AI-03947 Fix OpenProject Shared DB

Owner	Ⓡ Rigel Arcayan
Tags	Project
Created time	June 17, 2026 3:51 PM

Open Project Deployment with postgres-shared db.

I need help in creating an OpenProject Docker application with a seprated db from the app - postgres-shared. I've shared the infrastructure yml, env and yml with a working OpenProject. I'm running a working NPM for my dockerlab. Domain=blindcaveman.duckdns.org

I need a new env and compose files. The postgres db is already running with Inventree working as well.

At the near end of your response, please explain the concept so that I can understand,

To deploy OpenProject using your existing `postgres-shared` database, we need to strip out the internal database service from the OpenProject stack and bridge OpenProject into your existing `dockerlab-network`. This allows the application to communicate directly with your shared database and Nginx Proxy Manager (NPM) using internal Docker DNS names.

Here are your optimized, production-ready configuration files, followed by a beginner-friendly breakdown of how it all works.

1. The Dynamic Environment File (`.env`)

Save this as `.env` in your OpenProject directory.

Code snippet

```
TAG=17-slim
```

```
# 1. Routing & Security
OPENPROJECT_HTTPS=true
OPENPROJECT_HOST__NAME=blindcaveman.duckdns.org
OPENPROJECT_HSTS=true

# 2. Internal Port Binding (Listens on the VM, to be picked up by NPM)
PORT=8080

# 3. Production Security Keys
SECRET_KEY_BASE=a3f9e2b8c7d6e5f4a3b2c1d0e9f8a7b6c5d4e3f2a1b0c9d8e7f6a5b4c3d2e1f0
COLLABORATIVE_SERVER_SECRET=7f6e5d4c3b2a10fe9d8c7b6a5f4e3d2c1b0a9f8e7d6c5b4a3f2e1d0c9b8a7
f6e

# 4. Collaborative Text Feature Sync
COLLABORATIVE_SERVER_URL=wss://blindcaveman.duckdns.org/hocuspocus

# 5. Persistent Volume Mapping
OPDATA=opdata

# 6. Shared Database Connection (Targeting postgres-shared)
# Format: postgres://[user]:[password]@[container_name]/[database_name]
# NOTE: Replace 'YourSharedDbPasswordHere' with your actual ${DB_ROOT_PASSWORD} value
DATABASE_URL=postgres://postgres:YourSharedDbPasswordHere@postgres-shared/openproject?poo
l=20&encoding=unicode&reconnect=true

# 7. Performance Tuning
IMAP_ENABLED=false
RAILS_MIN_THREADS=4
RAILS_MAX_THREADS=16
```

> ⚠️ **Important Step**: OpenProject requires its own database instance or schema. Before starting this stack, log into your shared database (via pgAdmin or `docker exec`) and run the command: `CREATE DATABASE openproject;`. Also, make sure to replace `YourSharedDbPasswordHere` in the `.env` file with your actual database root password.

2. The Unified Docker Compose File (`docker-compose.yml`)

Save this as `docker-compose.yml` in the same directory. Note that the `db` service has been entirely removed, and all services now attach to your external `dockerlab-network`.

YAML

```
version: "3.8"

networks:
  # Hook directly into your existing infrastructure network
  dockerlab-network:
    name: dockerlab-network
    external: true

volumes:
  opdata:

# Reusable configuration blocks using Docker Extension Anchors
x-op-restart-policy: &restart_policy
  restart: unless-stopped

x-op-image: &image
  image: openproject/openproject:${TAG}

x-op-app: &app
  <<: [*image, *restart_policy]
  environment:
    OPENPROJECT_HTTPS: "${OPENPROJECT_HTTPS}"
    OPENPROJECT_HOST__NAME: "${OPENPROJECT_HOST__NAME}"
    OPENPROJECT_ADDITIONAL__HOST__NAMES: "web"
    OPENPROJECT_HSTS: "${OPENPROJECT_HSTS}"
    OPENPROJECT_RAILS__CACHE__STORE: "memcache"
    OPENPROJECT_CACHE__MEMCACHE__SERVER: "cache:11211"
    OPENPROJECT_RAILS__RELATIVE__URL__ROOT: "${OPENPROJECT_RAILS__RELATIVE__URL__ROOT}"
    OPENPROJECT_COLLABORATIVE__EDITING__HOCUSPOCUS__URL: "${COLLABORATIVE_SERVER_URL}"
    OPENPROJECT_COLLABORATIVE__EDITING__HOCUSPOCUS__SECRET: "${COLLABORATIVE_SERVER_SECRE
T}"
    DATABASE_URL: "${DATABASE_URL}"
    RAILS_MIN_THREADS: ${RAILS_MIN_THREADS}
    RAILS_MAX_THREADS: ${RAILS_MAX_THREADS}
    IMAP_ENABLED: "${IMAP_ENABLED}"
    SECRET_KEY_BASE: "${SECRET_KEY_BASE}"
  volumes:
    - ${OPDATA}:/var/openproject/assets
  networks:
    - dockerlab-network

services:
  cache:
    image: memcached:alpine
    <<: *restart_policy
    networks:
      - dockerlab-network

  proxy:
    build:
```

```
      context: ./proxy
      args:
        APP_HOST: web
    image: openproject/proxy
    <<: *restart_policy
    ports:
      - "${PORT}:80"
    depends_on:
      - web
    networks:
      - dockerlab-network

  web:
    <<: *app
    command: "./docker/prod/web"
    hostname: "${OPENPROJECT_HOST__NAME}"
    depends_on:
      - cache
      - seeder
    labels:
      - autoheal=true
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080${OPENPROJECT_RAILS__RELATIVE__UR
L__ROOT}/health_checks/default"]
      interval: 10s
      timeout: 3s
      retries: 3
      start_period: 30s

  autoheal:
    image: willfarrell/autoheal:1.2.0
    <<: *restart_policy
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock"
    environment:
      AUTOHEAL_CONTAINER_LABEL: autoheal
      AUTOHEAL_START_PERIOD: 600
      AUTOHEAL_INTERVAL: 30
    networks:
      - dockerlab-network

  worker:
    <<: *app
    command: "./docker/prod/worker"
    depends_on:
      - cache
      - seeder

  cron:
    <<: *app
    command: "./docker/prod/cron"
    depends_on:
```

```
      - cache
      - seeder

  seeder:
    <<: *app
    command: "./docker/prod/seeder"
    restart: on-failure

  hocuspocus:
    image: openproject/hocuspocus:17.5.0
    <<: *restart_policy
    environment:
      SECRET: "${COLLABORATIVE_SERVER_SECRET}"
      OPENPROJECT_URL: "http://web:8080"
      OPENPROJECT_HTTPS: "${OPENPROJECT_HTTPS}"
    networks:
      - dockerlab-network
```

3. Nginx Proxy Manager (NPM) Configuration

To expose OpenProject to the web securely via your subdomains, configure **two separate proxy hosts** inside your NPM admin UI:

Host 1: Main Application Interface

- **Domain Name:** `blindcaveman.duckdns.org`
- **Scheme:** `http`
- **Forward Hostname/IP:** `proxy` *(Since both stacks share the same network, NPM can target the OpenProject proxy container directly by name)*
- **Forward Port:** `80`
- **Block Common Exploits:** Enabled
- **SSL Tab:** Request a new SSL Certificate (or use your existing one), check **Force SSL** and **HTTP/2 Support**.

Host 2: WebSockets Sync Server (Hocuspocus)

- **Domain Name:** `blindcaveman.duckdns.org`
- **Custom Location Tab:** Add a custom location configuration to intercept WebSocket traffic:

- **Define Location:** `/hocuspocus`
- **Scheme:** `http`
- **Forward Hostname/IP:** `hocuspocus`
- **Forward Port:** `5000`
- **Click the Gear Icon (Advanced Settings) for this location and paste:**Nginx

```
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
```

Concept Breakdown for the Budding Cloud Architect

Welcome to the world of self-hosted cloud infrastructure! When transitioning from single, standalone applications to a consolidated home lab environment, understanding how containers talk to each other is crucial. Here are the core concepts driving your new setup:

1. External Docker Networks (The Shared Highway)

By default, when you run `docker compose up`, Docker creates an isolated, private network just for the containers listed in that specific file. They can talk to each other, but they are invisible to containers outside that file.

By declaring:

YAML

```
networks:
  dockerlab-network:
    external: true
```

We are telling OpenProject: *"Do not build a private island. Instead, drive onto the pre-existing highway called* `dockerlab-network`*."* Because your NPM, pgAdmin, and `postgres-shared` containers are already driving on that exact same highway, they can all see and talk to each other natively.

2. Embedded DNS (Container Name Resolution)

In a traditional computing setup, you connect to a database using an IP address (like `192.168.1.50`). In Docker, IP addresses change every single time a container restarts.

To solve this, Docker features an internal Domain Name System (DNS). When containers share a network, they can use each other's **container names** as web addresses. This is why your database string works seamlessly by targeting `postgres-shared` instead of an unstable IP address:

> `postgres://postgres:password@**postgres-shared**/openproject`

3. Consolidation vs. Isolation

Your previous OpenProject configuration spun up its own dedicated PostgreSQL instance. While highly isolated, running an individual database container for every application you host consumes significant system RAM and CPU overhead.

By migrating to `postgres-shared`, you have centralized your data layer. Your host VM runs one highly optimized database engine handling multiple databases simultaneously (e.g., Inventree, OpenProject, WordPress). This drastically minimizes resources while simplifying global data backups.