Engineer

Docker

プログラムを必要なものごと届けるための仕組み

どのようなものか

Dockerは、プログラム、ライブラリー、言語のランタイム、ファイルを、イメージという一つの成果物へまとめ、それをコンテナーという隔離したプロセスとして動かす道具一式です。Dockerfileというレシピから一度イメージを作り、同じものを、ノートPC、ビルドサーバー、本番で起動します。約束は、一文で表せます。テストしたものと動くものが、バイト単位で同じということです。

主な語彙は、四つの名詞で説明できます。Dockerfileは、`FROM`、`COPY`、`RUN`などの命令を書くテキストファイルです。イメージは、その命令から作る読み取り専用のひな型で、命令ごとに一層という、レイヤーの重なりとして保存します。小さな変更後の再ビルドでは、変わったレイヤーだけを作り直します。コンテナーは、イメージの実行中のインスタンスで、上に薄い書き込み可能な層を重ねます。Docker Hubやクラウドプロバイダーのレジストリーは、イメージをプッシュし、プルする保管場所です。

コマンドラインの下には、クライアントとサーバーがあります。`docker`コマンドはクライアントで、通常はUnixソケットを通じ、REST APIで、`dockerd`という常駐デーモンへ要求を送ります。デーモンは、イメージ、コンテナー、ネットワーク、ボリュームを管理します。コンテナー自体のライフサイクルは、containerdへ渡します。containerdは、既定でruncを使います。Linuxカーネルへ名前空間と制御グループを要求し、その中でプロセスを起動する、小さなプログラムです。つまりコンテナーは、見えるシステムの範囲を制限した、ホストのカーネル上の通常のプロセスです。仮想マシンではありません。macOSやWindowsのDocker Desktopは、共有するLinuxカーネルを用意するため、小さなLinux仮想マシンを動かします。

イメージ形式は、作った企業の枠を超えました。2015年6月、Docker、CoreOSなどがOpen Container Initiativeを立ち上げ、Dockerは、イメージ形式とruncを寄贈しました。そこから生まれた仕様によって、Dockerで作ったイメージ

が、Podman、containerd、クラウドのマネージドサービスでも、そのまま動きます。

概念を判断に生かす

「Dockerとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 ノートパソコン、テストサーバー、本番環境で同じ動作になるようにまとめ、忙しい週には処理能力を増やせるようにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Google Cloud・AWSでDockerアプリ20個を運用 — 20 デプロイし監視したコンテナアプリケーション

Google Cloud・AWSでDockerアプリ20個を運用 お問い合わせ →

なぜ重要なのか

役立つ問いは、抽象的な「Dockerを使うかどうか」ではありません。その処理に、イメージを用意する価値があるか、それともホスト上の通常のサービスなら、構成要素を減らして、同じ仕事ができるかです。

プログラムが環境を伴う場合、イメージには価値があります。正確に決めたパッケージ群が必要なPythonやRのアプリ、古いランタイムに固定した既存サービス、同じライブラリーの競合する版を必要とする二つのプログラムなどです。どれも手で再現するのは大変ですが、イメージからなら簡単です。ロールバックは、前のイメージを起動する作業です。新しい開発者は、一ページのセットアップ手順をたどる代わりに、一つのコマンドを実行します。仮想マシンとの比較も、反対側から同じ兼ね合いを見ます。コンテナーは、すぐに起動して一つのカーネルを共有するので、仮想マシンが数台しか入らないホストに、数十個が入ります。その代わり、隔離は弱くなります。

代償として、インフラの運用負担が生じます。デーモンは既定ではrootで動き、ネットワークを管理しますが、Dockerはrootlessモードにも対応しています。運用するか料金を払うレジストリー、セキュリティー修正を取り込むために再ビルドが必要なベースイメージ、ホストと規則の違うストレージ層が必要です。依存関係のない、一つの静的バイナリーが得るものは、ごくわずかです。サンドボックス化したsystemdのユニットでも、同じように実行できます。

DockerとPodmanの違いも、ここで重要になります。同じイメージを動かし、コマンドの多くも共通します。Dockerは、既定では特権を持つデーモンを使用します。Podmanは、コマンドかsystemdの子プロセスとして、コンテナーを直接

起動します。ノートPCでは、小さな違いです。すでにファイアウォールとプロセスの稼働管理を設計したサーバーでは、選択を決める違いになる場合があります。

実際の活用例

このサイトの仕事には、広くDockerが登場します。一方、一つのプロジェクトでは、意図的に採用しない判断をしました。

2022年から2024年には、Google CloudとAWSで、二十のコンテナー化したアプリを動かしました。Rで書いた分析アプリも含みます。正確なパッケージ群の再現が、難しい部分です。二十のアプリを、一貫した設定と監視でデプロイし、保守しました。安定性と再現性を保ち、クラウドをまたぐ問題の診断を速くしました。

さらに大きな例では、Ubuntuのホスト上で、ApacheとNginxを組み合わせた前段の背後に、百のDockerアプリがあり、それぞれ、有効なTLS証明書が必要でした。ACMEを使うワークフローが、証明書の発行、更新、デプロイを自動化しました。百のアプリすべてで、有効な証明書を保ち、証明書の手作業をなくしました。

第三の例は、逆の方向です。セキュリティー強化用のロールがファイアウォール規則を定めるホストでは、コンテナー層に、DockerではなくPodmanとQuadletを使いました。Dockerの既定のネットワーク設定は、カーネルのパケットフィルター規則を書き換え、ファイアウォール自体の規則より先に、コンテナーのポートを公開するためです。選択とともに理由を記録しているので、次に触る人にも、既定の選択肢を忘れたのではなく、検討したと分かります。

問題が起きるところ

Dockerでは、最初のコンテナーを動かすのは簡単です。問題は、その後に現れます。通常は、次の四つの形です。

ファイアウォールが遮断したポートに、インターネットから届きます。 公開するポートは、既定で、ホストのすべてのアドレスへバインドします。Dockerが書くパケットフィルター規則は、ufwなどの道具が見る前に、その通信をコンテナーへ送ります。実際は開いていても、ファイアウォールは、閉じていると報告します。ホスト内のリバースプロキシだけが必要とするなら、ループバックアドレスへ公開します。

```
# ufwの表示にかかわらず、外部からアクセス可能
docker run -d -p 8080:80 nginx:1.27

# ホスト自身からのみアクセス可能
docker run -d -p 127.0.0.1:8080:80 nginx:1.27
```

コンテナーと一緒に、データが消えます。 コンテナー自身の書き込み層に置いたものは、コンテナーを削除すると消えます。更新は、まさにその削除を行います。データベース、アップロードしたファイル、ほかの残すべきものは、名前付きボリュームか、バインドマウントへ置きます。そのボリュームには、別途バックアップが必要です。

再ビルドで、先週と違うものができます。 `python:3` や `latest` は、参照先が動くタグです。同じDockerfileでも、別の日には別のイメージを作る場合があります。ベースイメージを特定の版に固定し、厳密な再現が必要なら、ダイジェストに固定します。ベースイメージがセキュリティー修正を公開したら、意図して再ビルドします。

認証情報を、イメージへ入れて届けます。 ビルド引数と環境変数は、最終イメージに残ります。あるレイヤーへコピーしたファイルは、後の命令で消しても、そのレイヤーに残ります。プルできる人なら、読めます。ビルドへは、シークレットマウントで渡します。一つの `RUN` の段階だけに見せ、どのレイヤーへも書き込みません。

参考資料

- Dockerの文書 — エンジン、コマンドライン、Composeの、公式マニュアル、ガイド、リファレンスです。
- Dockerの概要 — プロジェクト自身による、クライアント、デーモン、イメージ、コンテナーの説明です。
- ポートの公開と対応付け — 公開ポートが、既定で外部へ開く理由と、localhostへバインドする方法です。
- パケットフィルターとファイアウォール — Dockerによるiptablesやnftables規則の書き込みと、ufwから通信が見えなくなる場所です。
- ビルドのシークレット — イメージへ残さずに、ビルドへ認証情報を渡す方法です。
- OCIイメージ形式仕様 — 一つのイメージを、Docker、Podman、containerdのいずれでも実行できるようにする、公開標準です。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「Dockerとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/docker/