Production Deployment Best Practices

📁 Files to NEVER Upload to Production

❌ Never Deploy These:

- `node_modules/` - Too large, not needed
- `.env` - Contains secrets (create manually on server)
- `.git/` - Git repository files
- `composer.json` / `composer.lock` - Development files
- `package.json` / `package-lock.json` - Development files
- `webpack.mix.js` / `vite.config.js` - Build configuration
- `README.md` - Documentation
- `tests/` - Testing files
- `.vscode/` / `.idea/` - IDE settings

✅ What SHOULD Be Deployed

Deploy These:

- **PHP files** - `.php`, `.html`
- **Built assets** - `public/css/`, `public/js/`, `public/dist/`
- **Images/Media** - `images/`, `uploads/`, `assets/`
- **Static files** - `.htaccess`, `robots.txt`, `favicon.ico`
- **Compiled vendor files** (if using Composer and pre-built)

🔧 Setup Guide

1. Frontend Assets (CSS/JS)

If you use npm (Webpack, Vite, etc.):

In your `package.json`, make sure you have a build script:

```
json
```

```
{
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "prod": "webpack --mode production"
  }
}
```

The workflow will:

1. Install `node_modules` (on GitHub Actions server)
2. Run `npm run build` (compiles CSS/JS)
3. Upload ONLY the compiled files (e.g., `dist/`, `public/build/`)
4. `node_modules` stays on GitHub Actions, never uploaded

2. Environment Variables (.env)

For Production:

Create `.env` manually on cPanel:

bash

```bash
# Login to cPanel → File Manager → Your project folder
# Create new file: .env

# Add your production values:
DB_HOST=localhost
DB_NAME=production_database
DB_USER=production_user
DB_PASS=SecurePassword123!

# API Keys
STRIPE_KEY=pk_live_xxxxx
MAIL_HOST=smtp.gmail.com
MAIL_USERNAME=your@email.com

# Environment
APP_ENV=production
APP_DEBUG=false
```

Security Tips:

- ✅ Use different values than development
- ✅ Set `APP_DEBUG=false` in production
- ✅ Use strong database passwords
- ✅ Never commit `.env` to GitHub

3. Database

Options:

Option A: Use cPanel phpMyAdmin

1. Export your local database
2. Import to cPanel MySQL database
3. Update `.env` with production DB credentials

Option B: Use Migrations (if applicable)

```bash
# SSH into cPanel
php migrate.php # or your migration script
```

4. File Permissions

After deployment, set proper permissions via cPanel:

```bash
# Folders: 755
chmod 755 uploads/
chmod 755 cache/

# Files: 644
chmod 644 index.php
chmod 644 config.php

# Writable folders: 775 or 777
chmod 775 uploads/
chmod 775 logs/
```

5. .htaccess Configuration

Make sure your `.htaccess` is properly configured:

```apache
# Example .htaccess
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php/$1 [L]

# Security
Options -Indexes
```

🚀 Deployment Workflow

Your Current Process:

1. ✅ **Develop locally** → Make changes
2. ✅ **Commit to GitHub** → `git push origin main`
3. ✅ **GitHub Actions runs** → Builds assets automatically
4. ✅ **Auto-deploys** → Uploads to cPanel via FTP
5. ⚠️ **Manual steps** → Create `.env` (first time only)

First Deployment Steps:

1. Push code to GitHub
2. Wait for GitHub Actions to complete
3. Go to cPanel and create `.env` file
4. Import database (if needed)
5. Set file permissions
6. Test your site!

Subsequent Deployments:

1. Just push to GitHub
2. Everything else is automatic! 🎉

📝 Common Scenarios

Scenario 1: Plain PHP + CSS/JS

- **Deploy**: All `.php`, `.css`, `.js`, `.html` files
- **Build**: Nothing (files are already ready)
- **Exclude**: `.git`, `.env`, development files

Scenario 2: PHP + Webpack/Vite

- **Deploy**: PHP files + compiled `dist/` or `public/build/`
- **Build**: `npm run build` creates production assets
- **Exclude**: `node_modules`, source files

Scenario 3: PHP + Composer

- **Deploy**: PHP files + `vendor/` folder (after composer install)
- **Build**: `composer install --no-dev` on GitHub Actions
- **Exclude**: `composer.json`, `composer.lock`

🔒 Security Checklist

□ `.env` file created manually on server
□ `APP_DEBUG=false` in production
□ Database credentials are production-specific
□ File permissions set correctly (755/644)
□ `.git` folder excluded from deployment
□ Error reporting disabled in production
□ HTTPS enabled (SSL certificate)
□ Sensitive folders protected (uploads, admin)

🐛 Troubleshooting

"My site shows errors after deployment"

- Check `.env` file exists and has correct values
- Verify database connection settings

- Check file permissions

"CSS/JS not loading"

- Verify asset paths in your HTML
- Check if `npm run build` ran successfully
- Clear browser cache

"Database connection failed"

- Confirm `.env` database credentials
- Check if database exists in cPanel
- Verify DB user has proper privileges

"Page not found / 404 errors"

- Check `.htaccess` file is uploaded
- Verify mod_rewrite is enabled
- Confirm file paths are correct