これからはじめるコンテナ

3ステップでコンテナを学ぼう

レッドハット株式会社
Sales/Solution Architect

コンテナを知る3つのステップ

1st ステップ

コンテナが
使われるのは
なぜなのか？

2nd ステップ

コンテナを
使うためには
何が必要なのか？

3rd ステップ

コンテナは
どのように
使われるのか？

コンテナが使われる理由

1st ステップ

コンテナが使われるのはなぜなのか？

2nd ステップ

コンテナを使うためには何が必要なのか？

3rd ステップ

コンテナはどのように使われるのか？

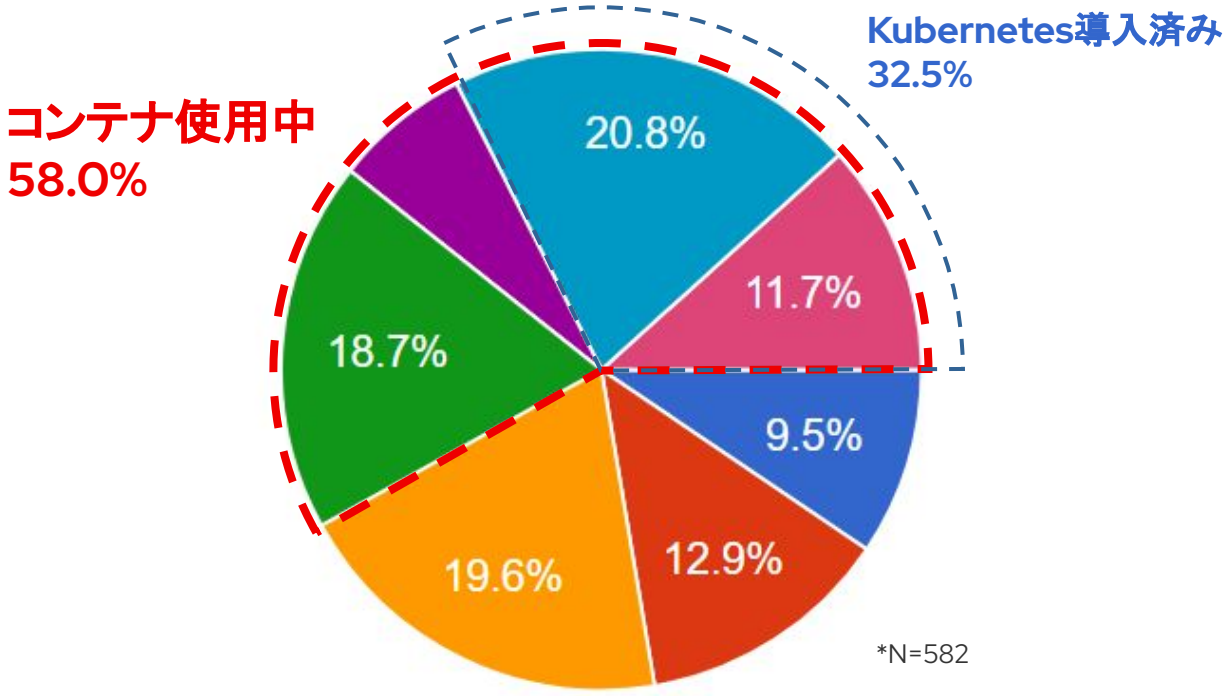
Red Hat

多くの企業で利用が進むコンテナ

半数を超える企業でコンテナが使用され、約1/3はKubernetes(コンテナ基盤ソフトウェア)を使用しています。

あなたの企業(もしくは支援先企業)ではKubernetesを利用していますか？

※Kubernetesのプロダクトは問いません。



* Red Hat K.K. - 日本のコンテナ市場2020調査 (2020/03)

Red Hat

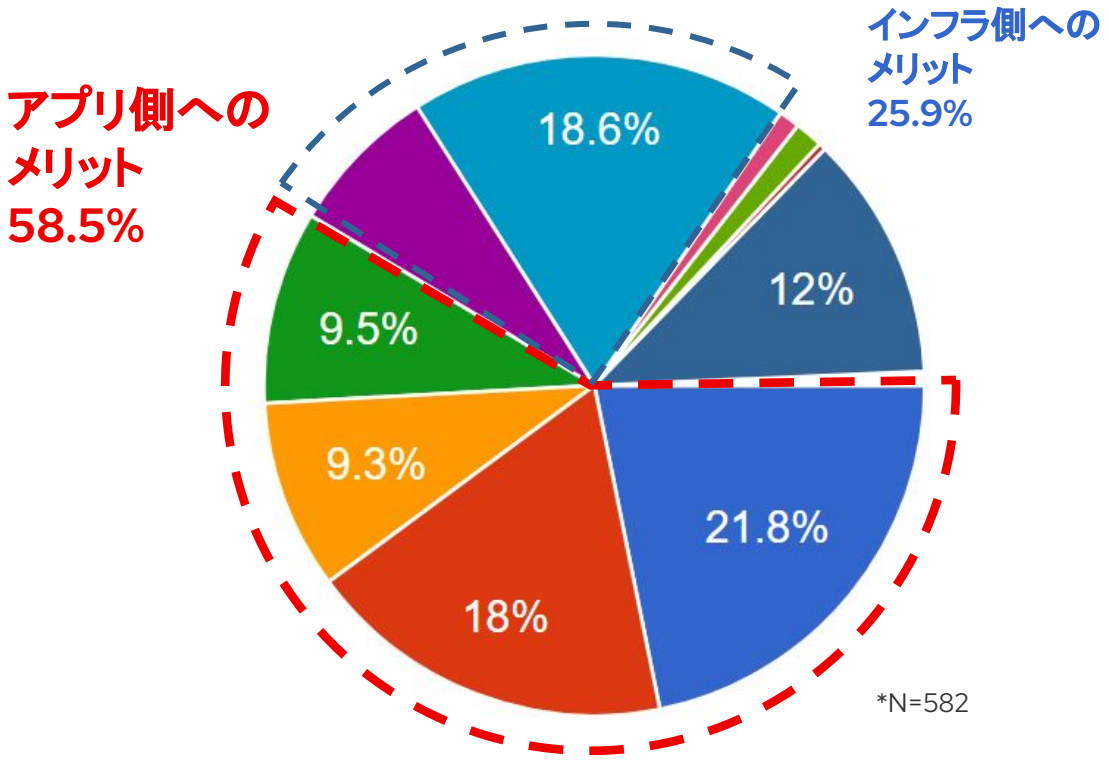
なぜコンテナを使うのか

コンテナを うまく使う ことで、ビジネスメリット を得られるから

コンテナに期待するビジネスメリット

コンテナを使うことによって様々なビジネスメリットが期待できます。

既存システムと比べ、コンテナ(Kubernetes)導入に期待する一番のビジネスメリットはなんですか？



- アプリケーション開発の生産性向上、アジリティ向上
- アプリケーション運用の効率化、コスト削減
- アプリケーションリリースサイクルのスピード向上
- アプリケーションのポータビリティ(可搬性)
- インフラリソースの集約率向上、コスト削減
- インフラリソース管理の運用自動化、プロセス改善
- SoE(IoTやAI、機械学習など)の促進
- SoRのアプリケーションモダナイゼーション
- SoRのデータ利活用/マイグレーション
- まだよくわからない

* Red Hat K.K. - 日本のコンテナ市場2020調査 (2020/03)

Red Hat

コンテナはアプリケーションの開発・運用を改善する

コンテナを使うことのメリットは、インフラよりもアプリケーションへのメリットが大きいと言えます。

アプリケーションのメリット

- **開発の生産性向上**
  - 自動的なテスト実施～成果物納品
  - 単一基盤での複数環境の準備
- **運用の効率化**
  - トラブル時の自動対応
  - 属人的作業の削減
- **リリースサイクルの短縮**
  - 修正・アップデートを頻繁にリリース
- **ポータビリティ**
  - 実行基盤を問わないアプリケーション

インフラのメリット

- **集約率の向上**
  - インフラリソース使用率の改善、コスト削減
- **運用の効率化**
  - インフラリソース提供の自動化
  - 属人的作業の排除

コンテナはアプリケーションの新しい姿である

コンテナはアプリケーションの視点でとらえるのがよいでしょう。

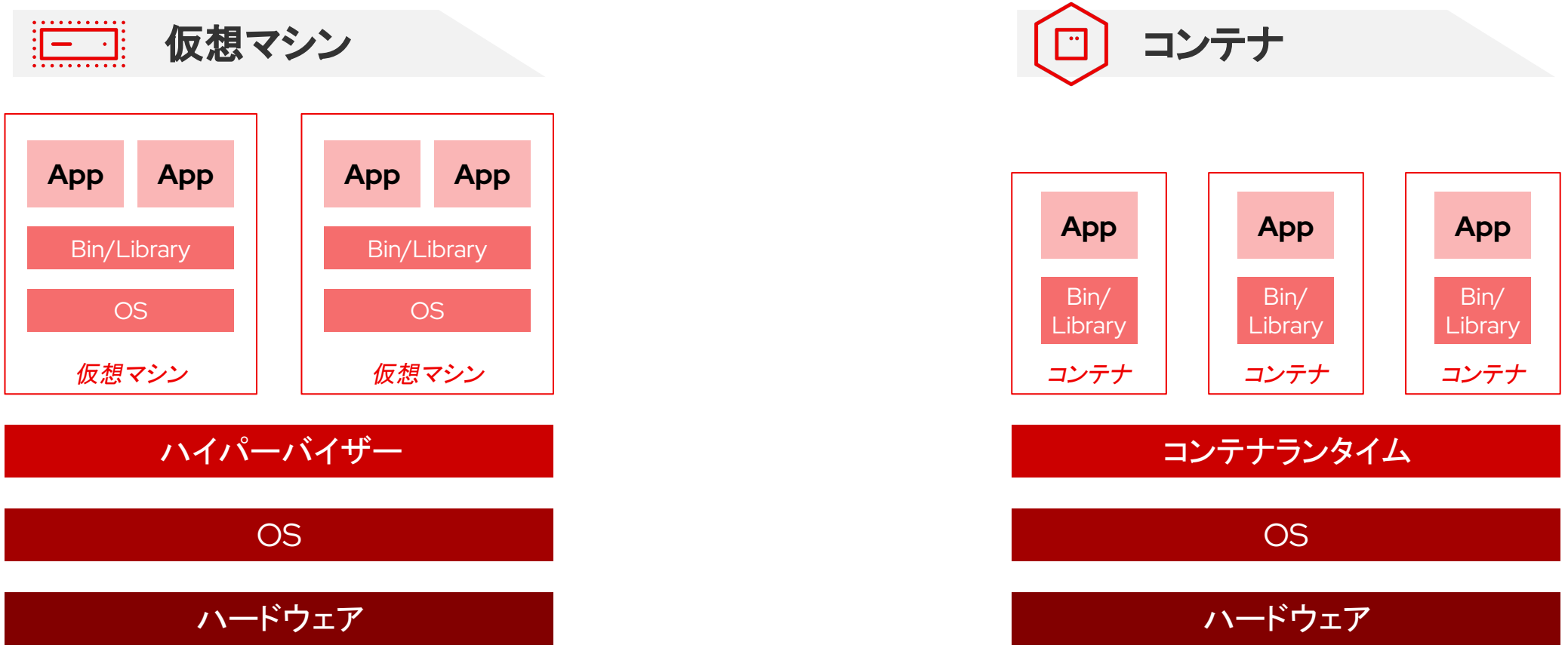
- コンテナを仮想マシンの延長と考えると、インフラ面でのメリットに注目してしまい、アプリケーション面でのメリットを置き去りにしがちです。
- コンテナをアプリケーションの新しい姿ととらえて、より大きなメリットを感じましょう。

コンテナとは一言で言えば何か？

アプリケーション本体と、
アプリケーションの実行に必要な**ライブラリ・依存関係**など、
必要最小限の要素をひとつにパッケージした姿

コンテナと仮想マシンの違い

コンテナはOSを持ちません。



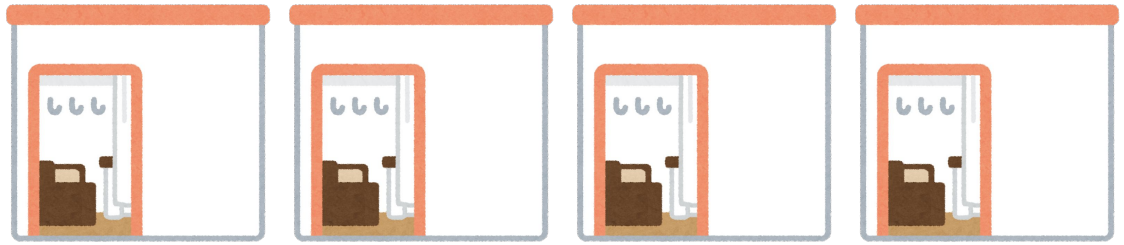
コンテナと仮想マシンの違い

コンテナはOSを持ちません。

仮想マシン

＝

個室



コンテナ

＝

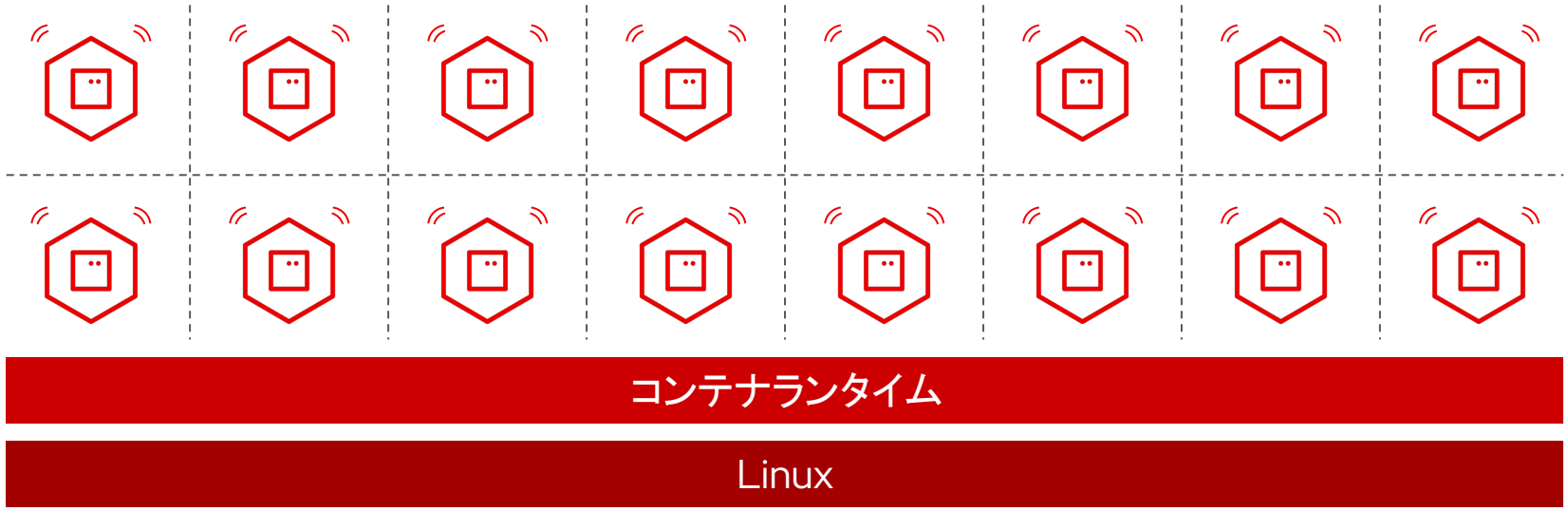
仕切りで区切られたスペース

Red Hat

コンテナの特徴

コンテナにはいくつもの特徴があります。

- **Linuxで稼働**
  - Linuxカーネルが持つ機能を利用する。
  - 1つのLinuxホストの上で複数のコンテナを同時に稼働できる。
- **隔離性**
  - Linuxホストのカーネルを共有するが、コンテナ同士は隔離され互いに競合しない。
    - コンテナ同士で通信可能にはできる。

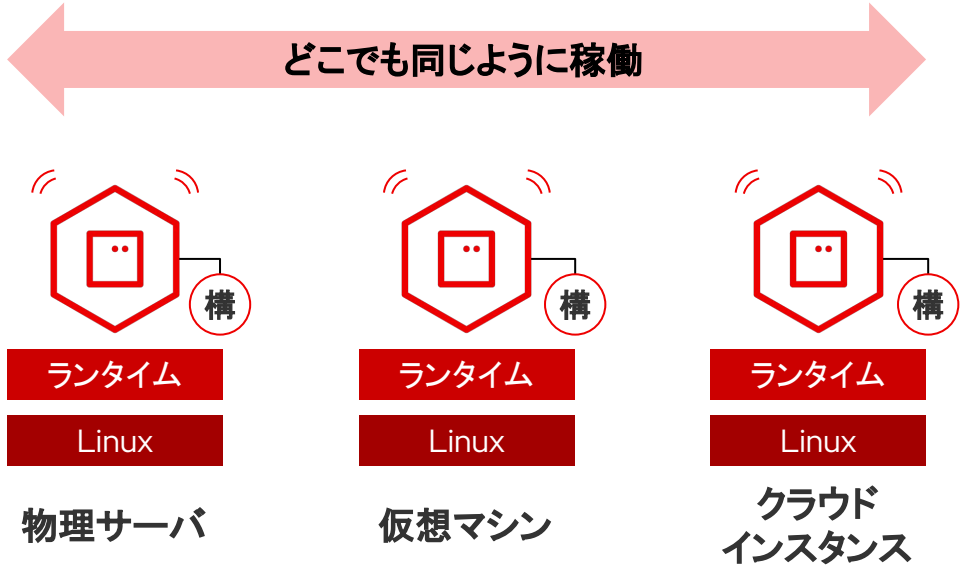


デモンストレーション

コンテナの特徴

コンテナにはいくつもの特徴があります。

- **可搬性 (Portable)**
  - どの環境でも同じように稼働する。
    - 環境に依存する構成情報はコンテナとは別で持つ。



- **軽量**
  - OSが無く、必要最小限の要素のみ持つ。
- **起動が高速**
  - OS起動時間を省略できる。

	仮想マシン	コンテナ
容量	1桁 ～ 2桁 GB	2桁 MB ～ 1桁 GB
起動時間	数分	数秒

Red Hat

デモ動画: コンテナの高速な起動

```
$ podman pull registry.access.redhat.com/ubi9/ubi
Trying to pull registry.access.redhat.com/ubi9/ubi:latest...
Getting image source signatures
Checking if image destination supports signatures
Copying blob 72d37ae8760a done
Copying config 8e9c11168e done
Writing manifest to image destination
Storing signatures
8e9c11168e6d9de29f6bbd7e59eca89f868cab89028f266dae17c684046b1479
$
$ podman images
REPOSITORY                          TAG         IMAGE ID      CREATED      SIZE
registry.access.redhat.com/ubi9/ubi  latest      8e9c11168e6d  3 weeks ago  219 MB
$
$ podman run -it registry.access.redhat.com/ubi9/ubi:latest
[root@206c03fedc9f /]#
[root@206c03fedc9f /]# cat /etc/os-release
NAME="Red Hat Enterprise Linux"
VERSION="9.1 (Plow)"
ID="rhel"
ID_LIKE="fedora"
VERSION_ID="9.1"
PLATFORM_ID="platform:el9"
PRETTY_NAME="Red Hat Enterprise Linux 9.1 (Plow)"
ANSI_COLOR="0;31"
LOGO="fedora-logo-icon"
CPE_NAME="cpe:/o:redhat:enterprise_linux:9::baseos"
HOME_URL="https://www.redhat.com/"
DOCUMENTATION_URL="https://access.redhat.com/documentation/red_hat_enterprise_linux/9/"
BUG_REPORT_URL="https://bugzilla.redhat.com/"

REDHAT_BUGZILLA_PRODUCT="Red Hat Enterprise Linux 9"
REDHAT_BUGZILLA_PRODUCT_VERSION=9.1
REDHAT_SUPPORT_PRODUCT="Red Hat Enterprise Linux"
REDHAT_SUPPORT_PRODUCT_VERSION="9.1"
[root@206c03fedc9f /]#
```

操作の内容

- RHEL9のコンテナイメージ(ubi9)を入手する。
- ubi9があることを確認する。
- ubi9からコンテナを起動しコンテナ内のシェルに入る。
- /etc/os-releaseファイルから、確かにRHEL9の環境であることが確認できる。
  コンテナであれば1分もしないうちにLinux環境が用意できることがわかる。

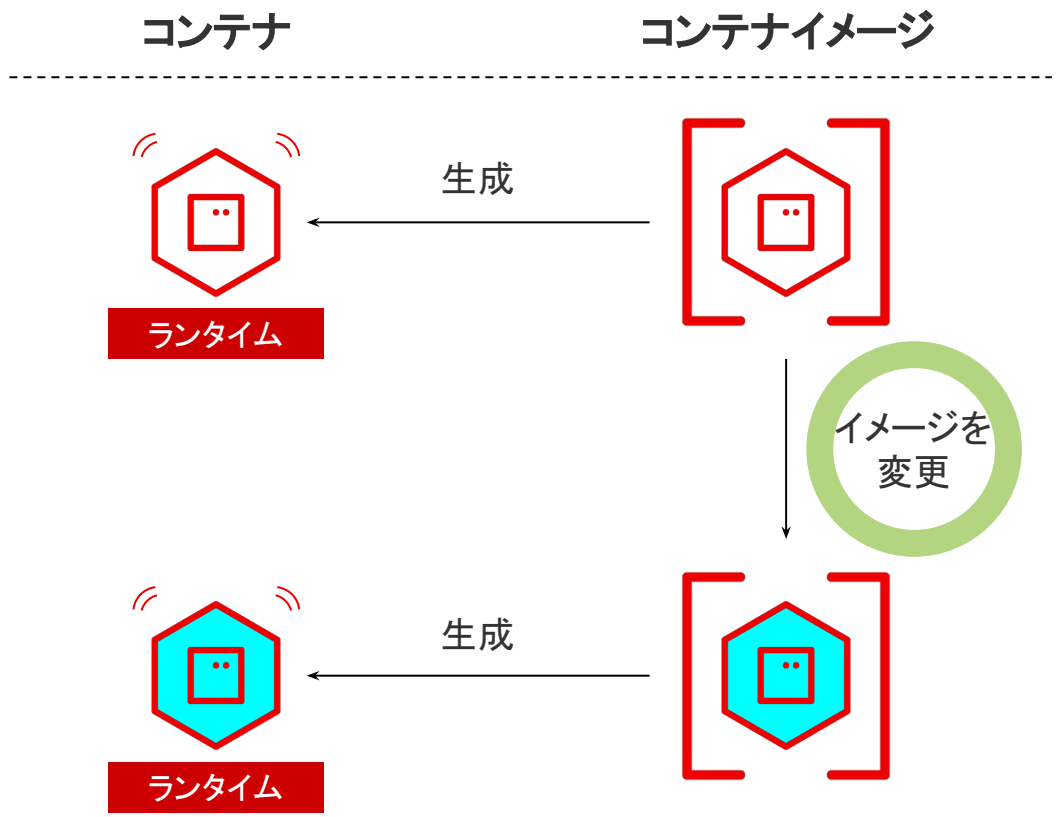
Red Hat

デモンストレーション

コンテナの特徴

コンテナにはいくつもの特徴があります。

- **イメージから生成**
  - コンテナイメージから複製して作られる。
- **不変性 (Immutable)**
  - 同じコンテナイメージから起動したコンテナは、毎回必ず同じものとなる。
- **揮発性 (Ephemeral)**
  - コンテナに加えた変更は、コンテナが停止すると失われる。
    - コンテナ自身に永続性は無い。
  - コンテナに変更を加えたい場合は、コンテナイメージを変更して新しくコンテナを起動する。
    - 古いコンテナは破棄する。



デモ動画: コンテナの不変性・揮発性

```
$ podman images
REPOSITORY                          TAG         IMAGE ID      CREATED      SIZE
registry.access.redhat.com/ubi9/ubi  latest      8e9c11168e6d  3 weeks ago  219 MB
$
$ podman run -it registry.access.redhat.com/ubi9/ubi:latest
[root@71b453c22df6 /]#
[root@71b453c22df6 /]# mkdir newdir
[root@71b453c22df6 /]# echo "HELLO, CONTAINER WORLD." > newdir/messagefile
[root@71b453c22df6 /]# ls -p newdir/
messagefile
[root@71b453c22df6 /]# cat newdir/messagefile
HELLO, CONTAINER WORLD.
[root@71b453c22df6 /]#
[root@71b453c22df6 /]# ls -p
afs/  bin  boot/  dev/  etc/  home/  lib  lib64  lost+found/  media/  mnt/  newdir/  opt/  proc/  root/  run/
 sbin  srv/  sys/  tmp/  usr/  var/
[root@71b453c22df6 /]# rm -rf boot/ home/ media/ mnt/ opt/ var/
[root@71b453c22df6 /]# ls -p
afs/  bin  dev/  etc/  lib  lib64  lost+found/  newdir/  proc/  root/  run/  sbin  srv/  sys/  tmp/  usr/
[root@71b453c22df6 /]# exit
exit
$
$ podman run -it registry.access.redhat.com/ubi9/ubi:latest
[root@99e682bc9d2d /]# ls -p
afs/  bin  boot/  dev/  etc/  home/  lib  lib64  lost+found/  media/  mnt/  opt/  proc/  root/  run/  sbin  sr
v/  sys/        tmp/  usr/  var/
[root@99e682bc9d2d /]# cat newdir/messagefile
cat: newdir/messagefile: No such file or directory
[root@99e682bc9d2d /]#
```

操作の内容

- RHEL9のコンテナイメージ(ubi9)を確認して起動し、シェルに入る。
- コンテナのルートディレクトリの中身を確認し、変更を加える。
  - 新規のディレクトリ(newdir)とファイル(messagefile)を作成する。
  - 既存のディレクトリをいくつか削除する。
- コンテナを終了し、再度同じコンテナを起動する。
- 変更を加える前の状態に戻っており、さきの変更は無かったことになっていることが確認できる。

Red Hat

コンテナを使うために必要なもの

1st ステップ

コンテナが
使われるのは
なぜなのか？

2nd ステップ

**コンテナを
使うためには
何が必要なのか？**

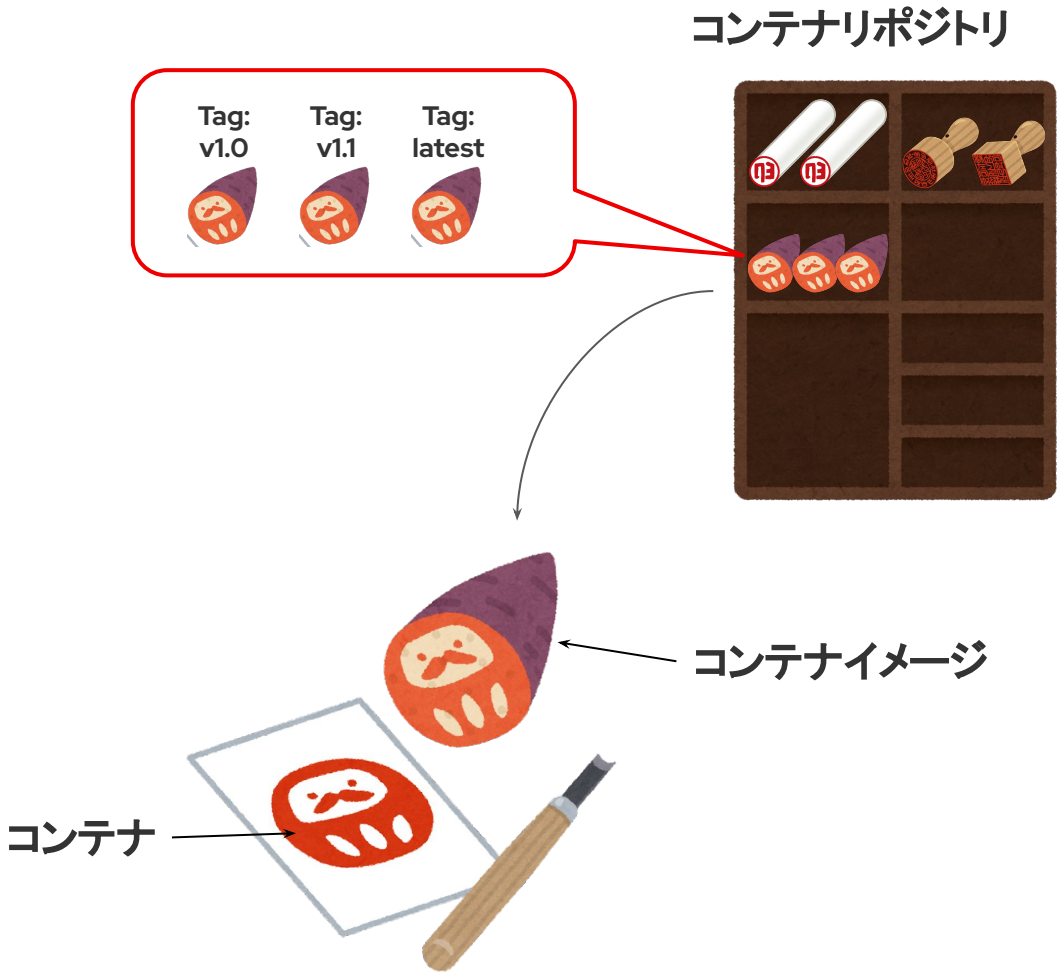
3rd ステップ

コンテナは
どのように
使われるのか？

コンテナイメージ

コンテナイメージはコンテナを生成する元となるものです。

- **コンテナの素**
  - あらゆるコンテナはイメージから作られる。
- **コンテナリポジトリで管理**
  - 同じイメージは系列立てて管理される。
  - それぞれのイメージはタグで区別される。
- **利用できるイメージ**
  - パブリックなイメージ
    - Web上で公開されている。
  - プライベートなイメージ
    - ユーザーが独自に作る。
- **イメージの作り方(イメージのビルド)**
  - 既存イメージに追加変更を加えてビルドする。
- **セキュリティには要注意**
  - 特にパブリックなイメージは「信頼できる提供元のイメージか」「脆弱性は無いか」など注意すること。

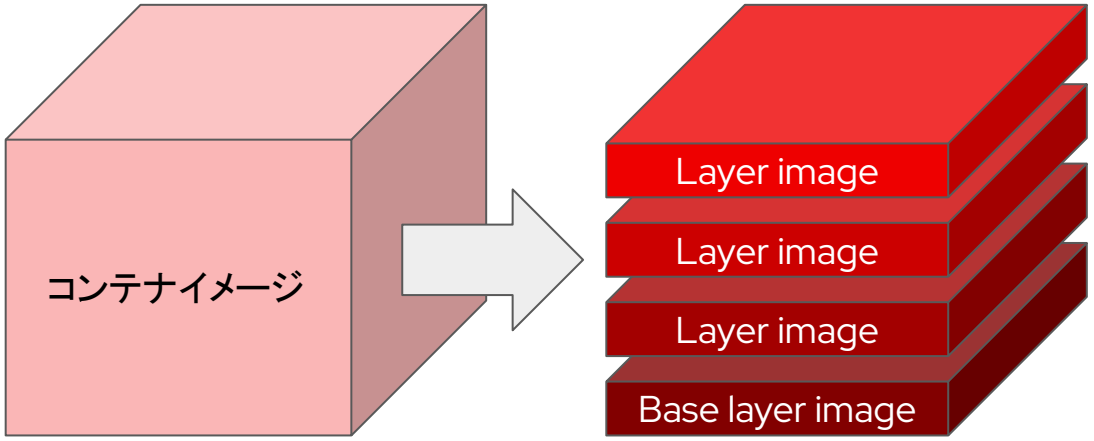


Red Hat

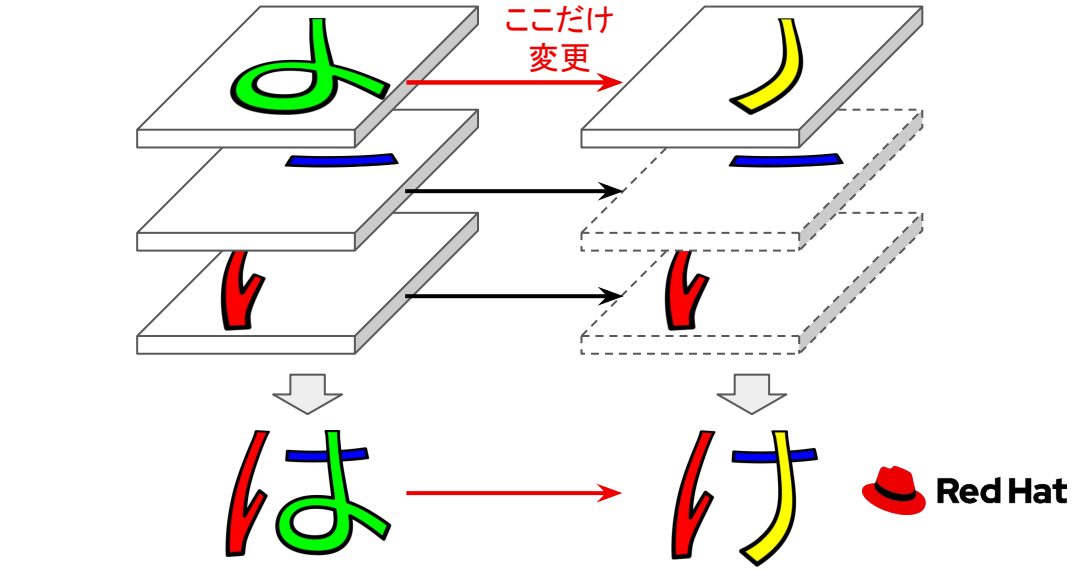
[参考] コンテナイメージのレイヤー構造

コンテナイメージはレイヤー構造を持ちます。

- **レイヤー構造**
  - コンテナイメージは複数のイメージの積み重なったレイヤー構造となる。



- **レイヤー構造の利点**
  - イメージを変更する際に、変更されたレイヤーのみを置き換えてビルドできる。
    - イメージに変更を加えても、イメージの容量は必要以上に膨らまない。
    - イメージの変更履歴が管理しやすい。



Red Hat

デモ動画: コンテナイメージのビルド

```
$ ls ./
Dockerfile
$ cat Dockerfile
FROM registry.access.redhat.com/ubi9/ubi

RUN mkdir newdir
RUN echo "HELLO, CONTAINER WORLD. From myimage." > newdir/messagefile
$
$ podman build -t myimage .
STEP 1/3: FROM registry.access.redhat.com/ubi9/ubi
STEP 2/3: RUN mkdir newdir
--> d82a426d9db2
STEP 3/3: RUN echo "HELLO, CONTAINER WORLD. From myimage." > newdir/messagefile
COMMIT myimage
--> 848cc2fe875c
Successfully tagged localhost/myimage:latest
848cc2fe875c3a5cb0e98c56527af1a8e147f37ed7352a79e950fb8bf7c403e1
$
$ podman images
REPOSITORY                          TAG         IMAGE ID      CREATED        SIZE
localhost/myimage                   latest      848cc2fe875c  4 seconds ago  219 MB
registry.access.redhat.com/ubi9/ubi  latest      8e9c11168e6d  3 weeks ago    219 MB
$
$ podman run -it localhost/myimage:latest
[root@63afe0527e3b /]#
[root@63afe0527e3b /]# ls -p
afs/  bin  boot/  dev/  etc/  home/  lib  lib64  lost+found/  media/  mnt/  newdir/  opt/  proc/  root/  run/
 sbin  srv/  sys/  tmp/  usr/  var/
[root@63afe0527e3b /]# cat newdir/messagefile
HELLO, CONTAINER WORLD. From myimage.
[root@63afe0527e3b /]#
```

操作の内容

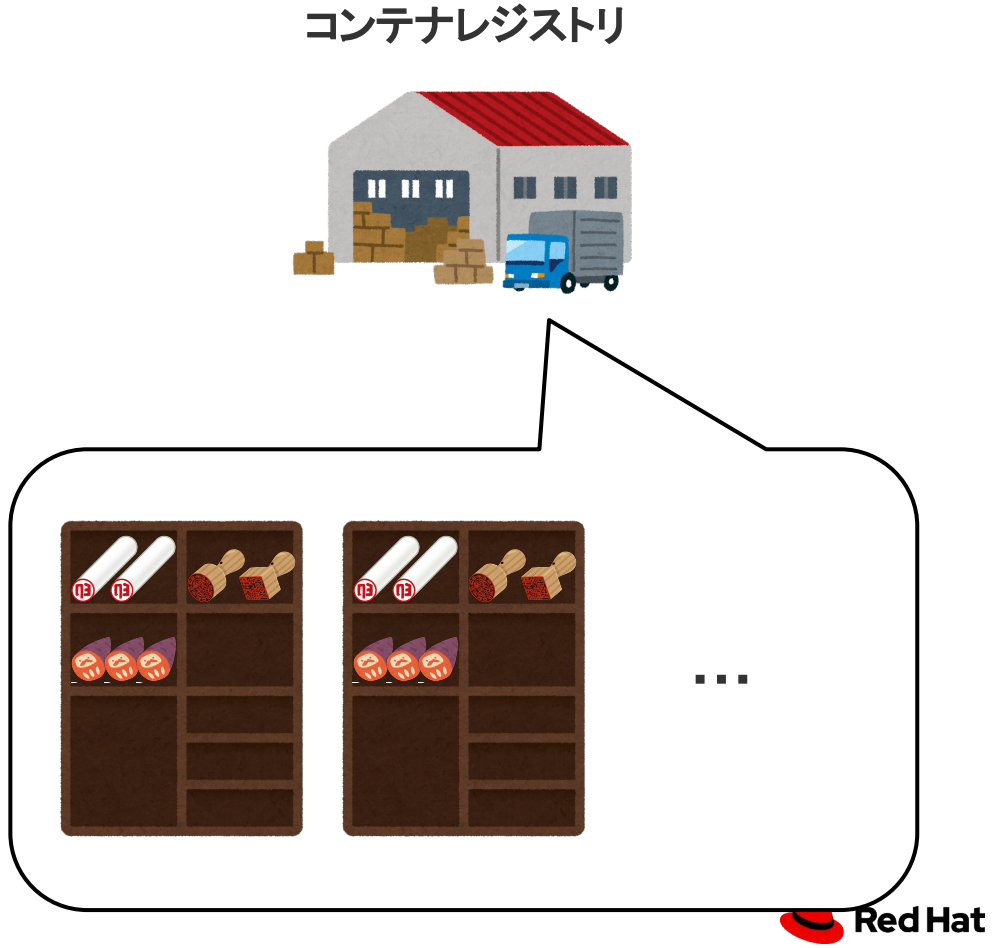
- Dockerfileの中身を確認する。
  - RHEL9のイメージ(ubi9)をベースとする。
  - ルートディレクトリに新規のディレクトリ (newdir)とファイル(messagefile)を作成する。
- Dockerfileからコンテナイメージをビルドする。
  - myimageとタグをつける
- コンテナイメージが作られていることを確認する
- 作成したイメージからコンテナを起動すると、Dockerfileに記述した通りに、newdirとmessagefileが作成されていることが確認できる。

Red Hat

コンテナレジストリ

コンテナレジストリはコンテナイメージを登録して共有する場所です。

- **コンテナイメージの保管場所**
  - コンテナイメージはレジストリで保管され、共有される。
    - コンテナレジストリからイメージをダウンロードすることを **“Pull”** と呼ぶ。
    - コンテナレジストリにイメージをアップロードすることを **“Push”** と呼ぶ。
- **使用できるコンテナレジストリ**
  - パブリックなレジストリ
    - Docker HubやQuay.ioなど、Web上で公開されたレジストリ。
  - プライベートなレジストリ
    - イメージを非公開にするため、独自に構築したり、クラウドのサービスを使うなどして準備する。



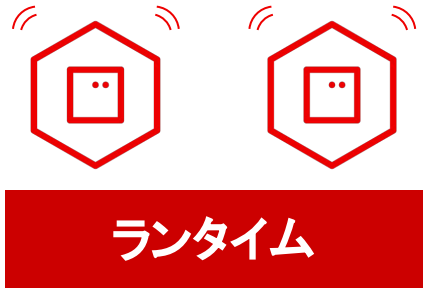
Red Hat

コンテナランタイム

コンテナランタイムはコンテナを稼働するために必要不可欠なものです。

- **コンテナが稼働するために必要なソフトウェア**
  - コンテナの作成・実行・停止・削除などのライフサイクルの管理をする。
  - ランタイムがなければコンテナは動かない。
- **有名なランタイム**
  - ローカル環境で使うとき
    - Docker, Podman
  - コンテナ基盤(Kubernetesなど)で使うとき
    - containerd, cri-o

※ ランタイムの役割や機能はおおむね同じだが、思想・内部アーキテクチャは変わるため微妙な違いはある。



ランタイムの役割

- コンテナのライフサイクル管理
- ハードウェアリソースの分離
- モニタリング・ロギング
- コンテナイメージのPull・管理
- コンテナイメージのビルド・Push　　...etc

Red Hat

コンテナの使いかた

1st ステップ

コンテナが
使われるのは
なぜなのか？

2nd ステップ

コンテナを
使うためには
何が必要なのか？

3rd ステップ

コンテナは
どのように
使われるのか？

なぜコンテナを使うのか

コンテナを**うまく使う**ことで、**ビジネスメリット**を得られるから

アプリケーション開発現場でのコンテナのうまい使いかた

モダンなアプリケーションの開発現場ではコンテナをうまく使っています。

▸ アプリケーション開発者の問題

- **開発・テスト環境を持つ際の問題**
  - 仮想マシン(VM)を使うが、開発機のスペックなどの都合で複数VMを持てない。
  - VMを並列して起動できず、テストのたびに停止・起動を行うため時間がかかる。
  - そもそも複数のVMの用意に時間と労力を要する。
- **複数人で開発する際の問題**
  - 環境の違いに起因する結合テストや統合テストでの失敗がありえる。
  - 標準化した環境を配布しようとしても、VMでは環境の変更管理が難しい。

▸ コンテナを使うことによる解決

- **複数の環境を持つことが容易**
  - コンテナはVMよりも少ないリソースで複数の環境を並列稼働できる。
  - 軽量で高速起動できるため、時間的効率が上がる。
  - 環境の用意はコンテナレジストリからPullするだけでよい。
- **全ての開発者が同じ環境を使用**
  - 不変性を持つコンテナイメージがベースになるため、人によって環境が違うことは起きない。
  - イメージはコンテナレジストリ側で自動的に変更管理される。

Red Hat

アプリケーション本番稼働でのコンテナのうまい使いかた

コンテナは様々なサービスの本番環境でも使われています。

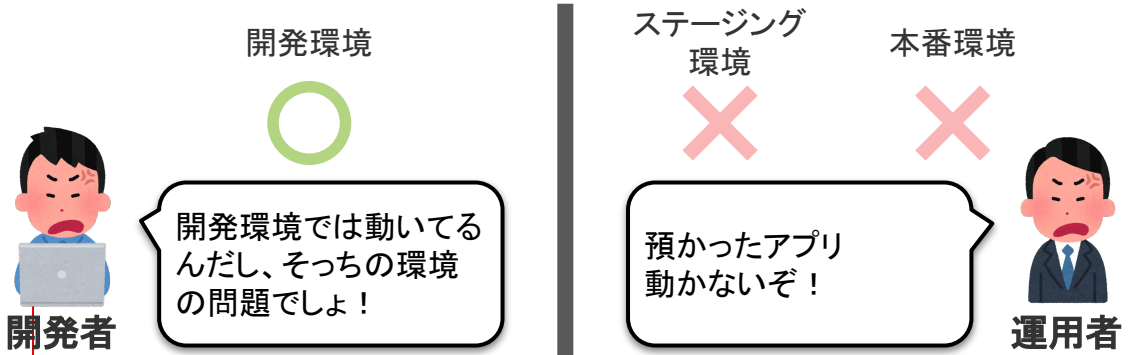
▸ アプリケーション本番稼働の問題

- **環境への依存**
  - 微妙な環境の違いによって、開発環境とステージング・本番環境で挙動が違うなどの問題が起こりえる。
- **システム基盤への依存**
  - 別の基盤(例えばオンプレミスからクラウド)へのアプリケーション移行が難しい。

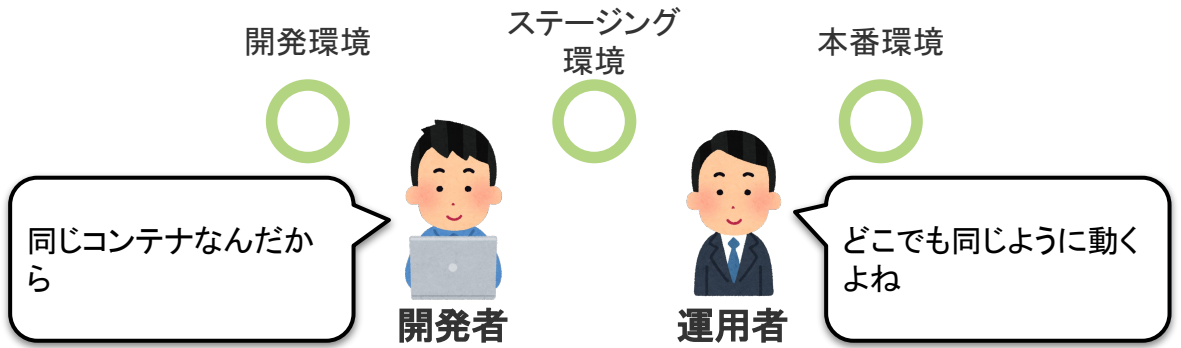
▸ コンテナを使うことによる解決

- **環境やシステム基盤への依存を極小化**
  - コンテナは基盤への依存性が低いため、環境ごとに挙動が違うことを防ぐことができる。
  - 基盤ごとの違いはコンテナ自体とは別で吸収できるため、アプリケーションの移行が比較的容易である。

コンテナ化されていないアプリケーション



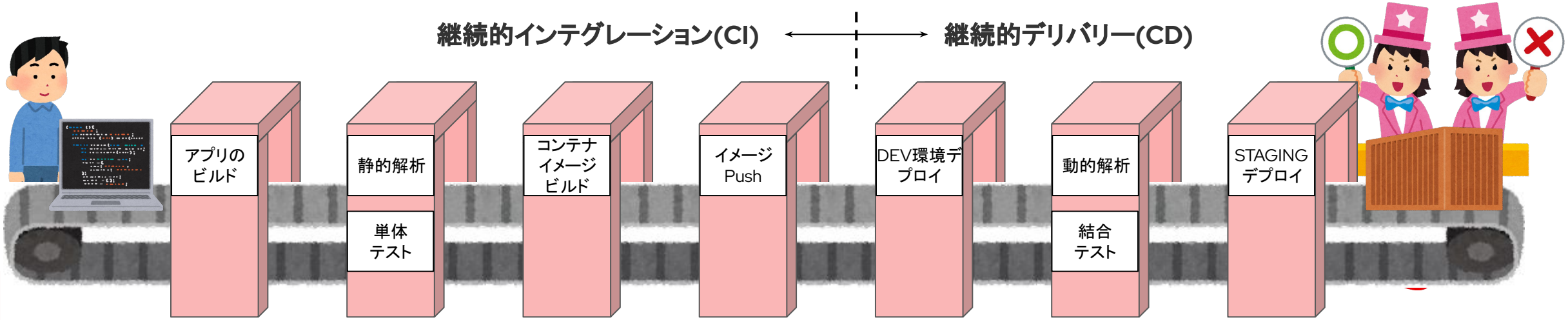
コンテナ化されたアプリケーション



開発～デリバリーまでのコンテナのうまい使いかた

コンテナを使うとアプリケーションの開発からデリバリーまでの作業を一気通貫で走らせることが容易になります。

- **継続的インテグレーションと継続的デリバリー(CI/CD)**
  - アプリケーションのビルドからコンテナのビルド、デプロイまでを自動的に行うようにする。
    - 途中で行うコードの解析やテストなども自動化する。
    - CI/CD用のソフトウェアを利用して、**“パイプライン”**を構成する
  - 人の作業を「アプリケーションのソースコードの投入」と「デプロイの承認」など最小限にして極力自動化することで、アプリケーションデリバリーまでの作業品質を均一化する。



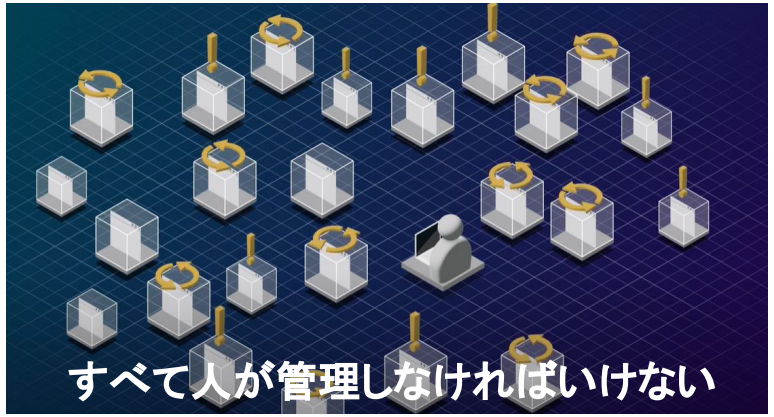
実稼働するシステムでコンテナをうまく使うには

コンテナオーケストレータはコンテナの管理・運用を自動化し、ランタイムだけでは足りないシステムの可用性や運用性を提供します。

コンテナオーケストレータ = Kubernetes を使う

▶ コンテナの管理・運用を自動化

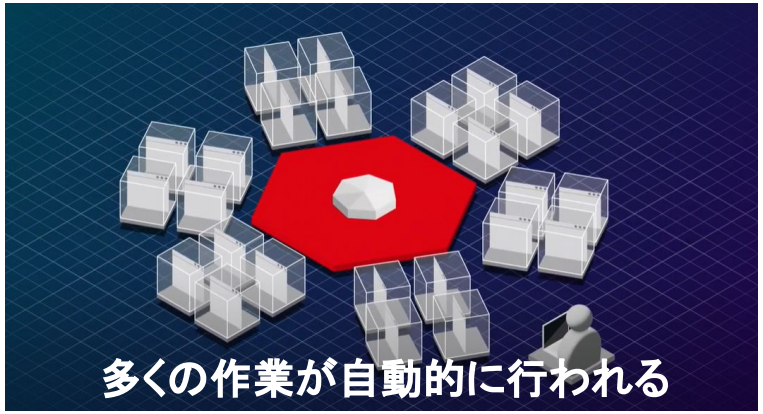
コンテナランタイムのみ



【人が行う作業】

- 属人的な障害復旧オペレーション
- 手動によるのコンテナ変更作業
- アプリケーションごとの設定管理
- 定期的な監視作業

コンテナオーケストレータを使用



【人が行う作業】

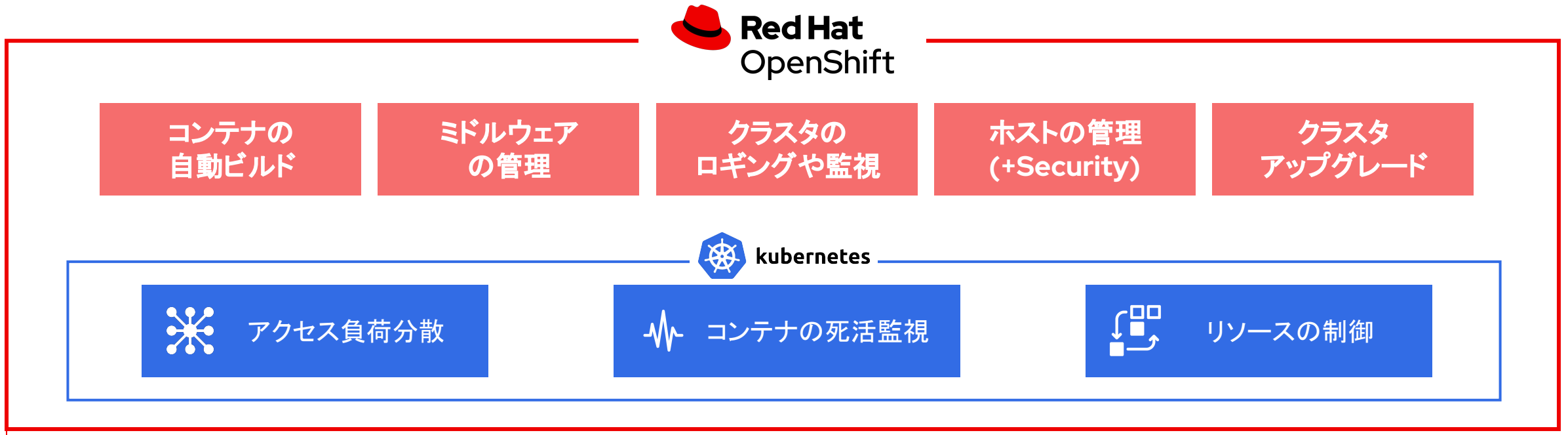
- ビジネス変化に応じた適切なリソース調整

Red Hat

Red Hat OpenShift

Red Hat OpenShiftはKubernetesを包含し、さらにビジネス価値に直結する機能を補完します。

- **Kubernetesでできること**
  - コンテナアプリケーションのデプロイやスケーリングに伴う操作を自動化し、運用負担を軽減する。
- **Kubernetesだけではできないことを補完するOpenShift**
  - アプリケーション開発と運用管理を効率化する機能を提供する。



まとめ

1st ステップ

コンテナが
使われるのは
なぜなのか？

2nd ステップ

コンテナを
使うためには
何が必要なのか？

3rd ステップ

コンテナは
どのように
使われるのか？

まとめ

- コンテナはインフラ側よりもアプリケーション側により大きなビジネスメリットを提供する。
- コンテナは仮想マシンと同じように見えるが、様々な異なる特徴がある。
- コンテナの特徴をうまく使うことで、アプリケーションの開発や運用を効率化できる。
- 実稼働するシステムでは可用性や運用性が必要となるため、コンテナオーケストレータを使うことが求められる。

Container / OpenShift ハンズオン環境のご紹介

本日の101でご紹介した内容を実際に試すことができるハンズオン環境を準備しています。

①ハンズオンのまとめページへアクセス

OpenShiftハンズオンへようこそ

目的

- Container（Docker）の操作を体験します
- OpenShiftの超初級的内容を体験します

ハンズオンに必要な環境

- ブラウザ（Firefox, Chrome）

コース数と難易度

- Container編：2コース　★☆☆☆☆
- OpenShift編：6コース　★☆☆☆☆

難易度	
とても易しい	★☆☆☆☆
易しい	★★☆☆☆
普通	★★★☆☆
少し難しい	★★★★☆
難しい	★★★★★

https://github.com/mayumi00/OpenShiftLearning/blob/main/README.md

コンテナやOpenShiftについて難易度ごとに複数のコンテンツを準備しています。

②コンテンツを選択しハンズオン実施

コースにより環境の立ち上げに最大20分程度かかります。起動したら説明に従いハンズオンを実施下さい。

*: 本ハンズオン環境はハンズオン以外の目的で使用することを想定していません。PoC等の目的でご利用することはお控え下さい。
**: ハンズオンコンテンツの内容については予告無く変更（公開の中止を含む）することがあります。あらかじめご了承下さい。

Red Hat

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat