Docker Compose を利用したお手軽サーバー構築

北陸先端科学技術大学院大学 先端科学技術研究科（青木研究室）

長谷川 央

2023-09-30

はじめに

研究・個人開発においてサービスを self-hosting すると便利なことが多々ある

例えば · · ·

GitHub クローンの gogs
完全にプライベートで安心してソースコードを管理できる

Nextcloud
Dropbox のようにファイルを共有できる
ストレージの増築が格安でできる＆維持費は電気代のみ

Mattermost
Slack のようにチャット・通話・画面共有ができる
完全にプライベート＆課金しなくてもデータが消えない

self-hosting できる OSS 一覧

GitHub に self-hosting できる OSS の一覧がある

`https://github.com/awesome-selfhosted/awesome-selfhosted`

実際に自分が Self-hosting しているサービス

mattermost
VPS に直接インストール（メモリが少ないので）
家族との連絡用に使用中

misskey
家鯖に Docker Compose でインストール
VPS からリバースプロキシ経由で家鯖に繋がるようにしている
（実は Cloudflare Tunnel を使えば直接、家鯖に繋げられる）

gogs
研究室の NAS に Docker Compose でインストール
研究のメモや実験データの管理に使用中

Self-hosting って結構面倒くさい？

1 つのサーバに Self-hosting のサービスを複数入れると、
依存パッケージのデータ・バージョン管理が面倒くさい

昔、インストールした DB に依存する新しいサービスをインストールしようとしたものの
パスワードを忘れていたりして、いじるにいじれない状態になりがち

また、サービスを他のサーバに移行しようとすると 1 日がかりの作業になる

→ Docker Compose を使えばこれらの問題を解決できる

Docker Compose とは？

複数の Docker コンテナを一括管理するためのツール

YAML ファイルを書いてコマンドを 1 つ実行するだけで、
複数のコンテナを連動させたサービスを立ち上げることができる
Self-hosting 系の OSS は、PostgreSQL などの DB を使用することが多いので とても便利

サービスの使用するデータは Host 側（コンテナの外）に保存できるので、
YAML ファイルとデータのディレクトリだけ移動させれば、
簡単に他のサーバに移行することができる

Docker Compose を使用する手順

1. Docker と Docker Compose をインストールする
2. YAML ファイルを書く
3. `docker-compose up -d` を実行する

YAML の例—Misskey

```yaml
version: "3"

services:
  web:
    build: .
    restart: always
    links:
      - db
      - redis
    depends_on:
      db:
        condition: service_healthy
      redis:
        condition: service_healthy
    ports:
      - "3000:3000"
    networks:
      - internal_network
      - external_network
    volumes:
      - ./files:/misskey/files
      - ./.config:/misskey/.config:ro
networks:
  internal_network:
    internal: true
  external_network:
```

```yaml
  redis:
    restart: always
    image: redis:7-alpine
    networks:
      - internal_network
    volumes:
      - ./redis:/data
    healthcheck:
      test: "redis-cli ping"
      interval: 5s
      retries: 20

  db:
    restart: always
    image: postgres:15-alpine
    networks:
      - internal_network
    env_file:
      - .config/docker.env
    volumes:
      - ./db:/var/lib/postgresql/data
    healthcheck:
      test: "pg_isready -U $$POSTGRES_USER -d $$POSTGRES_DB"
      interval: 5s
      retries: 20
```

https://github.com/misskey-dev/misskey/blob/develop/docker-compose.yml.example

YAML の例—gogs

```yaml
version: '3'
services:
  db:
    image: postgres:latest
    environment:
      - POSTGRES_USER=gogs
      - POSTGRES_PASSWORD=#####
      - POSTGRES_DB=gogs
    volumes:
      - ./dbdata:/var/lib/postgresql/data
    restart: always
  gogs:
    image: gogs/gogs
    ports:
      - "10880:3000"
      - "10022:22"
    depends_on:
      - db
    volumes:
      - ./data:/data
    restart: always
```

https://qiita.com/mkakh/items/e624c3e836a3994cf84f

※実は Bing Chat にほぼ書かせていて、僕が書いたのは Port 番号の設定とパスワードだけ

YAML で書く設定内容

YAML では `environment` の部分で DB の設定を書いているが、
これについては各コンテナの説明ページに書いてあるから自力で探さなくても良い
`https://hub.docker.com/_/postgres/`

YAML を書いた後

YAML を書いたら、あとはコマンドを 1 つ実行するだけ

```
# docker-compose up -d
[+] Running 3/3
 ✔ Network gogs_default   Created                                    0.1s
 ✔ Container gogs-db-1    Created                                    0.0s
 ✔ Container gogs-gogs-1  Created                                    0.1s
...
```

YAML で `restart: always` を設定していれば、
サーバを再起動しても自動で立ち上がってくれる

まとめ

あまりサーバーに関して詳しくなくても Docker Compose を使えば
お手軽にサービスを立ち上げられる

しかも、Docker を使っているから元の環境に影響を与えずに、色々と試すことができる

別のサーバーで本格運用したいときも、移行が簡単だから安心

ぜひ使ってみてください