コンテナ仮想化（Docker互換－Rancher Desktop）

目次

コンテナ仮想化（Docker互換－Rancher Desktop）

はじめに

「オープンソース」のソフトウェアはソースを公開していますが、特許や著作権、商標等の知的財産権を放棄しているわけではありません。指定のライセンス条件下でのみ利用することができ、条件が変更されたりソフトウェアの一部（コアのみ）だけがオープンソースになっている場合もあります。

Docker は Docker 社が GitHub に公開しているオープンソース（リポジトリが複数存在します）で、2022 年 2 月 1 日から条件付きで有償になりました。これに先立ち、Red Hat 社は「RHEL 8 から Docker コンテナーエンジンと、docker コマンドが削除されました。」と発表[1]し、コンテナ運用管理の主力ソフトウェア Kubernetes は技術的な理由で v1.20 以降 Docker を非推奨にしました[2]。

ミドルウェアやアプリをインストール済のプラットフォーム（コンテナ）をひな形イメージから簡単に作ることができ、公開されている（Docker Hub[3]）イメージが多数あるので、コンテナを使い続けることができる Docker 代替えソフトウェア群の開発が活発に行われています。

1. Docker の構成と有償化部分

Docker は Linux 環境を対象[4]に開発され、主要な機能に以下のものがあります（一部）。

・ひな形イメージからカスタマイズしたイメージの作成（コマンド：buid）
・コンテナの起動（コマンド：start, run）
・コンテナ間の連動やネットワーク管理（コマンド：docker-compose）
・コンテナの監視や削除、シェルを含むコマンド実行（コマンド：ps, rm, exec）

Docker はサーバの Docker デーモン(dockerd)でコンテナを操作したり監視を行い、操作コマンドと操作の内容をサーバに送信するクライアント(docker)で構成されています。

クライアントはホスト OS 用の実行可能ファイルで、Linux や Windows、Mac 等ホスト OS 毎の実行環境（Windows であれば exe ファイル）が必要になります。Docker 社は自社サイトで公開している実行環境毎のコマンドや GUI の監視ツールを同梱した“Docker Desktop”を有償化しました。

2. Docker の互換ソフトウェア

Docker はコンテナランタイム、イメージのビルド、ネットワーク等の機能で構成されています。現在では各機能毎に類似・互換可能なコンポーネントがそれぞれ（オーナー組織／リポジトリ）で開発されていて(containerd/containerd⇔opencontainers/runc、moby/buildkit⇔containers/buildah、moby/libnetwork⇔containernetworking/cni、…)、色々な組合せで Docker を代替えします。

---

[1] [Red Hat Customer Portal] 1.3. Docker を使用せずにコンテナーを実行
https://access.redhat.com/documentation/ja-jp/red_hat_enterprise_linux/8/html/building_running_and_managing_containers/con_running-containers-without-docker_assembly_starting-with-containers

[2] Don't Panic: Kubernetes and Docker https://kubernetes.io/blog/2020/12/02/dont-panic-kubernetes-and-docker/

[3] Docker Hub https://hub.docker.com/search?q=

[4] マイクロソフト社は Docker Desktop で Windows が動作する Windows コンテナを公開しています
https://learn.microsoft.com/ja-jp/virtualization/windowscontainers/about/

コンテナ仮想化（Docker互換－Rancher Desktop）

＜Docker 代替え候補＞

Podman[5] ：Docker が“対応外”となった RHEL 8 で推奨しています。コマンドが podman に変わりますが、ほぼ同一のサブコマンドが使えます。イメージの作成(buid)に使う Dockerfile は Docker のものを流用できますが、Docker-Compose の設定ファイルはコンテナを連動させる方法がポッド(Pod)に変わったため使えません。

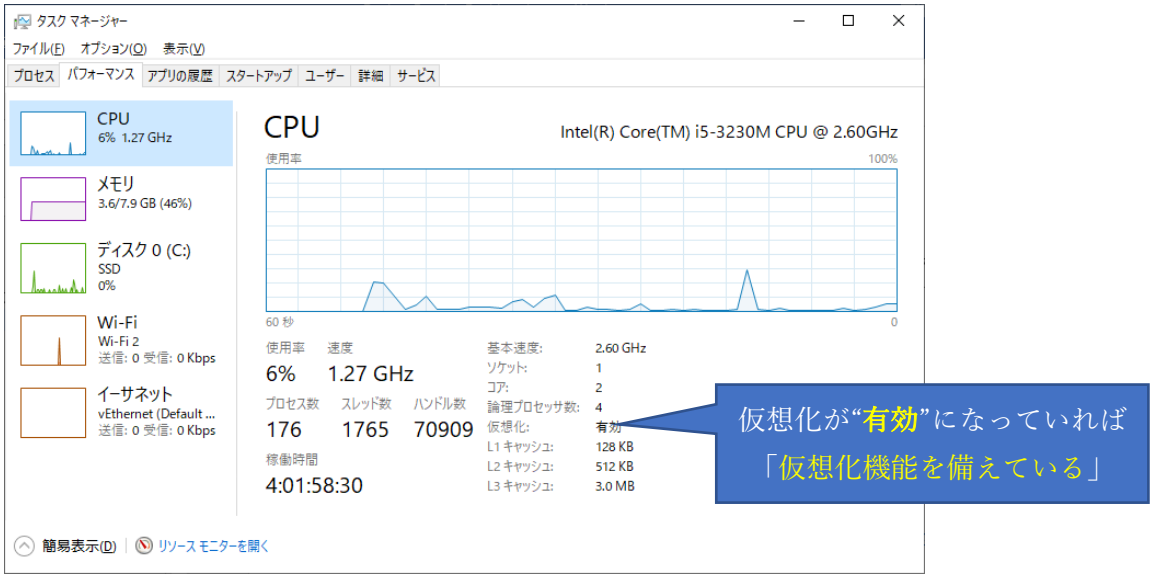
nerdctl[6] ：コマンドが docker から nerdctl に変わる以外は Docker と互換・機能追加があります。Widows 版の実行ファイルもありますが、他に以下の追加インストールが必要です。
追加インストールが必要なコンポーネント… containerd、BuidKit、CNI

Rancher Desktop[7]：nerdctl と Docker(moby)から選択でき、Docker(moby)を選択すると、docker コマンド（Docker-Compose を含む）がそのまま使えます。

3. Rancher Desktop のセットアップ

以下、Rancher Desktop のバージョン 1.8.1 を Windows 10 Pro 22H2 の環境で試行します。
Rancher Desktop 1.8.1 は Apache-2.0 ライセンス（商用利用可）です。
＜必要な環境＞Windows では Rancher Desktop には次のものが必要です[8]。

・Windows 10 ビルド 1909 以降。Home エディションもサポート対象です
・仮想化機能を備えたマシン
・永続的なインターネット接続



[5] Podman(ダウンロード Windows/Mac/Linux) https://github.com/containers/podman-desktop
[6] nerdctl https://github.com/containerd/nerdctl
[7] Rancher Desktop(ダウンロード) https://rancherdesktop.io/
(リポジトリ) https://github.com/rancher-sandbox/rancher-desktop/
[8] (Windows 環境) https://docs.rancherdesktop.io/getting-started/installation#windows

コンテナ仮想化（Docker互換－Rancher Desktop）

3.1. ダウンロード

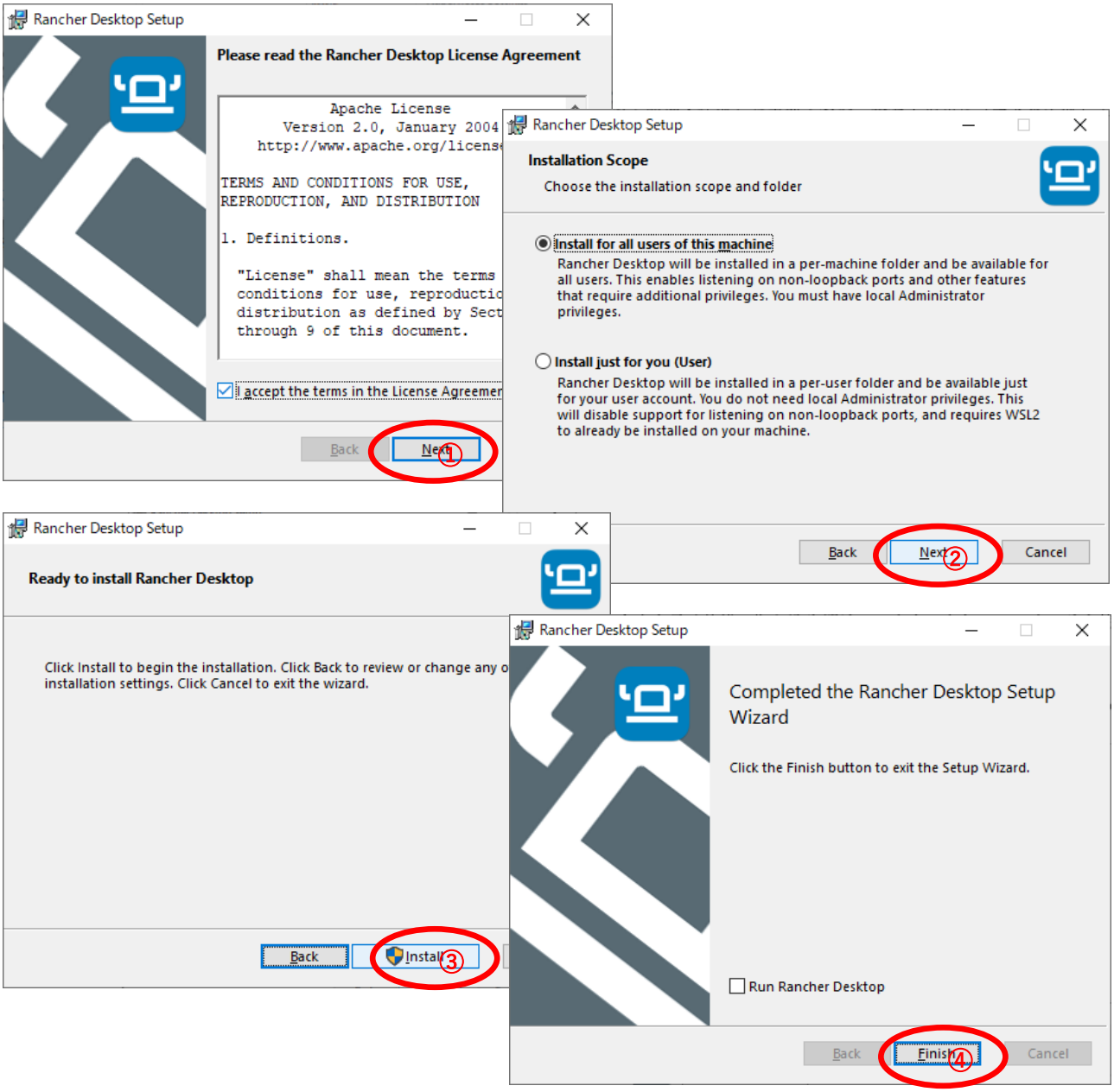
https://rancherdesktop.io/から Windows 用のインストーラをダウンロードします。

3.2. インストール

Rancher Desktop には Windows 上の Linux 用 Windows サブシステム（WSL2）[9]が必要で、これは Rancher Desktop セットアップの一部として自動的にインストールされます。

以下は、WSL2 を設定済の環境で Rancher Desktop のインストーラを実行した例です。

（1）インストール時の選択肢は②の画面の利用者の範囲（全利用者／インストール者のみ）だけで、どちらを選んでも機能に違いはでません。



[9] 手動でインストールする場合は管理者モードのコマンドプロンプトまたは PowerShell から wsl –install を実行します。 https://learn.microsoft.com/ja-jp/windows/wsl/install

コンテナ仮想化（Docker互換－Rancher Desktop）

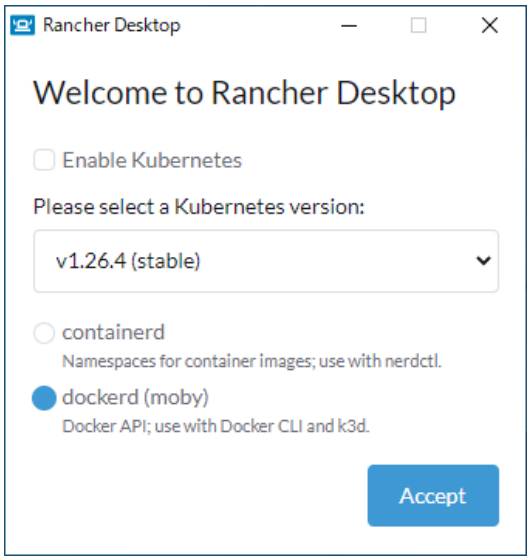
【インストールされるファイル…Windows 関連以外は省略しています】

```
C:¥PROGRAM FILES¥RANCHER DESKTOP
|   chrome_100_percent.pak
|   chrome_200_percent.pak
|   d3dcompiler_47.dll
|   ffmpeg.dll
|   icudtl.dat
|   libEGL.dll
|   libGLESv2.dll
|   LICENSE.electron.txt
|   LICENSES.chromium.html
|   Rancher Desktop.exe
|   resources.pak
|   snapshot_blob.bin
|   v8_context_snapshot.bin
|   vk_swiftshader.dll
|   vk_swiftshader_icd.json
|   vulkan-1.dll
|
+---locales
|       ja.pak   （他略）
\---resources
    \---resources
        +---darwin
        +---icons
        +---linux
        +---rancher-dashboard
        \---win32
            |   distro-0.34.tar
            |   wsl-helper.exe
            |
            +---bin
            |       docker-buildx.exe
            |       docker-compose.exe
            |       docker-credential-ecr-login.exe
            |       docker-credential-none.exe
            |       docker-credential-wincred.exe
            |       docker.exe
            |       helm.exe
            |       kubectl.exe
            |       kuberlr.exe
            |       nerdctl.exe
            |       rdctl.exe
            |
            \---internal
                    host-resolver.exe
                    host-switch.exe
                    privileged-service.exe
                    steve.exe
                    vtunnel.exe
```


（２）初回起動の構成選択

インストール後の初回起動時に containerd または docker(moby)のどちらを使うか選びます。docker(moby)を選択すると操作コマンド（CLI）は Docker と同様“docker”になります。

(Enable Kubernetes のオプションは、Kubernetes を使う場合に選択…説明は割愛します)

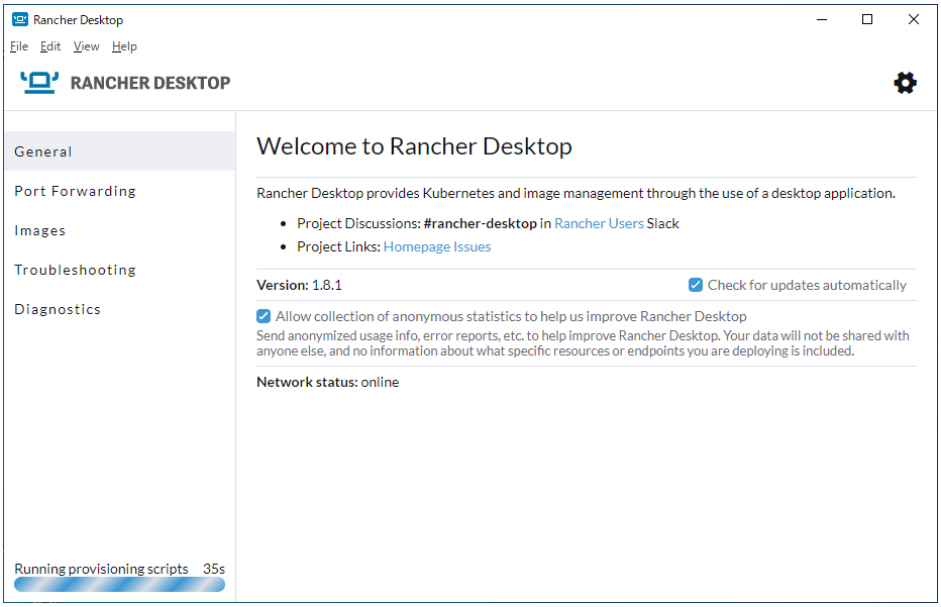


4. Rancher Desktop の起動と終了

（１）起動

仮想マシン（コンテナ）の起動やイメージの操作の前にインストーラが作ったショートカットか Windows スタートボタンから C:¥PROGRAM FILES¥RANCHER DESKTOP¥Rancher Desktop.exe を起動します。

または Rancher Desktop に同梱の rdctl.exe を使い、rdctl start コマンドでも起動できます。



画面左下に表示されたプログレスバーが消えたら起動完了で、以降 docker コマンドが使えます。

（２）終了

画面の File メニューから Exit を選択すると Rancher Desktop.exe 終了します（閉じるボタンで画面を消しても終了しません）。またはコマンドプロンプト等から rdctl shutdown と打鍵します。

コンテナ仮想化（Docker互換－Rancher Desktop）

5. カスタムイメージの作成

操作方法は Docker と同一です。Docker Hub からひな形になるイメージをダウンロードして自分の環境にカスタマイズしたイメージを作るために、以下の作業を行います。

（1）Dockerfile の作成

Docker で使用したものがそのまま使えます。

以下、Docker Desktop のバージョン 3 で使用したファイルの例(フォルダと Dockerfile を作る)。

```
C:¥TOOLS¥POSTGRES
   └Dockerfile
```

----Dockerfile の内容----

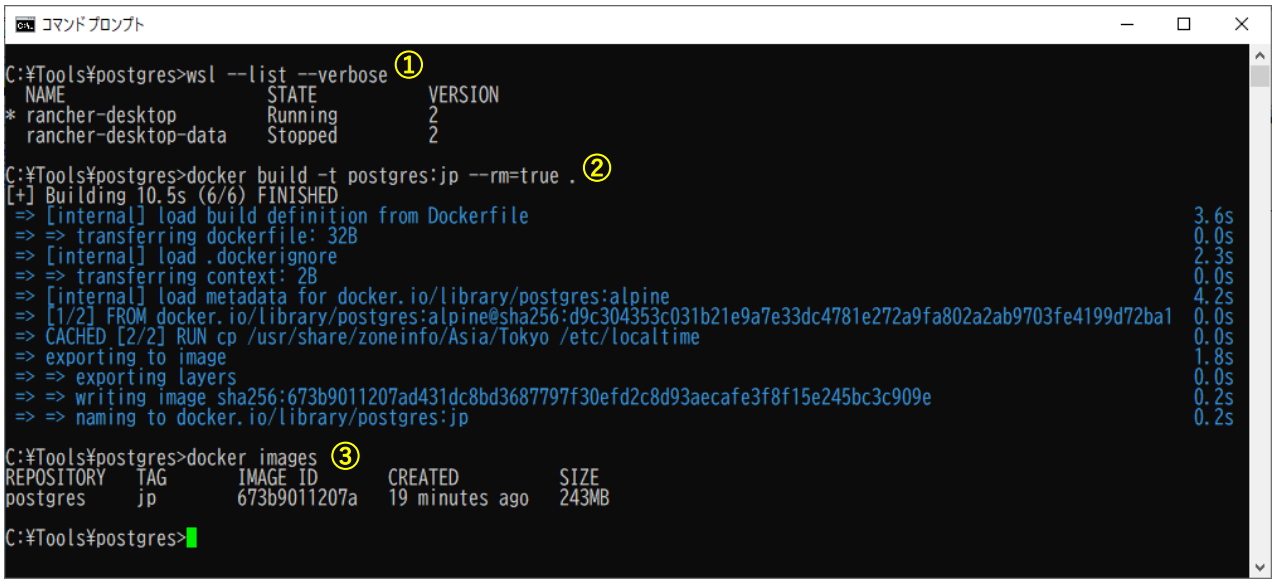
```
FROM postgres:alpine
# 日本語環境に設定… image 作成[ docker build -t postgres:jp --rm=true . ]
RUN cp /usr/share/zoneinfo/Asia/Tokyo /etc/localtime
ENV LANG ja_JP.utf8
-------------------------
```

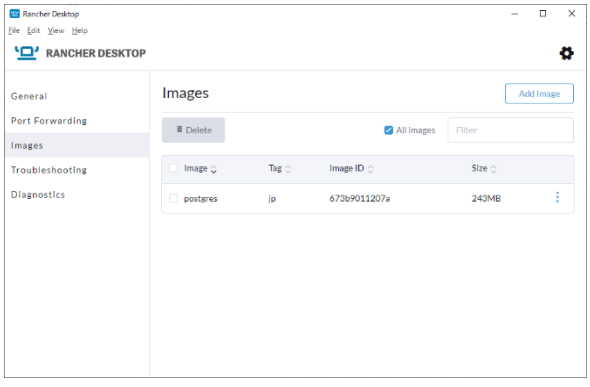
（2）docker build コマンドの実行

① wsl --list –verbose で rnacher-desktop が Running になっているのを確認します

② docker build -t *リポジトリ名:タグ* (--rm=true で中間イメージ削除) *Dockerfile のパス*
を実行するとイメージの取得が始まります

③ docker images コマンドで ②で指定した *リポジトリ名：タグ* が表示されれば成功です





作成されたイメージは Rancher Desktop の GUI からも確認したり操作できます。

コンテナ仮想化（Docker互換－Rancher Desktop）

6. イメージの起動（docker-compose コンテナ化）

Docker（ver3.1）で使用したものがそのまま使えます。

（１）docker-compose.yaml

起動環境を記述した docker-compose.yaml を作ります。

以下、Docker Desktop のバージョン３で使用したファイルの例

```
C:¥TOOLS¥POSTGRES
  ├docker-compose.yaml
  └Dockerfile
```

---- docker-compose.yaml の内容----

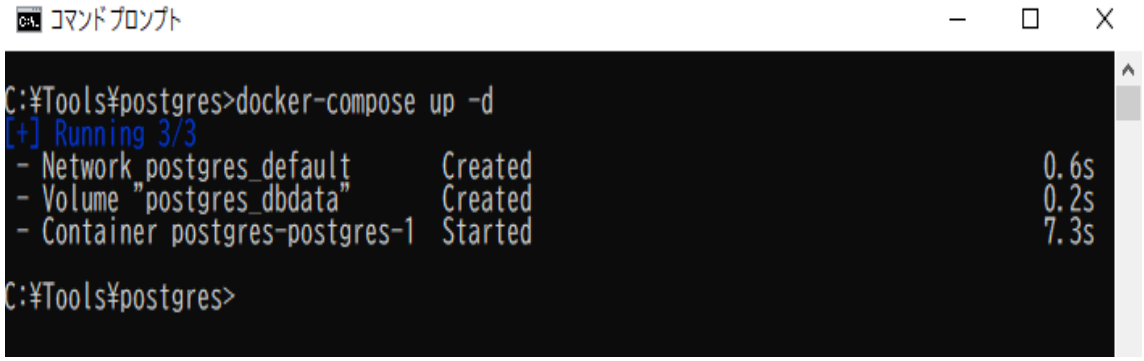
```
version: '3'
services:
  postgres:
    image: postgres:jp
    ports:
      - "5432:5432"
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin
    volumes:
      - dbdata:/var/lib/postgresql/data
      - ./docker-entrypoint-initdb.d:/docker-entrypoint-initdb.d

volumes:
  dbdata:
```

（２）docker-compose コマンドの実行

docker-compose.yaml ファイルのディレクトリに cd 後に以下のコマンドを実行します。

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



docker-compose.yaml で参照しているフォルダ docker-entrypoint-initdb.d は自動生成されます

ホスト OS とディレクトリやファイルが共用でき更新もできますが、利用者権限や inode に関係する機能は違いが出るのでコンテナ側からは更新しない方が無難です。

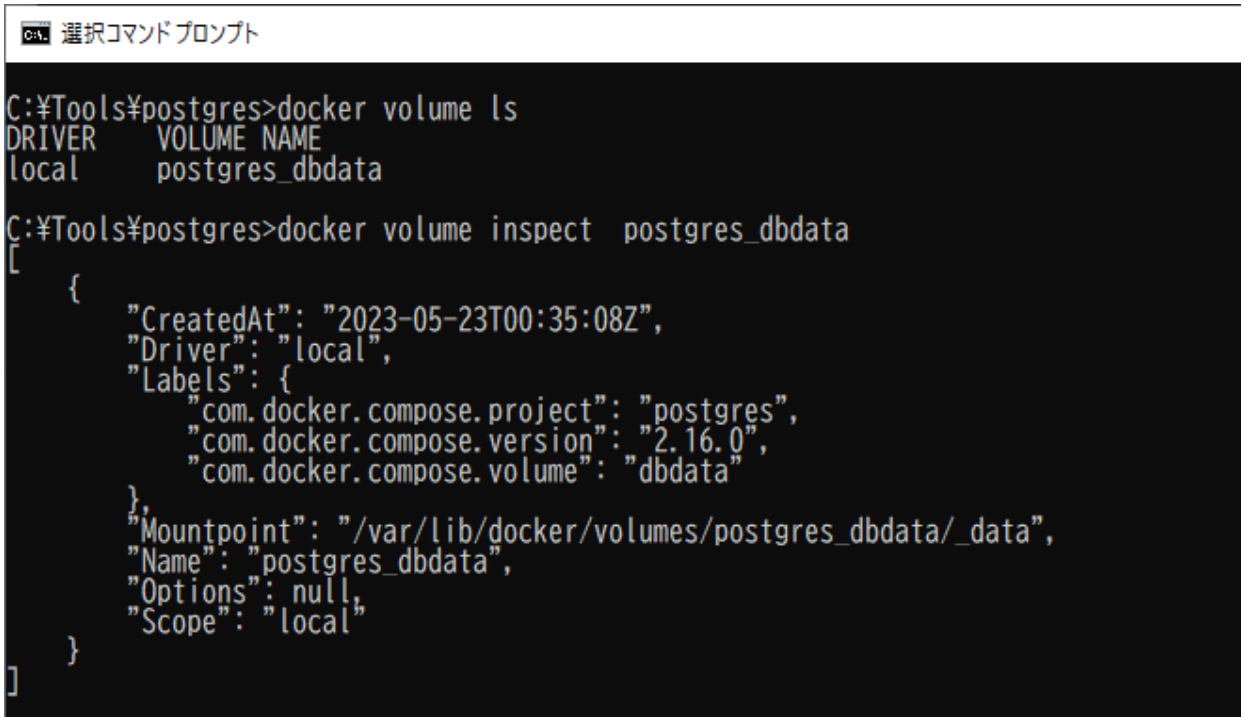
コンテナ仮想化（Docker互換－Rancher Desktop）

その他の点も Docker と同様に動作し、docker-compose.yaml に指定された資材が作られます。

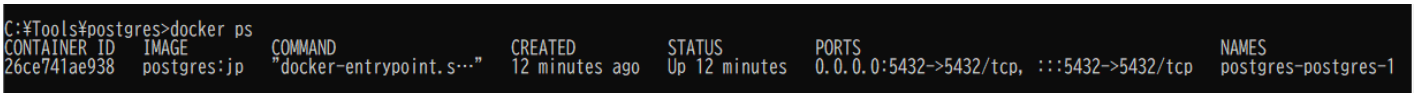
① 初回の docker-compose によるコンテナ生成で名前付ボリュームが作られ、以下のコマンドで確認できます。

・docker volume ls

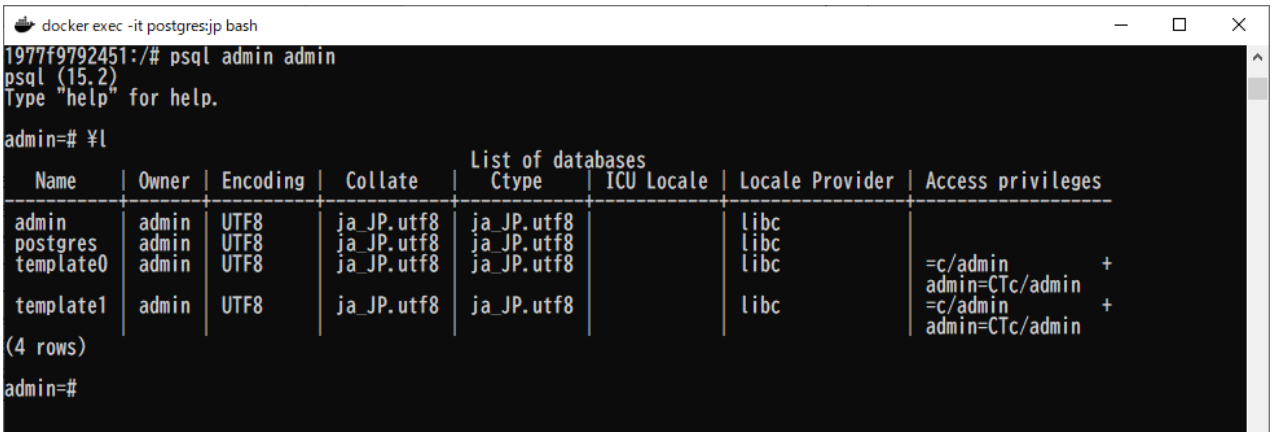
・docker volume inspect *ボリューム名*



② ports で指定したポートフォワーディングが有効になっているのは、docker ps コマンドで表示される「PORTS」で確認できます



③ PostgreSQL のコンテナは docker-compose.yaml に指定した POSTGRES_USER のデータベースが作られ、POSTGRES_PASSWORD に指定したパスワードでデータベースに接続できます。



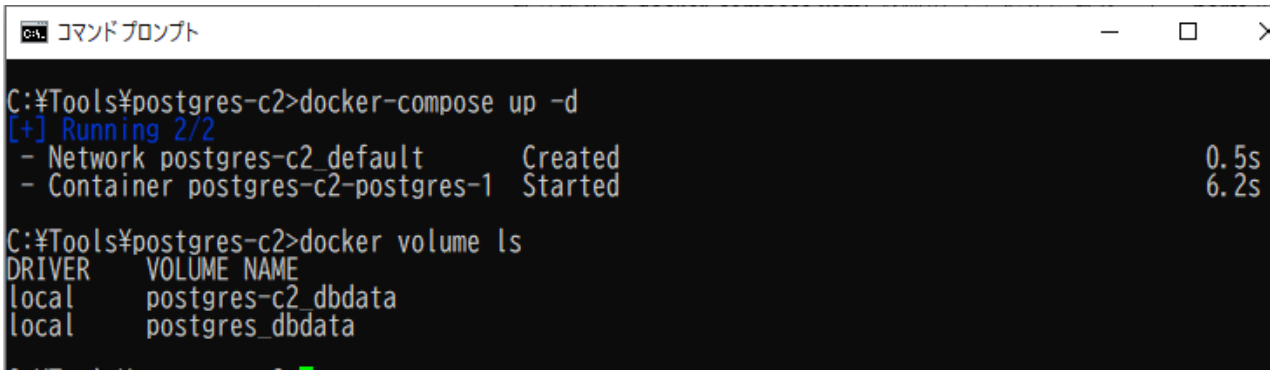
コンテナ仮想化（Docker互換－Rancher Desktop）

7. 同一イメージから複数コンテナの生成

コンテナは docker-compose.yaml を別のフォルダにコピーし、ports のホスト側（左）を重複しない値に書き替えるだけでコンテナを生成させることができます。

```
ports:
  - "nnnn:5432"
```

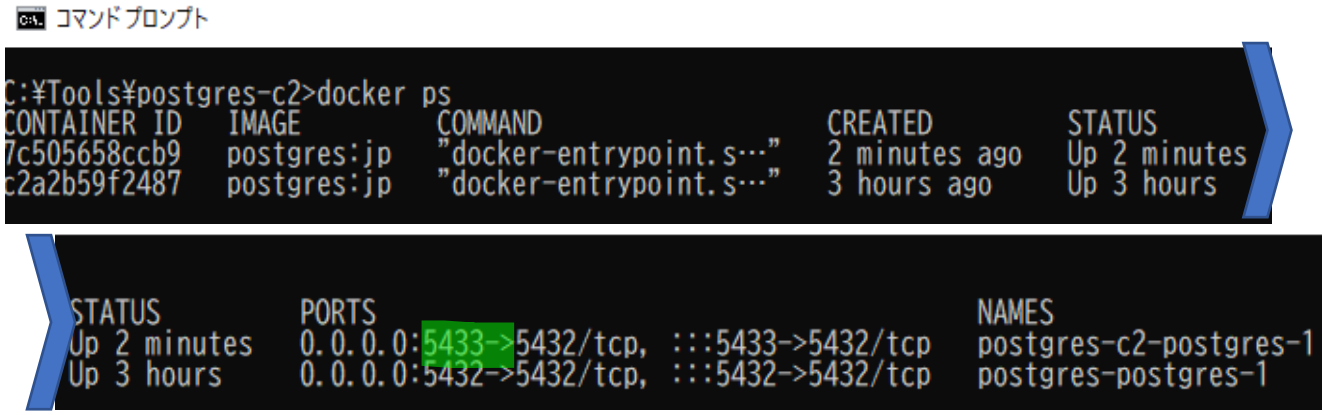
<ホストのポート>:<コンテナ・アプリのポート>



同一イメージから生成したコンテナ名

コンテナの名前はデフォルトでは、docker-compose.yaml の格納フォルダ名が先頭に付きます。

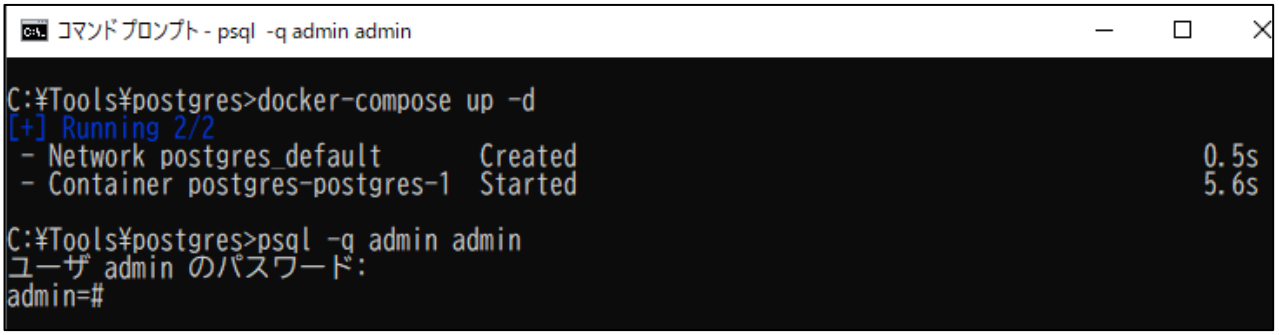
postgres-c2 という名前のフォルダに格納した場合は、"postgres-c2-"がイメージ名（タグ除く）の先頭に付加されたコンテナ名になります。



8. コンテナ上のサービス利用

コンテナ上のサービスは docker-compose.yaml の ports で指定したポート経由で利用します。

※この例では port パラメータを省略してホストの 5432 ポート（Postgres のデフォルト）から接続



9. コンテナの状態と停止・削除

（1）docker-compose

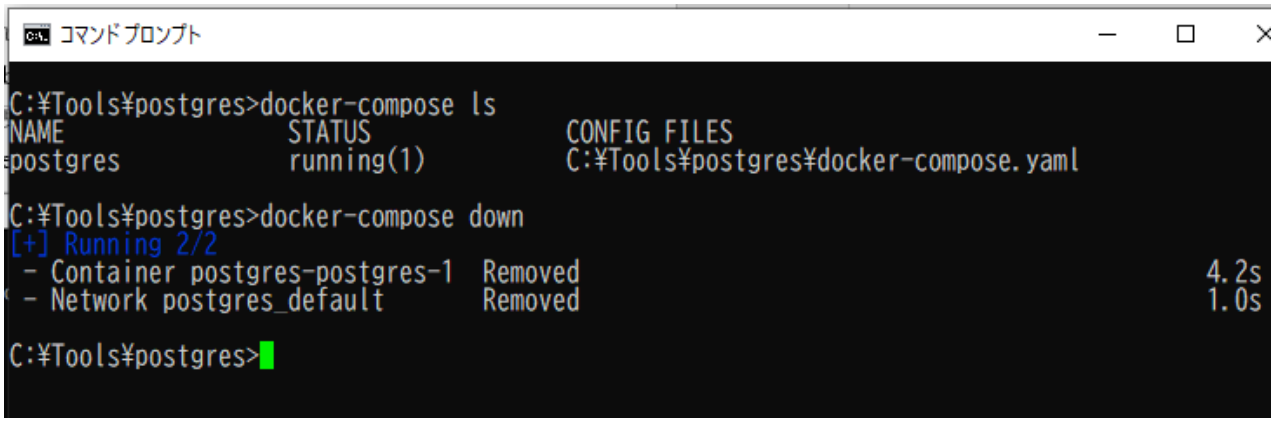
docker-compose up で生成・開始したコンテナは docker-compose のサブコマンドで監視・操作します。

リスト表示：docker-compose ls

停止・削除：docker-compose down

※down をオプションなしで実行すると削除されるのはコンテナとネットワークです。

docker-compose up で 生成されたコンテナは docker-compose down で削除されます



（2）docker ps -a/stop/rm

docker run コマンドで生成・起動したコンテナは、docker stop で停止します。

docker rm *ID or NAMES* でコンテナを消去すると docker ps -a で表示されなくなります

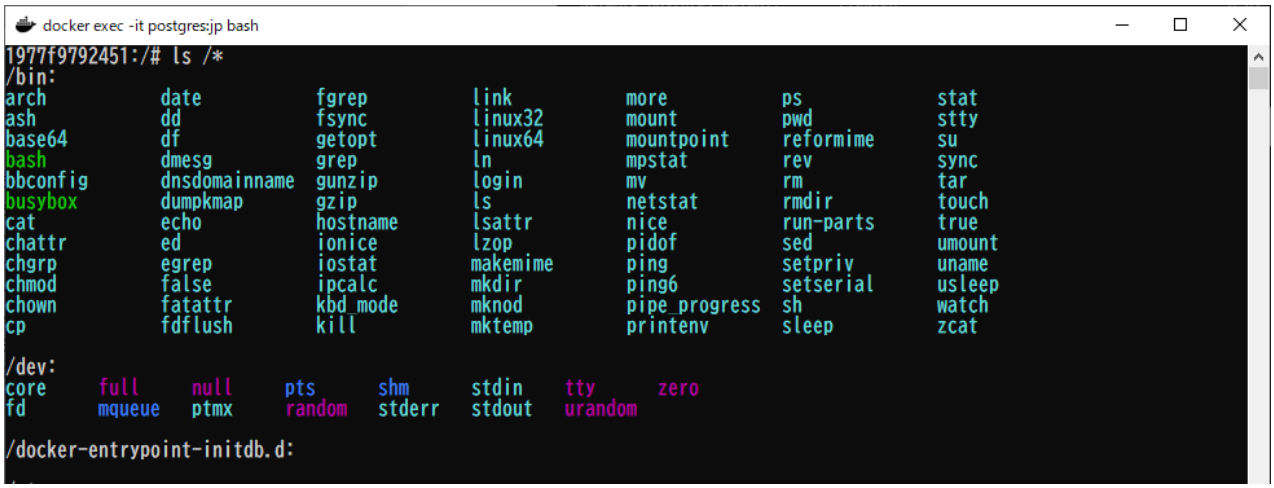
10. コンテナへのターミナル接続

`docker-compose up` の実行時に表示された Container 名を指定して docker exec -it コマンドで接続できます。

`start "docker exec -it postgres:jp bash"` `docker exec -it postgres-postgres-1 bash`

注）コマンド例の前半[start "docker exec -it postgres:jp bash"]は新しい画面を開くためのもので、

docker exec コマンドとは直接関係はありません



コンテナ仮想化（Docker互換－Rancher Desktop）

11. ホストとコンテナのネットワーク

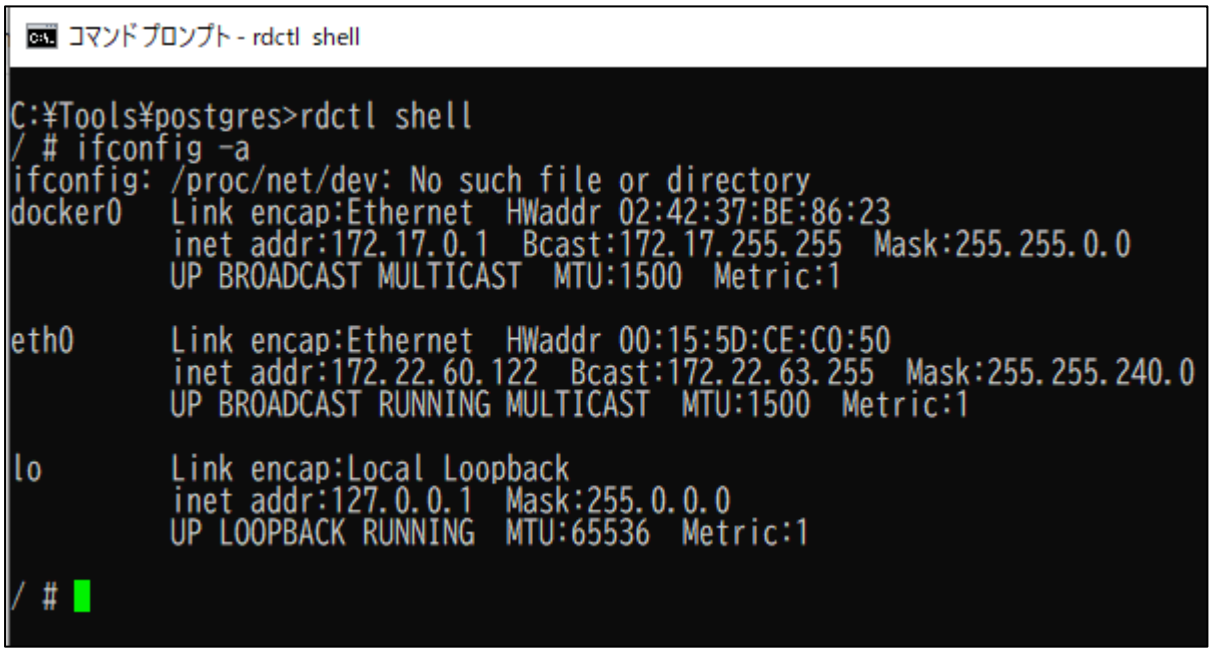
ネットワークに関連する機能は Windows OS も Docker、Rancher コンテナ側も更改を続けている[10][11][12]のでこの資料ではデフォルト設定に絞って記述しますが、Rancher Desktop 1.8.1 を Windows 10 Pro 22H2 以降の環境では変わっている可能性があります。

デフォルトでは WSL2 が作成した“イーサネット アダプター vEthernet (WSL)”（仮想スイッチ）を通してコンテナのネットワークとホスト側のネットワークを接続する構成になります。

11.1. ゲートウェイ

コンテナはゲートウェイを経由して仮想スイッチと繋がります。コンテナは固有の IP アドレスとゲートウェイの IP アドレスを持っています。ゲートウェイは rdctl shell コマンド（Rancher Desktop に同梱）で Rancher Desktop の vm に接続すると確認できます。

起動直後（コンテナが起動していない）のネットワーク・インタフェースは以下になっています。



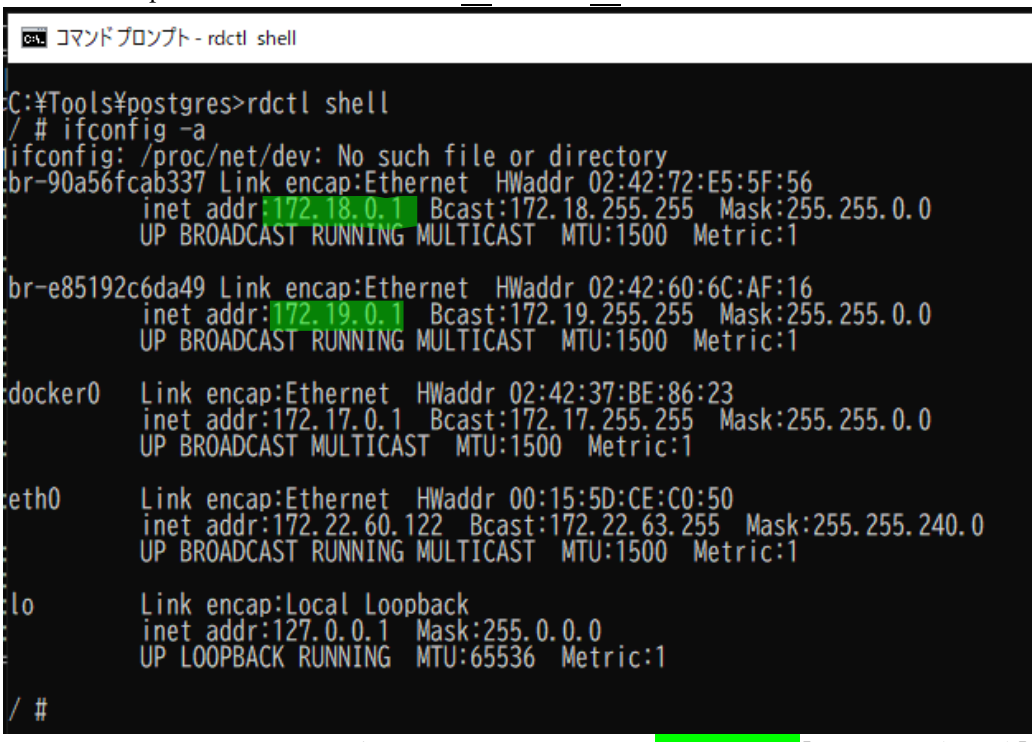
---

[10] Linux Containers on Windows https://learn.microsoft.com/ja-jp/virtualization/windowscontainers/container-networking/advanced#linux-containers-on-windows

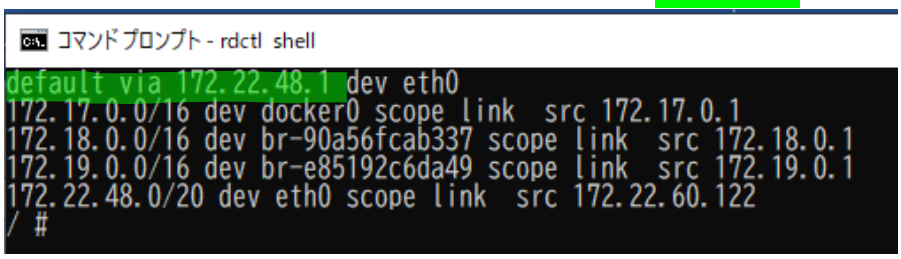
[11] Moby Linux VM では、Docker for Windows (Docker CE の製品) と共に DockerNAT スイッチを使用しています https://learn.microsoft.com/ja-jp/virtualization/windowscontainers/container-networking/advanced#moby-linux-vms-use-dockernat-switch-with-docker-for-windows-a-product-of-docker-ce

[12] Windows Containers side by side with Rancher Desktop - for windows containers implementation ideas #3999 https://github.com/rancher-sandbox/rancher-desktop/issues/3999
Runcher Desktop issues（ネットワーク関連の作業中の課題) https://github.com/rancher-sandbox/rancher-desktop/issues?q=is%3Aissue+is%3Aopen+network

docker-compose で起動すると、172.18.1 , 172.19.0.1 , … と順次にインタフェースが追加され、↙



コンテナからの下り通信はデフォルトゲートウェイ 172.22.48.1[vEthernet(WSL)]に送られます。



11.2. コンテナからホスト OS

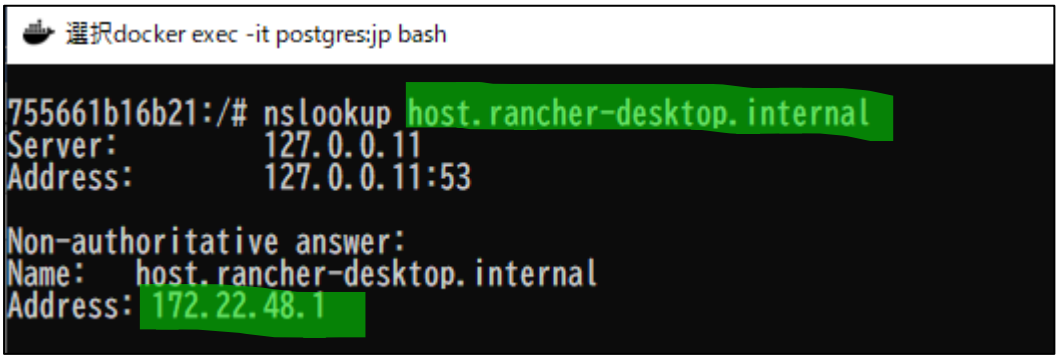
コンテナ側からホストにアクセスするには、以下の 2 つのホスト名を使うこととができます。

host.rancher-desktop.internal

host.docker.internal

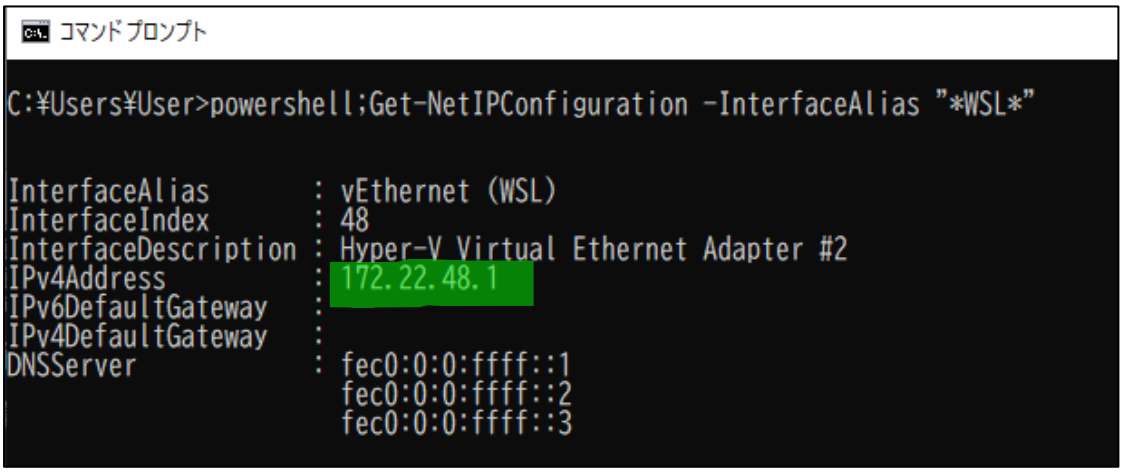
コンテナ内で上記ホスト名を照会（nslookup）すると下図のように表示されます。

Server : 127.0.0.11 は Docker が固定で使っている DNS サーバのアドレスで、DNS が返してきている Address : *172.22.48.1* は WSL 仮想スイッチ〔vEthernet (WSL)〕の IP アドレスで再起動により変動します。※nslookup のパラメータを host.docker.internal に変えても同一の値が返ってきます



コンテナ仮想化（Docker互換－Rancher Desktop）

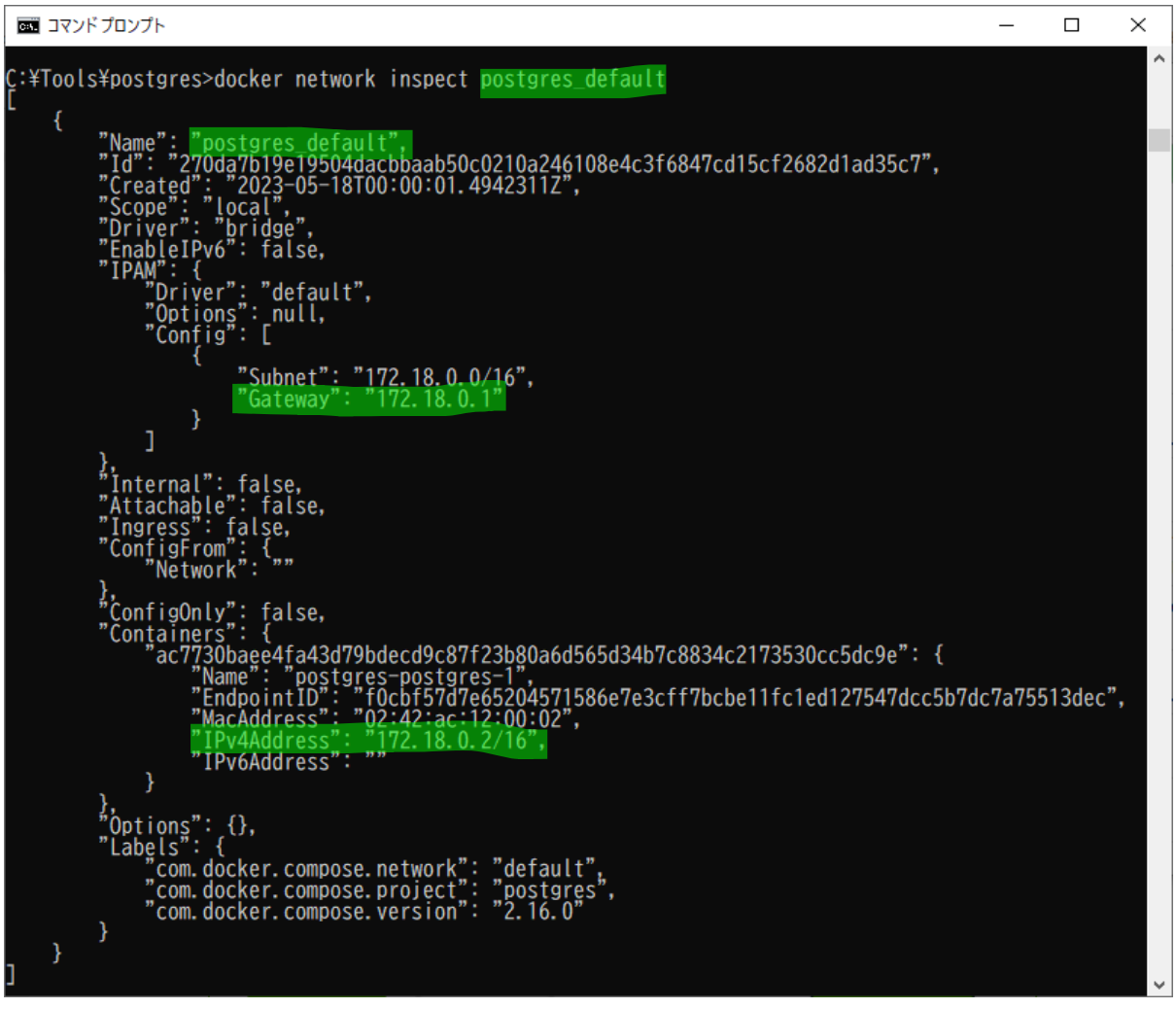
ホスト OS のコマンドプロンプトで powershell;Get-NetIPConfiguration -InterfaceAlias "*WSL*" を実行すると、WSL2 が構成した仮想スイッチが確認できます。コンテナはここを経由して外部ネットと通信します。（Hyper-V マネージャーから仮想スイッチマネージャーを起動しても表示されます）



※ DNSServer に IPv6 で初期値（実在しません）が入っていますが、問題ありません（以降に例）

docker-compose up コマンドで生成された（Network *ネットワーク名* Created）ネットワークは docker network ls で表示でき設定値は docker network inspect ***ネットワーク名*** で確認できます。

下図の例では、コンテナの IP アドレス :172.18.0.2(IPv4 のみ設定)、
デフォルトゲートウェー:172.18.0.1　に設定されていることが分かります。



コンテナ仮想化（Docker互換－Rancher Desktop）

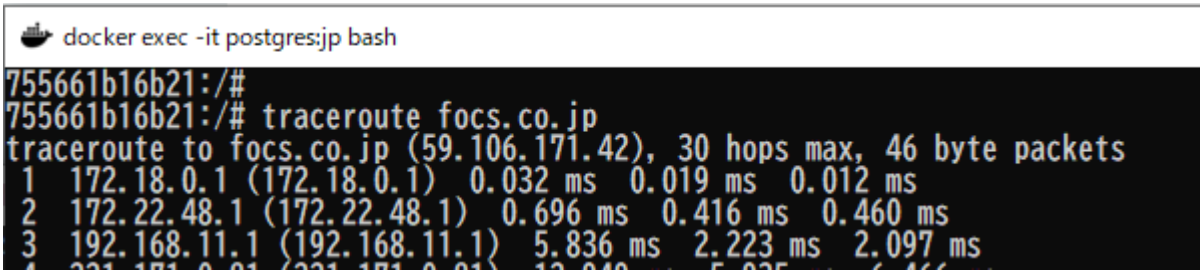
Rancher Desktop はデフォルトで、コンテナのプライベートネットワークを作りホスト側とは仮想スイッチの“イーサネット アダプター vEthernet (WSL)”経由で接続します。

vEthernet (WSL)は NAT[13]の機能を持っているので意識せずに外部ネットワーク／インターネット上のサーバに接続し、名前解決や http アクセス等ができます。

下図の例は、インターネット上のサーバ focs.co.jp に接続する経路を見たときの出力です。

172.18.0.1(デフォルトゲートウェイ) ⇒ 172.22.48.1(仮想スイッチ)

⇒ 192.168.11.1 ホスト OS が接続しているネットワーク



11.3. ホスト OS からコンテナ

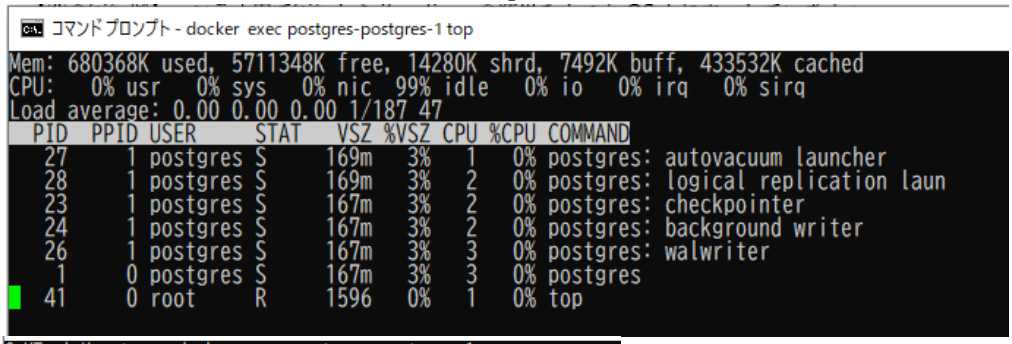
ホスト OS からコンテナのサービスに接続する仕組みとしては以下のものがあります。

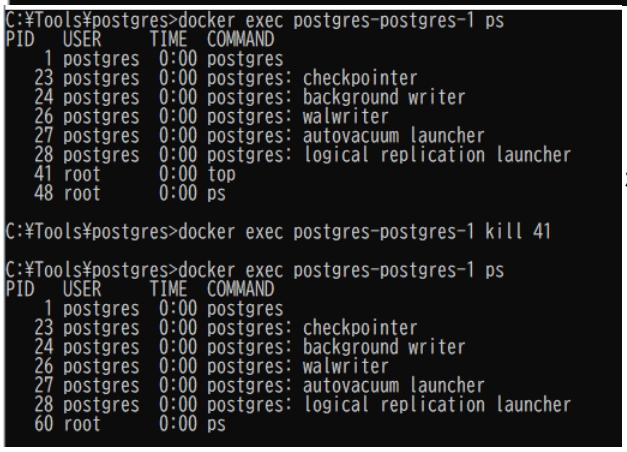
① ポートフォワーディングでコンテナのポートとホスト OS のポートを接続する

② docker exec *コンテナ名 コマンド* でコマンドの結果(stdout/err)をコマンドプロンプトへ

③ docker exec -it *コンテナ名* で bash 等に接続する（ターミナルで連続した操作ができます）

【②の実行例】コンテナ内で実行した top コマンドの結果をホスト OS 上に表示しています





【注意】

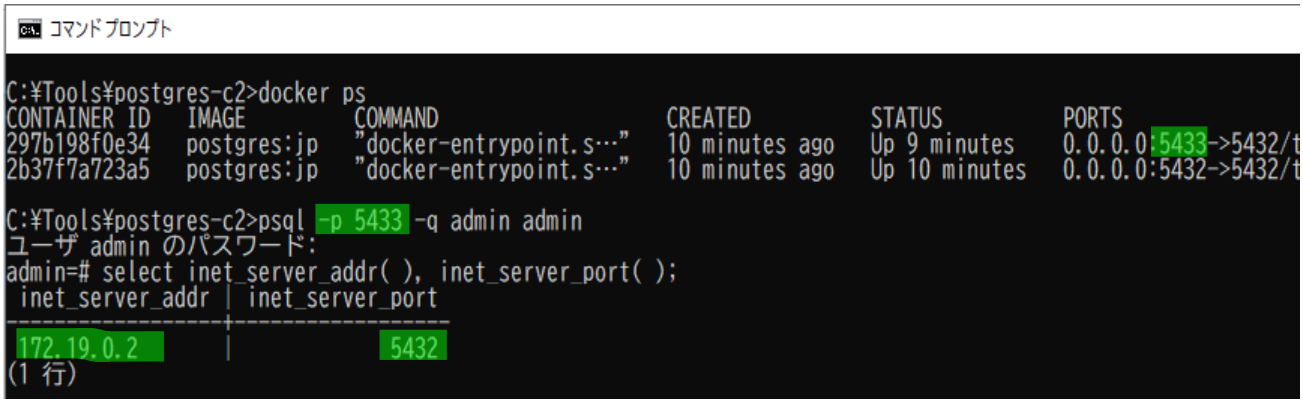
top コマンドのように**停止操作（q 打鍵等）が必要なコマンド**は ctrl＋c で切断後、ps コマンドで PID を確認して **kill *pid* コマンド**で終了させてください。プロセスが生き続けて資源の消費が続きます。

[13] Hyper-V ホストの管理 - NAT ネットワークの設定 https://learn.microsoft.com/ja-jp/virtualization/hyper-v-on-windows/user-guide/setup-nat-network

コンテナ仮想化（Docker互換－Rancher Desktop）

④ IP アドレスを使った接続（注意）

コンテナで起動しているサービスはポートフォワーディングで使うのが一番効果的（手軽で安全）ですが、IP アドレスを使ってサービスにリクエストを送ることもできます。



― コンテナの実 IP アドレスとポート番号

・docker ps でホストのポート 5433 に接続している postgres のコンテナがあることが分かります

・psql でホストの 5433 ポートでデータベースを開きます

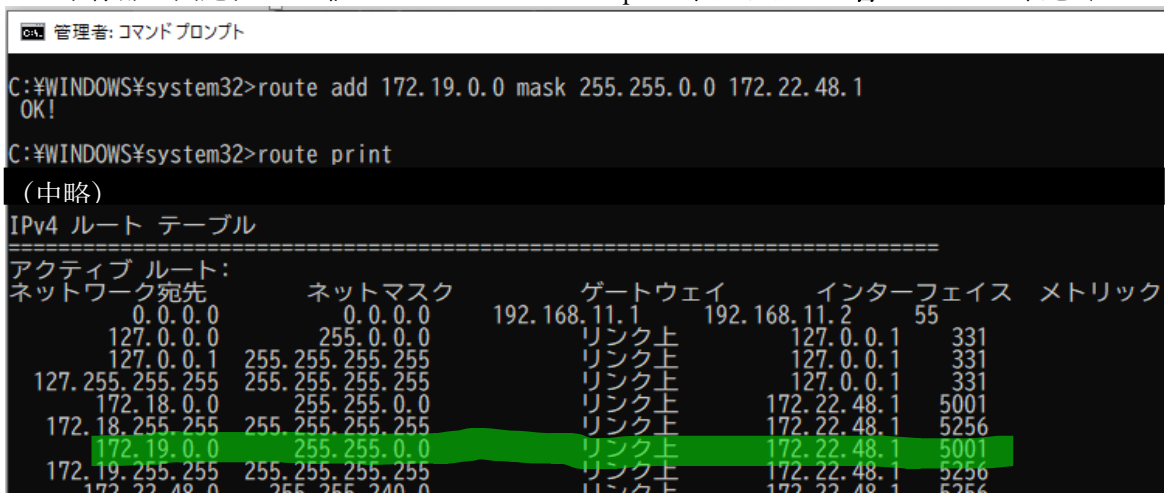
・データベースにサーバの IP とポート番号を問い合わせる(select inet_server_addr(), inet_server_port();)と DB サーバは IP アドレス 172.19.0.2、5432 ポートで稼働していることが分かります

― ネットワーク経路(route)の確認／存在しない場合に追加

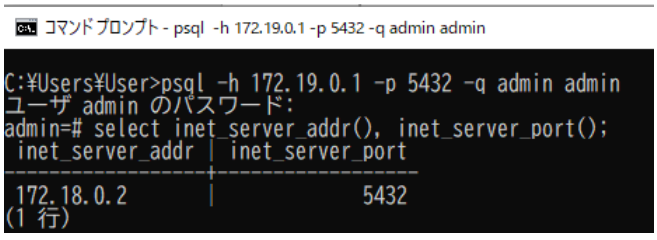
・コンテナの IP アドレスへのルートが登録されていないとポートフォワーディング以外では接続できません。IP アドレスで接続する場合は、管理者権限による root add が必要です

route add *サブネットアドレス* mask 255.255.0.0 *仮想スイッチ(WSL)のアドレス*

※下線部は固定、mask 値は docker network inspect ***ネットワーク名*** の Subnet 末尾 “/16”より



【注意】使用したバージョン（Rancher Desktop 1.8.1 / WSL2 プレインストール版）では、IP アドレスを使った接続は以下のように**不整合な環境になります**（172.19.0.1 で 172.18.0.2 に接続する）



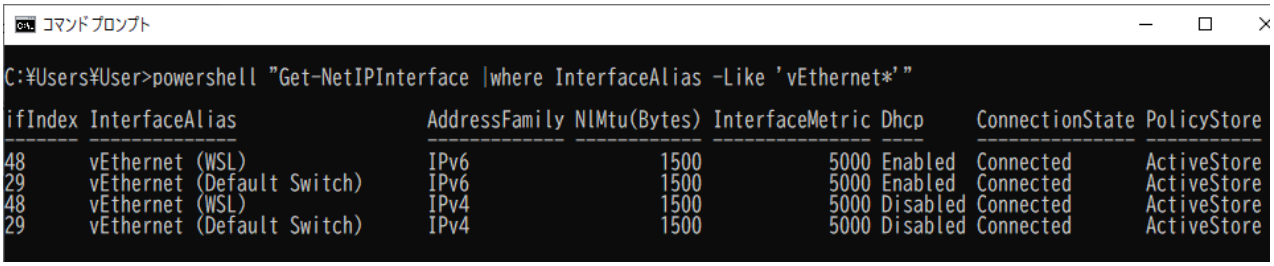
コンテナ仮想化（Docker互換－Rancher Desktop）

11.4. コンテナから Hyper-V 仮想マシン

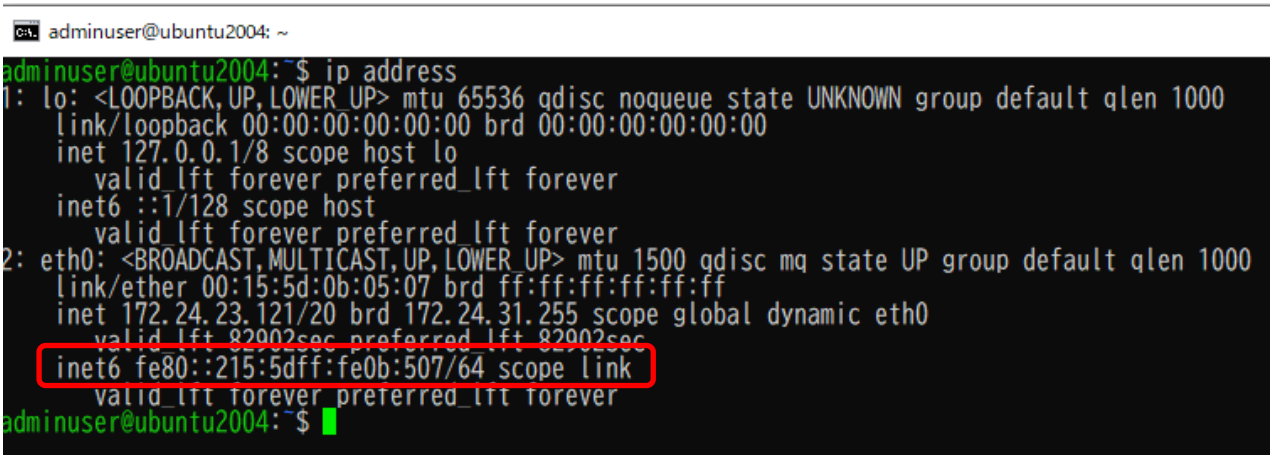
コンテナと Hyper-V 仮想マシンは各々別の仮想スイッチに繋がってるので、この間で通信を行う場合はパケットが通過できるように設定する必要があります。これは以下の手順で行います。

① WSL2 と Hyper-V マネージャーが使っている仮想スイッチの一覧を探します。仮想スイッチは”vEthernet”という文字列を含んでいるので、以下のコマンドで抽出します。

powershell "Get-NetIPInterface |where InterfaceAlias -Like 'vEthernet*'"

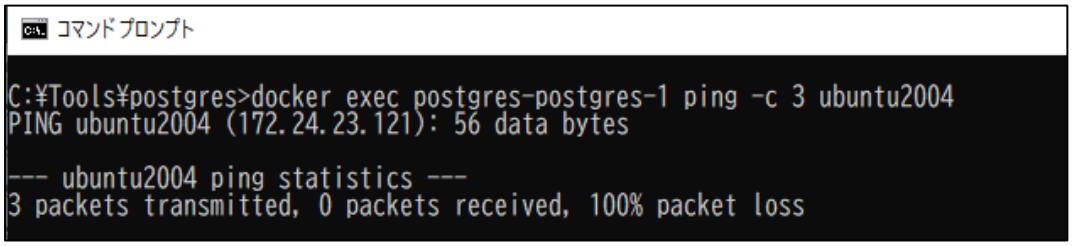


※ IPv6 と IPv4 のペアになっていますが、IPv6 のアドレスはデフォルトでは設定されていません。
Hyper-V 仮想マシンの eth0 の下から 2 行目に inet6 fe80::～ scope link と IPv6 アドレスが表示されていますが scope link はローカルリンクアドレスでスイッチやルータを越えられません



したがって、コンテナから ping コマンドを実行しても届きません。

（下図の ubuntu2004 は Hyper-V 仮想マシン。名前解決は成功しても ping は届いていません）



コンテナ仮想化（Docker互換－Rancher Desktop）

② 仮想スイッチ（vEthernet）の Forwading を Enabled にします

管理者権限で Poweshell を起動し、名前が vEthernet で始まるネットワークインタフェースを選んで Set-NetIPInterface コマンドレットを使って変更します。

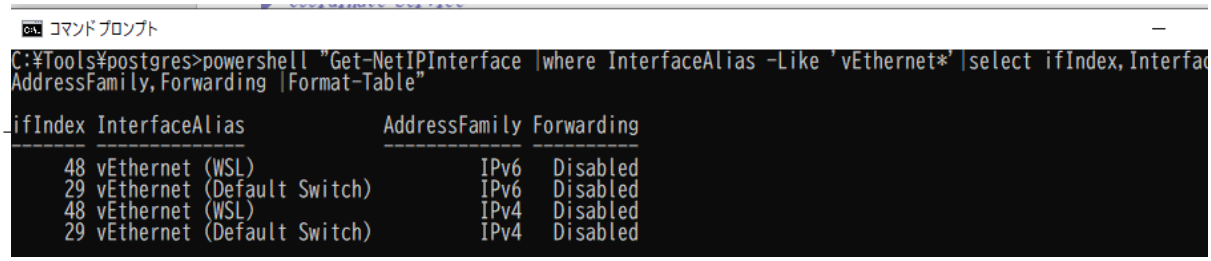
```
powershell "Get-NetIPInterface |where InterfaceAlias -Like 'vEthernet*'|Set-NetIPInterface -Forwarding Enabled"
```

【Set-NetIPInterfase -Forwarding Enabled の実行前後の状態確認】

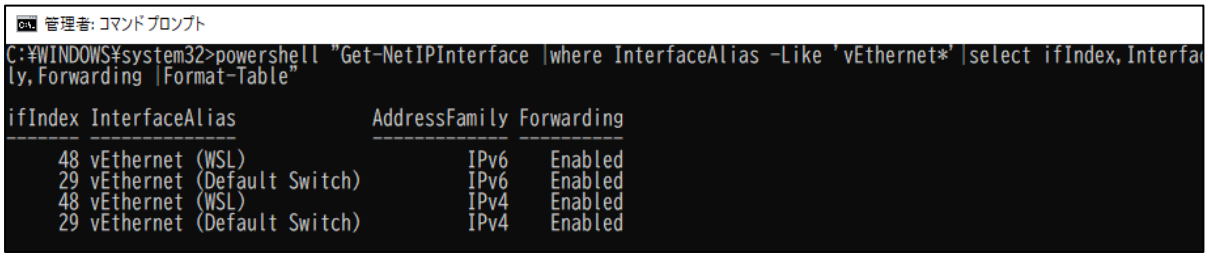
以下のコマンドで Forwarding の値を確認します

```
powershell "Get-NetIPInterface |where InterfaceAlias -Like 'vEthernet*'|select
ifIndex,InterfaceAlias,AddressFamily,Forwarding |Format-Table"
```

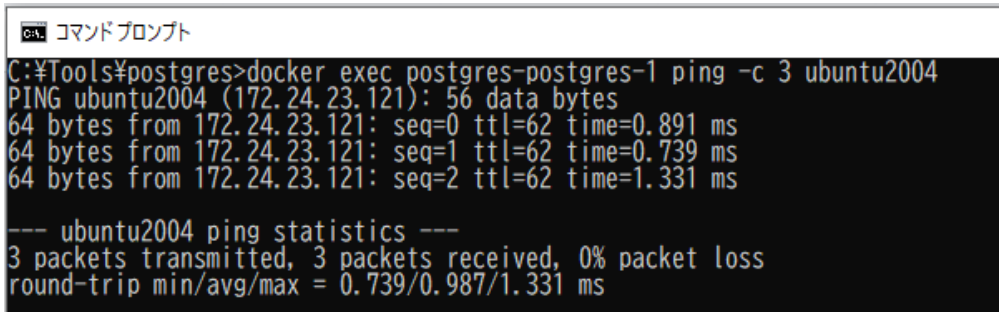
＜初期状態＞



＜変更後＞



Forwarding Enabled 後に ① と同じように ping を実行してみると、今回は成功します



※ 但し、仮想スイッチの状態変更は再起動により変更前に戻ります。

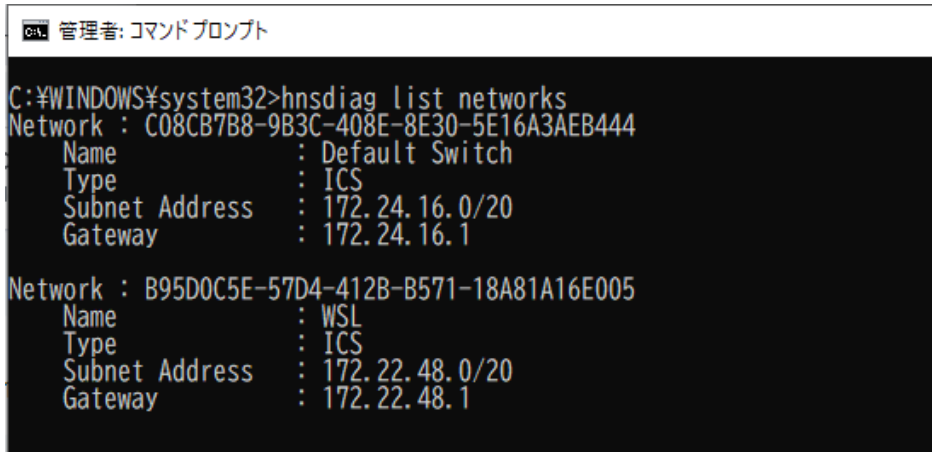
12. WSL2 とコンテナの注意点

WSL2 は 2023 年 5 月末時点で、9 世代のプレリリース版が並列管理され GitHub で公開され[14]ていて 2000 件近い作業中の管理項目(issues)が存在します。Docker（や代替え製品）を Windows で使う場合、コンテナ間やホスト、他 VM との通信環境として WSL を使いますが issues の数を見ても先々仕様が変わっていくことが予測されます。

特に、NAT や名前解決を実現する仮想スイッチは、以前から Windows に備わっていた ICS(Internet Connection Sharing)や、Hyper-V が利用する HNS(Host Networking Service)を組み合わせて実現している[15]ようですが、環境作成時に ICS の設定が壊れることがあります。

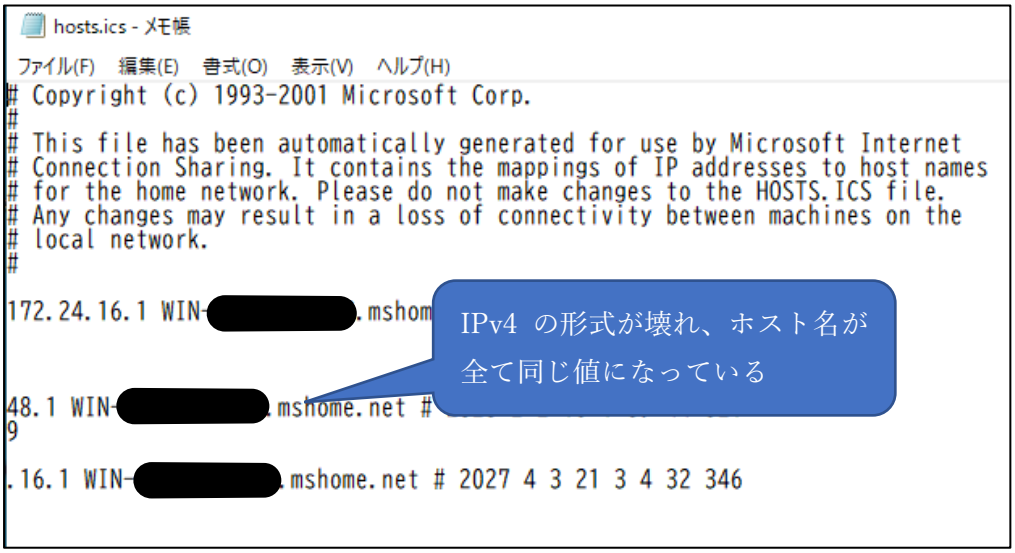
【HNS のネットワーク情報】※情報参照には管理者権限で hnsdiag list networks を実行

Type:ICS と表示され、ICS を参照していることが確認できる



【hosts.ics】

ICS が名前解決に使っている C:¥Windows¥System32¥drivers¥etc¥hosts.ics が仮想スイッチの登録時に IP アドレス(172.24.16.1 と 172.22.48.1)とホスト名の書き込みでフォーマットが壊れました



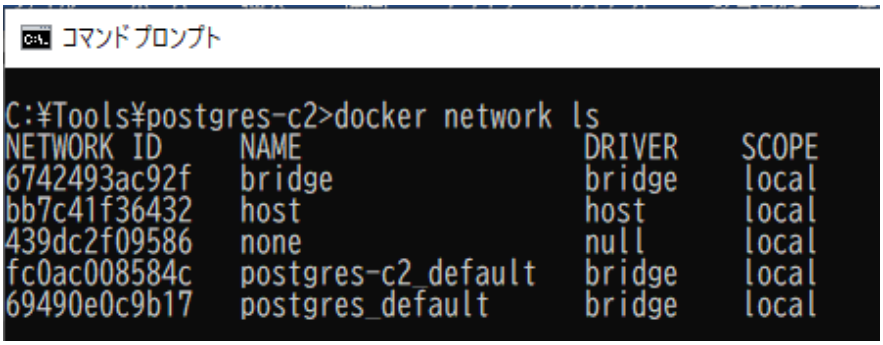
[14] microsoft/WSL https://github.com/microsoft/WSL/releases

[15] ICS が無効になっている場合の WSL 2 エラー https://learn.microsoft.com/ja-jp/windows/wsl/troubleshooting#wsl-2-errors-when-ics-is-disabled

コンテナ仮想化（Docker互換－Rancher Desktop）

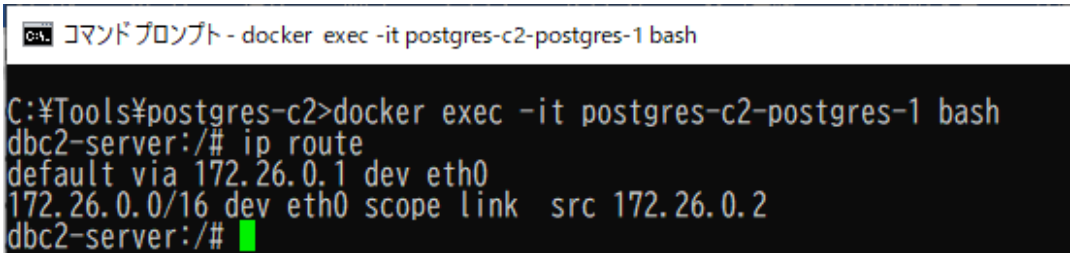
hosts.ics が壊れた場合 Hyper-V の vm で名前解決が正常にできなくなる可能性がありますが、今回使った環境は vm に Samba を入れてホスト名を公開していたので、Docker コンテナからも vm ホスト名の解決ができています。

次に、Rancher Desktop のネットワークに関しては、docker run で生成したものと docker-compose で生成したものではデフォルトで使われるネットワークが異なる（bridge と *コンテナ名*_default）ようですが、どちらもブリッジとサービスのペアになります（例 172.26.0.1 と 172.26.0.2）。



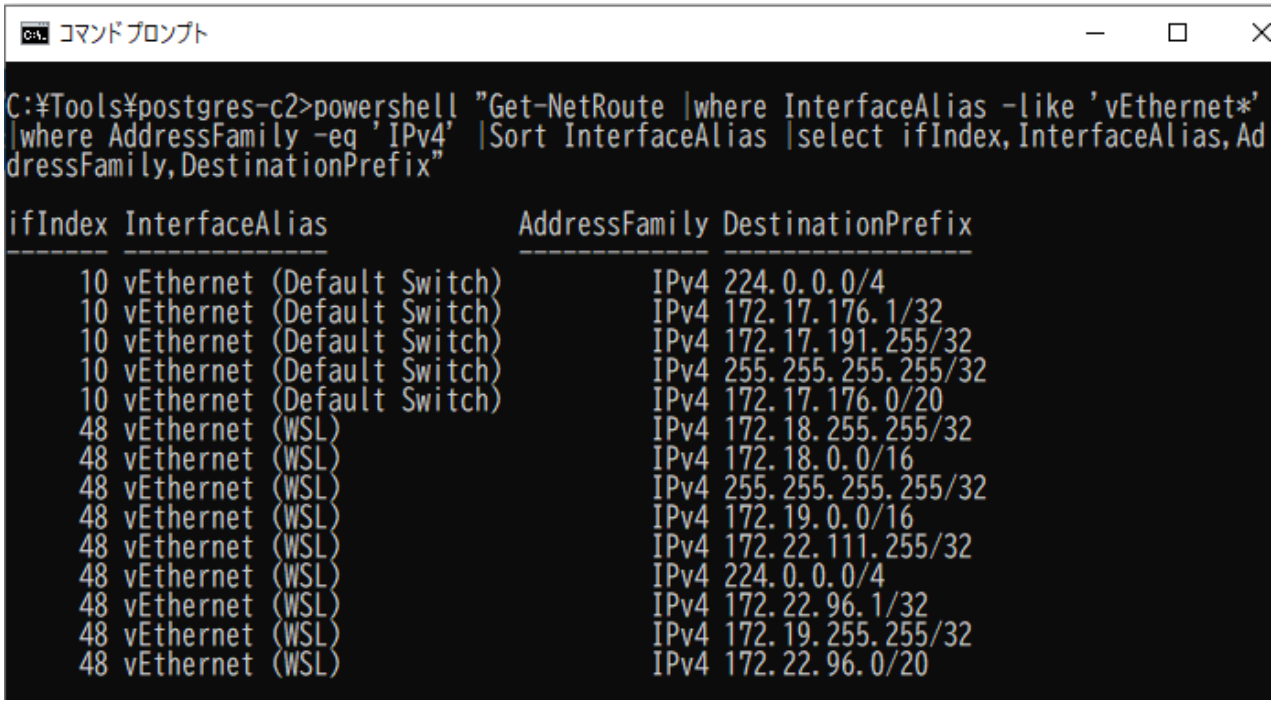
＜サービスの IP アドレスでアクセスできません＞

docker exec -it *コンテナ名* bash で接続する IP アドレスは n.n.n.2 ですが、このアドレスはホストから見つけることはできず、n.n.n.1 を指定することで接続できます。



＜ホスト OS 上の IP ルートは以下のコマンドで確認できます＞

powershell "Get-NetRoute |where InterfaceAlias -like 'vEthernet*'|where AddressFamily -eq 'IPv4' |Sort InterfaceAlias |select ifIndex,InterfaceAlias,AddressFamily,DestinationPrefix"



コンテナ仮想化（Docker互換－Rancher Desktop）

13. 使い方

WSL 2 と Rancher Desitop がデフォルトで作るネットワークが不可解だったので、同様に疑問を感じた人が自分で確認できるようにコマンド例をあげておきました。ネットワーク関係はコード内に直書きされている部分があったり、今後の改善や機能追加でこの資料とは変わっていく可能性があるので動きが変わったと感じたら、コマンドで環境を確認してみてください。

ただ、ネットワークの構成は解り辛くともデフォルトで動作する環境が作られますし、基本的な使い方の範囲（項番 9 迄）では多量のコマンドを覚えずとも十分に利用できます。インターネットに公開されているコンテナのイメージを使えば一定の動作確認が済んだ、モノによっては商用製品レベルの環境を簡単に作ることができるので、是非試してください。

以上