Complete Linux Server Setup & Deployment Guide

1. Server Preparation

Initial Server Setup (Ubuntu 20.04/22.04)

bash

```bash
# Update system
sudo apt update && sudo apt upgrade -y

# Install essential packages
sudo apt install -y curl wget git unzip software-properties-common

# Create a non-root user (if not exists)
sudo adduser deploy
sudo usermod -aG sudo deploy

# Setup SSH key authentication (recommended)
sudo -u deploy mkdir -p /home/deploy/.ssh
sudo -u deploy chmod 700 /home/deploy/.ssh
# Copy your public key to /home/deploy/.ssh/authorized_keys
```

Install Docker & Docker Compose

bash

```bash
# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Add user to docker group
sudo usermod -aG docker $USER
sudo usermod -aG docker deploy

# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.2/docker-compose-$(unam
sudo chmod +x /usr/local/bin/docker-compose

# Verify installation
docker --version
docker-compose --version
```

2. Firewall Configuration

bash

```bash
# Enable UFW firewall
sudo ufw enable

# Allow SSH
sudo ufw allow ssh

# Allow HTTP and HTTPS
sudo ufw allow 80
sudo ufw allow 443

# Check status
sudo ufw status
```

3. SSL Certificate Setup (Let's Encrypt)

bash

```bash
# Install Certbot
sudo apt install certbot python3-certbot-nginx -y

# Get SSL certificate (replace with your domain)
sudo certbot certonly --standalone -d yourdomain.com -d www.yourdomain.com

# Setup auto-renewal
sudo crontab -e
# Add this line:
# 0 12 * * * /usr/bin/certbot renew --quiet
```

4. Application Deployment

Clone and Setup Project

bash

```bash
# Switch to deploy user
sudo su - deploy

# Create application directory
sudo mkdir -p /var/www/myapp
sudo chown -R deploy:deploy /var/www/myapp
cd /var/www/myapp

# Clone your repository
git clone https://github.com/yourusername/yourproject.git .

# Make deploy script executable
chmod +x deploy.sh
```

Create Production Files

Create the necessary production configuration files:

1. **docker-compose.prod.yml** (use the one provided above)
2. **Dockerfile.prod** for both backend and frontend
3. **nginx.prod.conf** with SSL configuration

Run Deployment

bash

```bash
# Development deployment
./deploy.sh development

# Production deployment
./deploy.sh production yourdomain.com
```

5. Production Dockerfiles

Backend Dockerfile.prod

dockerfile

```
FROM php:8.2-apache

WORKDIR /var/www/html

# Install system dependencies
RUN apt-get update && apt-get install -y \
    git curl libpng-dev libonig-dev libxml2-dev zip unzip \
    libzip-dev libfreetype6-dev libjpeg62-turbo-dev \
    && docker-php-ext-configure gd --with-freetype --with-jpeg \
    && docker-php-ext-install pdo_mysql mbstring exif pcntl bcmath gd zip \
    && pecl install redis opcache \
    && docker-php-ext-enable redis opcache \
    && apt-get clean && rm -rf /var/lib/apt/lists/*

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

# Enable Apache modules
RUN a2enmod rewrite ssl headers

# PHP configuration for production
RUN echo "opcache.enable=1" >> /usr/local/etc/php/conf.d/opcache.ini \
    && echo "opcache.memory_consumption=128" >> /usr/local/etc/php/conf.d/opcache.ini \
    && echo "opcache.max_accelerated_files=4000" >> /usr/local/etc/php/conf.d/opcache.ini \
    && echo "opcache.revalidate_freq=2" >> /usr/local/etc/php/conf.d/opcache.ini

# Set Apache document root
ENV APACHE_DOCUMENT_ROOT="/var/www/html/public"
RUN sed -ri -e 's!/var/www/html!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/sites-available/*.conf
RUN sed -ri -e 's!/var/www/!${APACHE_DOCUMENT_ROOT}!g' /etc/apache2/apache2.conf /etc/apache2/c

# Copy application
COPY . /var/www/html

# Install dependencies and optimize
RUN composer install --no-dev --optimize-autoloader --no-interaction
RUN php artisan config:cache
RUN php artisan route:cache
RUN php artisan view:cache

# Set permissions
RUN chown -R www-data:www-data /var/www/html \
    && chmod -R 755 /var/www/html/storage \
    && chmod -R 755 /var/www/html/bootstrap/cache

# Health check
```

```
HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
    CMD curl -f http://localhost/api/health || exit 1

EXPOSE 80
CMD ["apache2-foreground"]
```

Frontend Dockerfile.prod

dockerfile

```
FROM node:18-alpine as build

WORKDIR /app

# Copy package files
COPY package*.json ./

# Install dependencies
RUN npm ci --only=production

# Copy source code
COPY . .

# Build application
ARG VITE_API_URL
ENV VITE_API_URL=$VITE_API_URL
RUN npm run build

# Production stage
FROM nginx:alpine

# Copy built application
COPY --from=build /app/dist /usr/share/nginx/html

# Copy nginx configuration
COPY nginx.conf /etc/nginx/conf.d/default.conf

# Add health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
    CMD wget --quiet --tries=1 --spider http://localhost || exit 1

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

6. Monitoring & Maintenance

Setup Log Rotation

bash

```
# Create logrotate configuration
sudo tee /etc/logrotate.d/myapp << EOF
/var/www/myapp/logs/nginx/*.log {
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 644 www-data www-data
    postrotate
        docker-compose -f /var/www/myapp/docker-compose.prod.yml exec nginx nginx -s reload
    endscript
}

/var/www/myapp/logs/laravel/*.log {
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 644 www-data www-data
}
EOF
```

Setup System Monitoring

bash

```bash
# Install htop for system monitoring
sudo apt install htop

# Install Docker system monitoring
docker run -d \
  --name=netdata \
  -p 19999:19999 \
  -v netdataconfig:/etc/netdata \
  -v netdatalib:/var/lib/netdata \
  -v netdatacache:/var/cache/netdata \
  -v /etc/passwd:/host/etc/passwd:ro \
  -v /etc/group:/host/etc/group:ro \
  -v /proc:/host/proc:ro \
  -v /sys:/host/sys:ro \
  -v /etc/os-release:/host/etc/os-release:ro \
  --restart unless-stopped \
  --cap-add SYS_PTRACE \
  --security-opt apparmor=unconfined \
  netdata/netdata
```

Backup Script

bash

```bash
#!/bin/bash

# Create backup script
sudo tee /var/www/myapp/backup.sh << 'EOF'
#!/bin/bash

BACKUP_DIR="/var/backups/myapp/$(date +%Y%m%d_%H%M%S)"
mkdir -p "$BACKUP_DIR"

# Backup database
docker-compose -f /var/www/myapp/docker-compose.prod.yml exec -T mysql \
    mysqldump -u root -p"$DB_ROOT_PASSWORD" --all-databases > "$BACKUP_DIR/database.sql"

# Backup storage
cp -r /var/www/myapp/backend/storage "$BACKUP_DIR/"

# Compress backup
tar -czf "$BACKUP_DIR.tar.gz" -C "$(dirname "$BACKUP_DIR")" "$(basename "$BACKUP_DIR")"
rm -rf "$BACKUP_DIR"

# Keep only last 7 days of backups
find /var/backups/myapp -name "*.tar.gz" -mtime +7 -delete

echo "Backup completed: $BACKUP_DIR.tar.gz"
EOF

chmod +x /var/www/myapp/backup.sh

# Setup daily backup cron job
echo "0 2 * * * /var/www/myapp/backup.sh" | sudo crontab -
```

7. Useful Commands

bash

```bash
# View application logs
docker-compose -f docker-compose.prod.yml logs -f

# Update application
git pull origin main
./deploy.sh production yourdomain.com

# Scale services
docker-compose -f docker-compose.prod.yml up -d --scale queue=3

# Monitor resources
docker stats

# Backup database manually
./backup.sh

# Laravel commands
docker-compose -f docker-compose.prod.yml exec laravel php artisan migrate
docker-compose -f docker-compose.prod.yml exec laravel php artisan cache:clear

# Restart specific service
docker-compose -f docker-compose.prod.yml restart nginx
```