ODCS（オープンデータカタログサービス）パッケージ

導入マニュアル

目次

1. 概要

ODCS（オープンデータカタログサービス）パッケージ（以下、パッケージ）は、オープンデータの提供に必要な Web アプリケーションを統合した Web アプリケーションです。

本パッケージは、平成 28 年度「地方公共団体のオープンデータ取組推進に係る具体的施策の調査」において、調査内容の妥当性を確認するために開発されました。その後、BODIK（ビッグデータ＆オープンデータ・イニシアティブ九州）を中心に機能の改良、追加などを実施しています。

本「ODCS（オープンデータカタログサービス）パッケージ 導入マニュアル」（以下、導入マニュアル）では、本パッケージの構築手順について説明します。

1.1 パッケージの機能

パッケージは大きく以下の 3 つの Web アプリケーションから構成されます。

- カタログサイト（ckan）
- ポータルサイト（wordpress）
- ダッシュボード（地図上でのデータの可視化機能）

カタログサイト

オープンデータの蓄積や提供に対し、国内外で広く利用されているオープンソースソフトウェアである、ckan を使用しています。

ポータルサイト

ポータルサイトの機能に対し、ホームページの提供などで広く利用されているオープンソースソフトウェアである、wordpress を使用しています。

ダッシュボード

カタログサイト上に公開されたオープンデータを即座に地図上で視覚化するための機能です。

また、パッケージは環境構築の容易性や導入手順の簡略化を目的として、コンテナ型の仮想化技術である Docker（*1）によって構成されています。

パッケージは各アプリケーション間の結合度を下げることで、新規にオープンデータに取り組まれる地方公共団体には新規パッケージ全体の導入、既にオープンデータに取り組まれている(カタログサイトのみ導入済み)地方公共団体にはダッシュボードのみを導入することが可能な構成となっています。

1.2 パッケージの構成

パッケージの構成は以下の通りです。

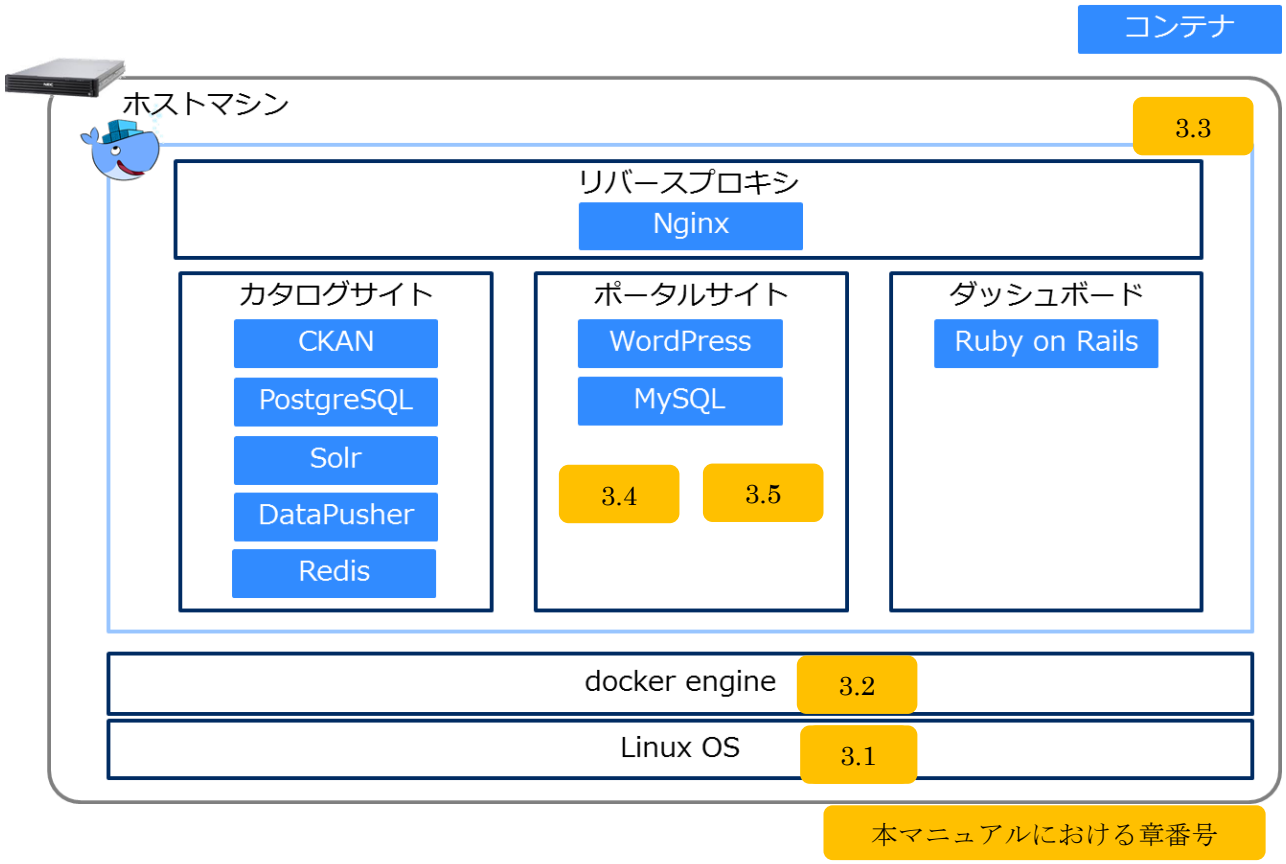


図 1　新規パッケージ構成

(*) Docker: https://www.docker.com/

1.3 実行環境

新規パッケージを構築する際の推奨サーバーは以下の通りです。

表 1　推奨サーバー要件

項目	新規パッケージ構築	ダッシュボードのみ構築
OS	CentOS 7.3	CentOS 7.3
CPU	4 コア	1 コア
RAM	4GB	2GB
HDD	100GB	50GB

導入マニュアルにて用いる構築スクリプトは、Linux カーネル 3.10 以上、Docker v17.3 以降の環境にて使用可能です。

新規パッケージの導入に際しては、外部のインターネットに接続できることを前提としています。

本導入マニュアルでは、上記環境に構築する際の手順を例に説明します。

※ インターネット外部接続が出来ない環境へのインストールにつきましては、4.1 項
「インターネット外部接続が出来ない環境へのインストールについて」を参照し、参考にして下さい。

2. 事前準備

新規パッケージを構築する前に以下の 7 点が必要になります。

1. サイト公開用のドメイン
2. ドメインに対する SSL 証明書の取得
3. Google Analytics トラッキング ID の取得
4. Google Maps API キーの取得
5. SMTP メールサーバー
6. ファイアウォールのポート設定
7. パラメータシートの作成

2.1 サイト公開用のドメイン

インターネットに公開するためのドメインについては、ckan、wordpress、map のそれぞれに対してドメインが必要になります。

ドメインの取得方法は取得機関によって異なるため、詳細手順は省略します。

以降の手順は、以下のドメインを取得している前提で記述しています。

表 2　取得するドメイン

機能	ドメイン(例)
Ckan	data-odpkg.example.co.jp
Wordpress	www-odpkg.example.co.jp
Map	map-odpkg.example.co.jp

2.2 ドメインに対する SSL 証明書の取得

https で公開するため、前述のドメインに対して証明書を発行する必要があります。

証明書の発行手順は取得機関によって異なるため、詳細手順は省略します。

本システムでは Nginx を採用しているため、以下のファイルが必要になります。

- .pem
- .key

取得したファイルはファイル名を、以下のファイル名に**リネーム**（変更）してください。また、以降の手順は、以下のファイルが準備できている前提で記述しています。

表 3　SSL 証明書

機能	ファイル名
ckan	data.pem data.key
wordpress	www.pem www.key
map	map.pem map.key

SSL 証明書の種類によっては変換が必要になります。

参考として、pfx 形式で発行された証明書を変換する手順を以下に記載します。 なお、本手順は構築で利用する CentOS 環境上で実施しても構いません。

【入力】

```
$ openssl pkcs12 -in input.pfx -nodes -out output.pem
$ openssl pkcs12 -in input.pfx -nocerts -nodes -out output.key
```

中間証明書のマージ

PEM 形式で使用する SSL 証明書ファイルでは、サーバー証明書と中間証明書をマージしたファイルを使用します。

通常、中間証明書は、サーバー証明書の発行機関のウエブサイトで提供されています。中間証明書をダウンロードし、cat コマンドなどで、上記２つのファイルをマージしたファイルを作成します。

2.3 Google Analytics トラッキング ID の取得

Google Analytics トラッキング ID の取得方法については、別紙「GoogleAnalytics、GoogleMapsAPI キー取得方法」をご確認ください。

以降の手順は、以下の Google Analytics トラッキング ID を取得している前提で記述しています。

表 4 Google Analytics トラッキング ID

機能	Google Analytics トラッキング ID (例)
ckan	UA-00000000-1
wordpress	UA-00000000-2
map	UA-00000000-3

2.4 Google Maps API キーの取得

Google Maps API キーの取得方法については、別紙「GoogleAnalytics、GoogleMapsAPI キー取得方法」をご確認ください。

以降の手順は、以下の Google Maps API キーを取得済という前提で記述しています。

表 5 Google Maps API キー

機能	Google Maps API キー（例）
map	dummygooglemapkey

2.5 SMTP メールサーバー

本システムからメール通知する際の SMTP メールサーバーを用意します。

本システムからアクセス可能であれば内部または外部のメールサーバーどちらでも構いません。

以降の手順は、以下のメールサーバーを準備している前提で記述しています。

表 6　SMTP メールサーバー

設定項目	設定値（例）	説明
SMTP_HOST	smtp.example.com	SMTP メールサーバー
SMTP_PORT	587	ポート番号
SMTP_AUTH	true	SMTP 認証の有効化
SMTP_SSL	Tls	暗号化方式
SMTP_STARTTLS	true	TLS 通信の有効化
SMTP_USER	example-user	認証用のユーザ
SMTP_PASS	example-pass	認証用のパスワード
MAIL_FROM	odpkg@example.com	送信元アドレス
ERROR_EMAIL	odpkg-admin@example.com	エラーメール送信先アドレス

2.6 ファイアウォールのポート設定

新規パッケージを導入するサーバーが以下の通信が可能になるよう、ファイアウォールを設定してください。

表 7　ファイアウォール設定

設定内容	設定項目	設定
インターネット(外部) ⇒ サーバー	ポート 80/tcp(http) ポート 443/tcp(https)	許可
イントラネット(内部) ⇒ サーバー	ポート 22/tcp(ssh)	許可
サーバー⇒ インターネット(外部)	ポート 80/tcp(http) ポート 443/tcp(https)	許可
サーバー⇒ メールサーバー	ポート 587/tcp(smtp)	許可
上記以外	許可	許可

2.7 パラメータシートの作成

本パッケージで使用するパラメーター一覧を用意しています。別紙「パラメータシート」をご確認ください。変更必須項目がすべて入力済みのものをご準備ください。

3. 構築手順

ここではパッケージの構築を行います。

3.1 OS 設定

タイムゾーンおよびロケール設定

タイムゾーンを設定します。
コマンドプロンプト ＄ に続けて以下のように入力しリターンキーを押して下さい。
【入力】

```
$ sudo timedatectl set-timezone Asia/Tokyo
```

このコマンド投入による画面の変化なく、コマンドプロンプト ＄ が表示されます。
ロケールを設定します。
コマンドプロンプト ＄ に続けて以下のように入力しリターンキーを押して下さい。
【入力】

```
$ sudo localectl set-locale LANG=ja_JP.utf8
```

コマンドプロンプト ＄ が表示されたら終了です。
インストール時に既に OS が日本設定になっている場合は、上記手順は不要です。

SELinux 無効化

SELinux の設定ファイル/etc/sysconfig/selinux を以下のように編集します。
(SELINUX=enforcing を SELinux=disabled に書き換えています)
【ファイル】

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of three two values:
#       targeted - Targeted processes are protected,
#       minimum - Modification of targeted policy. Only selected processes are protected.
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

ファイルを保存した後、反映するために OS の再起動が必要になります。

コマンドプロンプト $ に続け、以下のように入力しリターンキーを押すと OS が再起動されます。

【入力】

```
$ sudo reboot
```

OS ファイアウォール設定

OS が再起動されたら、本システムで必要なポート以外はアクセスできないようにするため、OS のファイアウォールの設定を行います。

コマンドプロンプト $ に続けて以下のように入力しリターンキーを押して下さい。

【入力】

```
$ sudo yum install firewalld
```

【出力】

```
読み込んだプラグイン:fastestmirror, langpacks
base                                                    | 3.6 kB  00:00:00
extras                                                  | 3.4 kB  00:00:00
openlogic                                               | 2.9 kB  00:00:00
updates                                                 | 3.4 kB  00:00:00
(1/5):base/7/x86_64/group_gz                            | 155 kB  00:00:00
(2/5):openlogic/7/x86_64/primary_db                     |  18 kB  00:00:00
(3/5):extras/7/x86_64/primary_db                        | 139 kB  00:00:00
(4/5):updates/7/x86_64/primary_db                       | 3.8 MB  00:00:00
(5/5):base/7/x86_64/primary_db                          | 5.6 MB  00:00:00
Determining fastest mirrors
依存性の解決をしています


[省略]
```

コンソールに以下のように「 Is this ok [y/d/N]: 」と表示されたら「ｙ」を入力しリターンキーを押して下さい。

【入力】

```
Is this ok [y/d/N]:y
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
[省略]


完了しました!
```

コマンドプロンプト $ に続けて以下のように入力しリターンキーを押して下さい。

【入力】

```
$ sudo systemctl enable firewalld
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
Created symlink from /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service to /usr/lib/systemd/system/firewalld.service.

Created symlink from /etc/systemd/system/basic.target.wants/firewalld.service to /usr/lib/systemd/system/firewalld.service.
```

コマンドプロンプト　$　に続けて以下のように入力しリターンキーを押して下さい。

【入力】

```
$ sudo systemctl start firewalld
$ sudo firewall-cmd --add-port 80/tcp –permanent
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
success
```

コマンドプロンプト　$　に続けて以下のように入力しリターンキーを押して下さい。

【入力】

```
$ sudo firewall-cmd --add-port 443/tcp --permanent
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
success
```

コマンドプロンプト　$　に続けて以下のように入力しリターンキーを押して下さい。

【入力】

```
$ sudo firewall-cmd –reload
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
success
```

コマンドプロンプト　$　に続けて以下のように入力しリターンキーを押して下さい。

【入力】

```
$ sudo firewall-cmd --list-all
```

上記コマンド投入後、コンソールに以下のように表示されれば OS の設定は完了です。

【出力】

```
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpv6-client ssh
  ports: 443/tcp 80/tcp
  protocols:
  masquerade: no
  forward-ports:
  sourceports:
  icmp-blocks:
  rich rules:
```

3.2 Docker のインストール

新規パッケージは、各コンポーネントを Docker コンテナとして構成しています。
新規パッケージの導入には、この Docker のインストールが必要となります。
以下に Docker のインストール手順ついて説明します。

まず Docker の実行に必要となる Linux ユーティリティモジュールを インターネット上の Linux パッケージ管理サイトからダウンロードします。
モジュールダウンロードは、コンソールから yum（ヤム）コマンドの投入により行われます。
モジュールダウンロードに先立ち、yum のインストールが必要となります。
コンソールのコマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。
【入力】

```
$ sudo yum install -y yum-utils
```

コンソールに以下のように出力されたら、次へ進みます。
【出力】

```
読み込んだプラグイン:fastestmirror, langpacks
Loading mirror speeds from cached hostfile
パッケージ yum-utils-1.1.31-40.el7.noarch はインストール済みか最新バージョンです
何もしません
```

モジュールダウンロードは、コンソールから yum コマンド投入により行われます。
コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。
【入力】

```
$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
読み込んだプラグイン:fastestmirror, langpacks
adding repo from: https://download.docker.com/linux/centos/docker-ce.repo
grabbing file https://download.docker.com/linux/centos/docker-ce.repo to /etc/yum.repos.d/docker-ce.repo
repo saved to /etc/yum.repos.d/docker-ce.repo
```

続けて、コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo yum install docker-ce
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
読み込んだプラグイン:fastestmirror, langpacks
docker-ce-stable                                          | 2.9 kB  00:00:00
docker-ce-stable/x86_64/primary_db                        | 3.5 kB  00:00:00
Loading mirror speeds from cached hostfile
依存性の解決をしています


[省略]
```

コンソールに以下のように「 Is this ok [y/d/N]: 」と表示されたら「ｙ」を入力しリターンキーを押して下さい。

【入力】

```
Is this ok [y/d/N]:y
```

【出力】

```
[省略]
```

コンソールに以下のように「上記の処理を行います。よろしいでしょうか？ [y/N] 」と表示されたら、「ｙ」を入力しリターンキーを押して下さい。

【入力】

```
上記の処理を行います。よろしいでしょうか？ [y/N]y
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
[省略]
完了しました!
```

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo systemctl enable docker
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
```

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo systemctl start docker
```

コマンドプロンプト $ が出力されたら、次へ進みます。

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

Docker が正常にインストールされたかの確認を行います。

【入力】

```
$ docker version
```

コンソールに、以下のように正常に表示されれば正常にインストールされています。

(バージョンはインストールを行う時期によって異なる場合があります)

【出力】

```
Client:
 Version:      17.03.0-ce
 API version:  1.26
 Go version:   go1.7.5
 Git commit:   3a232c8
 Built:        Tue Feb 28 08:10:07 2017
 OS/Arch:      linux/amd64

Server:
 Version:      17.03.0-ce
 API version:  1.26 (minimum version 1.12)
 Go version:   go1.7.5
 Git commit:   3a232c8
 Built:        Tue Feb 28 08:10:07 2017
 OS/Arch:      linux/amd64
 Experimental: false
```

続けて、現在ログインしているユーザが docker コマンドを使用できるように、ユーザを docker グループに追加します。

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo gpasswd -a `whoami` docker
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
ユーザ centos7 をグループ docker に追加
```

その後、再接続して以降の手順に進んでください。

【入力】

```
$ sudo systemctl restart docker
$ exit
```

docker-compose インストール

続けて、複数の docker コンテナ管理機能、docker-compose をインストールします。

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo sh -c 'curl -L https://github.com/docker/compose/releases/download/1.11.2/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose'
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   600    0   600    0     0    678      0 --:--:-- --:--:-- --:--:--   678
100 8066k  100 8066k    0     0   1832k      0  0:00:04  0:00:04 --:--:-- 2804k
```

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

コンソールにコマンドプロンプト $ が表示されたら、続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ docker-compose version
```

コンソールに、以下のように正常に表示されれば Docker のインストールは完了です。

【出力】

```
docker-compose version 1.11.2, build dfed245
docker-py version: 2.1.0
CPython version: 2.7.13
OpenSSL version: OpenSSL 1.0.1t  3 May 2016
```

3.3 パッケージ構築

構築にあたっては、新規パッケージのほかに、ckan、datapusher のパッケージを個別に導入する必要があります。

以下に各パッケージの導入手順を示します。

ソースコード取得

本パッケージで使用するファイル群は GitHub 上にアップロードされています。

ファイル群を取得するために git コマンドを使用します。

OS に git をインストールします。

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押して下さい。

【入力】

```
$ sudo yum install git
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
読み込んだプラグイン:fastestmirror, langpacks
Loading mirror speeds from cached hostfile
依存性の解決をしています


[省略]
```

コンソールに以下のように「 Is this ok [y/d/N]: 」と表示されたら「ｙ」を入力しリターンキーを押して下さい。

【入力】

```
Is this ok [y/d/N]:y
```

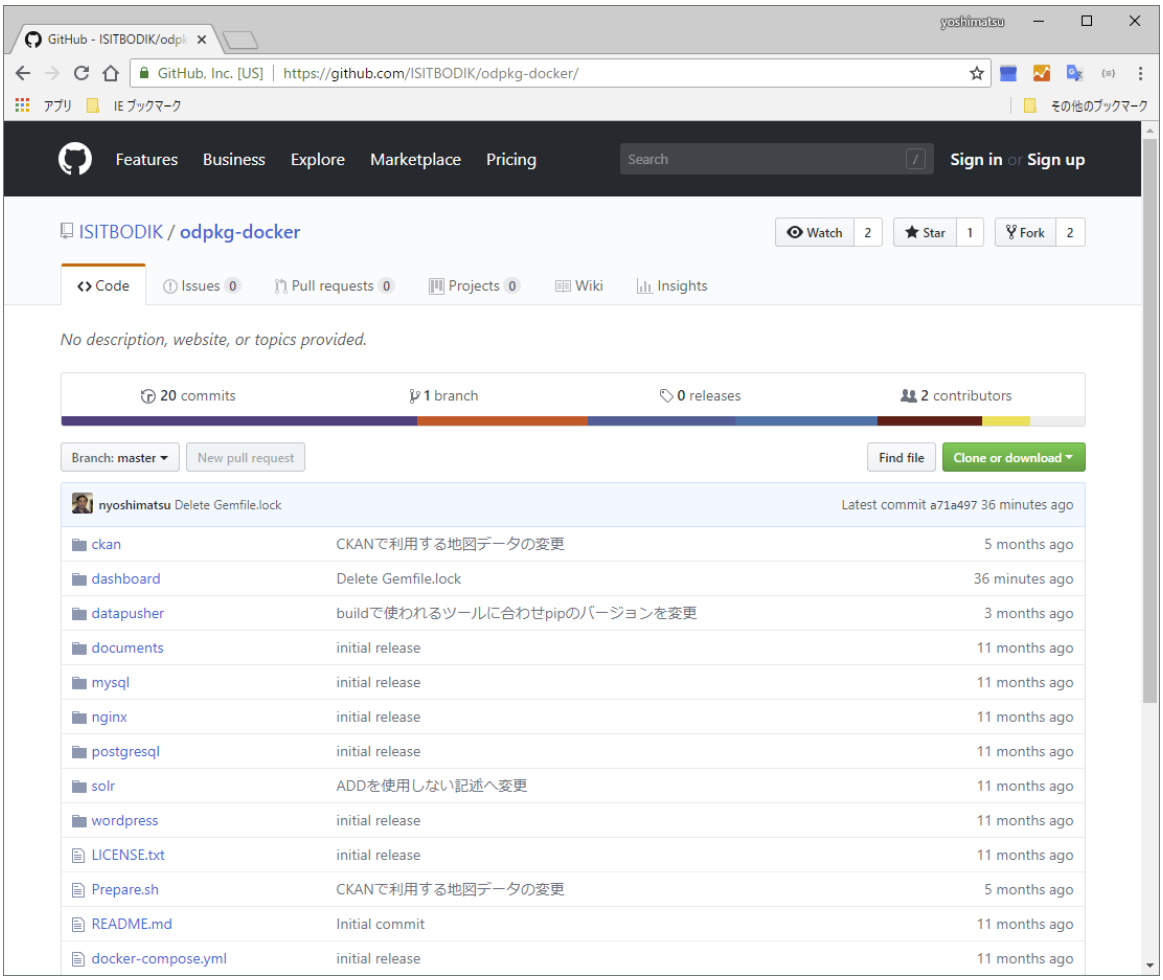
コンソールに以下のように出力されたら、次へ進みます。

【出力】

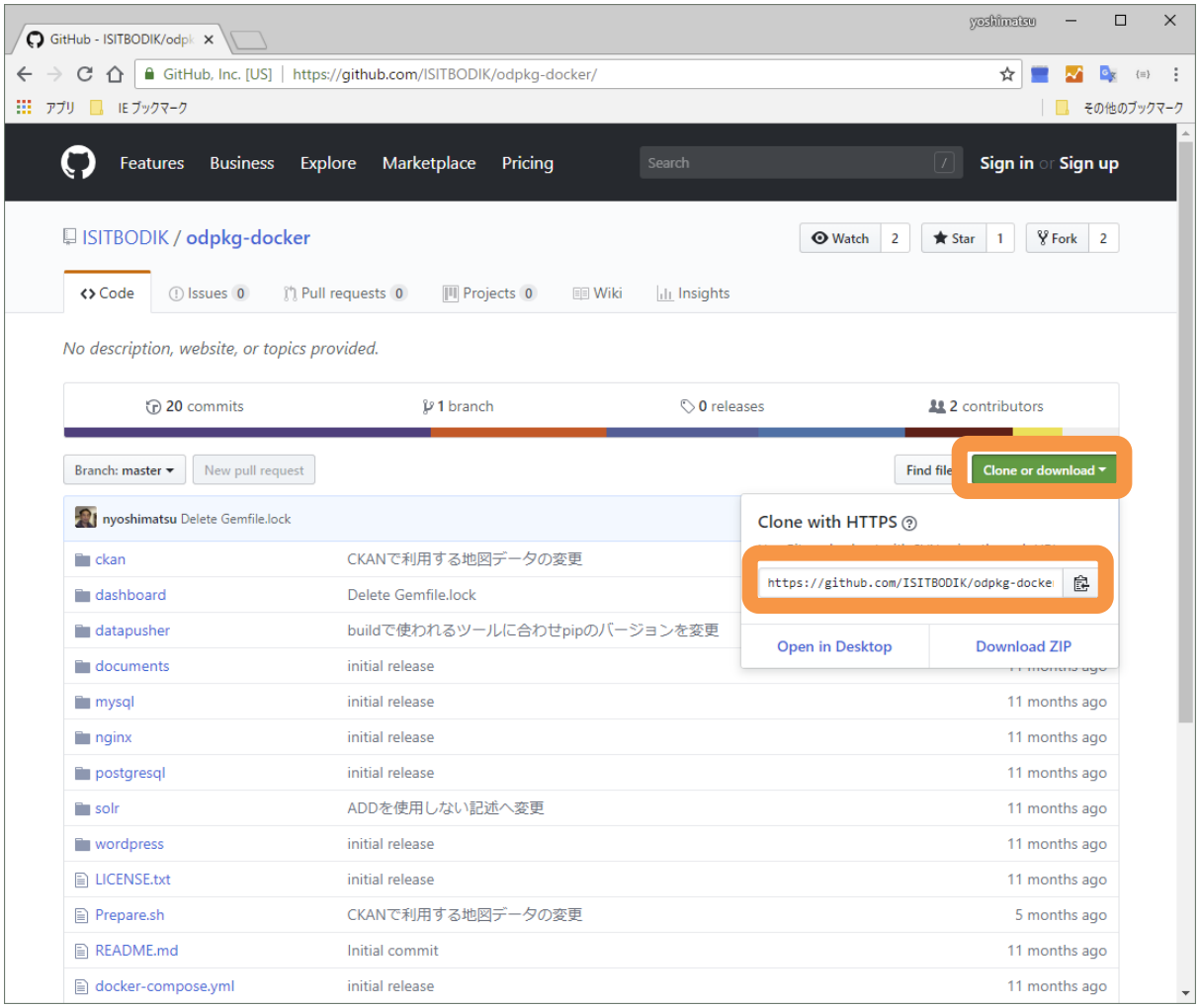
```
 [省略]
完了しました!
```

新規パッケージを公開している Web サイトをブラウザで開きます。以下の URL にアクセスしてください。

https://github.com/ISITBODIK/odpkg-docker/



「Clone or download」をクリックし、表示される URL を控えてください。



ファイル群をダウンロードするために git コマンドを入力します。
git コマンドの引数に控えておいた URL を使用します。

【入力】

```
$ cd ~
$ git clone https://github.com/bodik /～～～～～～～～
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
$ git clone https://github.com/ISITBODIK/odpkg-docker.git
Cloning into 'odpkg-docker'...
remote: Counting objects: 468, done.
remote: Total 468 (delta 0), reused 0 (delta 0), pack-reused 468
Receiving objects: 100% (468/468), 20.34 MiB | 4.58 MiB/s, done.
Resolving deltas: 100% (110/110), done.
```

ファイル群ダウンロードのフォルダ体系は以下となります。

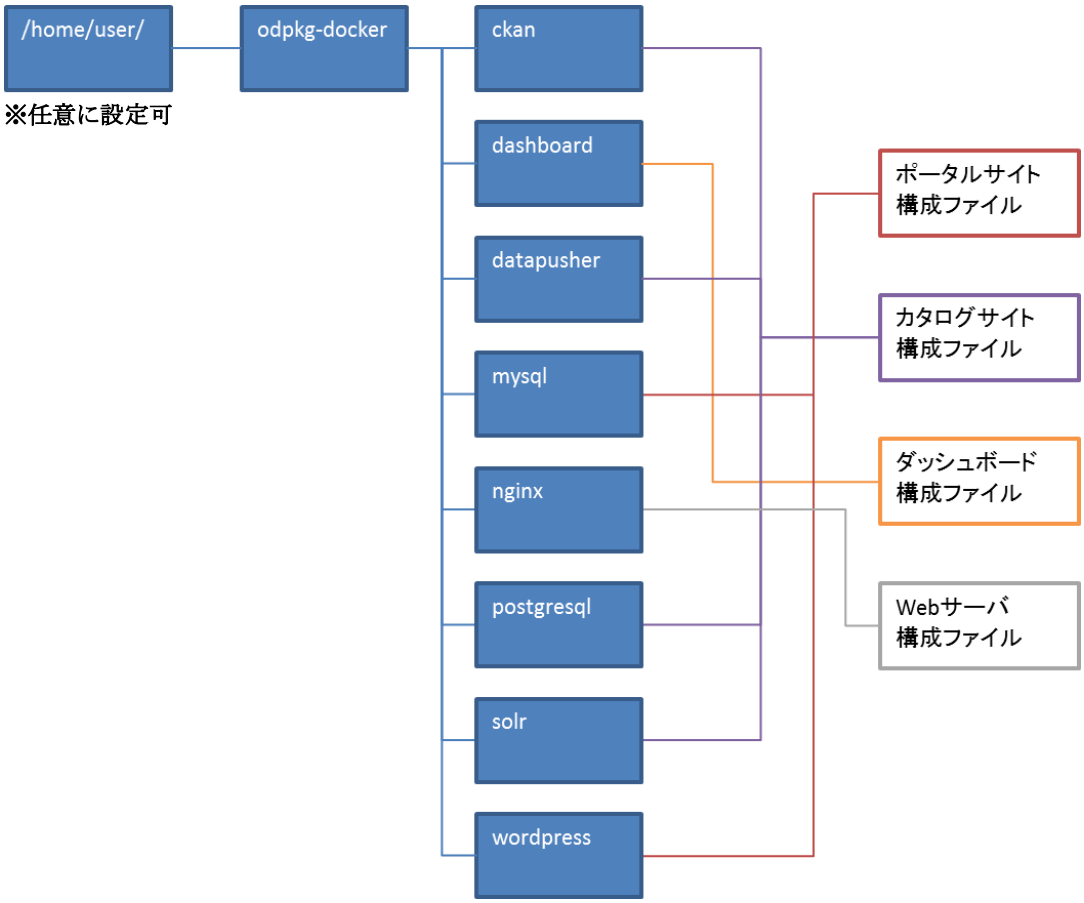


図 2　新規パッケージフォルダ体系

SSL 証明書配置

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押し、ディレクトリを作成します。

【入力】

```
$ mkdir ~/odpkg-docker/nginx/ssl
```

このディレクトリ配下に、事前準備にて作成した証明書をコピーし配置します。

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押します。

【入力】

```
$ ls –l ~/odpkg-docker/nginx/ssl
```

以下のように各ドメインの SSL 証明書が配置できていれば完了です。

※ファイル名は固定です。

【出力】

```
data.key
data.pem
map.key
map.pem
www.key
www.pem
```

各種ソースコード取得

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押しディレクトリを移動します。

【入力】

```
$ cd ~/odpkg-docker/ckan
```

コマンドプロンプト $ に続けて以下のように入力し、ckan のソースコードをダウンロードします。

【入力】

```
$ git clone -b ckan-2.5.3 https://github.com/ckan/ckan.git
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
Cloning into 'ckan'...
remote: Counting objects: 153109, done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 153109 (delta 23), reused 0 (delta 0), pack-reused 153046
Receiving objects: 100% (153109/153109), 123.40 MiB | 26.09 MiB/s, done.
```

```
Resolving deltas: 100% (111312/111312), done.
Note: checking out '0dfa2df5ea401ab8bf1e40668fb571304c7c32db'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:
git checkout -b new_branch_name
```

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押しディレクトリを移動します。

【入力】

```
$ cd ~/odpkg-docker/datapusher
```

コマンドプロンプト $ に続けて以下のように入力し、datapusher のソースコードをダウンロードします。

【入力】

```
$ git clone https://github.com/ckan/datapusher.git
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
Cloning into 'datapusher'...
remote: Counting objects: 1189, done.
remote: Total 1189 (delta 0), reused 0 (delta 0), pack-reused 1189
Receiving objects: 100% (1189/1189), 487.32 KiB | 0 bytes/s, done.
Resolving deltas: 100% (645/645), done.
```

コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押しディレクトリを移動します。

【入力】

```
$ cd datapusher
```

コマンドプロンプト $ に続けて以下のように入力し、datapusher のソースコードをダウンロードします。

【入力】

```
$ git checkout 96ae744f68f97aa2ff778eeb017af432986fdc00
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
Note: checking out '96ae744f68f97aa2ff778eeb017af432986fdc00'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 96ae744... Merge pull request #109 from opendatazurich/add-ssl-verification-option
```

パラメータ設定

~/odpkg-docker/odpkg.env を編集し、以下のパラメータを設定します。事前準備で作成した、パラメータシートと同様の値にします。

パラメータの編集は、メモ帳などのテキストエディタで可能です。

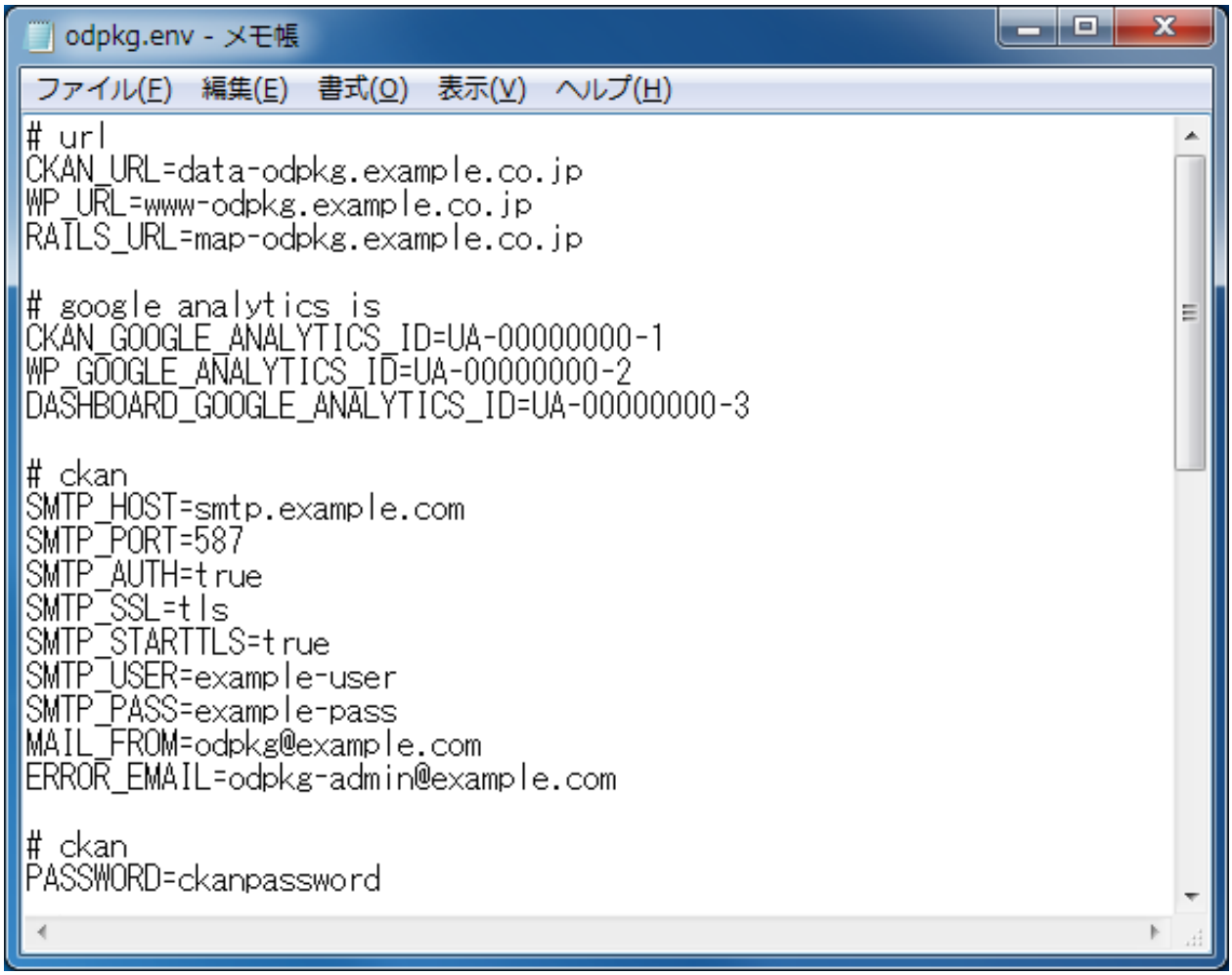


表 8　odpkg.env のパラメータ

カテゴリ	設定項目	デフォルト値	説明	変更必須
ドメイン	CKAN_URL	data-odpkg.example.co.jp	ckan サイトの URL	○
ドメイン	WP_URL	www-odpkg.example.co.jp	wordpress サイトの URL	○
ドメイン	RAILS_URL	map-odpkg.example.co.jp	map サイトの URL	○
サイト解析	WP_GOOGLE_ANALYTICS_ID	UA-00000000-1	wordpress サイト用の GoogleAnalyticsID	○
サイト解析	CKAN_GOOGLE_ANALYTICS_ID	UA-00000000-2	ckan サイト用の GoogleAnalyticsID	○
サイト解析	DASHBOARD_GOOGLE_ANALYTICS_ID	UA-00000000-3	map サイト用の GoogleAnalyticsID	○
メール	SMTP_HOST	smtp.example.com	SMTP メールサーバー	○
メール	SMTP_PORT	587	ポート番号	○
メール	SMTP_AUTH	true	SMTP 認証の有効化	○
メール	SMTP_SSL	tls	暗号化方式	○
メール	SMTP_STARTTLS	true	TLS 通信の有効化	○
メール	SMTP_USER	example-user	認証用のユーザ	○
メール	SMTP_PASS	example-pass	認証用のパスワード	○
メール	MAIL_FROM	odpkg@example.com	送信元アドレス	○
メール	ERROR_EMAIL	odpkg-admin@example.com	エラーメール送信先アドレス	○
ckan	PASSWORD	ckanpassword	ckan の管理者パスワード	○
ckan	POSTGRESQL_DB	ckan_default	ckan データベース名	
ckan	POSTGRESQL_USER	ckan_default	ckan データベースのユーザ名	
ckan	POSTGRESQL_PASS	ckan_default	ckan データベースのユーザパスワード	○
ckan	POSTGRESQL_DATASTORE_DB	datastore_default	ckan(datastore)データベース名	
ckan	POSTGRESQL_DATASTORE_USER	datastore_default	ckan(datastore)データベースのユーザ名	
ckan	POSTGRESQL_DATASTORE_PASS	datastore_default	ckan(datastore)データベースのユーザパスワード	○
wordpress	WORDPRESS_DB_HOST	mysql:3306	WordPress データベース接続先	
wordpress	WORDPRESS_DB_USER	odpkg	WordPress データベースのユーザ名	
wordperss	WORDPRESS_DB_PASSWORD	odpkg	WordPress データベースのユーザパスワードユーザパスワード	○
wordpress	MYSQL_ROOT_PASSWORD	rootpass	MySQL の root パスワード	○
wordpress	MYSQL_DATABASE	wordpress	WordPress 用データベース名	
wordpress	MYSQL_USER	odpkg	WordPress データベースのユーザ名（WORDPRESS_DB_USER と同様の値）ユーザ名	
wordpress	MYSQL_PASSWORD	odpkg	WordPress データベースのユーザパスワード（WORDPRESS_DB_PASSWORD と同様の値）	○
map	SECRET_KEY_BASE	dummysecretkeybase	Rails の SECRETKEY(任意の 32~64 文字の半角英数字)	○
map	GOOGLE_MAP_KEY	dummygooglemapkey	map サイトの Google Maps API キー	○
サイト	MAIN_COLOR	0097E0	各サイトのメインカラー	
サイト	SUB_COLOR	58C5F9	各サイトのサブカラー	

パラメータを反映させるために、コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押しディレクトリを移動します。

【入力】

```
$ cd ~/odpkg-doker
```

コマンドプロンプト $ に続けて以下のように入力し、パラメータを反映させます。

【入力】

```
$ sh Prepare.sh
```

コンソールに以下のように出力されたら、次へ進みます。

【出力】

```
setting production.yml
setting custom_option.ini
setting wordpress.sql
setting colors
create mount dir
change permission wp-content/
change permission ckan/data
```

パッケージ起動

パッケージを起動させるために、コマンドプロンプト $ に続けて以下のように入力し、リターンキーを押しディレクトリを移動します。

【入力】

```
$ cd ~/odpkg-doker
```

コマンドプロンプト $ に続けて以下のように入力し、パッケージを起動します。

【入力】

```
$ docker-compose up -d
```

初回起動時のみイメージのビルドを行うため、３０分程度時間を要します。(インターネットの環境次第で多少前後します)

以下のように表示されれば、コンテナの起動が完了です。

【出力】

```
Creating odpkgdocker_db_1
Creating odpkgdocker_solr_1
Creating odpkgdocker_redis_1
Creating odpkgdocker_mysql_1
Creating odpkgdocker_dashboard_1
```

```
Creating odpkgdocker_datapusher_1
Creating odpkgdocker_ckan_1
Creating odpkgdocker_wordpress_1
Creating odpkgdocker_nginx_1
```

コンテナの起動確認

コマンドプロンプト ＄ に続けて以下のように入力し、コンテナが正常に起動していることを確認します。

【入力】

```
$ docker ps
```

以下のように、STATUS”が”Up”と表示されていればコンテナが正常起動しています。

【出力】

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
ba38e53cdb02	odpkgdocker_nginx	"/bin/bash /entryp..."	49 seconds ago	Up 48 seconds	0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp	odpkgdocker_nginx_1
ca8fa165aec7	odpkgdocker_wordpress	"docker-entrypoint..."	54 seconds ago	Up 49 seconds	80/tcp	odpkgdocker_wordpress_1
0a4890b9ff70	odpkgdocker_ckan	"/sbin/my_init"	54 seconds ago	Up 45 seconds	80/tcp	odpkgdocker_ckan_1
2e4e44628e04	odpkgdocker_datapusher	"python datapusher..."	56 seconds ago	Up 54 seconds	8800/tcp	odpkgdocker_datapusher_1
5d64be52bcba	odpkgdocker_dashboard	"bundle exec rails..."	56 seconds ago	Up 55 seconds	3000/tcp	odpkgdocker_dashboard_1
dbbad67fb80c	odpkgdocker_mysql	"docker-entrypoint..."	56 seconds ago	Up 54 seconds	3306/tcp	odpkgdocker_mysql_1
fa07dca57f72	redis	"docker-entrypoint..."	56 seconds ago	Up 54 seconds	6379/tcp	odpkgdocker_redis_1
7ff07283102d	odpkgdocker_db	"/usr/local/bin/run"	56 seconds ago	Up 54 seconds	5432/tcp	odpkgdocker_db_1
00719d73e890	odpkgdocker_solr	"java -jar start.jar"	56 seconds ago	Up 54 seconds	8983/tcp	odpkgdocker_solr_1

3.4 WordPress の更新

WordPress を最新版に更新します。

[WordPress のアドレス]/wp-admin/ にアクセスし、WordPress にログインします。
WordPress の管理ユーザ情報は以下の通りです。

- ユーザ名　　：default
- パスワード　：P@ssw0rd

更新アイコンをクリックし、WordPress、プラグイン、テーマ、翻訳、それぞれ更新があった場合は、更新ボタン　　を押し、新しいバージョンに更新します。

3.5 WordPress 管理ユーザの更新

本パッケージ構築時、WordPress の管理ユーザ情報は以下の既定値となっている為、管理ユーザのパスワード変更を推奨します。

- ユーザ名　　: default
- パスワード　: P@ssw0rd

パスワードの変更方法については、別紙「新規パッケージ管理者ガイド」の「WordPress ログイン」をご確認ください。

以上で、パッケージの構築作業は完了です。

4 留意事項

4.1 インターネット外部接続が出来ない環境へのインストールについて

インターネット外部接続が出来ない環境へインストールを行う場合

パッケージのインストーラーは、活用している Docker、WordPress 等の標準的ソフトウェアについて導入時点での最新パッチを適用することを目的に、インストール時に必要なモジュール等についてインターネットを経由して外部から取り込むように構成されています。

新規パッケージを導入される地方公共団体様によってはセキュリティポリシーにより、インストール時の外部アクセスが制限されていたり、外部アクセスは可能でもポート番号に制限があり、インストーラーが求めるモジュールの取り込みが出来ない場合があります。
そうした場合には、以下の手順にて、インストールを行うことになります。

① クラウドサービス等を活用し、インターネット接続可能なサーバー環境を確保する。
② 本導入マニュアルを用い、①で確保したサーバー環境に新規パッケージのインストールを行う。
③ ②でのインストール作業が完了した直後、その環境を Docker の機能を用いてイメージファイルとしてエクスポート(外部保存)する。
④ 本来インストールを行うサーバー環境に、CentOS、および Docker のインストールを行う。
   - Docker のインストールに関しては次項を参照
⑤ 本来インストールを行うサーバー環境に、③でエクスポートしたイメージファイルを用いて、Docker のインポート機能を用い②の構築イメージを移行する。
   - Docker のエクスポート/インポートについては次々項を参照

Docker のインストールに関して

上記手順において、④の本来インストールを行うサーバー環境への Docker インストールには、必要モジュール等をあらかじめ外部インターネットからダウンロードして、DVD 等の媒体にコピーし、インストールする環境に持ち込み対応を行うことになります。

Docker インストールには、Docker 本体モジュールの他、様々な Linux ライブラリモジュールを必要となります。
Docker が必要とする Linux ライブラリモジュールは、CentOS のインストール時にあわせてインストールされるものもあれば、インターネット上にしか存在しないものもあります。
これは、CentOS をどのようにインストールしたかにより変わってきます。
ちなみに、最低限の構成で CentOS をインストールした際に必要になる Linux ライブラリモジュールが格納されている rpm 形式パッケージファイルは以下のものになります。

- audit-2.6.5-3.el7_3.1.x86_64.rpm
- audit-libs-2.6.5-3.el7_3.1.x86_64.rpm

- audit-libs-python-2.6.5-3.el7_3.1.x86_64.rpm
- checkpolicy-2.5-4.el7.x86_64.rpm
- device-mapper-1.02.135-1.el7_3.3.x86_64.rpm
- device-mapper-event-1.02.135-1.el7_3.3.x86_64.rpm
- device-mapper-event-libs-1.02.135-1.el7_3.3.x86_64.rpm
- device-mapper-libs-1.02.135-1.el7_3.3.x86_64.rpm
- device-mapper-persistent-data-0.6.3-1.el7.x86_64.rpm
- docker-ce-17.03.0.ce-1.el7.centos.x86_64.rpm
- docker-ce-selinux-17.03.0.ce-1.el7.centos.noarch.rpm
- dracut-033-463.el7.x86_64.rpm
- dracut-config-rescue-033-463.el7.x86_64.rpm
- dracut-network-033-463.el7.x86_64.rpm
- glib2-2.46.2-4.el7.x86_64.rpm
- initscripts-9.49.37-1.el7.x86_64.rpm
- kmod-20-9.el7.x86_64.rpm
- libaio-0.3.109-13.el7.x86_64.rpm
- libcgroup-0.41-11.el7.x86_64.rpm
- libgudev1-219-30.el7_3.7.x86_64.rpm
- libseccomp-2.3.1-2.el7.x86_64.rpm
- libselinux-2.5-6.el7.x86_64.rpm
- libselinux-python-2.5-6.el7.x86_64.rpm
- libselinux-utils-2.5-6.el7.x86_64.rpm
- libsemanage-2.5-5.1.el7_3.x86_64.rpm
- libsemanage-python-2.5-5.1.el7_3.x86_64.rpm
- libsepol-2.5-6.el7.x86_64.rpm
- libtool-ltdl-2.4.2-21.el7_2.x86_64.rpm
- lvm2-2.02.166-1.el7_3.3.x86_64.rpm
- lvm2-libs-2.02.166-1.el7_3.3.x86_64.rpm
- policycoreutils-2.5-11.el7_3.x86_64.rpm
- policycoreutils-python-2.5-11.el7_3.x86_64.rpm
- python-IPy-0.75-6.el7.noarch.rpm
- selinux-policy-3.13.1-102.el7_3.15.noarch.rpm
- selinux-policy-targeted-3.13.1-102.el7_3.15.noarch.rpm
- setools-libs-3.3.8-1.1.el7.x86_64.rpm
- systemd-219-30.el7_3.7.x86_64.rpm
- systemd-libs-219-30.el7_3.7.x86_64.rpm
- systemd-sysv-219-30.el7_3.7.x86_64.rpm

Docker のエクスポート/インポートに関して

③のインストール作業完了直後の環境の Docker のイメージファイルとしてのエクスポート(外部保存)については、以下の操作を行って下さい。

インストールを行ったサーバーのコンソールから、以下のコマンドを投入することでイメージファイルが出力されます。

```
docker save イメージ名 > イメージファイル名
```

ここで、イメージ名とイメージファイル名は以下のようになります。

	イメージ名	イメージファイル名
1	odpkg-docker_ckan	ckan.tar
2	odpkg-docker_datapusher	datapusher.ter
3	odpkg-docker_solr	solr.tar
4	Redis	redis.tar
5	odpkg-docker_db	db.tar
6	odpkg-docker_wordpress	wordpress.tar
7	odpkg-docker_mysql	mysql.tar
8	odpkg-docker_dashboard	dashboard.tar
9	odpkg-docker_nginx	nginx.tar

上記 9 つの docker イメージを、各々 docker save コマンドでエクスポートして下さい。

※コマンド投入イメージ odpkg-cocker_ckan のエクスポート

```
docker save odpkg-docker_ckan > ckan.tar
```

⑤の本来インストールを行うサーバー環境へのインポートについては、以下の操作を行って下さい。

本来インストールを行うサーバーのコンソールから、以下のコマンドを投入することでイメー ジファイルがインポート(移行展開)されます。

```
docker load < イメージファイル名
```

ここでイメージファイル名は、上記表のイメージファイル名となります。
上記表 9 つのイメージファイルのインポートが完了すればインストール完了です。