロジェクトに導入する。

```
$ oc apply -f https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-templates/eap74/eap74-openjdk17-image-stream.json
imagestream.image.openshift.io/jboss-eap74-openjdk17-openshift created
imagestream.image.openshift.io/jboss-eap74-openjdk17-runtime-openshift created

$ oc get is
NAME                                  IMAGE REPOSITORY                                                                                  TAGS         UPDATED
jboss-eap74-openjdk17-openshift           default-route-openshift-image-registry.apps-crc.testing/myproj/jboss-eap74-openjdk17-openshift           7.4,latest   29 seconds ago
jboss-eap74-openjdk17-runtime-openshift   default-route-openshift-image-registry.apps-crc.testing/myproj/jboss-eap74-openjdk17-runtime-openshift   7.4,latest   25 seconds ago

$ for resource in eap74-amq-persistent-s2i.json eap74-amq-s2i.json eap74-basic-s2i.json eap74-https-s2i.json eap74-sso-s2i.json ; \
  do oc apply -f https://raw.githubusercontent.com/jboss-container-images/jboss-eap-openshift-templates/eap74/templates/${resource}; done
template.template.openshift.io/eap74-amq-persistent-s2i created
template.template.openshift.io/eap74-amq-s2i created
template.template.openshift.io/eap74-basic-s2i created
template.template.openshift.io/eap74-https-s2i created
template.template.openshift.io/eap74-sso-s2i created

$ oc get template
NAME                       DESCRIPTION                                                                   PARAMETERS      OBJECTS
eap74-amq-persistent-s2i   An example JBoss Enterprise Application Platform application using Red Hat JB...   39 (10 blank)   14
eap74-amq-s2i              An example JBoss Enterprise Application Platform application using Red Hat JB...   37 (10 blank)   13
eap74-basic-s2i            An example JBoss Enterprise Application Platform application. For more inform...   20 (5 blank)    8
eap74-https-s2i            An example JBoss Enterprise Application Platform application configured with...    30 (11 blank)   10
eap74-sso-s2i              An example JBoss Enterprise Application Platform application Single Sign-On a...   50 (21 blank)   10
```

これらは`openshift`名前空間にすでに存在することもあるが古かったりするので独自に取得する。また自分の名前空間に置くことで直接編集することもできる。

何を行ったのか

- ImageStream
  - Red HatのレジストリにあるOpenJDK 17を使ったEAPのビルダーイメージおよびランタイムイメージへの参照
- Template
  - 最終的に必要なのはDeploymentConfigやServiceなどのOpenShiftが解釈できるリソースだが、それらを直接書くのではなくテンプレートとしてパラメタ化したもの
    - DeploymentConfigの変わりにDeploymentが推奨になったが、当面は使い続けられる
  - 似た技術としてHelmやKustomizeなどがあるが、Templateは単純なので直感的に理解できる
  - `oc describe template <name>` や `oc get template <name> -o yaml` で中身を確認するとよい

oc nw-appコマンドによるビルドとデプロイ

GitHubのリポジトリのアカウント名は適宜読み替えること。

```
$ oc new-app --template=eap74-basic-s2i \
  -p APPLICATION_NAME=myapp \
  -p IMAGE_STREAM_NAMESPACE=$(oc project -q) \
  -p EAP_IMAGE_NAME=jboss-eap74-openjdk17-openshift:latest \
  -p EAP_RUNTIME_IMAGE_NAME=jboss-eap74-openjdk17-runtime-openshift:latest \
  -p SOURCE_REPOSITORY_URL=https://github.com/onagano-rh/helloworld-html5 \
  -p SOURCE_REPOSITORY_REF=main \
  -p CONTEXT_DIR=""

# `oc logs -f buildconfig/myapp-build-artifacts` でビルドの様子を、
# ビルドの正常終了後は `oc logs -f buildconfig/myapp` デプロイの様子を確認できる
```

エラー時は `oc delete all -l application=myapp` で関連リソースを削除してやり直す。もしくはプロジェクトごと再作成してもよい（`oc delete project myproj`）。

Nexusを用意してそれを `MAVEN_MIRROR_URL` に設定しないと毎回ライブラリをインターネットからダウンロードすることになるのでビルドが遅くなる点には注意が必要。

デプロイされたアプリにアクセス

ビルドもデプロイも正常に終了すると、デプロイようのポッドのStatusが"Running"になる。Routeを確認してブラウザでアクセスしてみる。

```
$ oc get pod
NAME                          READY   STATUS      RESTARTS   AGE
myapp-1-6thw9                 1/1     Running     0          5m14s
myapp-1-deploy                0/1     Completed   0          5m17s
myapp-2-build                 0/1     Completed   0          6m26s
myapp-build-artifacts-1-build   0/1     Completed   0          10m

$ oc get route
NAME    HOST/PORT                          PATH   SERVICES   PORT    TERMINATION     WILDCARD
myapp   myapp-myproj.apps-crc.testing             myapp      <all>   edge/Redirect   None

# ブラウザで https://myapp-myproj.apps-crc.testing/ にアクセスしてみる。
```

コードの修正と再ビルドの例 1/2

デフォルトのhelloworld-html5はhttpでの動作を前提にしているので、OpenShiftのhttpsでのアクセスでは動作しない。このバグを修正しGitHubにコミットして、新しいコードを使ったビルドおよびデプロイを行う。

```
$ vi src/main/webapp/index.html
(jqueryをダウンロードする部分(22行目)のプロトコルをhttpsに変更する)
$ git add src/main/webapp/index.html
$ git commit -m "jQueryをhttpではなくhttpsで取得するよう変更"
$ git push
```

コードの修正と再ビルドの例 2/2

```
# BuildConfigの名前を確認
$ oc get bc

# 再ビルド (--followによりログも確認できる)
$ oc start-build myapp-build-artifacts --follow

# 新しいビルドが正常終了すると、イメージの変更を検出し自動で新しいデプロイも始まる
$ oc get pod -w
```

GitHubのリポジトリでWebhookを設定すれば手動で`oc start-build`する必要はなくなるが、OpenShiftがインターネットからアクセス可能である必要がある。

その他のQuickstartsプロジェクトについて

- kitchensink-jsp
  - フロント側の技術としてJSPを採用
  - DBアクセスを行うJPAも使っておりhelloworld-html5より本格的
  - デフォルトでは内臓DBのH2を使う
- todo-backend
  - DBとしてOpenShift上に別ポッドとしてPostgreSQLを立てる構成でより本格的
  - これ自体はバックエンドのみで、外部のサイト が提供するSPAをフロントエンドとして使う
    - クロスサイトの呼び出し例になっており、より本格的なマイクロサービスの例と言える

参考: EAP 7.4の他の選択肢について

- EAP 8.0
  - 2023年末現在、ベータ版はあるがまだ未リリース
  - Javaのソースコードのパッケージ名に変更がある
- Quarkus
  - Springの対抗馬としてRed Hatが中心となって開発している
  - EAPやSpringよりも軽量で圧倒的に使いやすい
  - CDIやJPA、JAX-RSといった基本APIはJava EEと同じで、EAPから移行しやすい

参考: 他のプログラミング言語の例

Todoアプリのようなシンプルな共通の課題を、様々な言語やフレームワークで実装して比較するというおもしろい試みがある。

- https://todobackend.com/
  - バックエンド部分の様々な実装がある
- https://todomvc.com/
  - フロントエンド部分を特にJavaScriptの様々なフレームワークで実装した例
  - 前述の todobackend.com のバックエンドに繋がるわけではない
- https://codebase.show/projects/realworld
  - Todoアプリよりかなり本格的な"Conduit"というブログサイトの実装集