Dockerコンテナを動かしてみよう！

株式会社びぎねっと
宮原 徹

講師 自己紹介

- 株式会社びぎねっと 代表取締役社長兼CEO
  - 日本仮想化技術株式会社 代表取締役社長兼CEOでもある

私とLPI-Japan

- LinuCの試験開発の活動に参加しています
  - 新しい試験範囲に加わった仮想化・クラウドなどは専門です
- LPI-Japan発行のメールマガジンに寄稿しています
- LPI-Japan主催各種セミナーの講師を務めています
- 標準教科書シリーズの執筆などをしていました

Linux技術者認定「LinuC（リナック)」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、オープンソースへの理解など、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

Dockerコンテナを動かしてみよう！

コンテナの特徴

- 1コンテナ1プロセスで動作する
  - 仮想マシンと比べてハイパーバイザーのオーバーヘッドが無いのでより高速
- ≒コンテナ内で1プロセス（1タスク）が動作する
  - サービスプロセスと並行で対話型シェルを動かすなどの考え方は基本的に無い（シングルタスク）
  - コンテナの中であれこれ動作させるのには向かない
  - WebアプリならWeb APサーバで1コンテナ、DBサーバで1コンテナという感じ
- Dockerイメージは（ルート）ファイルシステム
  - chrootを思い浮かべるとちょうどいい

各方式の比較

	ベアメタル(物理)	仮想マシン	コンテナ
性能	最も速い	I/Oが遅い	速い
OS	1つだけ	複数種類を混在可能	カーネルは1つだけだが、ディストリビューションは混在可能
リソース使用	システムで専有	メモリの無駄が多い	OSカーネルは1つで効率が良い
柔軟性	硬直的	非常に柔軟	単機能向け
主な用途	高速なDBなど	従来型の業務システム	Webサービスのフロントエンド等、同一のものを大量配備する必要があるもの

Dockerを動かす方法

- Docker Desktopを使う
  - WindowsとMacに対応
  - Linux版はベータ版で仮想マシン上では動作しない
  - GUIが便利
- LinuxにDocker関連パッケージをインストール
  - シンプルに動かせる
  - よりサーバー向き？

- その他、いろいろなソリューションがあるので、目的に応じて

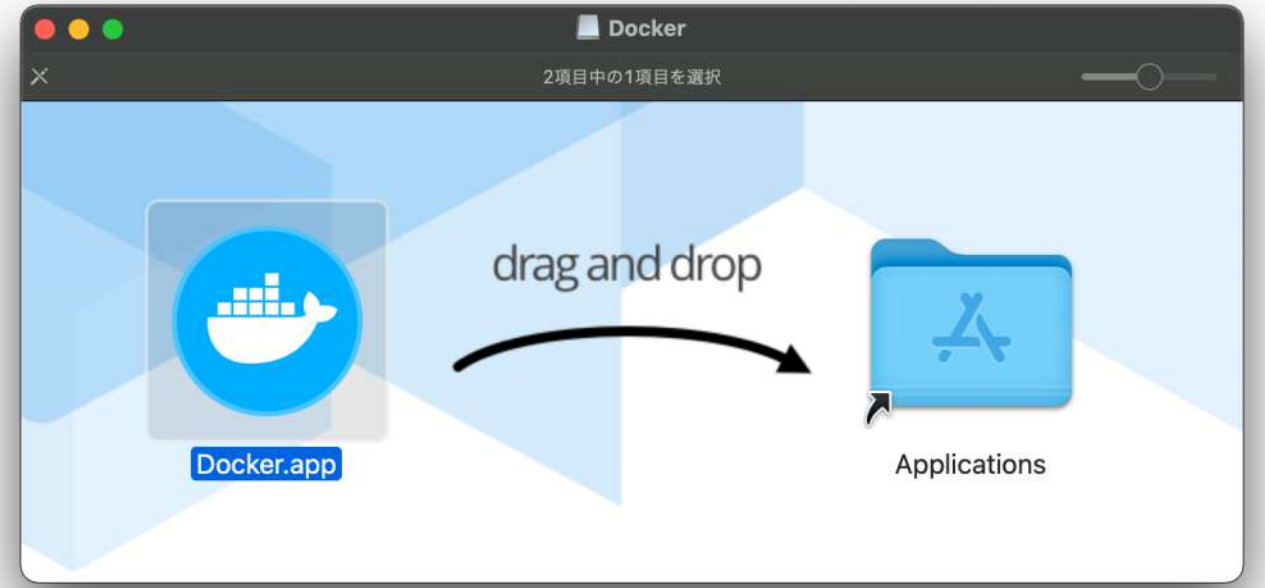
DOCKER DESKTOPのインストール

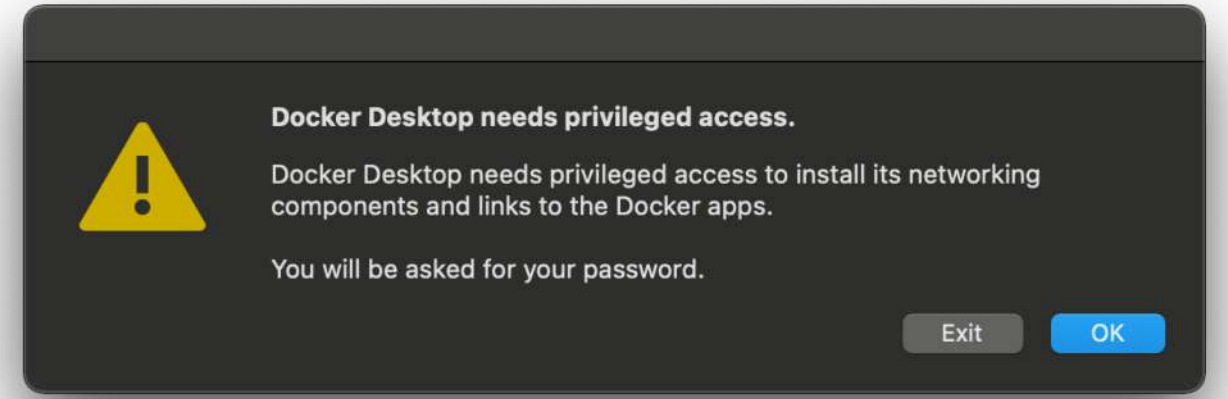
Docker Desktopをインストール

- 手元で使いやすいDocker Desktopをインストール
  - 学習目的としてインストールしてください
  - 商用利用は有償（小規模企業は例外あり）
- システム要件はドキュメントで確認
  - https://docs.docker.jp/desktop/
- WindowsはWSL2上でDockerコンテナが動作
  - Windows HomeでもWSL2は動作するのでインストール可能
- macOSは10.13(High Sierra)以降が必要
  - Apple Siliconも対応
- Linuxの場合でもコマンドラインは共通なので、関連パッケージをインストールしてDockerコンテナの実行環境を用意してください
  - 物理マシン上でDocker Desktop for Linuxベータ版でも可

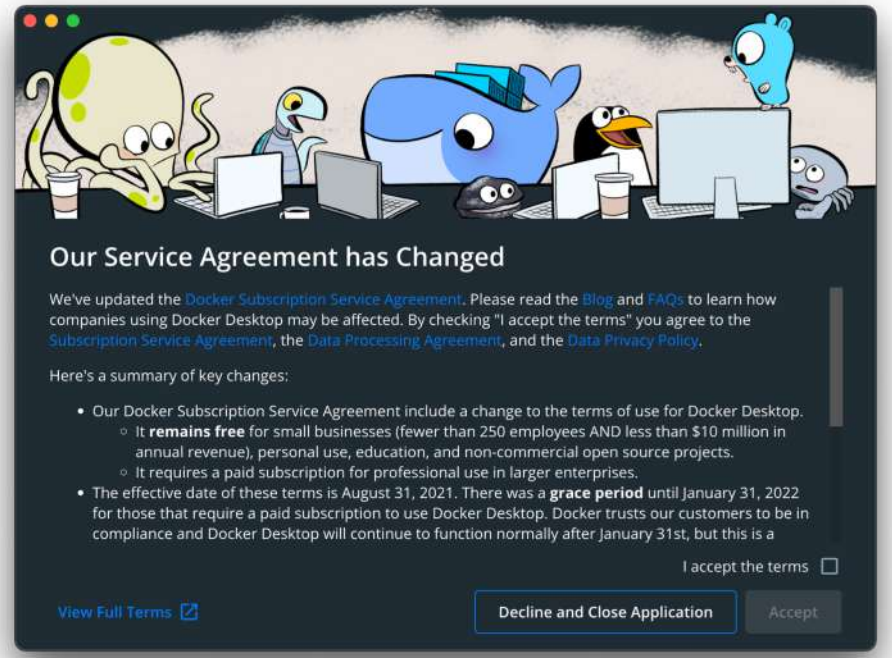
Docker Desktop for Macのインストール

- ダウンロード後、実行ファイルをコピー
- 実行ファイルを起動すると管理者権限を要求されるので認証
- 使用権許諾を確認
- （アプリケーションが正しく起動しないので再度起動）





管理権限が必要



使用権許諾を確認

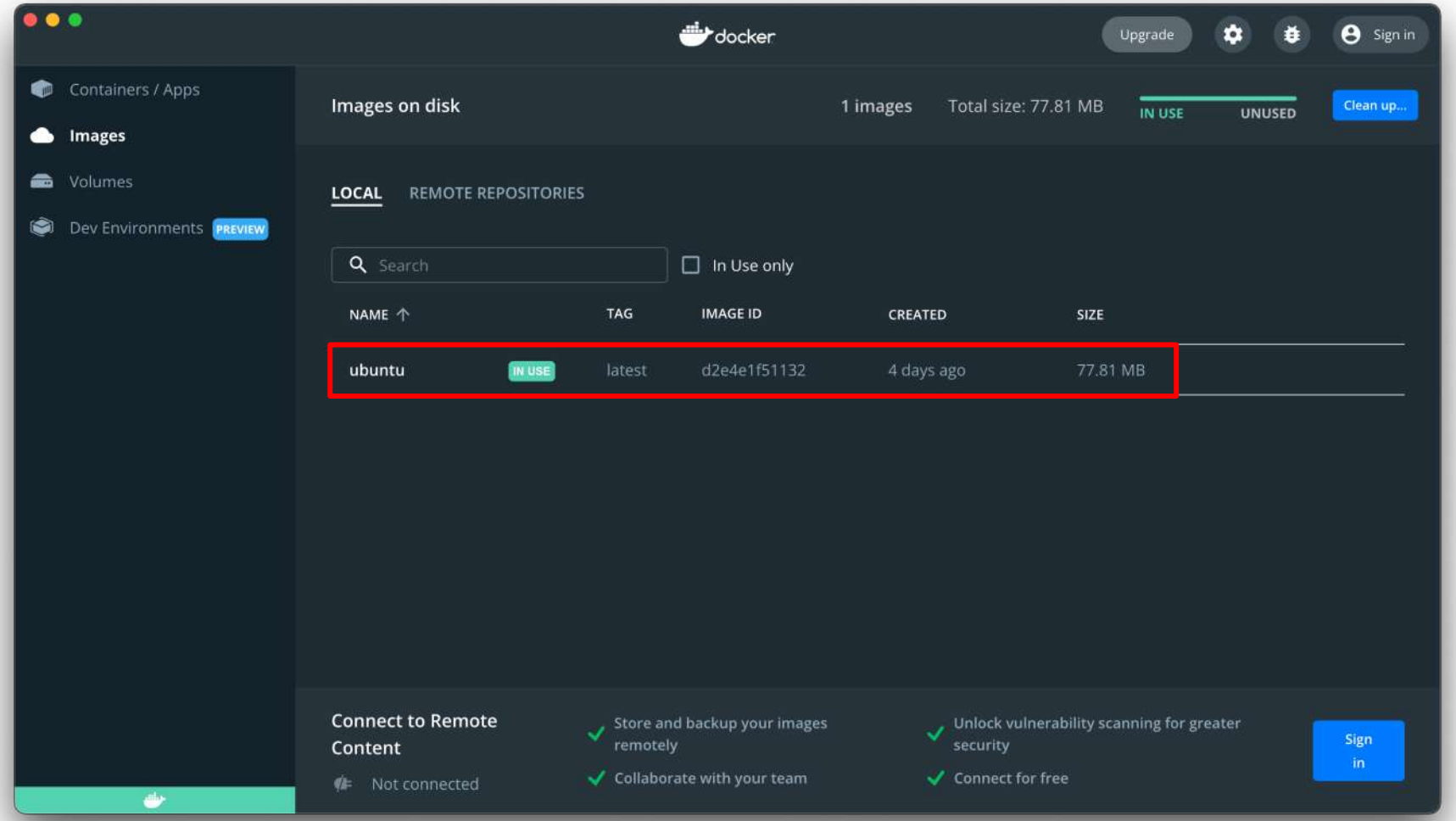
Dockerイメージをダウンロード

1. UbuntuのDockerイメージを検索
   - # docker search ubuntu
2. UbuntuのDockerイメージをダウンロード
   - # docker pull ubuntu
   - 最新版（latest）がダウンロードされる
3. Dockerイメージを確認
   - # docker images
   - イメージはIDで識別

コマンド実行例

```
% docker search ubuntu
NAME                      DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
ubuntu                    Ubuntu is a Debian-based Linux operating sys…   14171     [OK]
（略）
% docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
125a6e411906: Pull complete
Digest: sha256:26c68657ccce2cb0a31b330cb0be2b5e108d467f641c62e13ab40cbec258c68d
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
% docker images
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
ubuntu       latest    d2e4e1f51132   4 days ago   77.8MB
```

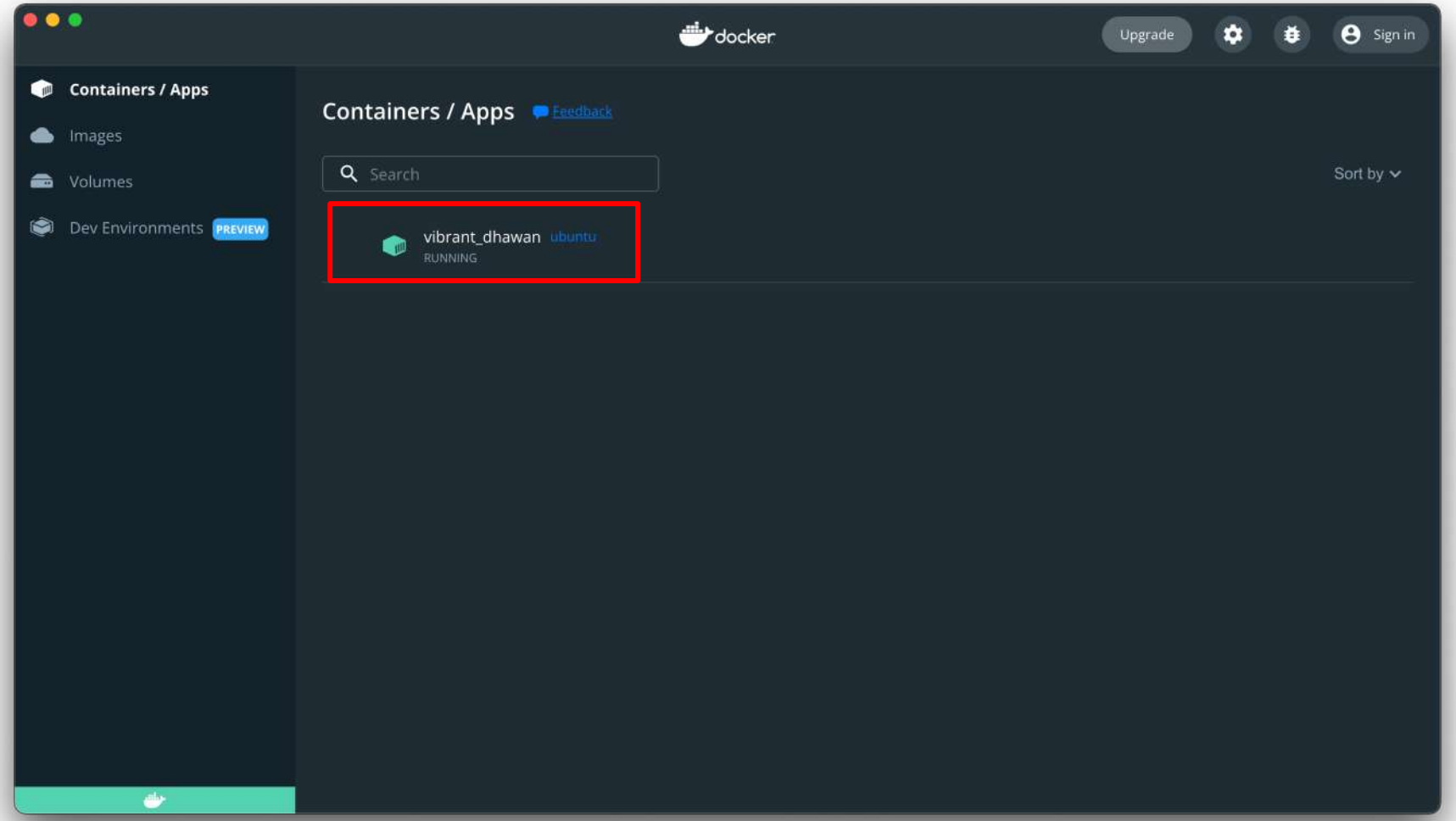
GUIでイメージを確認



Dockerコンテナを実行（基本）

1. Ubuntuイメージでコンテナを実行
   - # docker run -it ubuntu bash
   - -i, --interactive=true|false
   - -t, --tty=true|false
2. 実行中のコンテナを確認（ホスト）
   - コンテナ内ではなくホスト側で実行（別ターミナル起動など）
   - # docker ps -a
   - -a, --all=true|false
   - 一意のIDと名前が割り当てられる
3. コンテナから抜けるには exit コマンド

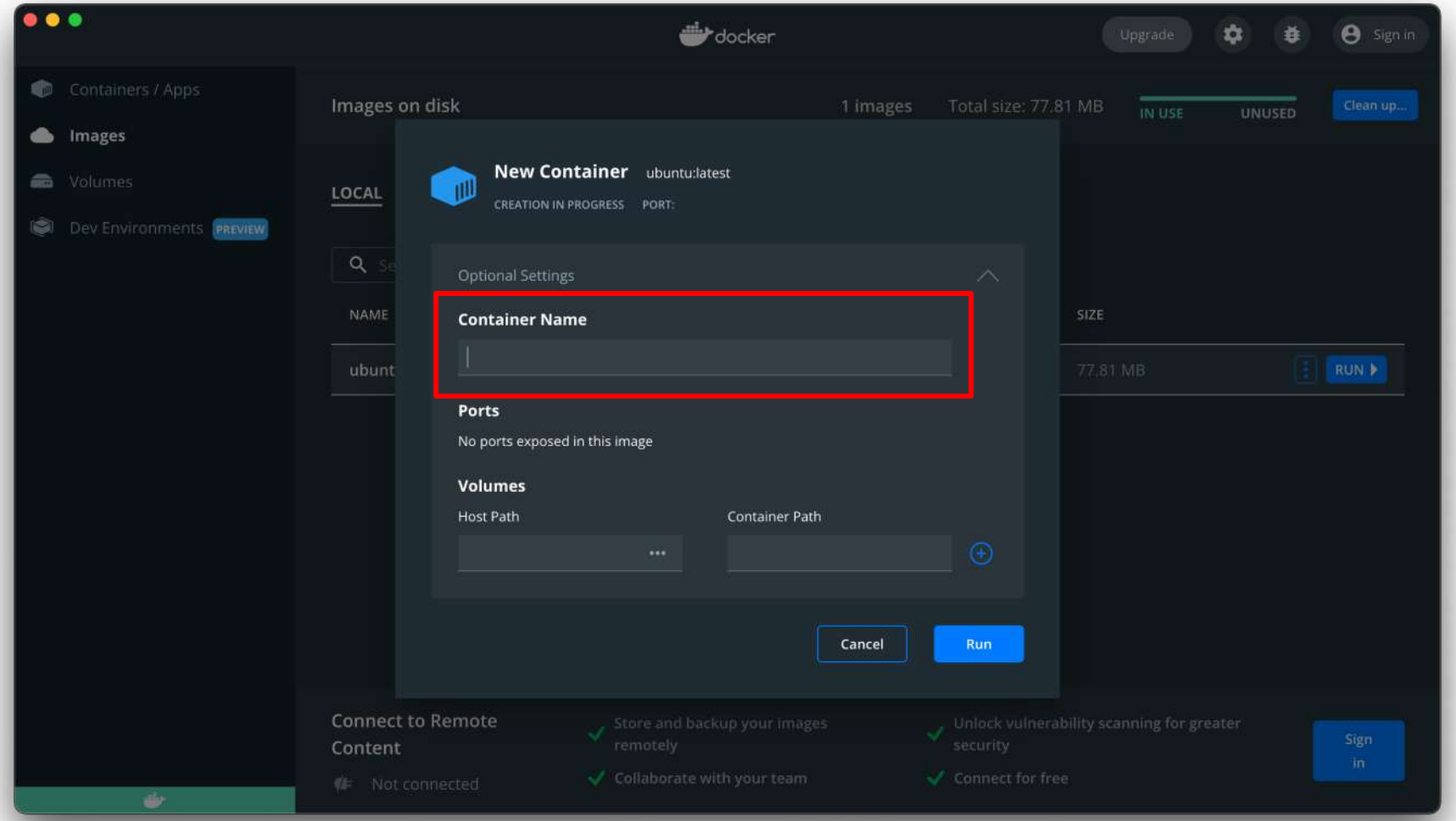
GUIで実行しているコンテナを確認



Dockerコンテナ実行（応用）

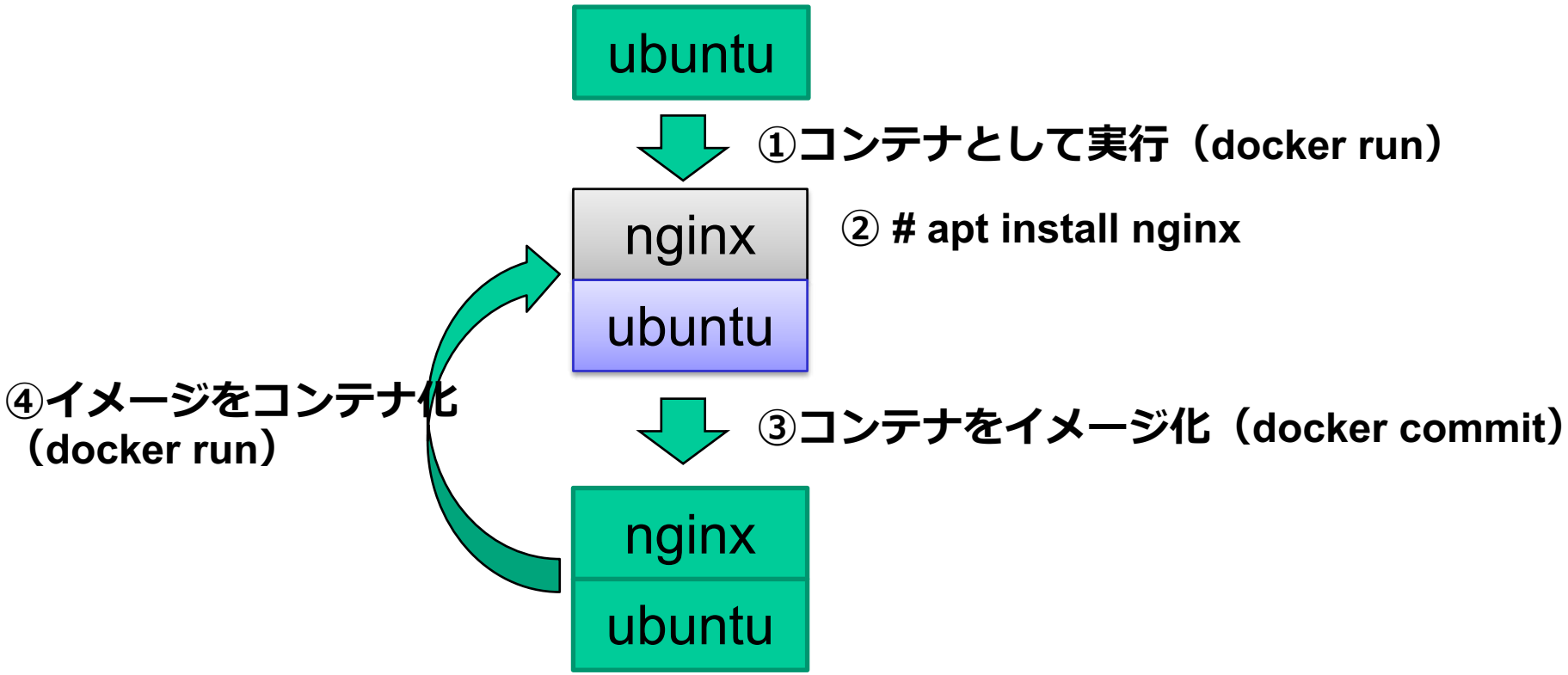
1. コンテナ名を付ける
   - # docker run --name=名前 イメージ名 コマンド
2. シェルからコマンドを実行させる
   - # docker run イメージ名 /bin/bash -c "コマンド"
3. より複雑な実行時処理を行いたい場合はDockerfileを記述する（後述）
   - どこまで複雑にするかはケースバイケース？

GUIでコンテナを起動



DOCKERコンテナのライフサイクル

イメージとコンテナの変移



コンテナにNginxをインストール

1. コンテナを起動（ホスト）
   - # docker run -it --name=nginx ubuntu bash
2. apt updateを実行
   - # apt update
3. Nginxをインストール
   - # apt install nginx
4. curlコマンドをインストール（Nginxの動作確認用）
   - # apt install curl
5. Nginxを起動（バックグラウンド動作）
   - # nginx
6. 動作確認
   - # curl localhost
   - サンプルHTMLが表示されればNginxが動作している
   - Nginxの停止は nginx -s quit を実行

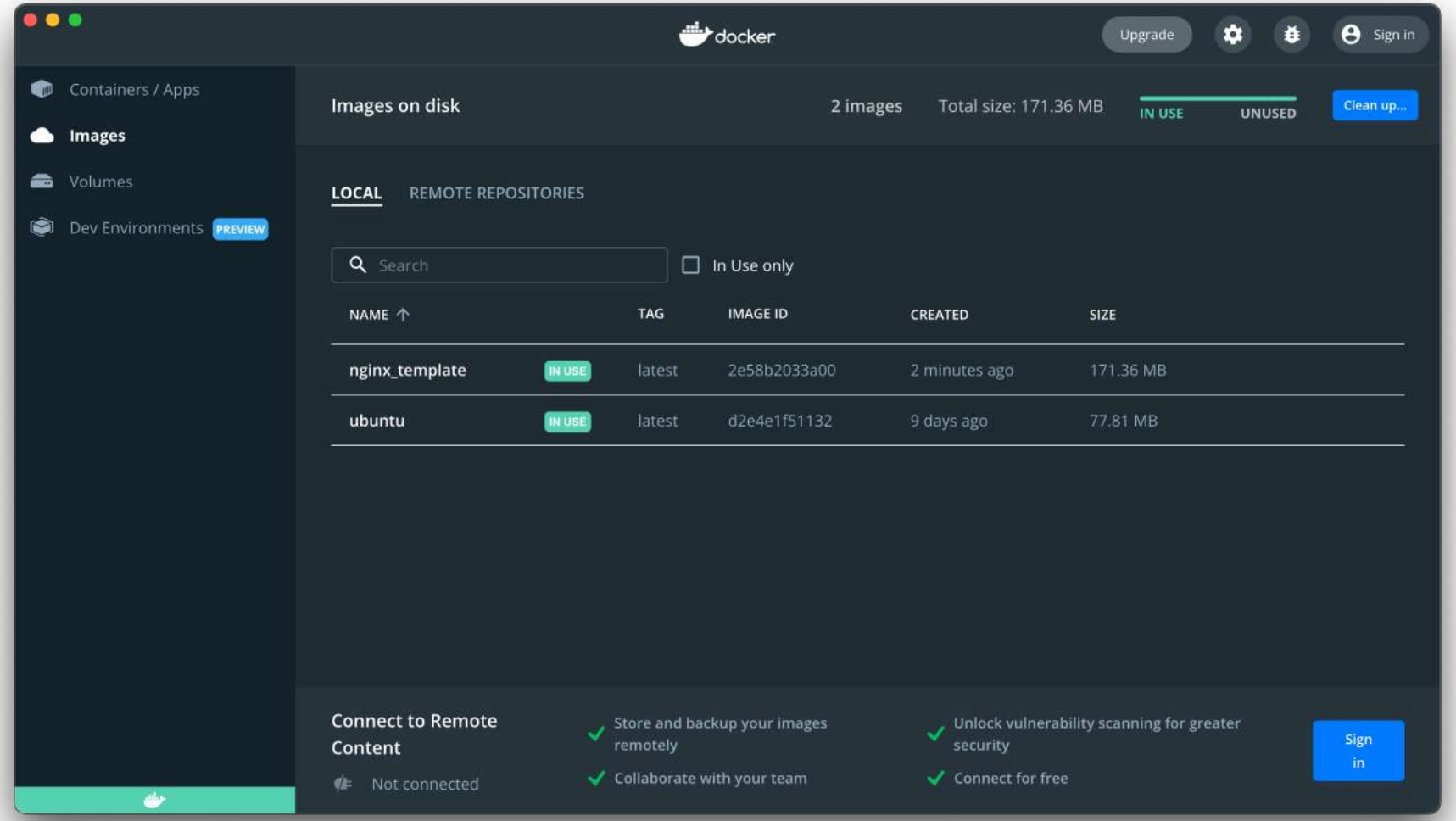
コンテナをイメージ化

1. Nginxをフォアグラウンド実行するように設定
   - コンテナ内でNginxをバックグラウンド起動するとコンテナが停止してしまう
   - このコマンドはコンテナ内で実行すること
   - # echo "daemon off;" >> /etc/nginx/nginx.conf
2. コンテナから抜ける
   - # exit
3. コンテナをイメージ化する（ホスト）
   - # docker commit nginx nginx_template
   - nginxは実行していたコンテナ名
   - nginx_templateは保存するイメージ名

■コンテナイメージが追加されたことを確認（ホスト）

- # docker images

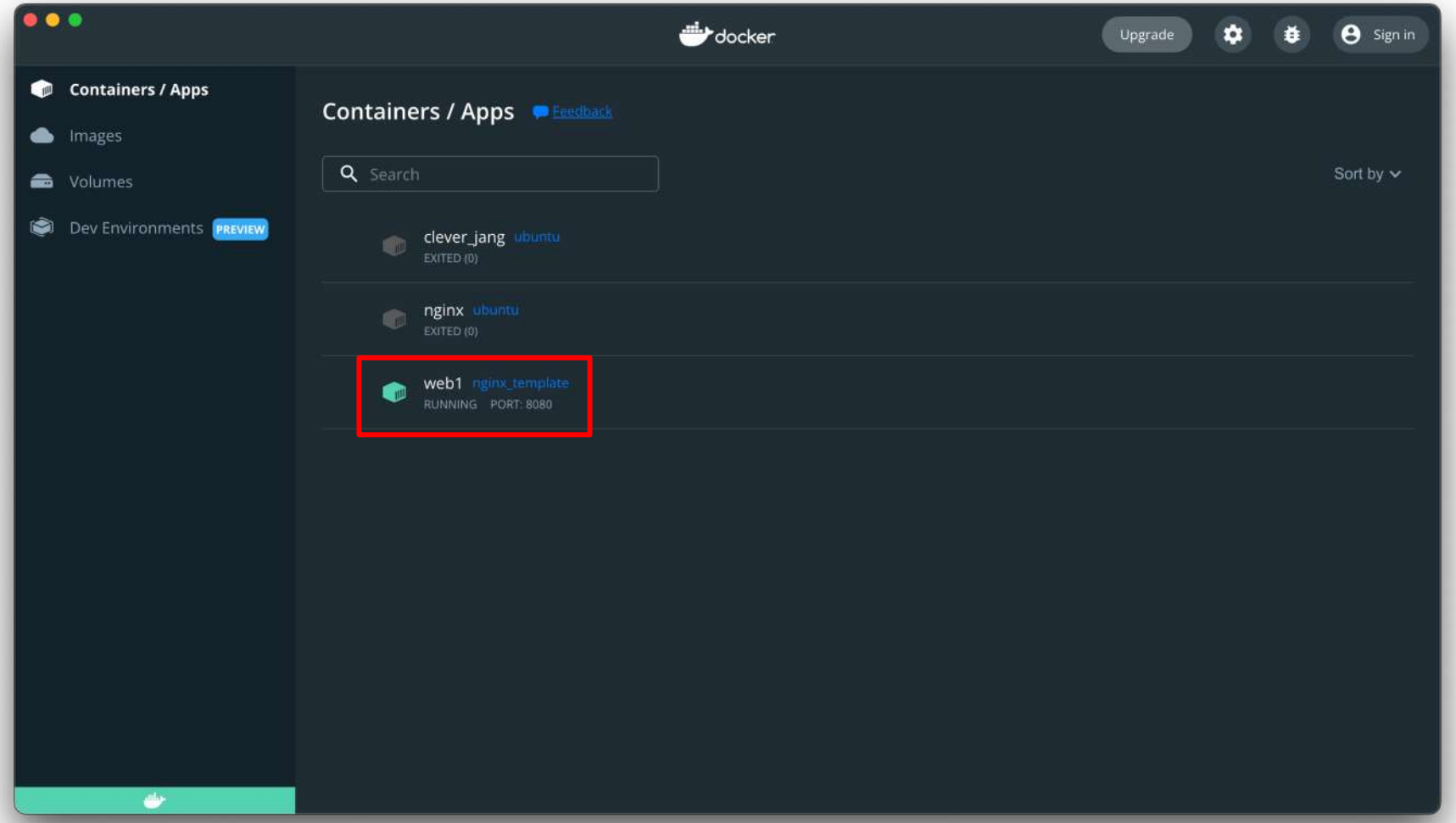
GUIでイメージを確認



新たなイメージからコンテナ実行

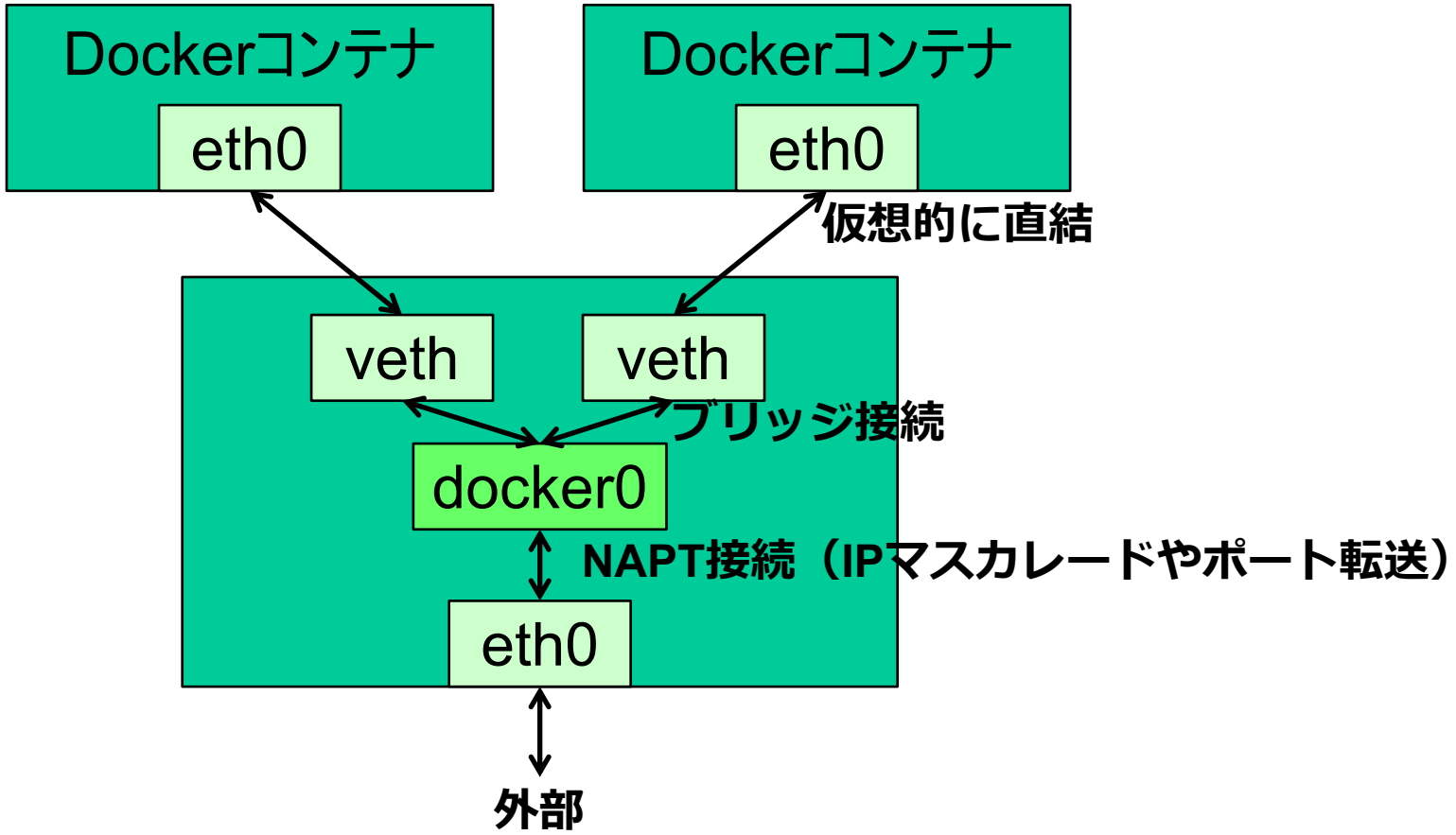
1. コンテナでNginxを実行し、ホストのポート8080番をコンテナのポート80番に紐付ける（ホスト）
   - # docker run -d -p 8080:80 --name=web1 nginx_template nginx
   - -d, --detach=true|false
2. ポートが紐付いていることを確認（ホスト）
   - # docker ps -a
   - # curl localhost:8080
   - NginxのサンプルページのHTMLが表示されればOK

GUIで実行しているコンテナを確認

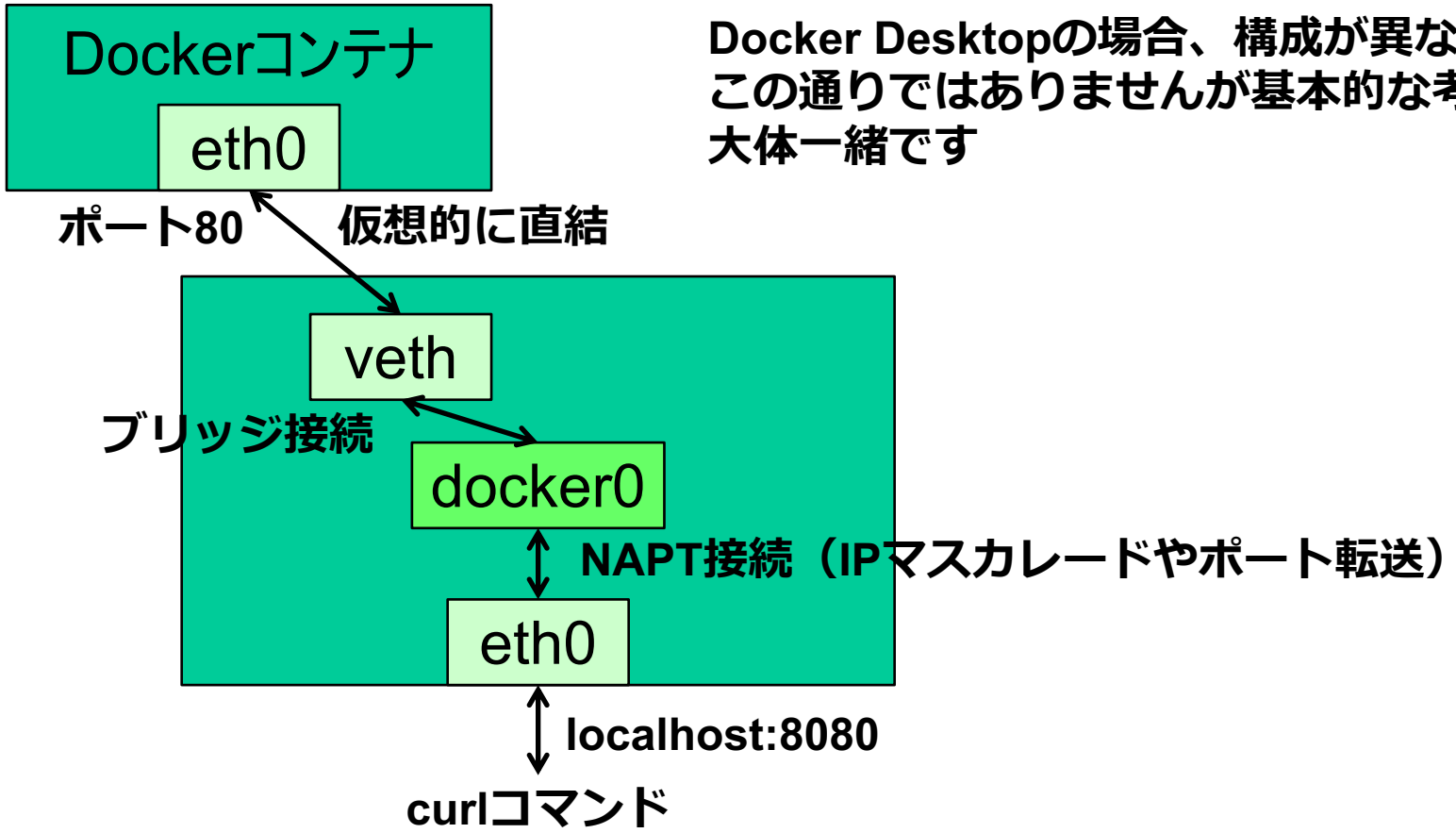


DOCKERのネットワーク

Dockerのネットワーク構成図（Linuxの場合）



今回のケースを当てはめてみると



Docker Desktopの場合、構成が異なるので
この通りではありませんが基本的な考え方は
大体一緒です

自動化への第一歩

DOCKERFILEを書いてみる

Dockerfileに書ける内容

命令	内容
FROM	元となるイメージ
LABEL	イメージにメタデータを付与
RUN	docker build時にコマンド実行
CMD	コンテナ実行時にコマンド実行（実行時上書き可）
EXPOSE	ポートの紐付け
ENV	環境変数の指定
ADD	ファイル、ディレクトリの追加
COPY	ファイル、ディレクトリの追加（URL指定不可・解凍不可）
ENTRYPOINT	コンテナ実行時にコマンド実行（実行時上書き不可）
VOLUME	ボリュームのマウント
USER	実行ユーザの指定
WORKDIR	作業ディレクトリの指定
ONBUILD	作成したイメージを元にしたdocker build終了後にコマンドを実行

Dockerfileの使用例（1）

- Dockerfileを作成する
  - ホームディレクトリ以外のディレクトリに作成すること

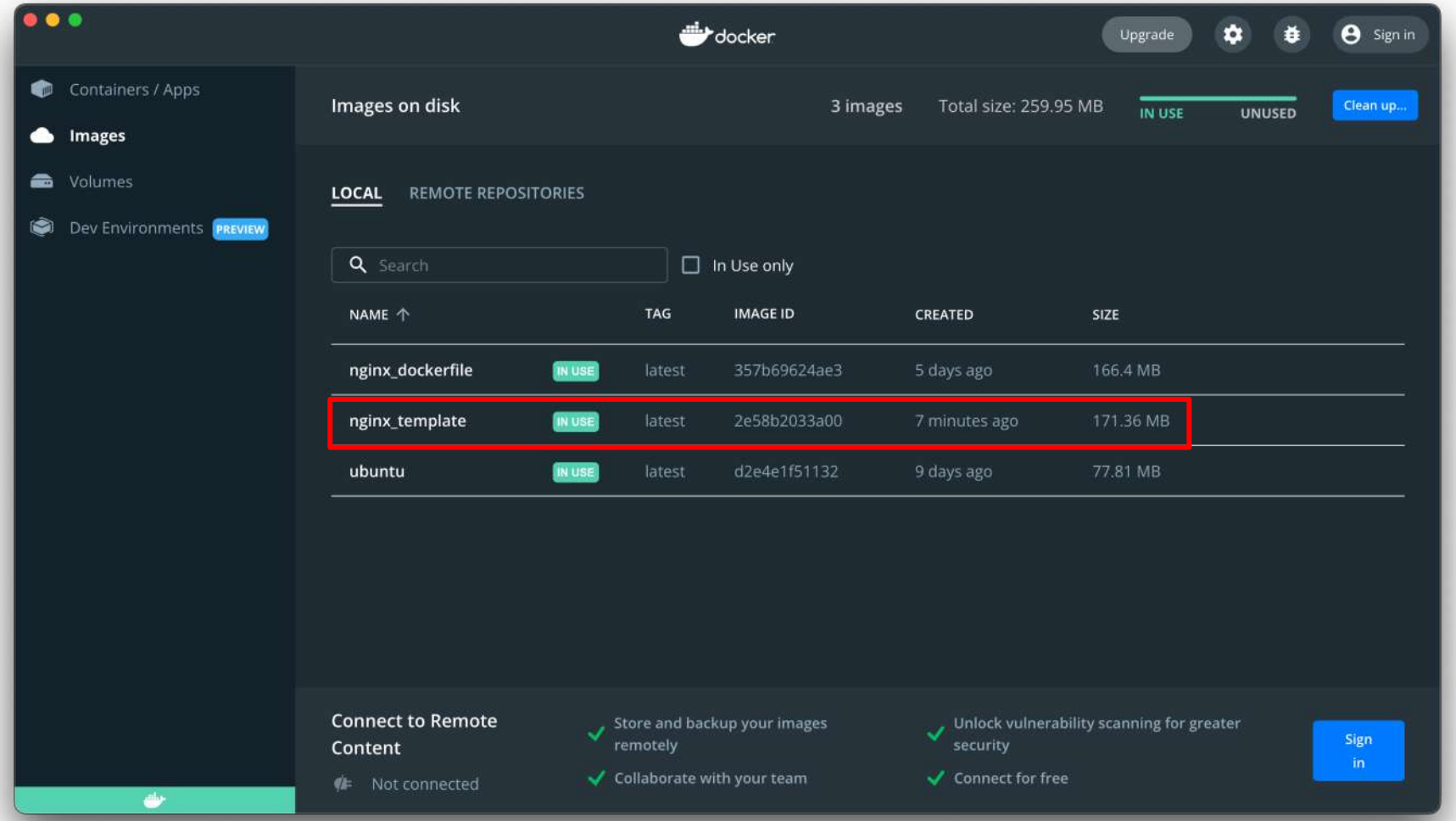
```
FROM ubuntu:latest
RUN apt update -y
RUN apt install nginx -y
ENTRYPOINT /usr/sbin/nginx -g "daemon off;"
```

  - ENTRYPOINTでNginxのフォアグラウンド起動を指定
- docker buildでDockerイメージを作る
  - # docker build -t nginx_dockerfile .　←.（ドット）を忘れずに
  - 最後の「. 」はカレントディレクトリのDockerfileを参照するために指定

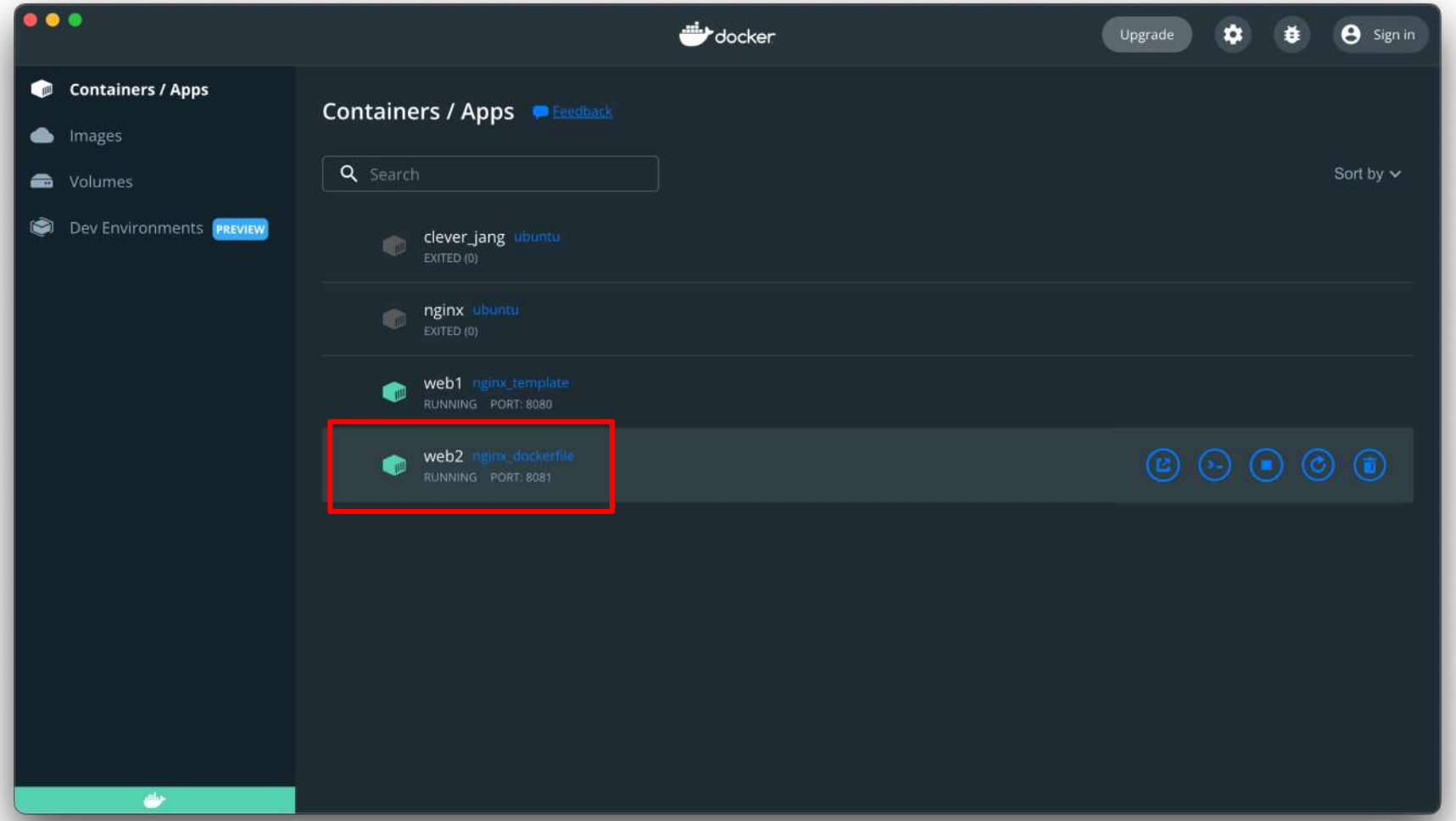
Dockerfileの使用例（2）

- Dockerイメージが作成されたことを確認
  - # docker images
- Docker buildで作成したイメージでコンテナを実行する
  - # docker run -d -p 8081:80 --name=web2 nginx_dockerfile
  - # curl localhost:8081
  - ENTRYPOINTでnginxの起動を指定しているのでコンテナで実行するコマンドの指定は不要
  - Nginxのフォアグラウンド起動は起動時オプション-gでパラメータを動的に与えています

GUIでイメージを確認



GUIで実行しているコンテナを確認



まとめ

- Dockerコンテナの実行環境を構築して、基本的なDockerコンテナの扱い方について解説しました
- コンテナは実体が見えにくいため、Docker DesktopのGUIを状態把握のために活用しましょう
- コンテナのライフサイクルについてきちんと理解しましょう
- Dockerfileで自動化に入門