Saya akan membantu Anda membuat container dengan 3 service tersebut. Mari kita mulai bertahap dari instalasi Laravel dengan Composer.

Langkah 1: Setup Laravel dengan Composer

Pertama, buat direktori project dan install Laravel:

```
# Buat direktori project
mkdir laravel-docker-app
cd laravel-docker-app

# Install Laravel menggunakan Composer
composer create-project laravel/laravel .

# Atau jika ingin versi spesifik
# composer create-project laravel/laravel . "10.*"

# Set permission untuk storage dan bootstrap/cache
chmod -R 775 storage
chmod -R 775 bootstrap/cache

# Generate application key (biasanya sudah otomatis)
php artisan key:generate
```

Langkah 2: Konfigurasi Environment Laravel

Setelah Laravel terinstall, kita perlu mengkonfigurasi file `.env` untuk PostgreSQL:Perhatikan bahwa `DB_HOST=postgres` menggunakan nama service yang akan kita definisikan di Docker Compose nanti.

Apakah Anda ingin melanjutkan ke langkah berikutnya untuk membuat Dockerfile dan konfigurasi Docker Compose? Atau ada yang perlu diperjelas dari setup Laravel ini terlebih dahulu?

```
# File .env
APP_NAME=LaravelApp
APP_ENV=local
APP_KEY=base64:your-generated-key
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

# Database Configuration untuk PostgreSQL
DB_CONNECTION=pgsql
DB_HOST=postgres
DB_PORT=5432
DB_DATABASE=laravel_db
```

```
DB_USERNAME=laravel_user
DB_PASSWORD=laravel_password

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailpit
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=
AWS_USE_PATH_STYLE_ENDPOINT=false

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_HOST=
PUSHER_PORT=443
PUSHER_SCHEME=https
PUSHER_APP_CLUSTER=mt1

VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
VITE_PUSHER_HOST="${PUSHER_HOST}"
VITE_PUSHER_PORT="${PUSHER_PORT}"
VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

Langkah 3: Membuat Dockerfile untuk Laravel

Sekarang kita buat Dockerfile untuk aplikasi Laravel:

```
# Dockerfile
FROM php:8.2-fpm

# Set working directory
WORKDIR /var/www

# Install system dependencies
RUN apt-get update && apt-get install -y \
    git \
    curl \
    libpng-dev \
    libonig-dev \
    libxml2-dev \
    libpq-dev \
    zip \
    unzip \
    nginx

# Clear cache
RUN apt-get clean && rm -rf /var/lib/apt/lists/*

# Install PHP extensions
RUN docker-php-ext-install pdo pdo_pgsql mbstring exif pcntl bcmath gd

# Get latest Composer
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer

# Copy existing application directory contents
COPY . /var/www

# Copy existing application directory permissions
COPY --chown=www-data:www-data . /var/www

# Change current user to www-data
USER www-data

# Install dependencies
RUN composer install --no-interaction --optimize-autoloader --no-dev

# Generate application key
RUN php artisan key:generate

# Cache configuration and routes
RUN php artisan config:cache
RUN php artisan route:cache

# Expose port 9000 and start php-fpm server
EXPOSE 9000
CMD ["php-fpm"]
```

Langkah 4: Konfigurasi NGINX

Buat konfigurasi NGINX untuk melayani aplikasi Laravel:

```
# docker/nginx/default.conf
server {
    listen 80;
    server_name localhost;
    root /var/www/public;
    index index.php index.html index.htm;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        fastcgi_pass laravel:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }

    location ~ /\.ht {
        deny all;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires 1y;
        add_header Cache-Control "public, immutable";
        try_files $uri =404;
    }

    error_log /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
}
```

Langkah 5: Docker Compose Configuration

Sekarang kita buat file docker-compose.yml yang menggabungkan ketiga service:

```
# docker-compose.yml
version: '3.8'

services:
  # Laravel Application
  laravel:
    build:
```

```
      context: .
      dockerfile: Dockerfile
    container_name: laravel_app
    restart: unless-stopped
    working_dir: /var/www
    volumes:
      - ./:/var/www
      - ./storage:/var/www/storage
    networks:
      - laravel_network
    depends_on:
      - postgres

  # Nginx Service
  nginx:
    image: nginx:alpine
    container_name: nginx_server
    restart: unless-stopped
    ports:
      - "80:80"
    volumes:
      - ./:/var/www
      - ./docker/nginx:/etc/nginx/conf.d
    networks:
      - laravel_network
    depends_on:
      - laravel

  # PostgreSQL Database
  postgres:
    image: postgres:15
    container_name: postgres_db
    restart: unless-stopped
    environment:
      POSTGRES_DB: laravel_db
      POSTGRES_USER: laravel_user
      POSTGRES_PASSWORD: laravel_password
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
    networks:
      - laravel_network

# Networks
networks:
  laravel_network:
    driver: bridge

# Volumes
volumes:
  postgres_data:
    driver: local
```

Langkah 6: Struktur Direktori dan File Tambahan

Buat struktur direktori yang diperlukan:

```
# Buat struktur direktori
mkdir -p docker/nginx

# Copy konfigurasi NGINX ke direktori yang tepat
# Buat file docker/nginx/default.conf dengan isi konfigurasi NGINX di atas

# Tambahkan ke .gitignore
echo "
# Docker
docker-compose.override.yml
.env.docker

# Database
postgres_data/
" >> .gitignore

# Buat file .dockerignore
cat > .dockerignore << 'EOF'
.git
.env
docker-compose*.yml
README.md
.dockerignore
Dockerfile
node_modules
vendor
storage/app/*
storage/framework/cache/*
storage/framework/sessions/*
storage/framework/views/*
storage/logs/*
bootstrap/cache/*
EOF
```

Langkah 7: Script untuk Menjalankan Container

Buat script untuk memudahkan menjalankan dan mengelola container:

```
#!/bin/bash
# run.sh

# Function untuk menjalankan container
start_containers() {
    echo "🚀 Starting Laravel Docker containers..."
```

```
    # Build dan jalankan container
    docker-compose up -d --build

    echo "⏳ Waiting for containers to be ready..."
    sleep 10

    # Jalankan migrasi database
    echo "📊 Running database migrations..."
    docker-compose exec laravel php artisan migrate --force

    echo "✅ Containers are ready!"
    echo "🌐 Application is available at: http://localhost"
    echo "🗄️  Database is available at: localhost:5432"
}

# Function untuk menghentikan container
stop_containers() {
    echo "🛑 Stopping containers..."
    docker-compose down
}

# Function untuk melihat logs
show_logs() {
    docker-compose logs -f
}

# Function untuk masuk ke container Laravel
enter_laravel() {
    docker-compose exec laravel bash
}

# Function untuk menjalankan Artisan command
artisan() {
    docker-compose exec laravel php artisan "$@"
}

# Function untuk menjalankan Composer
composer_cmd() {
    docker-compose exec laravel composer "$@"
}

# Menu
case "$1" in
    start)
        start_containers
        ;;
    stop)
        stop_containers
        ;;
    logs)
        show_logs
        ;;
    shell)
```

```
        enter_laravel
        ;;
    artisan)
        shift
        artisan "$@"
        ;;
    composer)
        shift
        composer_cmd "$@"
        ;;
    *)
        echo "Usage: $0 {start|stop|logs|shell|artisan|composer}"
        echo ""
        echo "Commands:"
        echo "  start     - Build and start containers"
        echo "  stop      - Stop containers"
        echo "  logs      - Show container logs"
        echo "  shell     - Enter Laravel container"
        echo "  artisan   - Run Artisan commands"
        echo "  composer  - Run Composer commands"
        echo ""
        echo "Examples:"
        echo "  $0 start"
        echo "  $0 artisan migrate"
        echo "  $0 composer install"
        exit 1
        ;;
esac
```

Langkah 8: Menjalankan Setup

Sekarang mari kita jalankan setup lengkapnya:

```
# Pastikan Anda berada di direktori project Laravel
cd laravel-docker-app

# Buat direktori untuk NGINX
mkdir -p docker/nginx

# Copy konfigurasi NGINX (pastikan file default.conf sudah dibuat)
# File ini harus berisi konfigurasi NGINX yang sudah dibuat sebelumnya

# Berikan permission untuk script
chmod +x run.sh

# Jalankan container
./run.sh start

# Atau jalankan manual:
# docker-compose up -d --build
```

```
# Cek status container
docker-compose ps

# Jika ada masalah, cek logs:
# docker-compose logs

# Test aplikasi
curl http://localhost

# Untuk masuk ke container Laravel:
# docker-compose exec laravel bash

# Untuk menjalankan migrasi:
# docker-compose exec laravel php artisan migrate

# Untuk menghentikan:
# docker-compose down
```

Rangkuman Setup Lengkap

Struktur direktori akhir Anda akan terlihat seperti ini:

```
laravel-docker-app/
├── app/
├── bootstrap/
├── config/
├── database/
├── docker/
│   └── nginx/
│       └── default.conf
├── public/
├── resources/
├── routes/
├── storage/
├── vendor/
├── .env
├── .dockerignore
├── .gitignore
├── composer.json
├── docker-compose.yml
├── Dockerfile
└── run.sh
```

Untuk menjalankan:

1. `./run.sh start` - Membangun dan menjalankan semua container
2. Akses aplikasi di `http://localhost`
3. Database PostgreSQL tersedia di `localhost:5432`

Perintah berguna:

- `./run.sh stop` - Menghentikan container
- `./run.sh logs` - Melihat logs
- `./run.sh shell` - Masuk ke container Laravel
- `./run.sh artisan migrate` - Menjalankan migrasi
- `./run.sh composer install` - Install dependencies

Apakah ada yang perlu dijelaskan lebih detail atau ada masalah dalam implementasinya?