LinuC

# LinuC レベル 1 Version10.0
# 技術解説無料セミナー
2023/7/22 開催

# コンテナを使った複数の Linux学習環境の構築方法と使用方法


INTERNOUS インターノウス株式会社
（LPI-Japanアカデミック認定校）

竹本 季史


LPI-JAPAN

# 会社紹介、自己紹介

## ■会社紹介：インターノウス株式会社

- ●人材紹介サービス、人材派遣/SESサービス、IT未経験者の教育及び就職支援サービス、法人研修サービス
- ●未経験からインフラエンジニアやプログラマーになりたい方へ、無料で研修と就職支援サービスを行っています。
  https://engineercollege.jp/lp/

## ■自己紹介：竹本 季史(たけもと ときふみ)

- ●IT業界で約10年間勤務後、インターノウス株式会社エンジニアカレッジ講師。
- ●これまで約900人を未経験者からエンジニアに養成。Linuxサーバー(メール、OpenSSH、シェルスクリプト、DB、監視、演習)を担当。
- ●LinuCレベル1バージョン10.0の差分教材で「仮想マシン・コンテナの概念と利用」を執筆。

# LinuCレベル１／レベル２ Version10.0とは

## LinuCとは

クラウド時代の即戦力エンジニアであることを証明するLinux技術者認定

### ✓現場で「今」求められている新しい技術要素に対応

- オンプレミス／仮想化・コンテナを問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

### ✓全面的に見直した「今」身につけておくべき技術範囲を網羅

今となっては使わない技術やコマンドの削除、アップデート、新領域の取り込み

### ✓Linuxの範疇だけにとどまらない領域までカバー

セキュリティや監視など、ITエンジニアであれば必須の領域もカバー

# クラウドを活用できるITエンジニアに必須の技術がまとまっている

**AWSなどのパブリッククラウドを活用するための技術**

間が欠けている状態

**オンプレミスのサーバーサイドLinux技術**

**【今まで／その他】**

**AWSなどのパブリッククラウドを活用するための技術**

**仮想マシン／コンテナ技術、クラウドセキュリティ、アーキテクチャ、ほか**

**オンプレミスのサーバーサイドLinux技術**

LinuC Version10.0

# 本セミナーについて

- LinuCでは、複数のディストリビューションに関する出題がありますが、初学者にとっては学習環境を整えることがハードルとなっています。
- そこで、本セミナーではコンテナを使用した複数のLinux学習環境の構築方法と使用方法について解説します。
- 仮想マシン上にDockerをセットアップすれば、複数のコンテナを使い分けることができるため、RedHat系だけでなく、Debian系のディストリビューションについても学習することができます。
- シェルスクリプトを使うことで容易に学習環境を使い分ける方法も解説します。

# 本セミナーについて

- 本日の内容はLinuCの下記の試験範囲が該当しますが、
  資格対策というよりは「コンテナで何が可能なのか？」というイメージを掴むことが主題です。

  - LinuC101
    - 1.01.2 仮想マシン・コンテナの概念と利用
    - 1.04.1 apt コマンドによるパッケージ管理
    - 1.04.3 yumコマンドによるパッケージ管理
    - 1.06.2 シェルスクリプト

  - LinuC201
    - 2.06.2 Dockerコンテナとコンテナイメージの管理

# 本セミナーについて

- 本セミナー講師の実行環境
  - OS：Windows 11 x86-64
  - 仮想マシン：VMWare Workstation 17 Player
  - LinuxOS：CentOS 7.9 minimal
  - コンテナ：Docker-CE 24.0.4

# 仮想マシンとコンテナの違い①

## 仮想マシン

- ハイパーバイザ（仮想化ソフトウェア）上に仮想的なハードウェア（仮想マシン）を構築する
- 各仮想マシンには、ハードウェア本体のCPU、メモリー、ストレージなどのリソースを分割して割り当てる
- 仮想マシンにはそれぞれOSが必要

### Type1 (ベアメタル型)

- ホストOSが不要
- ハードウェア上にハイパーバイザを搭載
- 例：VMWare ESXi、KVM、Xen、Hyper-V

### Type2 (ホスト型)

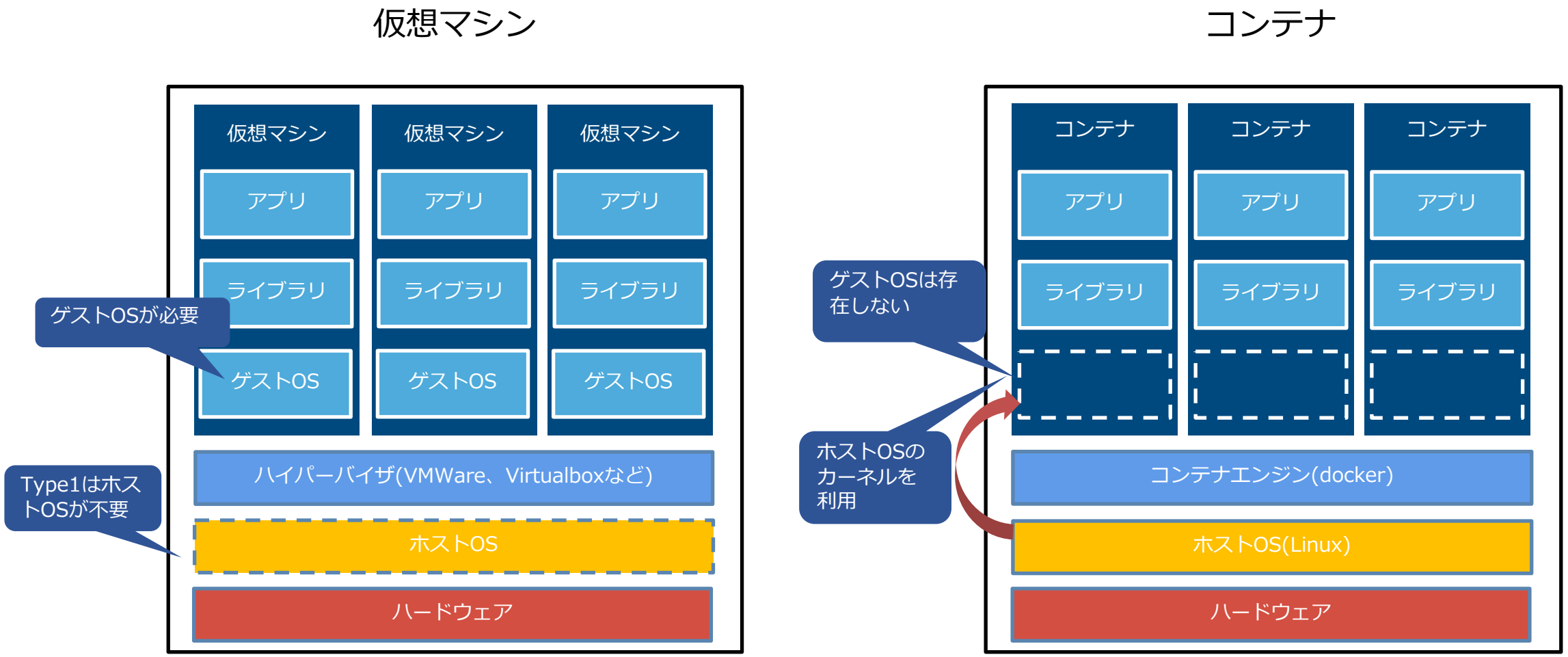
- ホストOSが必要
- ホストOS上にハイパーバイザを搭載
- 例：VMware Workstation Player、Virtualbox

## コンテナ

- コンテナはコンテナエンジン上にホストOSのプロセスを分離したコンテナを作成・実行する
- 各コンテナはホストOSのカーネルを利用することで個別のOSは不要

- 代表例：Docker、Podman

# 仮想マシンとコンテナの違い②



# 仮想マシンとコンテナの違い③

コンテナと仮想マシンの違いをまとめると、下表のようになります

| | ハイパーバイザ型 | コンテナ型 | 備考 |
|---|---|---|---|
| ホストと異なるOSを使用 | ○（可） | ×（不可） | • コンテナはホストOSのLinuxカーネルを利用するため、コンテナにはWindowsなど**Linuxと異なるOSは使用不可** |
| 起動速度 | △（遅） | ○（速） | • コンテナはホストOSのプロセスとして起動するため、**起動が速い** |
| CPU・メモリ消費量 | △（大） | ○（少） | • コンテナはOSの起動がないため、**CPU・メモリ消費量は少ない** |
| 環境の作り直し | △（難） | ○（簡単） | • コンテナはOS関連のファイルが含まれないためイメージファイルが軽量であり、環境の作り直しが**素早く簡単にできる** |
| 他環境へ移行 | △（難） | ○（簡単） | • コンテナはファイルサイズが小さいため、**他環境への移行が容易** |

# コンテナをLinux学習に利用するメリット・デメリット

## メリット

- Dockerコマンドを多用するため、Linuxの学習に加えてコンテナの操作に慣れることができる
- コンテナは軽量で起動が迅速であり、システム全体を仮想化するVMWare、Virtualboxなどの仮想マシンよりもパフォーマンスやリソースの面で仮想マシンより優れている
- 誤ってシステムの重要なファイルを削除しても、簡単に元の状態に戻すことができる
- 簡単に複数のLinuxディストリビューションを使い分けることができる
- 授業や研修で使用するコンテナイメージを講師と受講生の間で共有することで、使用する環境を一致させることができる

## デメリット

- コンテナでは、ファイルシステムやデバイスはホストOSを通じて使用するため、ファイルシステムの作成や変更、デバイスの操作などのコマンドは使用できない
- psコマンドでは、コンテナ内のプロセスしか見えないため、ホストOS全体のプロセスの状況は把握できない

**今回の構成では、仮想マシン上にコンテナを作るので、コンテナのデメリットを補うことが可能**

# Dockerのインストール

Dockerのインストールの手順を実行します。
今後のコマンド実行はrootユーザーで行うことを前提としています。

```
# yum install -y yum-utils （yum-utilsパッケージをインストール）
  インストール:
    yum-utils.noarch 0:1.1.31-54.el7_8

# yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
  （docker-ceリポジトリを登録）
  repo saved to /etc/yum.repos.d/docker-ce.repo
# yum install -y docker-ce （docker-ceをインストール）
  インストール:
   docker-ce.x86_64 3:24.0.4-1.el7

# rpm -qa | grep docker-ce | sort （docker-ceのインストールを確認）
  docker-ce-24.0.4-1.el7.x86_64
  docker-ce-cli-24.0.4-1.el7.x86_64
  docker-ce-rootless-extras-24.0.4-1.el7.x86_64
# docker -v （docker-ceのバージョンを確認）
  Docker version 24.0.4, build 3713ee1
```

# Dockerの起動と自動起動の設定

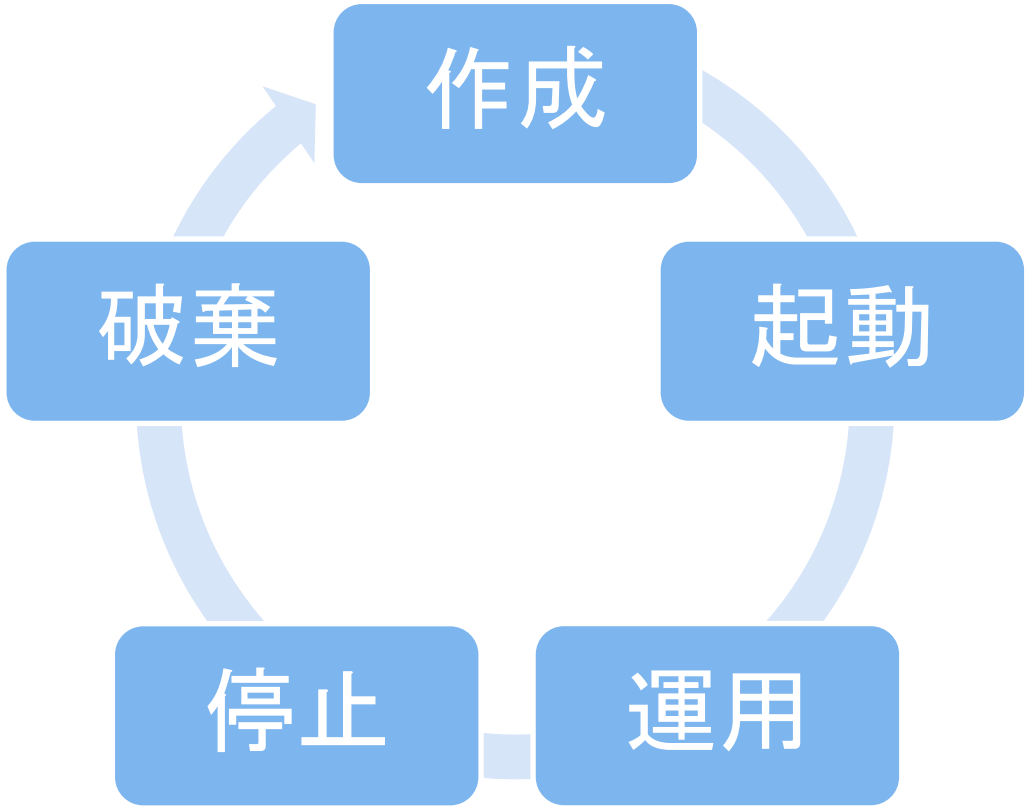
```
# systemctl start docker
(Dockerサービスを起動)

# systemctl status docker
(Dockerサービスが起動していることを確認)
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; vendor preset: disabled)
   Active: active (running) since 日 2023-07-16 11:37:20 JST; 6s ago

# systemctl enable docker
(DockerサービスがLinux起動時に自動的に起動するように設定)
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.
# systemctl is-enabled docker
(自動起動が設定されていることを確認)
enabled
```

# コンテナのライフサイクル

コンテナは右図のように、
「作成→起動→運用→停止→破棄」を繰り返しながら、運用をしていきます。

ひとつのコンテナをアップデートしながら大事に使っていくという思想ではなく、新しいアップデートされたコンテナに入れ替えていくというイメージとなります。

この後、コンテナの各ライフサイクルで使用するコマンドを見ていきます。



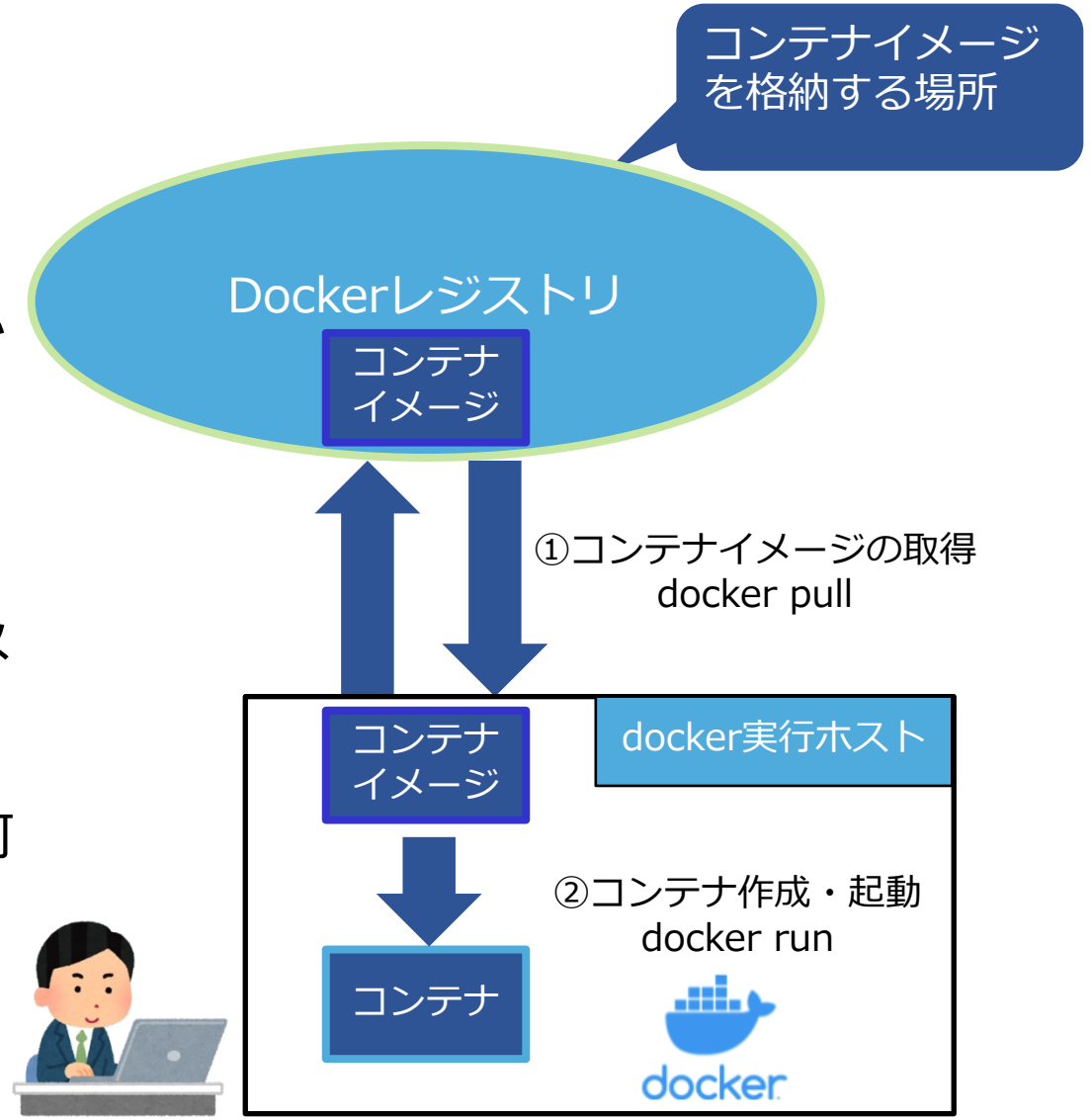
# コンテナ作成方法①（Docker Hubからイメージ取得）

コンテナを0から作成するのは大変です。
レジストリからコンテナイメージを取得して、コンテナを作成する方法があります。

① コンテナイメージファイルをDockerレジストリから取得
② docker実行ホストでコンテナイメージからコンテナを作成

DockerHubというレジストリには多数のコンテナイメージが登録されています。

企業内で独自のDockerレジストリを作成することも可能です。



# 主要なDockerコマンド

| コマンド | 例 | 説明 |
|---|---|---|
| docker pull 〈イメージ名〉 | docker pull centos:7 | 指定したコンテナイメージを取得する |
| docker build 〈Dockerfileのパス〉 | docker build ./ | Dockerfileからコンテナイメージを作成する |
| docker images | docker images | ローカルにあるコンテナイメージを確認 |
| docker run 〈イメージ名〉 | docker run -it centos:7 | コンテナイメージから指定のコンテナを作成して起動する |
| docker attach 〈コンテナID/コンテナ名〉 | docker attach centos7 | 指定したコンテナに接続する |
| docker start 〈コンテナID/コンテナ名〉 | docker start centos7 | 停止しているコンテナを起動する |
| docker stop 〈コンテナID/コンテナ名〉 | docker stop centos7 | 起動中のコンテナを停止する |
| docker ps | docker ps -a | 起動中のコンテナのみを出力(オプションなし)<br>全ての状態のコンテナを出力（-a） |
| docker rm 〈コンテナID/コンテナ名〉 | docker rm centos7 | 指定したコンテナを破棄 |
| docker rmi 〈イメージ名〉 | docker rmi centos:7 | コンテナイメージを削除 |

# Dockerコマンド詳細 docker run

docker run - コンテナを取得、作成、起動する

## docker run〈イメージ名〉

例文：　docker run -itd --name centos7 -h centos7 centos:7

-itは、シェル操作をする時に必ずつけると覚えてください。

| オプション | 意味 |
|---|---|
| -i | コンテナに操作端末（キーボード）を接続する |
| -t | 特殊キーを使用可能にする |
| -d | デタッチモード(バックグラウンド)で起動する。起動後操作の必要がない時に使用。 |
| --name | コンテナ名を指定 |
| -h | コンテナのホスト名を指定 |
| --rm | コンテナ終了時にコンテナを自動的に削除 |

# コンテナ作成方法① DockerHubからコンテナイメージ取得（CentOS）

DockerHubからコンテナイメージを取得してコンテナを作成できます。
ここではCentOS7のコンテナイメージを取得してコンテナを作成します。

```
# docker pull centos:7 （DockerHubからcentos7のコンテナイメージを取得）
Status: Downloaded newer image for centos:7
docker.io/library/centos:7

# docker images （取得したコンテナイメージを確認）
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
centos       7         eeb6ee3f44bd   22 months ago    204MB

# docker run -itd --name centos7 -h centos7 centos:7 （コンテナイメージcentos:7からコンテナを起動）
93cfaa07f720a56b9ad0b1ee2805466b252926fddc7a92f0cf5bd2c5791139f8

# docker ps （centos7のコンテナが起動していることを確認）
CONTAINER ID   IMAGE      COMMAND       CREATED          STATUS          PORTS     NAMES
93cfaa07f720   centos:7   "/bin/bash"   23 seconds ago   Up 22 seconds             centos7
```

イメージのファイルサイズが小さいのは、必要最低限のコマンドしか入っていないため

英数字の羅列は起動したコンテナのコンテナID

Up=コンテナ起動中

# コンテナ作成方法① DockerHubからコンテナイメージ取得（CentOS）

```
# docker attach centos7 （centos7コンテナに接続）
[root@centos7 /]#

# cat /etc/os-release （コンテナのOS情報を確認）
PRETTY_NAME="CentOS Linux 7 (Core)"

# less
bash: less: command not found

# ls -l
total 12
-rw-r--r--.   1 root root 12114 Nov 13  2020 anaconda-post.log
lrwxrwxrwx.   1 root root     7 Nov 13  2020 bin -> usr/bin
drwxr-xr-x.   5 root root   360 Jul 16 02:40 dev
drwxr-xr-x.   1 root root    66 Jul 16 02:40 etc
```

プロンプトのホスト名がcentos7に変化=コンテナに接続できた

lessコマンドが入っていない（イメージサイズを抑えるため）

日付が英語で記載されている

# コンテナ作成方法① DockerHubからコンテナイメージ取得（CentOS）

```
# rpm
RPM バージョン 4.11.3
Copyright (C) 1998-2002 - Red Hat, Inc.
このプログラムは GNU GPL の下で自由に配布できます。
# yum
読み込んだプラグイン:fastestmirror, langpacks
いくつかのコマンドを指定する必要があります
Usage: yum [options] COMMAND
# exit（ホストOSに戻る）
[root@localhost ~]#
```

RedHat系のパッケージ管理コマンドが使用できる

ホストOSに戻った

# コンテナ作成方法① DockerHubからコンテナイメージ取得（Ubuntu）

DockerHubから別のLinuxディストリビューションとしてUbuntuのコンテナイメージを取得してコンテナを作成します。

```
# docker pull ubuntu （DockerHubからubuntu最新版のコンテナイメージを取得）
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

# docker images （取得したコンテナイメージを確認）
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
ubuntu       latest    5a81c4b8502e   2 weeks ago    77.8MB

# docker run -it --name ubuntu -h ubuntu ubuntu:latest （コンテナイメージubuntu:latestからコンテナを起動）
root@ubuntu:/#

# cat /etc/os-release （コンテナのOS情報を確認）
PRETTY_NAME="Ubuntu 22.04.2 LTS"
```

CentOSよりイメージサイズが小さい

# コンテナ作成方法① DockerHubからコンテナイメージ取得（Ubuntu）

```
# dpkg -l
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                    Version                                 Architecture Description
+++-=======================-=======================================-============-===================================
==================================
ii  adduser                 3.118ubuntu5                            all          add and remove users and groups

# apt
apt 2.4.9 (amd64)
Usage: apt [options] command

apt is a commandline package manager and provides commands for
searching and managing as well as querying information about packages.
It provides the same functionality as the specialized APT tools,
like apt-get and apt-cache, but enables options more suitable for
interactive use by default.

# exit （ホストOSに戻る）
[root@localhost ~]#
```

Debian系のパッケージ管理コマンドが使用できる

# コンテナ作成方法② Dockerfileでコンテナイメージをカスタマイズ

コンテナイメージは必要最低限のファイルで構成されているため、サイズが小さいというメリットがありますが、その反面、使用したいコマンドがない、日本語が使えないなどの課題があります。

そこで、DockerHubにあるコンテナイメージを学習環境に適した状態にするために、Dockerfileでパッケージをインストールしたり、設定をカスタマイズすることで、Linux学習やLinuC対策に使うことができます。

Dockerfileを元にコンテナを作成するためには、docker buildコマンドを使用します。

# Dockerコマンド詳細 docker build

## docker build – Dockerfileからコンテナイメージを作成する

```
docker build <Dockerfileのパス>
```

例文：　`docker build ./ -t centos7c`

| オプション | 意味 |
|---|---|
| -t | 作成するコンテナイメージの名前をつける |

# コンテナ作成方法② Dockerfileでコンテナイメージをカスタマイズ

カスタマイズ済みのDockerfileを使用してコンテナイメージを作成します

```
# mkdir docker（専用のディレクトリを作成）

# cd docker（ディレクトリ移動）

# curl -O https://raw.githubusercontent.com/tokifumi/docker/main/Dockerfile
（カスタマイズ済みのDockerfileをgithubから取得）

# ls（Dockerfileを確認）

# cat Dockerfile（Dockefileの内容を確認）

# docker build ./ -t centos7c（DockerfileからcentosTcというイメージを作成）

# docker images（centos7cのコンテナイメージができていることを確認）

# docker run -it centos7c（centos7cのコンテナイメージからコンテナ起動）

# less /etc/os-release（lessが使用できることを確認）

# ls -l（日付が日本語表示であることを確認）

# exit（ホストOSに戻る）
```

# コンテナ作成方法② Dockerfileでコンテナイメージをカスタマイズ

## Dockerfile

```
# ベースイメージ
FROM centos:7

# ロケールを日本語に変更とyesコマンドの無効化
RUN localedef -f UTF-8 -i ja_JP ja_JP.UTF-8 && ¥
ln -sf /usr/share/zoneinfo/Asia/Tokyo /etc/localtime && ¥
echo -e "export LANG=ja_JP.UTF-8 LANGUAGE=ja_JP:ja TZ=Asia/Tokyo¥nalias yes='printf '¥''¥''' >> /etc/bashrc && ¥
sed -i 's/LANG="en_US.UTF-8"/LANG="ja_JP.UTF-8"/g' /etc/locale.conf && ¥
# yum.confで日本語環境、ドキュメントをインストールするように変更
sed -i -e '/override_install_langs.*/d' -e '/tsflags.*/d' /etc/yum.conf

# 各種コマンドインストール
RUN rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7 && ¥
yum clean all && ¥
yum install -y less sudo psmisc which wget httpd mlocate bzip2¥
vim-enhanced vim-filesystem vim-common ¥
man-db man-pages-ja man-pages man-pages-overrides && ¥
# 日本語化、マニュアル作成のため全パッケージの再インストール
yum -y reinstall $(yum list installed | cut -f1 -d " " | sed 's/¥..*$//g' | grep -v '^[L|I].*ed$' | sed -z 's/¥n/
/g')

# manドキュメント追加
RUN mandb
```

# コンテナ作成方法② Dockerfileでコンテナイメージをカスタマイズ

Dockerfile

```
# コンテナ内ルートユーザーにパスワード設定
RUN echo "root:centos" | chpasswd

# ログインユーザーを作成
ARG UID=1000
RUN useradd -m -u ${UID} usera && ¥
# 作成したログインユーザーにパスワードを設定
echo "usera:usera" | chpasswd

# 作成したログインユーザーに切り替える
USER ${UID}
WORKDIR /home/usera/
RUN mkdir ダウンロード デスクトップ ビデオ 画像 テンプレート ドキュメント 音楽 公開

# コンテナ実行時のデフォルトコマンド
CMD ["/bin/bash"]
```

# コンテナによるLinux学習環境の事例デモ

弊社のインフラエンジニア養成機関であるエンジニアカレッジでは、Linux学習に先ほどのDockerfileでカスタマイズしたCentOS7コンテナと通常のUbuntuコンテナを利用しています。ここでは学習環境のデモをお見せします。

- エンジニアカレッジ「クラウド仮想環境」の構成
  - Amazon Web Service
  - 仮想マシン：EC2インスタンス(t3.micro)
  - ストレージ：30GB
  - LinuxOS：Almalinux 8.6
  - コンテナ：Docker 20.10.21

# シェルスクリプトを使用してコンテナを使い分ける

dockerコマンドはオプションが多いため、始めのうちは覚えるのが大変です。
慣れてきたら、コマンド入力の効率化を図るためにシェルスクリプトの作成がお勧めです。

次のスライドではDockerfileでカスタマイズしたCentOSとUbuntuの起動ができるシェルスクリプトを作成します。

# シェルスクリプトを使用してコンテナを使い分ける

## selectlinux.sh

```
# pwd（dockerディレクトリであることを確認）

# curl -O https://raw.githubusercontent.com/tokifumi/docker/main/selectlinux.sh
 （github上のselectlinux.shを取得）

# ls（selectlinux.shがあることを確認）

# cat selectlinux.sh（ファイル内容を確認）

# chmod u+x selectlinux.sh（所有者に実行権限を付与）

# ls -l selectlinux.sh（実行権限付与の確認）

# ./selectlinux.sh（シェルスクリプトの実行）
```

# シェルスクリプトを使用してコンテナを使い分ける

## selectlinux.sh

```
#!/bin/bash

while true
do
echo "起動するLinuxを選択してください"
echo "1)CentOS"
echo "2)Ubuntu"
read -p "1か2で選択してください: " ANS

case $ANS in
1)
echo -e "CentOSを起動します¥n"
docker run --rm -it --name centos7c-$(date +%s) -h centos7c centos7c
break
;;

2)
echo -e "Ubuntuを起動します¥n"
docker run --rm -it --name ubuntu-$(date +%s) -h ubuntu ubuntu:latest
break
;;

*)
echo -e "無効な選択肢です¥n"
continue
;;
esac
done
```

while trueで終了ステータスが0である間は繰り返します

日付のシリアル値を利用してコンテナ名を一意にしています

--rmオプションでコンテナをexitした時に自動的に削除するようにしています

breakで終了ステータスを1にして繰り返しを終了します

continueで終了ステータスを0にするので、再度繰り返します

# コンテナの便利な使い方

**デモ1**
コンテナのシェル操作中にキーボードのCtrlを押しながらpとqを連続で押すことで、コンテナから切断（デタッチ）できます。
ホストOSのシェルに戻るので、別のコンテナを起動できます。
元のコンテナに戻るときは、docker attach （アタッチ）で接続します。

**デモ2**
誤ってコンテナ内の/etcにある重要なファイルを削除してしまっても、コンテナを破棄して、コンテナイメージからコンテナを作り直すことで元の状態に戻すことができます。

# まとめ

- コンテナを使用すれば、容易にLinux学習環境を構築できる
- 複数のLinuxディストリビューションが簡単に利用できる
- Dockerfileでコンテナを学習内容に合わせてカスタマイズできる
- シェルスクリプトを使用して、簡単にコンテナを使い分けることができる

# コンテナに関する技術解説セミナー

コンテナに関しては、他の講師の方や私の過去のセミナーでも扱っていますので併せてご覧いただければ幸いです。

- 仮想マシン・コンテナの概念と利用
- コンテナを体験してみよう！
- コンテナ技術・Dockerの解説（Linux学習）
- コンテナの仕組み/Dockerコンテナとコンテナイメージの管理
- コンテナ技術について