Dockerコンテナ内からDockerを使うことについて

April 30, 2019

Docker
docker

Docker内からDockerを使う目的

CI用途で使う人が多いのではないでしょうか？ CIでビルドなりテストなりやるとき、毎回同じ環境で実行したいと思う サーバ上で直接ジョブ動かしちゃうと、ゴミが残る可能性があったりサーバに直接ジョブ実行に必要なjavaなりrubyなりを突っ込む必要が出てくるので、どんどん汚れていく

Dockerでビルドやテストを実行すれば毎回終わったら消せばいいので、クリーンにできるよねーって話 CIサーバを直接yumなりでインストールして、そこからDocker使うのであれば特に問題ないけど 最近はCIサーバ自体をDockerコンテナで立ち上げて、さらに中でDockerを呼ぶというのがよく使われている気がする その時のやり方の話です。

なぜDockerコンテナ内からDocker使えないか

前提としてDockerの仮想化の仕組みは今までのHypervisorとかと少し違って ベースとなるイメージや、ホストのカーネルなど多くの部分を共有して、コンテナごとに少ない書き込み領域を用

意することで、高速起動, 軽量などを実現しています.

TIS IT Holdings Group | Go Beyond

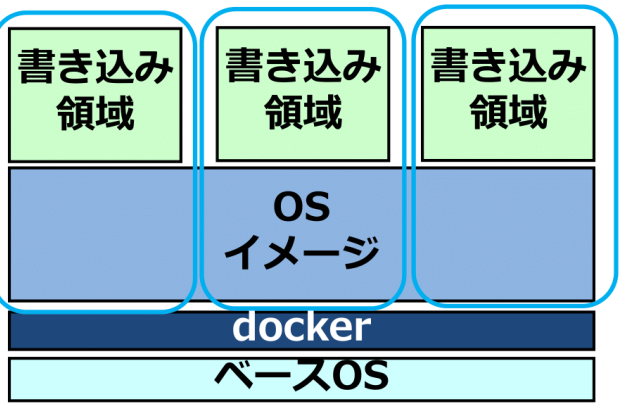
1. dockerとは？

③ コンテナはリソース消費量が少ない

● 同一OSのコンテナを多数起動する場合、OSイメージのファイルは各コンテナで共有される

各コンテナごとに書き込み領域を提供することで独自性を確保

OSイメージは同一OSのコンテナで共有するため、システム内に１つ存在すればよい



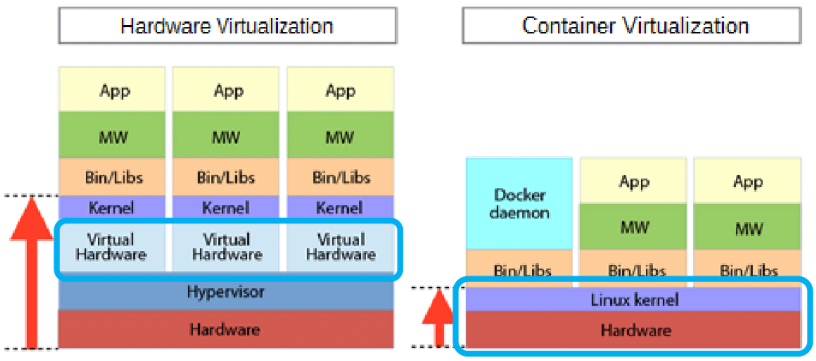
同一OSのコンテナを多数起動する場合、各コンテナの差分データのみがハードディスク上に保持されるため、**ハードディスクの消費量が大きく削減**できる

そしてセキュリティのため、コンテナ内からいくつかの操作は制限されます Dockerの操作は、Dockerデーモンに対して、クライアントを利用して接続しています. Dockerデーモンはホストに対して下の制約に引っかかることをやっているので、コンテナ内からは起動できないです

2．VMとdockerの違いとは？

② Dockerのコンテナのサーバ仮想化

- アクセスが制限されたファイルシステムをマウントしたプロセス内で独立した実行環境を構築



出典:Research at http://research.worksap.com/research/docker-20140724-2/

ベースOSと完全に分離してないため、**セキュリティー上の制約**が存在する。さらに**ハードウェアやKernelに影響するプロダクトは利用できない**場合がある

2．VMとdockerの違いとは？

できないことの一例

- fdiskコマンドやSoftware iSCSIやDRBDなどディスクデバイスに対して操作が発生する製品
  - dockerコンテナ内からでは/dev/sdxなどのデバイスを参照することも操作することもできない
- systemctlなどのサービスを制御するコマンド
  - RHEL 7.x系ではサービスの起動、停止を行うsystemcltコマンドがコンテナ内では使用できない
- rebootなどのサーバを再起動するコマンド
  - systemctl不可を再起動で回避することもできない

docker環境では**ハードウェアはベースOS上で管理**し、コンテナ内で利用するサービスは**コンテナ生成時にdockerfile等でインストール**を行うのが通常手段である

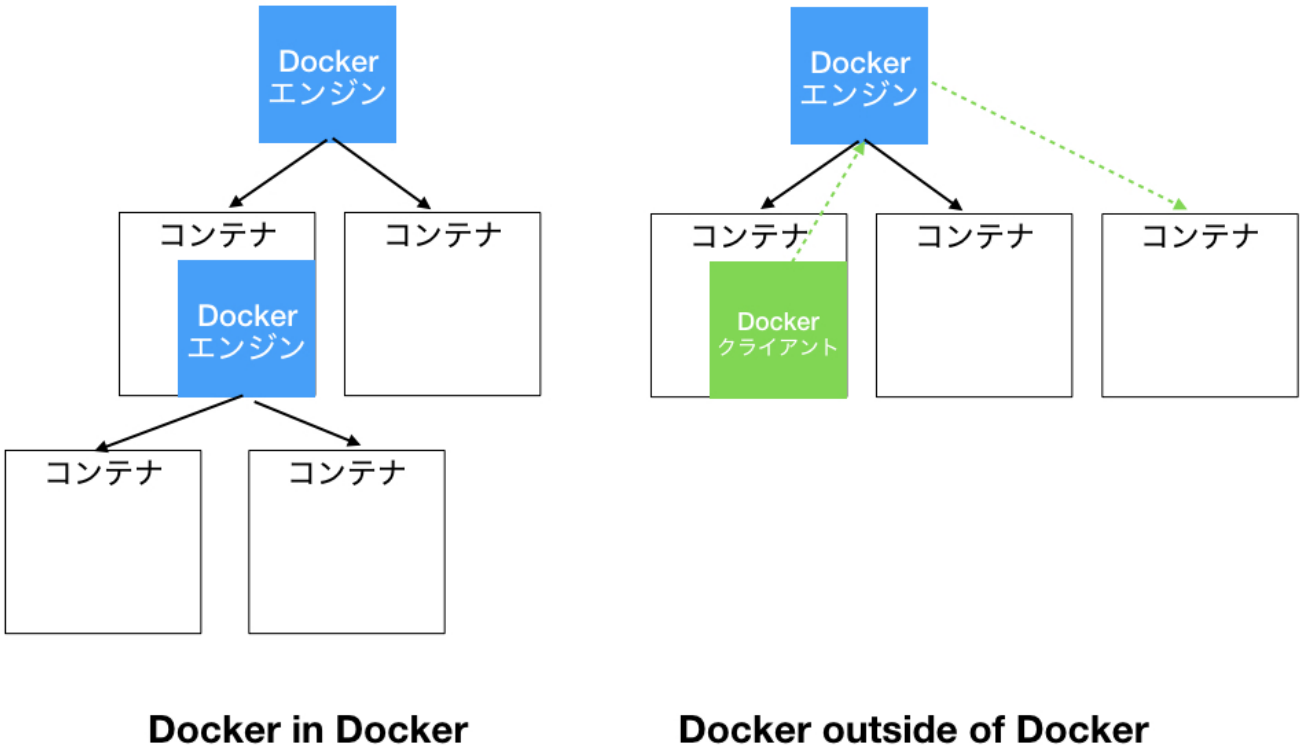
上記スライドの引用元

DockerコンテナからDockerを使うには？

2種類の方法があります

- 権限を付与して、コンテナ内からホストのリソースを自由に扱える権利を持たせる(–privilegedオプションをつける) – Docker in Docker(DinD)
- ホストのDockerデーモンのソケットファイルをvolume接続(マウント)して、そことやりとりするようにする – Docker outside of Docker (DooD)

イメージとしては下のような感じ



参考元

簡単な違い

参考1

わかりやすい違いとしては、DinDではホストと、コンテナで完全にデーモンが分かれるので イメージ・コンテナ管理なども別々になります, これは管理しやすいという利点でもありますが、ディスク容量を大量に消費する要因にもなります

DooDではデーモンを共有するため、ホストとコンテナ内で共通のイメージ・コンテナ管理を行います。 利点は↑と逆ですね

シェルで確認してみましょう

DinD

```
$ sudo docker run --privileged --name dind -d docker:dind
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
cba616b2250c        docker:dind         "dockerd-entrypoint.…"   8 seconds ago       Up 7 sec
// dockerコンテナに入ってみる
$ docker exec -it dind sh
// dockerコンテナ内でイメージ起動
/ # docker run -d -p 80:80 --name webserver nginx

// nginxのコンテナしか見えない
/ # docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
99e66f209820        nginx               "nginx -g 'daemon of…"   3 seconds ago       Up 2 sec

// ホストに戻って確認
/ # exit

// dindコンテナしか見れない
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
cba616b2250c        docker:dind         "dockerd-entrypoint.…"   4 minutes ago       Up 4 mir

// 消しておく
$ docker stop cba616b2250c && docker rm cba616b2250c
```

DooD

```
// doodコンテナ起動
$ sudo docker run -v /var/run/docker.sock:/var/run/docker.sock -it --name dood -d docker
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
4cfa187ac191        docker              "docker-entrypoint.s…"   3 seconds ago       Up 1 sec

// コンテナに入る
akasetnoMacBook-puro% docker exec -it dood sh

// コンテナ内からホストで動いているコンテナ観れる
```

```
/ # docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
4cfa187ac191        docker              "docker-entrypoint.s…"   18 seconds ago      Up 16 se

// コンテナ起動
/ # docker run -d -p 80:80 --name webserver nginx
/ # docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
be56745b1af4        nginx               "nginx -g 'daemon of…"   7 seconds ago       Up 6 sec
4cfa187ac191        docker              "docker-entrypoint.s…"   52 seconds ago      Up 51 se

// ホストに戻る
/ # exit
// ホストからもコンテナ内で起動したコンテナが見れる
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
be56745b1af4        nginx               "nginx -g 'daemon of…"   14 seconds ago      Up 12 se
4cfa187ac191        docker              "docker-entrypoint.s…"   59 seconds ago      Up 57 se
```

どっちを使えばいいの？

参考2

CI用途に関してはDooDを使うのが好ましいと思います. DinDの開発者自身がブログでDinDのCI利用について述べています https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/

ざっと要点

- そもそものDinDの用途はDockerの開発プロセス高速化のためだった
- DinDは次の問題がある
  - SELinuxとかをホストとコンテナで別設定にしていると、クラッシュする可能性がある
  - ホストとコンテナで別々のファイルシステムを使っているとクラッシュする可能性がある
  - /var/lib/dockerはdockerデーモンの専用領域みたいなものだから、別デーモン作って触らせると何が起きても知らないよ
- CIをやりたいならDooDでいいんじゃない？
  - ホストとDockerデーモンを共有することで、ビルドごとにイメージのキャッシュが消えたりがなくなると思うよ
  - 上の問題点も解決すると思うよ