Get Puzzles

An open source, self hosted puzzle game built from your photos.

- Install Get Puzzles

Install Get Puzzles

https://pt.opensourceisawesome.com/videos/embed/47UYTxE57s8SH2cjFVi4Qt

Setup a Server

Install Updates

Update and upgrade your server's packages using the following commands:

- For Ubuntu/Debian: `sudo apt update && sudo apt upgrade -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf update -y`

Create a Non-Root User

Create a non-root user with superuser (sudo) privileges:

1. Add a new user using `adduser <username>`
2. Set the password for this user.
3. Enter the relevant information (optional)
4. Enter 'Y', then press Enter.
5. Add the user to the "sudo" group:
- For Ubuntu/Debian: `usermod -aG sudo <username>`
- For RedHat/CentOS/Fedora/Alma/Rocky: `usermod -aG wheel <username>`

Ubuntu / Debian

```
usermod -aG sudo <username>
```

RedHat / CentOS / Fedora / Alma / Rocky

```
usermod -aG wheel <username>
```

Now, you can log out of the system, and log back in as your new non-root super user.

Install Docker and Docker Compose

Install Docker and Docker Compose on your server:

1. Install the curl utility:
- For Ubuntu/Debian: `sudo apt install curl -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf install curl -y`
2. Run the command to install Docker and Docker Compose:

`curl https://get.docker.com | sh`

Add Your User to the Docker Group

Add your non-root user to the docker group so you can use Docker commands without sudo:

`sudo usermod -aG docker <username>`

Now log out and back into your system, and you are ready to setup your application.

Install Get Puzzles

First, let's setup our folder structure. If you've already done this, continue on, but if not, this can help you run multiple applications in Docker (Podman) on a single device.

`mkdir -p docker/get_puzzles`

Now we'll move into the directory we just made:

`cd docker/get_puzzles`

We need to create two files. A compose file for our application configuration, and an environment file for our applications environment variables. First, let's create our compose file, which we'll call "compose.yaml"

`nano compose.yaml`

In the editor that opens, copy and paste the content below:

```
services:
  # ── PostgreSQL ──────────────────────────────────────────────────────
  postgres:
    image: postgres:16-alpine
    container_name: puzzle-postgres
    environment:
      POSTGRES_USER: ${DB_USER:-puzzle_user}
      POSTGRES_PASSWORD: ${DB_PASSWORD:-puzzle_password}
      POSTGRES_DB: ${DB_NAME:-puzzle_db}
    volumes:
```

```
      - ./postgres_data:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql:ro
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${DB_USER:-puzzle_user} -d ${DB_NAME:-puzzle_db}"]
      interval: 5s
      timeout: 5s
      retries: 5
    restart: unless-stopped
    networks:
      - puzzle-net
   # ── MinIO (S3-compatible object storage) ──────────────────────────────
  minio:
    image: minio/minio:latest
    container_name: puzzle-minio
    environment:
      MINIO_ROOT_USER: ${MINIO_ACCESS_KEY:-minio_admin}
      MINIO_ROOT_PASSWORD: ${MINIO_SECRET_KEY:-minio_password}
    volumes:
      - ./minio_data:/data
    command: server /data --console-address ":9001"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:9000/minio/health/live"]
      interval: 5s
      timeout: 5s
      retries: 5
    restart: unless-stopped
    networks:
      - puzzle-net
# ── ClamAV (virus scanner) ────────────────────────────────────────────────
  clamav:
    image: clamav/clamav:latest
    container_name: puzzle-clamav
    volumes:
      - ./clamav_data:/var/lib/clamav
    healthcheck:
      test: ["CMD-SHELL", "/usr/local/bin/clamdcheck.sh || exit 1"]
      interval: 10s
      timeout: 10s
      retries: 10
      start_period: 120s
```

```
    restart: unless-stopped
    networks:
      - puzzle-net
# ── Application (combined frontend + backend) ──────────────────────────
 app:
    image: bmcgonag/get-puzzles:latest
    container_name: puzzle-app
    environment:
      NODE_ENV: production
      # ── Database ──
      DB_HOST: postgres
      DB_PORT: 5432
      DB_USER: ${DB_USER:-puzzle_user}
      DB_PASSWORD: ${DB_PASSWORD:-puzzle_password}
      DB_NAME: ${DB_NAME:-puzzle_db}
      # ── MinIO ──
      MINIO_ENDPOINT: minio
      MINIO_PORT: 9000
      MINIO_USE_SSL: "false"
      MINIO_ACCESS_KEY: ${MINIO_ACCESS_KEY:-minio_admin}
      MINIO_SECRET_KEY: ${MINIO_SECRET_KEY:-minio_password}
      MINIO_BUCKET: ${MINIO_BUCKET:-puzzle-uploads}
      # ── ClamAV ──
      CLAMAV_HOST: clamav
      CLAMAV_PORT: 3310
      # ── Frontend / CORS ──
      FRONTEND_URL: ${FRONTEND_URL:-http://localhost}
      NEXT_PUBLIC_API_URL: ${NEXT_PUBLIC_API_URL:-http://localhost/api}
      PORT: "3000"
      BACKEND_PORT: "3001"
    ports:
      - "${APP_PORT:-8080}:80"
    depends_on:
      postgres:
        condition: service_healthy
      minio:
        condition: service_healthy
      clamav:
        condition: service_healthy
    restart: unless-stopped
```

```
    networks:
      - puzzle-net

networks:
  puzzle-net:
    driver: bridge
```

Looking through the configuration, you'll see multiple services being setup. We have Postgres, the database where we store all the relevant puzzle information, users, etc. Next, we have Minio, an object storage system made for storing the images your puzzles are based on. Then we have ClamAV. This is to scan the images being uploaded to make sure there is nothing malicious in them in an attempt to protect your server. Finally, we have App, this is the actual Get Puzzles application, which utilizes the other services to give your the working app.

Save the file with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Next, we need to create the environment variable file. All those values in the configuration above that have "${SOME_CAPITAL_LETTERS}" format, are variables that allow you to setup your system in a way which is secure, and allows you to run the system with your own domain or sub-domain.

Let's create that file, called ".env". The dot (.) in front of the filename means it's hidden, and you need to use special flags to see it when listing the files in the directory.

`nano .env`

Now, copy and paste the following into that file:

```
# POSTGRES DATA
DB_USER: puzzle
DB_PASSWORD:puzzle_password
DB_NAME:-puzzle_db

# MINIO DATA
MINIO_ACCESS_KEY: minio_admin
MINIO_SECRET_KEY: minio_password
MINIO_BUCKET: puzzle-uploads

# APP
FRONTEND_URL: https://localhost
NEXT_PUBLIC_API_URL: http://localhost/api
APP_PORT: 8080
```

In this file, you'll want to change the values (particularly for passwords and secrets) to long strong passwords and secrets in order to keep your system safe.

You'll want to replace `localhost` with an IP address for your server, or the domain / subdomain you'll be using to access your server (e.g. puzzle.yourdomain.net).

Save this file with CTRL + O, then press Enter to confirm, and exit the editor with CTRL + X.

Setup the initial SQL script needed.

In the same directory we will add a file called 'init.sql'. This file will have the script needed to get the system setup initially. You only need it for the first time the app runs. You can do one of the following:

Directly download the file using:

`wget` `https://forge.routemehome.org/brian/get_puzzles/raw/branch/main/init.sql`

Then skip down to the Pull Image section.

Or, create the file with:

`nano init.sql`

then copy and paste the following script code into that file:

```
-- Users table
CREATE TABLE IF NOT EXISTS users (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    email VARCHAR(255) UNIQUE NOT NULL,
    username VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255),
    role VARCHAR(20) NOT NULL DEFAULT 'player' CHECK (role IN ('player', 'admin')),
    email_verified BOOLEAN NOT NULL DEFAULT FALSE,
    totp_secret VARCHAR(255),
    encryption_salt VARCHAR(255),
    ip_address VARCHAR(45),
    last_active_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    status VARCHAR(20) NOT NULL DEFAULT 'active' CHECK (status IN ('active', 'banned')),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Sessions table for Lucia Auth
```

```
CREATE TABLE IF NOT EXISTS sessions (
    id VARCHAR(255) PRIMARY KEY,
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL
);

-- Uploaded photos table
CREATE TABLE IF NOT EXISTS uploaded_photos (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    file_path VARCHAR(500),
    encrypted_file_path VARCHAR(500),
    encrypted_dek TEXT,
    iv VARCHAR(255),
    salt VARCHAR(255),
    original_filename VARCHAR(255),
    mime_type VARCHAR(100),
    file_size INTEGER,
    width INTEGER,
    height INTEGER,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    deleted_at TIMESTAMP WITH TIME ZONE
);

-- Puzzles table
CREATE TABLE IF NOT EXISTS puzzles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    uploaded_photo_id UUID REFERENCES uploaded_photos(id) ON DELETE SET NULL,
    piece_count INTEGER NOT NULL,
    cut_style VARCHAR(50) NOT NULL DEFAULT 'classic' CHECK (cut_style IN ('classic',
'geometric', 'spiral')),
    has_rotation BOOLEAN NOT NULL DEFAULT FALSE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Puzzle progress table
CREATE TABLE IF NOT EXISTS puzzle_progress (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    puzzle_id UUID REFERENCES puzzles(id) ON DELETE SET NULL,
```

```
    status VARCHAR(50) NOT NULL DEFAULT 'in_progress' CHECK (status IN ('in_progress',
'solved')),
    state_json JSONB,
    hints_used INTEGER NOT NULL DEFAULT 0,
    started_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    completed_at TIMESTAMP WITH TIME ZONE
);

-- Wall items table for gallery
CREATE TABLE IF NOT EXISTS wall_items (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    uploaded_photo_id UUID REFERENCES uploaded_photos(id) ON DELETE SET NULL,
    puzzle_id UUID REFERENCES puzzles(id) ON DELETE SET NULL,
    x_pos FLOAT NOT NULL DEFAULT 0,
    y_pos FLOAT NOT NULL DEFAULT 0,
    target_width FLOAT NOT NULL DEFAULT 200,
    target_height FLOAT NOT NULL DEFAULT 200,
    z_index INTEGER NOT NULL DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Audit logs table
CREATE TABLE IF NOT EXISTS audit_logs (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID REFERENCES users(id) ON DELETE SET NULL,
    action VARCHAR(100) NOT NULL,
    entity_type VARCHAR(50),
    entity_id VARCHAR(255),
    details JSONB,
    ip_address VARCHAR(45),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Admin settings table
CREATE TABLE IF NOT EXISTS admin_settings (
    key VARCHAR(100) PRIMARY KEY,
    value TEXT NOT NULL,
```

```
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- SMTP settings table
CREATE TABLE IF NOT EXISTS smtp_settings (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    host VARCHAR(255) NOT NULL,
    port INTEGER NOT NULL,
    username VARCHAR(255) NOT NULL,
    password_encrypted TEXT NOT NULL,
    encryption_type VARCHAR(10) NOT NULL DEFAULT 'tls' CHECK (encryption_type IN ('tls',
'ssl', 'none')),
    from_email VARCHAR(255) NOT NULL,
    from_name VARCHAR(255),
    is_active BOOLEAN NOT NULL DEFAULT FALSE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- IP blocks table
CREATE TABLE IF NOT EXISTS ip_blocks (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    ip_address VARCHAR(45) NOT NULL UNIQUE,
    reason TEXT,
    blocked_by UUID REFERENCES users(id) ON DELETE SET NULL,
    expires_at TIMESTAMP WITH TIME ZONE,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Password reset tokens table
CREATE TABLE IF NOT EXISTS password_reset_tokens (
    token VARCHAR(255) PRIMARY KEY,
    user_id UUID NOT NULL UNIQUE REFERENCES users(id) ON DELETE CASCADE,
    expires_at TIMESTAMP WITH TIME ZONE NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Social sharing tables
CREATE TABLE IF NOT EXISTS shared_puzzles (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    puzzle_id UUID NOT NULL REFERENCES puzzles(id) ON DELETE CASCADE,
    token VARCHAR(64) UNIQUE NOT NULL,
    allow_gallery_view BOOLEAN DEFAULT FALSE,
    view_count INTEGER DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    revoked_at TIMESTAMP WITH TIME ZONE
);

CREATE TABLE IF NOT EXISTS shared_galleries (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    user_id UUID NOT NULL REFERENCES users(id) ON DELETE CASCADE,
    token VARCHAR(64) UNIQUE NOT NULL,
    view_count INTEGER DEFAULT 0,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    revoked_at TIMESTAMP WITH TIME ZONE
);

-- Indexes for performance
CREATE INDEX IF NOT EXISTS idx_users_email ON users(email);
CREATE INDEX IF NOT EXISTS idx_users_username ON users(username);
CREATE INDEX IF NOT EXISTS idx_uploaded_photos_user_id ON uploaded_photos(user_id);
CREATE INDEX IF NOT EXISTS idx_puzzles_photo_id ON puzzles(uploaded_photo_id);
CREATE INDEX IF NOT EXISTS idx_puzzle_progress_user_id ON puzzle_progress(user_id);
CREATE INDEX IF NOT EXISTS idx_puzzle_progress_puzzle_id ON puzzle_progress(puzzle_id);
CREATE INDEX IF NOT EXISTS idx_wall_items_user_id ON wall_items(user_id);
CREATE INDEX IF NOT EXISTS idx_audit_logs_user_id ON audit_logs(user_id);
CREATE INDEX IF NOT EXISTS idx_audit_logs_created_at ON audit_logs(created_at);
CREATE INDEX IF NOT EXISTS idx_shared_puzzles_token ON shared_puzzles(token);
CREATE INDEX IF NOT EXISTS idx_shared_puzzles_user_id ON shared_puzzles(user_id);
CREATE INDEX IF NOT EXISTS idx_shared_puzzles_puzzle_id ON shared_puzzles(puzzle_id);
CREATE INDEX IF NOT EXISTS idx_shared_galleries_token ON shared_galleries(token);
CREATE INDEX IF NOT EXISTS idx_shared_galleries_user_id ON shared_galleries(user_id);
```

Save the file with CTRL + O, then press Enter to confirm. Now exit the nano editor with CTRL + X.

Now we are ready to start pulling images, and getting the application started.

Pull the Images

Now we'll pull the images needed to run our application.

```
docker compose pull
```

Start the Application Containers

Once those images have pulled, we'll start our application containers with:

```
docker compose up -d
```

You can view the logs with:

```
docker compose logs -f
```

If you prefer, you can concatenate the commands to just start the containers and follow the logs as the system starts:

```
docker compose up -d && docker compose logs -f
```

Once the logs settle, you'll want to navigate to the IP address of your application server with the port you set in the .env file. If you set port 8020, and your application is running on a machine with IP 192.168.1.21, then you'd go to

http://192.168.1.21:8020

in your browser. Additionally, if you're setting up a domain / sub-domain, then you'll want to navigate to that name, and ensure your web server (NGinX or Apache) config is working. Finally, if you are setting up a Reverse Proxy (e.g. Pangolin, Netbird, Tailscale, NGinX Proxy Manager, etc), thne you'll want to make sure you use the proper IP and port to access your system.