Docker クイックリファレンス

コンテナ、イメージ、ボリューム、ネットワーク、Compose

基本操作

コンテナの実行

```
docker run nginx              # run image
docker run -d nginx           # detached (background)
docker run -p 8080:80 nginx   # map port
docker run --name web nginx   # named container
docker run -it ubuntu bash    # interactive shell
```

基本コマンド

docker ps	実行中のコンテナー覧
docker ps -a	全コンテナー覧（停止済みを含む）
docker images	ローカルイメージ一覧
docker pull nginx	レジストリからイメージをダウンロード
docker info	システム全体の情報

コンテナ管理

ライフサイクル

docker start <id>	停止中のコンテナを起動
docker stop <id>	正常停止（SIGTERM）
docker kill <id>	強制停止（SIGKILL）
docker restart <id>	コンテナを再起動
docker rm <id>	停止済みコンテナを削除
docker rm -f <id>	強制削除（実行中でも可）

検査＆デバッグ

docker logs <id>	コンテナのログを表示
docker logs -f <id>	ログをリアルタイムで追跡
docker exec -it <id> bash	実行中コンテナにシェルで接続
docker inspect <id>	コンテナの詳細メタデータ（JSON）
docker top <id>	コンテナ内の実行中プロセス
docker stats	リアルタイムのリソース使用量

ファイルのコピー

```
docker cp file.txt <id>:/app/    # host → container
docker cp <id>:/app/log.txt ./   # container → host
```

イメージ

ビルド＆タグ付け

```
docker build -t myapp .          # build from Dockerfile
docker build -t myapp:v2 .       # with tag
docker tag myapp user/myapp:v2   # retag image
```

公開

```
docker login
docker push user/myapp:v2
docker pull user/myapp:v2
```

イメージ管理

docker images	全ローカルイメージの一覧
docker rmi <image>	イメージを削除
docker image prune	未使用のダングリングイメージを削除
docker system prune	未使用データを全て削除
docker history <image>	イメージのレイヤー履歴を表示

Dockerfile

主要な命令

FROM node:20	ベースイメージ
WORKDIR /app	作業ディレクトリを設定
COPY . .	ファイルをイメージにコピー
RUN npm install	ビルド時にコマンドを実行
CMD ["node", "app.js"]	実行時のデフォルトコマンド
EXPOSE 3000	リッスンポートのドキュメント化
ENV NODE_ENV=production	環境変数の設定
ARG VERSION=latest	ビルド時変数
ENTRYPOINT ["python"]	固定の実行ファイル（CMD は引数）

Dockerfile の例

```
FROM node:20-alpine
WORKDIR /app
COPY package*.json ./
RUN npm ci --production
COPY . .
EXPOSE 3000
CMD ["node", "server.js"]
```

ボリューム

永続ストレージ

```
docker volume create mydata
docker run -v mydata:/app/data nginx
docker run -v $(pwd):/app nginx  # bind mount
```

ボリュームコマンド

docker volume ls	ボリューム一覧
docker volume inspect <v>	ボリュームの詳細
docker volume rm <v>	ボリュームを削除
docker volume prune	未使用ボリュームを削除

ネットワーク

ネットワーク基本

```
docker network create mynet
docker run --network mynet --name api nginx
docker run --network mynet --name db postgres
```

ネットワークコマンド

docker network ls	ネットワーク一覧
docker network inspect <n>	ネットワークの詳細
docker network connect <n> <c>	コンテナをネットワークに接続
docker network rm <n>	ネットワークを削除

同一ネットワーク上のコンテナはコンテナ名で通信できる

Docker Compose

compose.yaml の例

```
services:
  web:
    build: .
    ports: ["3000:3000"]
    depends_on: [db]
  db:
    image: postgres:16
    environment:
      POSTGRES_PASSWORD: secret
    volumes: [pgdata:/var/lib/postgresql/data]
volumes:
  pgdata:
```

Compose コマンド

docker compose up	全サービスを起動
docker compose up -d	バックグラウンドで起動
docker compose down	コンテナを停止・削除
docker compose down -v	ボリュームも削除
docker compose build	イメージを再ビルド
docker compose logs -f	全サービスのログを追跡
docker compose ps	実行中のサービス一覧
docker compose exec web bash	サービスにシェルで接続

便利なパターン

クリーンアップコマンド

```
docker system prune -a          # remove all unused
docker container prune           # remove stopped
docker image prune -a            # remove unused images
```

クイックレシピ

一時コンテナ	**docker run --rm -it alpine sh**
ポート確認	**docker port <id>**
環境変数	**docker run -e KEY=val image**
環境変数ファイル	**docker run --env-file .env image**
再起動ポリシー	**docker run --restart unless-stopped image**
リソース制限	**docker run --memory 512m --cpus 1 image**