\Orchestrating a brighter world NEC

Java アプリケーションのコンテナ開発における CI/CDの実践ガイド

2022年12月6日　第1.0版

日本電気株式会社
スマートエッジ事業部門

\Orchestrating a brighter world

NECは、安全・安心・公平・効率という社会価値を創造し、
誰もが人間性を十分に発揮できる持続可能な社会の実現を目指します。

目次

はじめに

本書に関する注意事項
背景と目的
表記ルール

本書に関する注意事項

1. WebOTXは日本電気株式会社の商標登録です。
2. OracleとJavaは、Oracle Corporation及びその子会社、関連会社の米国及びその他の国における商標登録です。
3. Cloud Native Computing Foundationは、米国及びその他の国においてThe Linux Foundationの商標登録です。
4. 本書の一部または全部を無断に転載することを禁じます。
5. 本書に関しては将来予告なしに変更することがあります。
6. 弊社の許可なく複製、改変することを禁じます。
7. 実行した結果の影響については、弊社は一切責任を負いません。

背景と目的

◆ 背景

NECは2022年9月にWebOTXシリーズ最新版であるWebOTX Application Server V11.1をリリースしました。
このバージョンではDockerfileを使用せずに少ない手順で高品質なコンテナイメージの生成を行う
「WebOTX Builder」を新たに提供しています。
これにより、従来よりも容易かつ効率的にCI/CDが実現できるようになりました。

◆ 目的

本資料では「WebOTX Builder」を活用してコンテナ開発・運用を自動化したCI/CD環境の構築方法を説明します。
CI/CD環境の導入や検討の方や、WebOTX Application Server Express V11.1 for Containerの利用者または検討中の方に特に有用です。

◆ 対象読者

- コンテナ運用におけるCI/CDに興味がある方
- コンテナに関する基本的なコマンドの知識がある方

表記ルール

◆ パスの表記

本資料では、パスの表記を”/”としています。

◆ 用語定義

用語・略語	意味
アプリケーション	Java言語で実装されたアプリケーション
アプリコンテナ	アプリケーションが配備され実行可能な状態または実行中のコンテナ
アプリイメージ	アプリコンテナの生成元となるコンテナイメージ

◆ プレースホルダの表記

プレースホルダ	説明
${JAVA_HOME}	Javaのインストールディレクトリ
${APP_GIT_URL}	サンプルアプリケーションのリポジトリURL

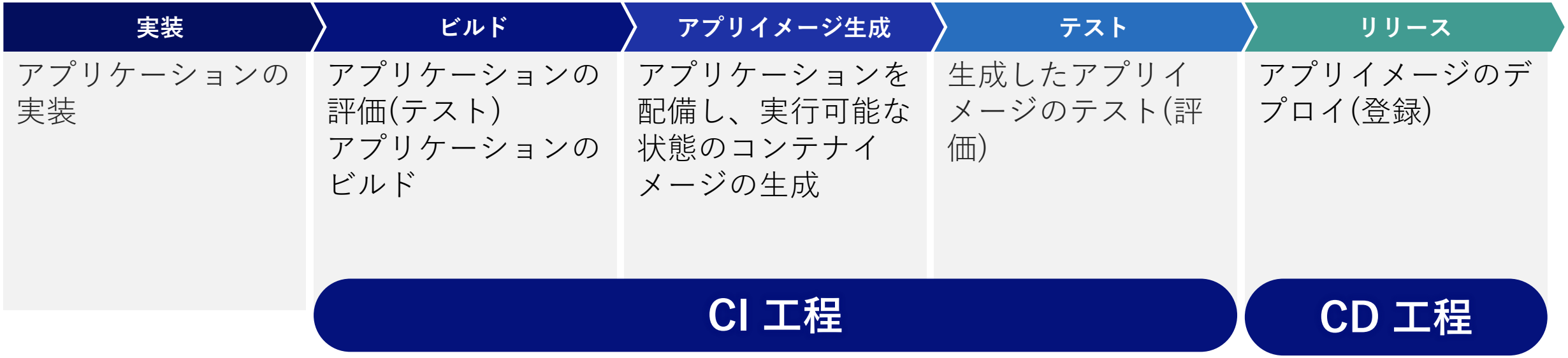
WebOTXが提供するサンプルアプリケーションの
リポジトリURLは**後日公開**

従来のコンテナ開発と運用

コンテナ開発の流れ
CI工程の課題
課題解決アプローチ

コンテナ開発の流れ

一般的にコンテナ開発は、以下の流れで行います。
説明の都合上、要件定義、設計フェーズは割愛しています。
本資料では、CI工程をメインとして説明します。



CI工程の課題

CI環境構築後、以下の工程内の要素におけるメンテナンスが必要となります。

ビルド

- アプリケーションのビルド環境
  - コンテナイメージ内環境を考慮する必要がある
- アプリケーションの改修に伴う、ビルド処理の変更
  - 例：Javaアプリケーションの実装変更に伴う、ビルドコマンド(引数も含む)の変更
- ビルド用ソフトウェアのアップグレード

アプリイメージ生成

- Dockerfile
  - ベースイメージ（種類やバージョンの選択）
  - 機密情報（プロキシ情報や証明書ファイルなど）の利用と排除
  - エントリポイントの設定
  - 内包ミドルウェアのバージョンやセキュリティ設定
  - イメージサイズの軽量化

課題解決アプローチ

Before

課題のポイント

- ビルド用環境のメンテナンス
- Dockerfileのメンテナンス

After

解決のポイント

- ビルド環境構築とビルド処理の自動化
- アプリイメージ生成処理を簡素化(属人性の排除)
  - アプリイメージ生成ツールと設定ファイルの活用

従来課題を解決するCI

Cloud Native Buildpacksとは
CI構成の比較
従来との比較
Cloud Native Buildpacksの構成要素
アプリイメージ生成手順

Cloud Native Buildpacksとは

これら従来課題の解決方法の一つにCloud Native Buildpacksが挙げられます。

本プロジェクトは、CNCFというクラウドネイティブコンピューティング技術を推進する財団によりホストされています。

◆ 機能

ユーザは自身のソースコードを与えるだけで、OCI仕様に準拠したアプリイメージを生成することができます。

生成したアプリイメージは、ベースOSのイメージとビルドされたユーザアプリケーション、

そして必要に応じてアプリケーションを実行するミドルウェア等のランタイムや

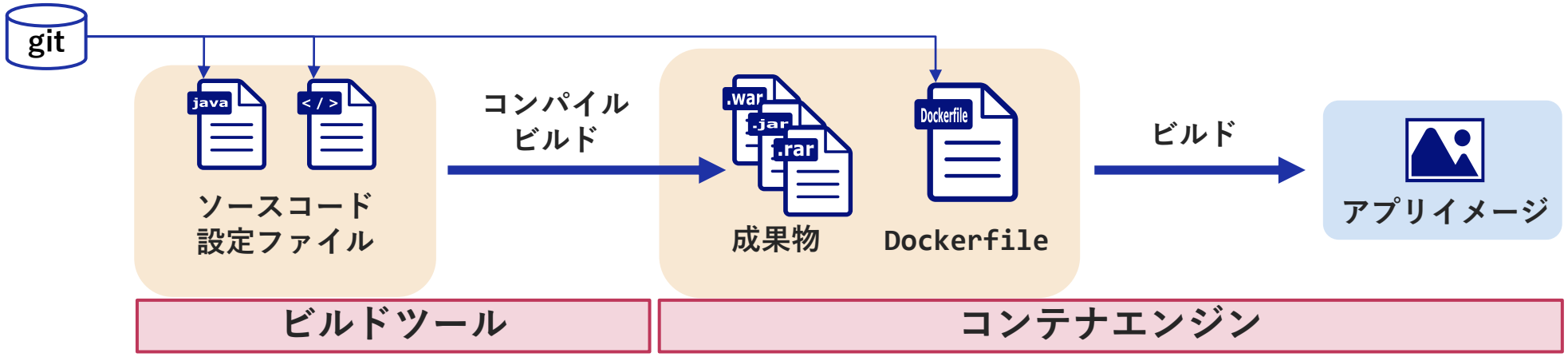
アプリケーションの依存ライブラリが追加されるため、

コンテナランタイムがインストールされたあらゆる環境で実行することができます。

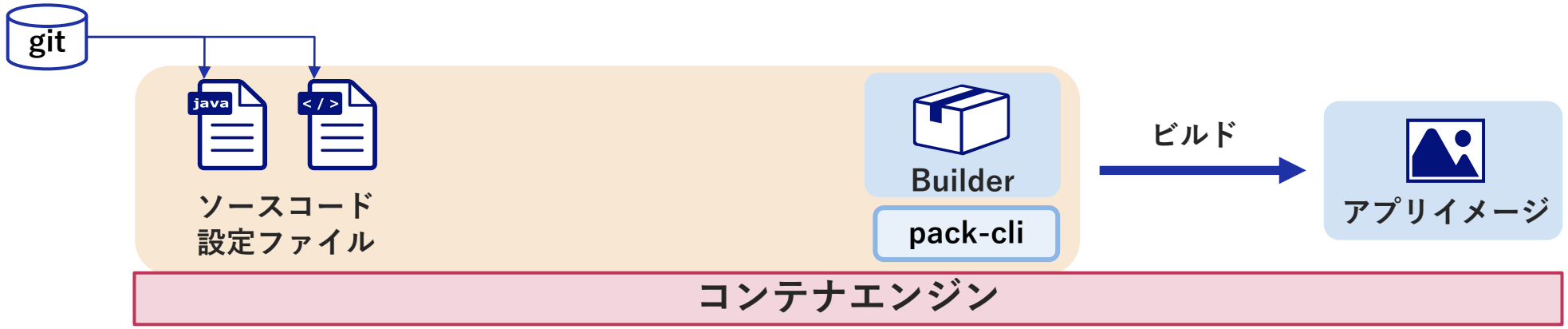
上記機能を実現するためのツール(pack-cli)が提供されています。

CI構成の比較

◆ 従来の構成



◆ Cloud Native Buildpacks(pack-cli)を使用した構成



従来手法との比較

コンテナイメージ生成処理(手法)における、従来手法とCloud Native Buildpacksを使用した手法の比較は次の表の通りです。

（◎: 非常に優れている、〇: 優れている、△: 課題がある）

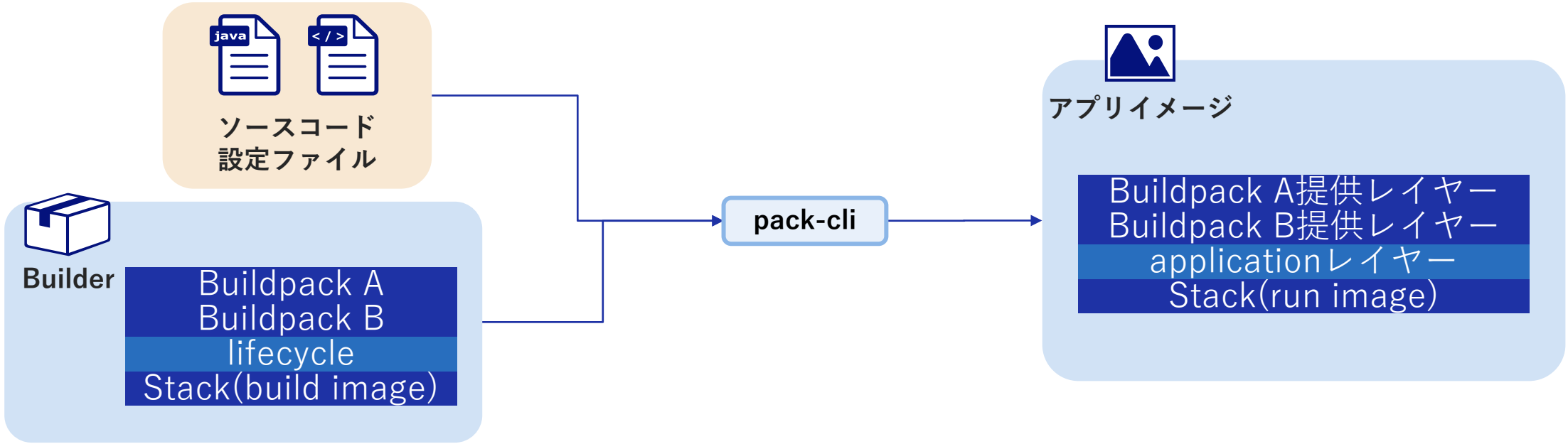
メンテナンス対象	従来手法	Cloud Native Buildpacks	Cloud Native Buildpacksのメリット
ビルド環境	△	◎	ビルド環境を自動で構築し、容易に変更することができます。
Dockerfile	△	◎	Dockerfileを使用しないため、Dockerfileのメンテナンスが必要ありません。
Builderイメージ (詳細は後スライドを参照)	-	〇	Builderイメージは提供元でメンテナンスがされるため、利用者がメンテナンスを行う必要ありません。

従来手法と比べCloud Native Buildpacksを使用した手法は、
利用者のメンテナンス対象を減らし、よりアプリケーション開発への専念を可能とします。

Cloud Native Buildpacksの構成要素

Cloud Native Buildpacksは、主に以下の要素で構成されています。

構成要素	意味
Buildpack	アプリケーションのソースコードに対し検証や依存関係の解決、コンパイル、実行環境のインストール等の作業を行うコンテナイメージ
Stack	コンテナのベースイメージ。 ビルド環境で使用されるbuild imageと実行環境で使用されるrun imageの2種類が存在する
Builder	アプリイメージ生成のために必要な要素(Buildpack, Stackなど)を全て含むコンテナイメージ



アプリイメージの生成手順

アプリイメージ生成コマンド | Cloud Native Buildpacks (pack-cli)

pack-cliのインストール

pack-cliを予めインストールしておく必要があります。
インストール方法は、以下のドキュメントを参照ください。
https://buildpacks.io/docs/tools/pack/
必要に応じてpack-cliをパスに設定ください。
以降の説明では、パスに設定されていること前提として記載しています。

```
$ pack build <image>  --path <アプリケーションのソースコードパス> ¥
                      --builder <Builderイメージ名> ¥
                      --env <環境変数名>=<環境変数値> ¥
                      --volume <ホスト上のパス>:<ビルド用コンテナ上のパス>
```

オプション	説明
--path	アプリケーションのソースコードを格納しているルートディレクトリパスを指定
--builder	使用するBuilderイメージを指定
--env	ビルド用コンテナ内に設定する環境変数とその値を指定
--volume	ビルド用コンテナ内にマウントするディレクトリの指定 [:(コロン)区切り]

アプリイメージの生成手順

アプリイメージの生成例

実際にコンテナイメージ生成手順をサンプルアプリケーションやBuilderイメージを使用して紹介する資料を用意しています。

「Appendix｜アプリイメージ生成手順」に記載しておりますので、是非ご参照ください。

説明中で使用しているサンプルアプリケーションやBuilderは公開しておりますので、ダウンロードした上でご利用いただけます。

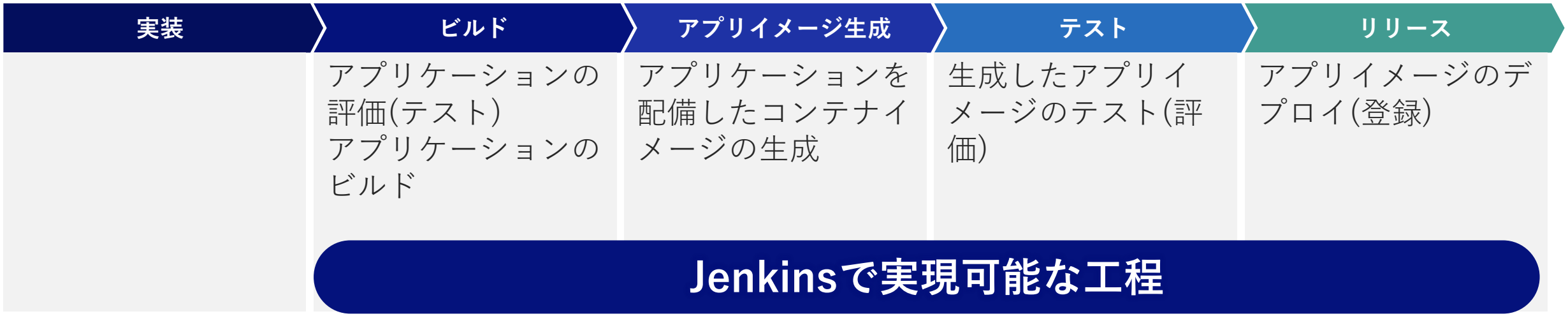
Jenkinsを用いたコンテナ開発と運用の自動化

主要な自動化方法
一般的なJenkinsの構成例
JenkinsとCloud Native Buildpacksの構成

主要な自動化方法

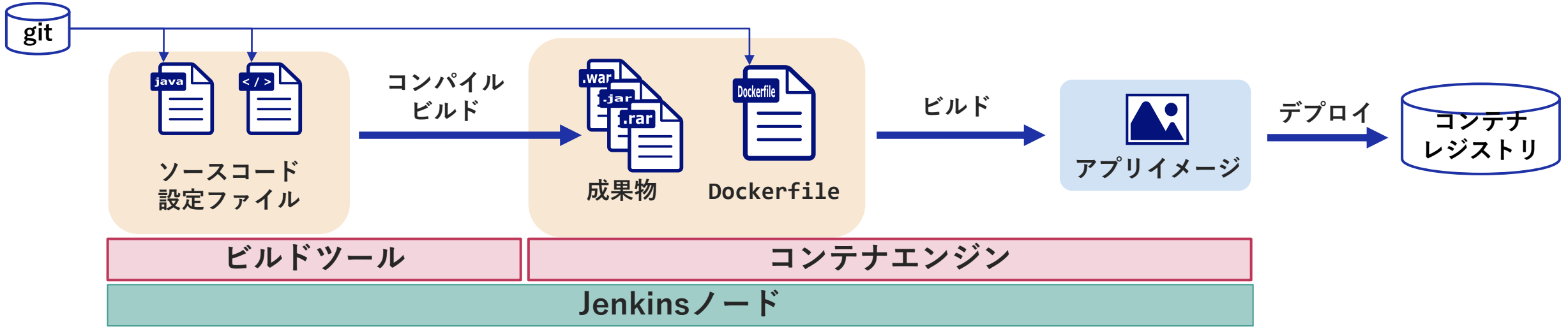
コンテナ開発と運用の自動化において、広く普及しているOSSの一つにJenkinsが挙げられます。

Jenkinsを使用することで、アプリケーションの開発に専念することが可能となります。



一般的なJenkinsの構成例

一般的なJenkins構成では、Jenkinsノードにビルドツールとコンテナエンジン環境を構築し、アプリイメージ生成の一連の流れを実現します。



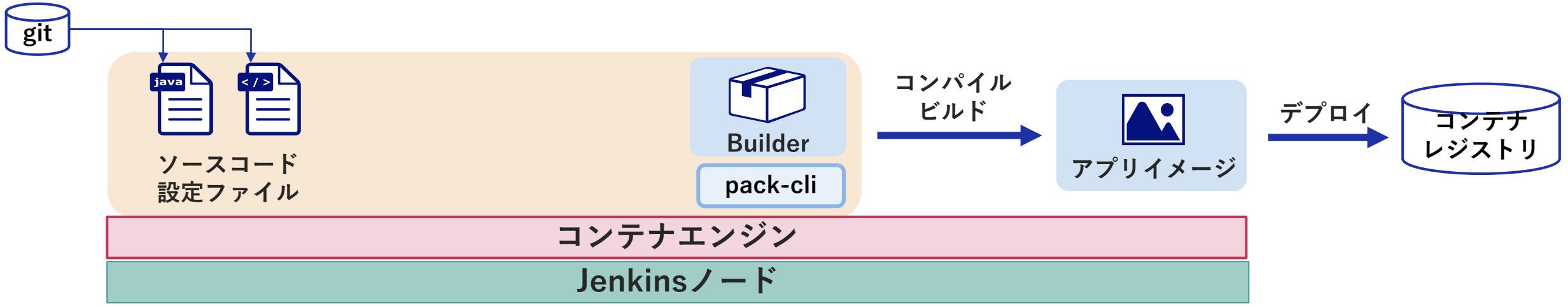
必要な構成要素

- ビルドツール
- コンテナエンジン(DockerやPodmanなど)
- Dockerfile

JenkinsとCloud Native Buildpacksの構成(1/12)

構成図

本章では、JenkinsとCloud Native Buildpacks(pack-cli)を使用した以下の構成図を実現する構築手順を説明します。



必要な構成要素

- コンテナエンジン(DockerまたはPodman)
- pack-cli
- Builderイメージ

不要な構成要素

- ビルドツール
- Dockerfile

JenkinsとCloud Native Buildpacksの構成(2/12)

Jenkinsのインストール

1. Jenkinsのインストール・起動・設定

   インストール及び起動・設定方法は、Jenkins公式のドキュメントをご参照ください。

   Jenkinsのインストールドキュメント
   https://www.jenkins.io/doc/book/installing/

2. Jenkinsの確認

   http://localhost:8080/にアクセスし、Jenkins画面が表示されてるのを確認してください。

JenkinsとCloud Native Buildpacksの構成(3/12)

Jenkinsノード環境構築

3. コンテナエンジンのインストール

コンテナエンジンは、DockerまたはPodmanを使用することができます。

インストール及び設定方法は、OS及びコンテナエンジンのドキュメントをご参照ください。

Dockerのインストールドキュメント
https://docs.docker.com/engine/install/

4. pack-cliのインストール

インストール及び設定方法は、Cloud Native Buildpacks公式のドキュメントをご参照ください。

pack-cliのインストールドキュメント
https://buildpacks.io/docs/tools/pack/

5. WebOTX Builderのインストール

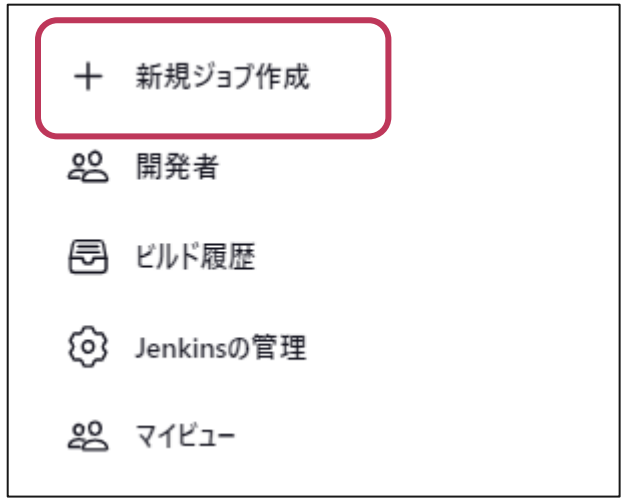
「Appendix | アプリイメージの生成手順　~WebOTX Builder~」をご参照ください。

JenkinsとCloud Native Buildpacksの構成(4/12)

Jenkinsジョブ構築

6. 新規Jenkinsジョブの作成

Jenkinsログイン後の左ペインから「新規ジョブ作成」を選択します。



7. ジョブ情報の入力/選択

ジョブ名を入力します。

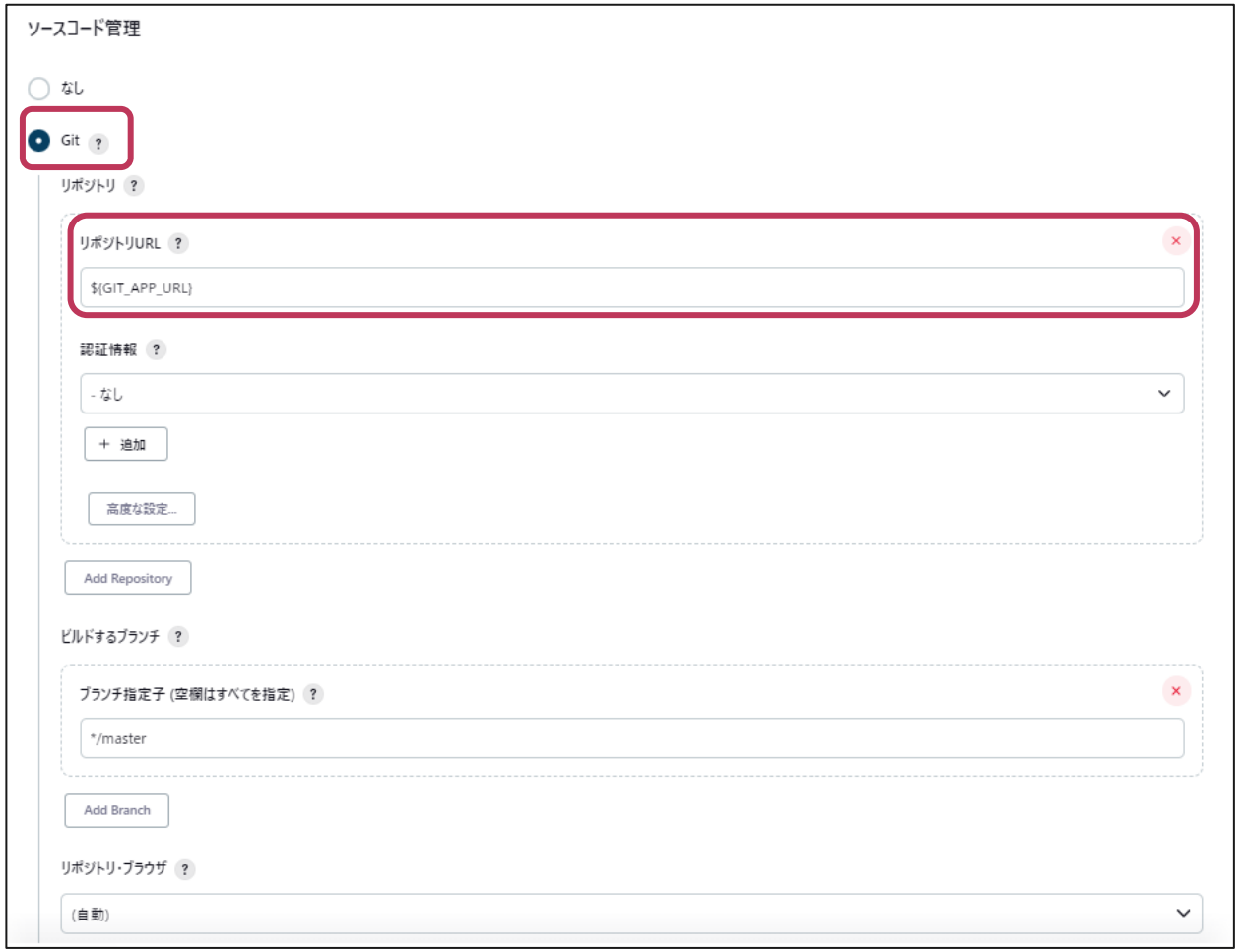
「フリースタイル・プロジェクトのビルド」を選択します。

JenkinsとCloud Native Buildpacksの構成(5/12)

Jenkinsジョブ構築

8. ソースコード設定

「Git」を選択します。
リポジトリURLに${GIT_APP_URL}を指定します



ソースコード管理の選択肢にGitがない場合

Jenkinsプラグインから以下のGit用プラグインをインストール必要があります。
https://plugins.jenkins.io/git/

Jenkinsプラグインのインストールは、
Jenkinsの管理 > プラグインの管理 > 利用可能タブ
より検索とダウンロード可能です。

※ダウンロード後、Jenkinsの再起動が必要です。

JenkinsとCloud Native Buildpacksの構成(6/12)

Jenkinsジョブ構築

9. ビルド処理設定｜アプリイメージ生成

「シェルの実行」を選択します。

シェルスクリプトにてアプリイメージ生成コマンドを記述します。

JenkinsとCloud Native Buildpacksの構成(7/12)

Jenkinsジョブ構築

10. ビルド処理設定｜アプリイメージのデプロイ

「シェルの実行」を選択します。

シェルスクリプトにてアプリイメージのデプロイコマンドを記述します。

※<repository>はコンテナレジストリに登録しているレポジトリ名パスを指定します。

JenkinsとCloud Native Buildpacksの構成(8/12)

ビルドジョブのパラメータ化

前スライドに続き、追加機能としてビルドジョブのパラメータ化を紹介します。

Jenkinsジョブのビルドパラメータ化(環境変数やファイルの受け渡し)を有効にすることで、以下の項目における変更を容易かつ動的に実現可能です。

ビルドパラメータ対象	変更内容
Builderイメージ	アプリイメージにおけるベースイメージ・実行基盤・内包ソフトウェア
ビルドツール	ビルド処理・ビルドツールのバージョン
アプリイメージ	イメージ名・タグ名

ビルドジョブをパラメータ化することで、ソフトウェアのバージョンアップやアプリケーションビルドの変更などを柔軟に実現できます。

本章ではMavenインストーラファイルのパラメータ化を例に説明しています。

JenkinsとCloud Native Buildpacksの構成(9/12)

ビルドジョブのパラメータ化 | Jenkinsジョブ追加設定

前スライドで構築したJenkinsジョブに追加で設定行います。

11. General設定 | ビルドのパラメータ化

「ビルドのパラメータ化」を有効化します。

パラメータの追加から「文字列」を選択します。

設定値には、名前(環境変数)、デフォルト値を指定します。

Mavenインストーラファイルを指定できる環境変数(BP_MAVEN_URL)とMaven3.6.3のインストーラファイルダウンロード用URLを使用しています。

JenkinsとCloud Native Buildpacksの構成(10/12)

ビルドジョブのパラメータ化 | Jenkinsジョブ追加設定

12. ビルド処理設定 | アプリイメージ生成

既存のビルド処理（アプリイメージ生成コマンド）に、Jenkinsノードに設定された環境変数を適用するオプション(--env)を追記します。

Jenkinsノードの環境変数(BP_MAVEN_URL)の値をビルド環境にも適用させています。

JenkinsとCloud Native Buildpacksの構成(11/12)

ビルドジョブのパラメータ化 | Jenkinsジョブ実行

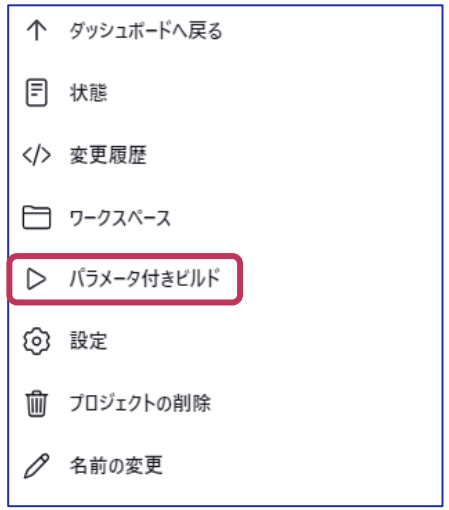
13. Jenkinsジョブ実行

「パラメータ付きビルド」を選択します。

環境変数値をビルド時に書き換えることで、

使用するMavenインストーラファイルを変更することができます。

ビルドのパラメータを使用し、ビルドツールやアプリイメージの
内包ソフトウェアの変更を実現できます。



JenkinsとCloud Native Buildpacksの構成(12/12)

ビルドジョブのパラメータ化 | その他の設定(環境変数)

WebOTX Builderでは、他にも以下の主な環境変数に対応しています。同様にビルドジョブのパラメータとして設定することで、ビルド時にカスタム・使用することができます。

環境変数	説明
BP_JDK_URL	JDKインストーラファイルを指定できます。
BP_MAVEN_BUILD_ARGUMENTS	Mavenプロジェクト(アプリケーション)のビルドコマンドの引数を指定できます。
BP_MAVEN_CLEAR_M2_CACHE	キャッシュしているMavenリポジトリをクリアできます。
BP_MAVEN_POM_FILE	MavenプロジェクトのPOMファイルを指定できます。

Cloud Native Buildpacksのキャッシュ機能により、カスタム環境が前回構築した環境と同様の場合は再利用され、アプリイメージの生成時間を短縮できます。

おわりに

本ホワイトペーパーのおさらい

本ホワイトペーパーのおさらい

Cloud Native Buildpacks(pack-cli)で実現できること

- アプリケーション開発に専念できる
  - ビルドツールのメンテナンスが不要
  - Dockerfileのメンテナンスが不要
- Builderによる品質の高いアプリイメージの生成
  - アプリイメージの軽量化
  - 機密情報等の漏洩防止

JenkinsとCloud Native Buildpacks(pack-cli)連携の強み

- 実行毎に以下の項目を容易に変更できる（ビルドパラメータと併用）
  - ビルド環境
  - アプリイメージのベースイメージ
  - アプリイメージの内包ソフトウェア
  - ビルド処理

Appendix

WebOTX Application Server コンテナ製品
アプリイメージ生成手順

WebOTX Application Serverコンテナ製品

◆ WebOTX Application Server

拡張性に優れた業務システム構築を実現するアプリケーションサーバです。

コストパフォーマンスに優れた小規模モデルからシステムの安定稼動を実現する高信頼な大規模モデルまで、長期に渡って安心してお使いいただける、企業システムの基盤インフラとして最適なミドルウェア製品です。

エンタープライズ・ソフトウェアの標準仕様「Jakarta EE」(認証取得予定)を核に、最新のJakarta EE 8に対応し高性能・高信頼なアプリケーションサーバとして進化を続けています。

本製品の詳細(マニュアル)は以下のリンクをご参照ください。

- https://docs.nec.co.jp/sites/default/files/webotx_manual_v111/WebOTX/111/html/asindex.html

◆ WebOTX Application Server for Container

上記製品のコンテナ版となる製品です。

本製品の詳細(マニュアル)は以下のリンクをご参照ください。

- https://docs.nec.co.jp/sites/default/files/webotx_manual_v111/WebOTX/111/html/install/linux/install_guide_linux.html

アプリイメージ生成手順

WebOTX Builder

◆ 概要

2022年9月にWebOTX Builder(11.10.00.00)をリリースしました。本Builderを使用することで、WebOTX Application Serverにアプリケーションが配備されたコンテナイメージを生成することができます。

WebOTX Builderのインストールドキュメント
https://www.support.nec.co.jp/View.aspx?id=9010110432

◆ 機能

WebOTX Builderは以下のBuildpack(機能)を内包しています。

Buildpack	機能
CA Certificates	CA証明書を適用する
JDK	JDK環境を構築する
Maven	Maven環境を構築し、アプリケーションをコンパイル・パッケージ化する
WebOTX	WebOTXフルプロファイル版(Jakarta EE 8対応)をインストールし、アプリケーションを配備する
WebOTX Uber JAR	WebOTXマイクロサービスプロファイル版(MicroProfile 4.1対応)とアプリケーションを配備する

アプリイメージ生成手順

アプリイメージ生成

1. サンプルアプリケーションフォルダに移動

```
$ git clone ${APP_GIT_URL}
$ cd application
```

2. アプリイメージ生成コマンドを実行

```
$ pack build app-image --builder webotx/full-profile-ubi8-builder:11.10.00.00 ¥
                       --env BP_JDK_URL=<JDK installer URL> ¥
                       --volume ./bindings:/platform/bindings
```

プロキシ環境内での実行

本コマンドでは、JDKやMavenのインストーラファイルを外部から(HTTPS通信で)ダウンロードする処理が行われます。
そのため、プロキシ環境内で実行する場合は、以下のパスにプロキシサーバのCA証明書(.crtまたは.pem)を格納しておく必要があります。
./bindings/certificates/

アプリイメージ生成手順

アプリコンテナ起動

3. 生成したアプリイメージからアプリコンテナ起動

本アプリイメージには365日間利用可能なお試しライセンスが登録されています。

```
$ docker run -p 8080:8080 app-image
```

しかし、本番利用する場合は正規ライセンスを登録する必要があります。

正規ライセンスを登録するには、起動時に環境変数OTX_LICENSEにWebOTX Application Server Expressのライセンスキーをカンマ(,)区切りで指定します。

```
$ docker run -p 8080:8080 --env OTX_LICENSE=<ライセンスキー> app-image
```

参考情報

参考情報
改版履歴

参考情報

Cloud Native Buildpacks

- https://buildpacks.io/: トップページ
- https://buildpacks.io/docs/tools/pack/: pack-cliインストールドキュメント

WebOTX Manual V11.1 | Cloud Native Buildpack対応

- https://docs.nec.co.jp/sites/default/files/webotx_manual_v111/WebOTX/111/html/use/container/cnb_container.html#container_image_create_cnb: コンテナイメージの生成(Cloud Native Buildpacksを使用する場合)

Docker

- https://docs.docker.com/engine/install/: インストールドキュメント

Jenkins

- https://www.jenkins.io/doc/book/installing/: インストールドキュメント

改版履歴

版数	発行日	改版内容
1.0	2022/12/6	新規作成


Orchestrating a brighter world
NEC