



Red Hat
Enterprise Linux 8

コンテナに デーモン入れて .service 化

2019 年 9 月 27 日
レッドハット株式会社
森若 和雄



Red Hat

概要

- **Application Stream とコンテナ**
- **コンテナイメージを入手する または 作成する**
- **データを置くディレクトリを作成、コンテナに対応づける**
- **systemd の .service ファイルを作成**

Red Hat Enterprise Linux 8 の特徴 : Application Stream

一部ソフトウェアに複数バージョンを提供

- ほとんどのソフトウェアは従来どおり RHEL8 のライフサイクルに渡り維持
- DB, 言語処理系, Web サーバ等の一部パッケージに、RHEL 本体から独立したライフサイクルで複数バージョンを提供します
- 新しいバージョンのソフトウェアを利用でき、それらを前提にした ISV 製品などを活用できます

Ver x.3 を 3 年

Ver x.5 を 3 年

Ver x.7 を 3 年

RHEL 8 の 10 年ライフサイクル

Application Stream(AppStream)

実は RHEL7 までにも類似の仕組みがありました

RHEL 7 まで提供されていた Red Hat Software Collections および Red Hat Developer Toolset の後継です

同時に複数のバージョンをインストールできないコンポーネントもあります

同時に利用したいときはコンテナを利用して分離します

今回のテーマ

RHEL 8.0 同梱で独立したライフサイクルがあるもの

Application Stream	Retirement Date
authd 1.4.4	May 2021
container-tools 1	May 2021
dotnet 2.1	Aug 2021
git 2.18	May 2021
httpd 2.4	May 2024
Identity Management DL1	May 2024
mariadb 10.3	May 2023
maven 3.5	May 2022
mercurial 4.8	May 2022
mysql 8	Apr 2023
nginx 1.14	May 2021

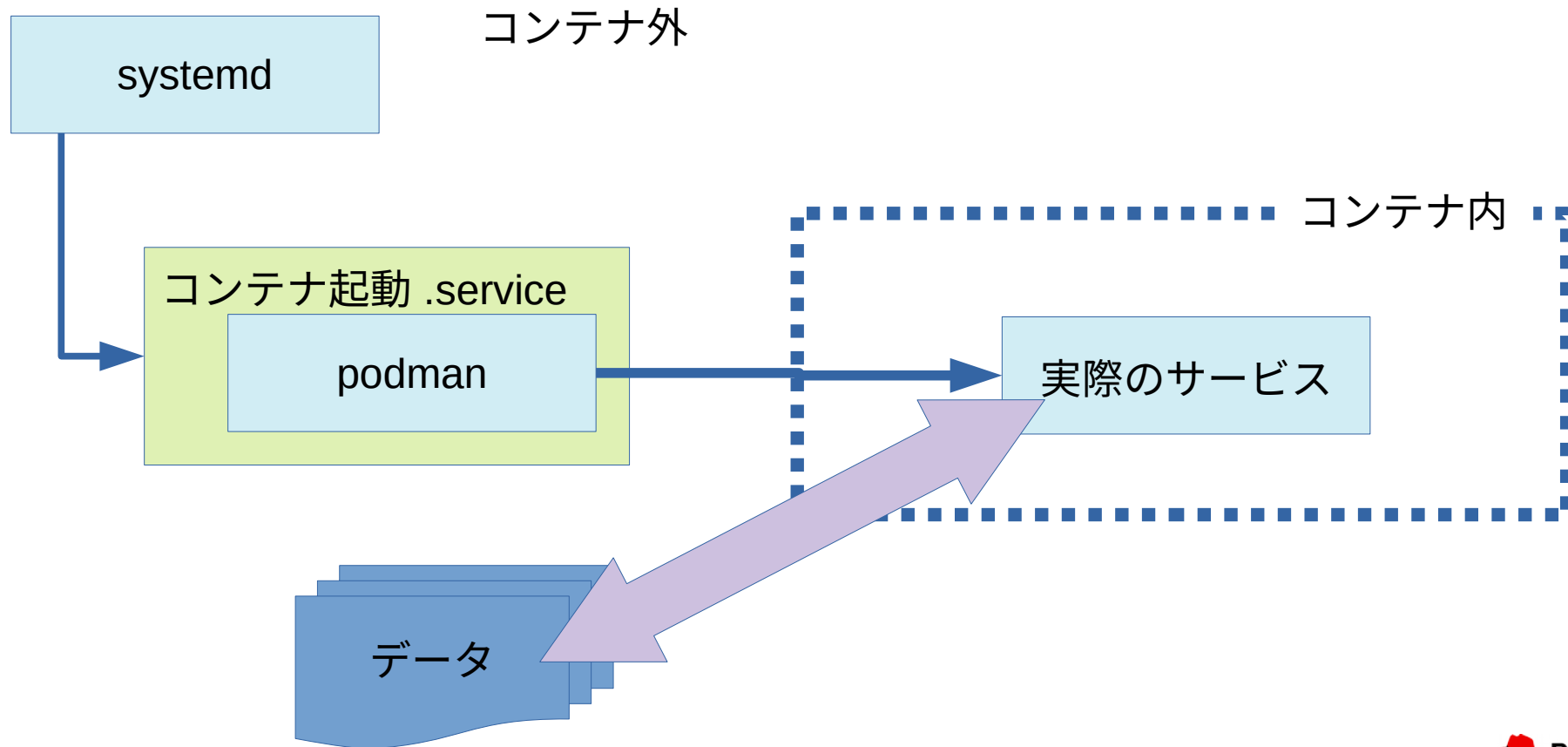
Application Stream	Retirement Date
openjdk 1.8.0	Jun 2023
openjdk 11	Oct 2024
perl 5.24	May 2021
php 7.2	May 2021
postgresql 10	May 2024
postgresql 9.6	Nov 2021
python 2.7	Jun 2024
redis 5	May 2022
ruby 2.5	Feb 2021
scala 2.1	May 2022
swig 3	May 2022

Hat

おおまかな方針

1. コンテナイメージを入手する または 作成する
2. データを置くディレクトリを作成、コンテナに対応づける
3. systemd の .service ファイルを作成し、自動的に起動する

おおまかな方針の図



コンテナイメージを
入手する
または
作成する

背景 : Universal Base Image(UBI)

課題 : コンテナイメージとしてソフトウェアを配布したい

ISV 製品などのコンテナイメージを作成する際に、RHEL のイメージをベースにすると自由に再配布できない。でも CentOS などを使うとサポートできない。

対策 : Red Hat Enterprise Linux の一部を抜きだして再配布可能な UBI として公開 (RHEL8 GA と同時)

3 種類のコンテナの base image と、ソフトウェア配布で必要になりそうな一部パッケージを UBI として公開しました。

UBI は制限なく変更・再配布でき、Red Hat のコンテナ基盤 (RHEL や OpenShift) 上で動作させるとサポートの対象になります。

UBI をベースにすることで ISV が自社製品を含むコンテナイメージを配布でき、サポートの問題もなくなります。

RHEL8 のコンテナイメージはどこにある？

Red Hat Container Catalog に登録されている

<https://access.redhat.com/containers/>

Product は「Red Hat Enterprise Linux」と「Universal Base Image」の2種類

RHEL 8 のコンテナベースイメージは「ubi8」

RHEL 8 のコンテナベースイメージは Universal Base Image (ubi) 8 です

探しても「rhel8」という名前のイメージはありません

AppStream で同時に複数バージョンをインストールできないコンポーネントのほとんどは ubi8/ または rhel8/ 以下のコンテナイメージとして提供されています

レジストリは registry.redhat.io

UBI は registry.access.redhat.com から入手できますが、
RHEL 特有のイメージは registry.redhat.io からのみ入手できます。

コンテナでの提供有無

Application Stream	container
authd 1.4.4	×
container-tools 1	×
dotnet 2.1	○
git 2.18	×
httpd 2.4	○
Identity Management DL1	×
mariadb 10.3	○
maven 3.5	×
mercurial 4.8	×
mysql 8	○
nginx 1.14	○
nodejs 10	○

Application Stream	container
openjdk 1.8.0	×
openjdk 11	×
perl 5.24	×
php 7.2	○
postgresql 10	○
postgresql 9.6	○
python 2.7	○
redis 5	○
ruby 2.5	○
scala 2.1	×
swig 3	×
varnish 6	○

RHEL8 のイメージを探して使い方を見る

検索や一覧はコマンドラインが便利

```
# podman login registry.redhat.io ← Red Hat Network のアカウントでログイン  
# podman search registry.redhat.io/ubi8  
# podman search registry.redhat.io/rhel8
```

利用方法は Red Hat Container Catalog を見るのが便利

気になるイメージのメタデータを skopeo で見ると、“url” ラベルに Red Hat Container Catalog の URL があります

```
# skopeo inspect docker://registry.redhat.io/rhel8/mariadb-103
```

```
"url": "https://access.redhat.com/containers/#/registry.access.redhat.com/rhel8/mariadb-103/images/1-37",
```

Red Hat Container Catalog

各コンテナイメージのページに
使い方や環境変数の説明がまとまっている

The screenshot shows the Red Hat Container Catalog search results for 'rhel8/mariadb-103 MariaDB 10.3'. The search bar at the top contains the text 'rhel8/mariadb-103 MariaDB 10.3'. Below the search bar, there is a table with columns: DATE PUSHED, IMAGE ADVISORY, SIGNING, SIZE, and DOCKER IMAGE ID. The table shows one result: 'a month ago', 'RHBA-2019:1985', 'Signed', '143.5 MB', and 'fclaf1bbcb7b'. Below the table, there are tabs for 'Security', 'Change Summary', 'Package List', and 'Dockerfile'. The 'Security' tab is selected, showing a 'Health Index' section with a green bar and a message: 'This image does not have any unapplied Critical or Important security updates. The Container Health Index analysis is based on RPM packages signed and created by Red Hat, and does not grade other software that may be included in a container image.'

The screenshot shows the detailed page for 'rhel8/mariadb-103 MariaDB 10.3'. The page has a navigation bar with tabs: 'Overview', 'Get This Image', 'Tech Details', 'Support', and 'Tags'. The 'Overview' tab is selected. The 'Description' section states: 'This container image provides a containerized packaging of the MariaDB mysqld daemon and client application. The mysqld server daemon accepts connections from clients and provides access to content from MySQL databases on behalf of the clients. You can find more information on the MariaDB project from the project Web site (<https://mariadb.org/>).' The 'Usage' section states: 'For this, we will assume that you are using the MariaDB 10.3 container image from the Red Hat Container Catalog called rhel8/mariadb-103. If you want to set only the mandatory environment variables and start the database in a shell directory.' The 'Most recent tag' section shows 'Updated a month ago' and '1-37'. The 'Health Index' section shows a green bar and 'A'. The 'Security' section is partially visible.

コンテナになっていないものはどうする？

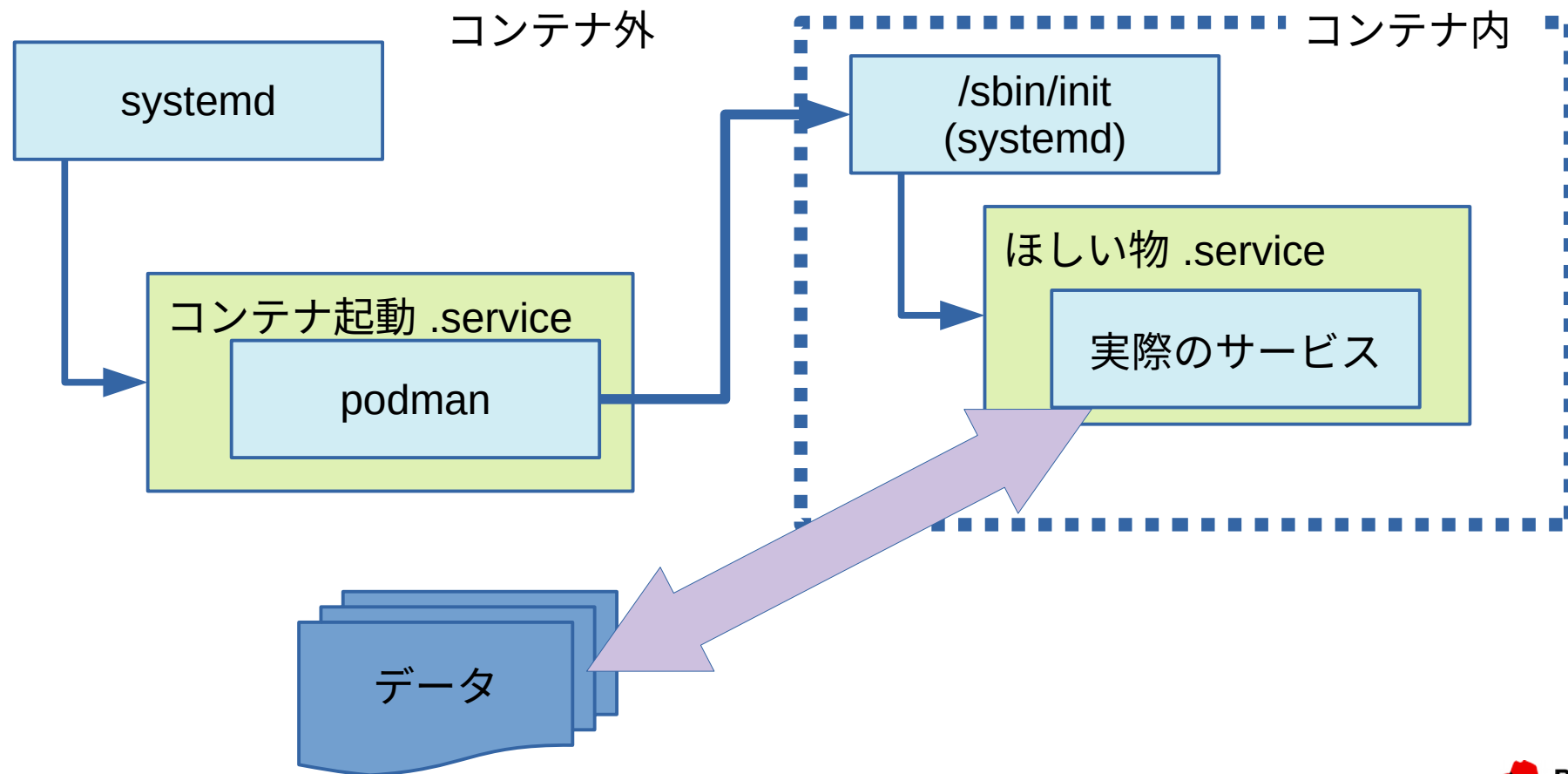
**言語処理系やライブラリを分けたいなら ubi8/ubi をベースに
コンテナを作成する**

必要なソフトウェアを導入

サービスなら ubi8/ubi-init をベースにコンテナを作成する

パッケージをそのまま使うなら systemd から起動するのが簡単。イメージ内でパッケージのインストールと `systemctl enable` までおこなう。

ubi-init を利用する場合のイメージ



独自のイメージ作成の概要

ベースイメージ取得

```
# podman login ← 既に login していれば不要  
# podman pull registry.redhat.io/ubi8/ubi-init
```

対話的に作成する

```
# buildah from registry.redhat.io/ubi8/ubi-init  
ubi-init-working-container ← 作業用コンテナ名  
# buildah run ubi-init-working-container コマンド ← 構築用のコマンド実施  
# buildah config ubi-init-working-container XXXX ← entrypoint などの設定  
# buildah commit ubi-init-working-container myname/myserver ← イメージ登録
```

バッチ的に作成する

```
# buildah bud -t Dockerfile . ← build using dockerfile の略で” bud”
```


実行例

SELinux の bool 値設定

```
# setsebool -P container_manage_cgroup on
```

イメージ取得

```
# podman login
```

```
# podman pull registry.redhat.io/ubi8/ubi-init
```

イメージ作成

```
# buildah from registry.redhat.io/ubi8/ubi-init
```

```
# buildah run ubi-init-working-container yum install httpd
```

```
# buildah run ubi-init-working-container systemctl enable httpd
```

```
# buildah copy ubi-init-working-containers index.html /var/www/html/index.html
```

```
# buildah config --port 80 --cmd "/usr/sbin/init" ubi-init-working-container
```

```
# buildah commit ubi-init-working-container kmoriwak/httpd
```

コンテナ実行

```
# podman run -d -p 8000:80 localhost/kmoriwak/httpd
```

データを置く ディレクトリを作成、 コンテナに対応づける

永続化データをホストに置くためのディレクトリ

ディレクトリを作成する

ごく普通に mkdir 等でデータを置くディレクトリを用意します

ディレクトリで指定すべき **permission** が コンテナにより異なります

特定ユーザでの読み書きができること、root での読み書きができること、
利用するユーザを引数で指定できそれと揃える必要があるものなど
イメージの内容次第で条件が異なります

コンテナと対応づけて実行する

podman の実行時に対応づける

ホストのディレクトリとコンテナ内のディレクトリを対応づけます

例:

```
# podman run -d -v /home/httpd:/var/log/httpd:Z -v /dev/log:/dev/log -p 80:80  
localhost/kmoriwak/httpd
```

SELinux のタイプづけ

コンテナと共有するディレクトリに SELinux のタイプ container_file_t をつけます
podman run のボリューム指定で **:Z** (1つのコンテナで占有) または **:z** (複数のコンテナで共有) オプションを付与すると自動的にタイプづけを行います

コンテナで SELinux は何のために利用される？

SELinux はコンテナ同士の分離、コンテナとホストの分離のために利用されます

コンテナ内から別のコンテナやホスト側のリソースを操作できる脆弱性があってもそのような操作を防止できる場合があります

<https://www.redhat.com/en/blog/latest-container-exploit-runc-can-be-blocked-selinux>

ログについての考慮

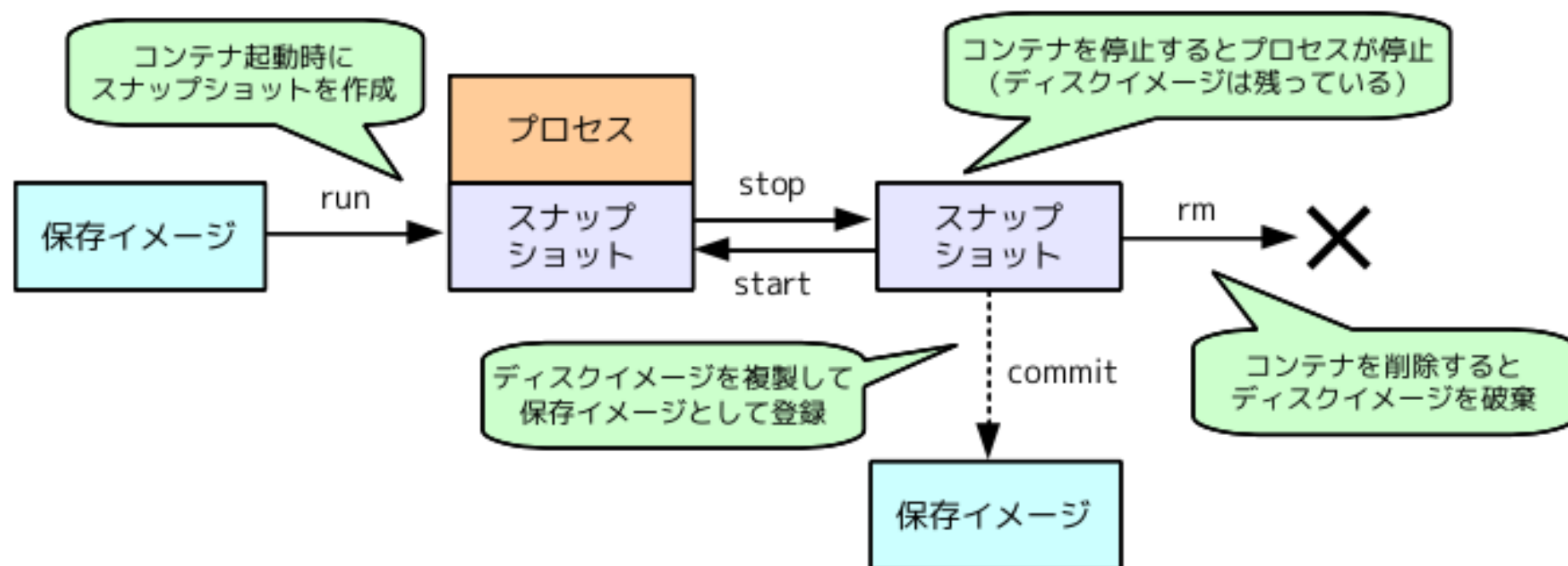
ログをホスト側へ送る方法いろいろ

アプリケーションのログ出力方式により方針が変わります。

- 標準入出力にログを出力する。コンテナのログとしてホスト側に出力される。コンテナ環境ではこの仕組みが一番よく使われる。
- リモート転送
 - アプリケーションがリモートの syslog サーバへの転送に対応していればそれを利用。
 - rsyslogd または journald をコンテナ内で起動してリモート転送する。
- ファイル / デバイス共有
 - ログを保存する専用ディレクトリがある場合にはディレクトリを共有する
 - /dev/log を共有してホスト側の systemd-journald にログ出力させる。(この方式ではどのコンテナ由来かわからなくなる点に注意。)

systemd の .service ファイルを用意

コンテナのライフサイクル



コンテナを systemd のサービスにする

おおまかな方針

1. コンテナに名前をつけて起動する
2. podman start / stop を行う .service ファイルを作成する
3. systemd のサービスとしてコンテナを管理する

コンテナに名前をつけて起動する

コンテナ起動

コンテナに名前をつけて起動する

例:

# podman run -d \	バックグラウンドで起動
--name mariadb_server \	コンテナに名前「mariadb_server」をつける
-p 3306:3306 \	ホストの port をコンテナの port に対応づけ
-v /home/db:/var/lib/mysql:Z \	ホストとコンテナのディレクトリ対応づけ
localhost/kmoriwak/mariadb	

コンテナ停止

# podman stop -t 10 mariadb_server	SIGTERM 送信後 10 秒待って SIGKILL
------------------------------------	-----------------------------

podman start / stop を行う .service ファイルを作成する

コンテナの起動・終了を行う .service ファイルを作成する

例 : /etc/systemd/system/mymariadb.service

[Unit]

Description=mariadb container

After=network.target

[Service]

ExecStart=/usr/bin/podman start -a mariadb_server

ExecStop=/usr/bin/podman stop -t 10 mariadb_server

[Install]

WantedBy=multi-user.target

サービスの管理

.service ファイルの読み込み

```
# systemctl daemon-reload
```

サービスの起動

```
# systemctl enable --now mymariadb
```

ログの参照

```
# journalctl -u mymariadb
```

停止・削除

```
# systemctl stop mymariadb  
# podman rm mariadb_server
```

実行例

- 前準備

```
# setsebool -P container_manage_cgroup on
# mkdir /home/db ; chown -R mysql:mysql /home/db
# podman login
```

- 公開イメージを取得

```
# podman pull registry.redhat.io/rhel8/mariadb-103
```

- コンテナ実行（環境変数などは Container Catalog を参照）

```
# podman run -d --name mariadb_server -e MYSQL_USER=mysql \
-e MYSQL_PASSWORD=mypassword -e MYSQL_DATABASE=db -v /home/db:/var/lib/mysql/data \
-p 3306:3306 rhel8/mariadb-103
# podman stop -t 10 mariadb_server
```

- .service ファイル用意

```
# cat > /etc/systemd/system/mymariadb.service
# systemctl daemon-reload
```

- 起動

```
# systemctl enable --now mymariadb
```

- 停止 & 削除

```
# systemctl stop mymariadb
# podman rm mariadb_server
```

まとめ

なぜコンテナをサービス化したいか？

- Application Stream で複数バージョンを同時に使いたい
- 同じサービスを複数起動したい など

RHEL コンテナの配布

- UBI, RHEL, いくつかの既存コンテナ

ディレクトリ作成時の注意

コンテナを systemd でのサービスにする

参考情報

- 公式ドキュメント「コンテナの構築、実行、および管理」
<https://red.ht/2ZTKug7>
- Red Hat Container Catalog
<https://access.redhat.com/containers/>

Thank You