Docker Compose

Monolithic Deployment

If you prefer not to deploy any additional containers, you can use the following docker-compose configuration to start a monolithic service using the monolith image:

```
services:
  monolith:
    container_name: beaver-iot
    image: milesight/beaver-iot:latest
    restart: always
    ports:
      - "80:80"
      # Websocket
      - "8083:8083"
      # If you want to use the built-in MQTT broker, you need to
map port 1883
      - "1883:1883"
    environment:
      # Configure database connection (using h2 as default)
      - "DB_TYPE=h2"
      - "SPRING_DATASOURCE_URL=jdbc:h2:file:~/beaver-
iot/h2/beaver;AUTO_SERVER=TRUE"
      - "SPRING_DATASOURCE_USERNAME=sa"
      - "SPRING_DATASOURCE_PASSWORD="
      - "SPRING_DATASOURCE_DRIVER_CLASS_NAME=org.h2.Driver"
    volumes:
```

```
    # Persist database data and log files
    - "./beaver-iot/:/root/beaver-iot/"
```

Separate Frontend and Backend Deployment

If you need to deploy frontend and backend containers separately, you can use the following docker-compose configuration to deploy `nginx`, `web`, and `api` containers:

```
services:
  nginx:
    container_name: beaver-iot-nginx
    image: nginx:stable-alpine3.20-slim
    restart: always
    ports:
      - "80:80"
    volumes:
      # Nginx config files should be prepared by yourself
      - "./nginx/nginx.conf:/etc/nginx/nginx.conf"
      - "./nginx/conf.d/:/etc/nginx/conf.d/"
  web:
    container_name: beaver-iot-web
    image: milesight/beaver-iot-web:latest
    restart: always
  api:
    container_name: beaver-iot-api
    image: milesight/beaver-iot-api:latest
    restart: always
    environment:
      # Configure database connection (using h2 as default)
```

```
      - "DB_TYPE=h2"
      - "SPRING_DATASOURCE_URL=jdbc:h2:file:~/beaver-
iot/h2/beaver;AUTO_SERVER=TRUE"
      - "SPRING_DATASOURCE_USERNAME=sa"
      - "SPRING_DATASOURCE_PASSWORD="
      - "SPRING_DATASOURCE_DRIVER_CLASS_NAME=org.h2.Driver"
    volumes:
      # Persist database data and log files
      - "./beaver-iot/:/root/beaver-iot/"
```

> **TIP**
>
> You can also extract the page files from the `/web` path in the web container and host them on another HTTP server.

Using Postgres Database

If you wish to use a Postgres database instead of an H2 database, simply modify the environment variables for the `monolith` or `api` container in the above configurations:

```
services:
  nginx:
    container_name: beaver-iot-nginx
    image: nginx:stable-alpine3.20-slim
    restart: always
    ports:
      - "80:80"
      - "443:443"
    volumes:
      # Nginx config files should be prepared by yourself
```

```
      - "./nginx/nginx.conf:/etc/nginx/nginx.conf"
      - "./nginx/conf.d/:/etc/nginx/conf.d/"
  web:
    container_name: beaver-iot-web
    image: milesight/beaver-iot-web:latest
    restart: always
  api:
    container_name: beaver-iot-api
    image: milesight/beaver-iot-api:latest
    restart: always
    environment:
      # Configure database connection
      - "DB_TYPE=postgres"
      -
"SPRING_DATASOURCE_DRIVER_CLASS_NAME=org.postgresql.Driver"
      - "SPRING_DATASOURCE_USERNAME=postgres"
      - "SPRING_DATASOURCE_PASSWORD=postgres"
      - "SPRING_DATASOURCE_URL=jdbc:postgresql://beaver-iot-
postgresql:5432/postgres"
    volumes:
      # Persist log files
      - "./beaver-iot/logs/:/root/beaver-iot/logs/"
      # Load integrations
      - "./beaver-iot/integrations/:/root/beaver-
iot/integrations/"
  postgresql:
    container_name: beaver-iot-postgresql
    image: postgres:17.0-alpine3.20
    restart: always
    ports:
      - "5432:5432"
    environment:
      - "POSTGRES_USER=postgres"
      - "POSTGRES_PASSWORD=postgres"
      - "POSTGRES_DB=postgres"
      - "PGDATA=/var/lib/postgresql/data/pgdata"
```

```
    volumes:
      - "./postgresql/:/var/lib/postgresql/data/"
```

> **TIP**
>
> You can deploy the Postgres database anywhere as long as you configure the `SPRING_DATASOURCE_URL` correctly.

Edit this page