# 🚀 Django Production Deployment (Step-by-Step)
### Docker + PostgreSQL + GitHub Actions (CI/CD) + Linode + Nginx + Gunicorn + Custom Domain + SSL

This repository demonstrates how to deploy a **Django application** from local development to **production** using:
- Django
- Docker & Docker Compose
- PostgreSQL
- GitHub Actions (CI/CD)
- Linode VPS
- Nginx
- Gunicorn
- Custom Domain
- SSL (Let's Encrypt)

You will go step-by-step from:
**Local → Docker → GitHub → Linode → Domain → HTTPS**

---

## Prerequisites
Install the following on your system:
- Git
- Python 3.10+
- pip
- Docker Desktop
- VS Code (recommended)

=========================================================

**Clone, Install Required Package and Run Locally**

=========================================================

## 📦 Step 1 — Clone the Project
```sh
git clone https://github.com/dev-rathankumar/django_clickmart
cd django_clickmart
```
## Step 2 - Remove Git history
```sh
**ls -Force** - list down dir
window - **cmd /c "rd /s /q .git"**
linux - rm -rf .gituy
```
"rm -rf .git" - This wipes your commit history & remote. Now it is just files in your local computer, not a repo.

---

## Create your own GitHub repository
Go to GitHub → Click New Repository → Name: django-clickmart
## Re-initialize Git
```sh
git init
git add .
git commit -m "Initial project setup"
git branch -M main
git remote add origin https://github.com/Manoj-Bhandarkar/django_auto_deploy.git
git push -u origin main
```
Now you have the full source code in your own repo.

---

## Run Django Locally (Without Docker)
**Create virtual environment**
```sh
cd backend-drf
python -m venv env
source env/bin/activate     # Mac / Linux
# OR
env\Scripts\activate        # Windows
```

**Install dependencies**
```sh
pip install -r requirements.txt
python.exe -m pip install --upgrade pip
```
**Create ```.env``` file** -- gitignore hidden file - no go to repository
- secreate information like db info passwordemail password.
- create new file .env in backend-drf folder

**.env**
```sh
DEBUG=True
SECRET_KEY=8k(cww2yxuz1lx)he7u=0kw#)mhi0@6k7qx!-ul)s31)=!5j
# django secret key generator use any website – https://djeccrety.in

**# Database Settings - install postgres db and pgadmin client**
**Open pgadmin > create database > name: clickmart_local_db > save**
# Postgress Configuration
DB_NAME= clickmart_local_db
DB_USER= postgres
DB_PASSWORD=root
DB_HOST=localhost
DB_PORT=5432

# Email Configuration
EMAIL_HOST_USER=developer.manojbhandarkar@gmail.com
EMAIL_HOST_PASSWORD=<PASSWORD> # USE APP PASSWORD IF YOU ARE USING GMAIL
```

---

After that Create database tables and run the Django server
```sh
python manage.py migrate
python manage.py runserver
```

---

Create ```.env``` file inside /frontend/ directory and write:
Frontend > .env create file

-------

**.env**

-------

```sh
VITE_SERVER_BASE_URL=http://127.0.0.1:8000/api/v1
```
And run the frontend – React
>>> frontend >
```sh
npm install
npm run dev
```
Go to http://localhost:5173/
Optional: You can now create superuser and add some products.

>>> py manage.py createsuperuser
**Check working**
Go to http://localhost:5173/
Go to http://127.0.0.1/admin

Settings.py
STATIC_URL = 'static/'
STATIC_ROOT = BASE_DIR /'staticfiles'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]

==========================================================

*Dockerizing the Project*

==========================================================

---

## *9 - Dockerfile for Backend*

---

*## Install and verify Docker and Docker Compose*

```sh
docker --version
docker compose version
```

*## Create Dockerfile for backend*
*Create a new file "Dockerfile" inside /backend-drf/ folder*
*>>> **Backend**-drf > **Dockerfile** - no extension*

---

*Backend Docker Container*

---

~~~sh
# Purpose: A Dockerfile is a step-by-step instruction file that tells Docker how to build and run our application.
FROM python:3.10-slim

ENV PYTHONUNBUFFERED=1        #debugging        # it shows the log in terminal
ENV PYTHONDONTWRITEBYTECODE=1           # stop creating .pyc file

WORKDIR /app                            # root directory inside docker

# update the linux software instances
RUN apt-get update && apt-get install -y --no-install-recommends \
   gcc \
   libpq-dev \
   && rm -rf /var/lib/apt/lists/*       # remove cache

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000

CMD ["gunicorn", "clickmart_main.wsgi:application", "--bind", "0.0.0.0:8000", "--workers", "3" , "--timeout", "180"]
```sh

---

*The instruction breaks down into four main parts:*
***1. apt-get update***
*Purpose: Updates the local list of available packages from the software repositories.*
*Why here: Base Docker images are often stripped of these lists to save space, so you must run this before you can install anything.*
***2. apt-get install -y --no-install-recommends gcc libpq-dev***
*-y: Automatically answers "yes" to all prompts, allowing the build to proceed without user interaction.*
*--no-install-recommends: Prevents apt from installing "recommended" but non-essential packages. This can reduce the final image size by up to 60%.*
*Packages: Installs gcc (a C compiler) and libpq-dev (libraries for PostgreSQL), which are commonly needed to compile Python or Node.js database drivers.*
***3. && (Command Chaining)***
*Purpose: This ensures that the next command only runs if the previous one succeeded.*
*Crucial Tip: Chaining these into one RUN instruction creates a single image layer. If you ran rm in a separate RUN line later, the files would still exist in the previous layer, and your image wouldn't actually get smaller.*
***4. rm -rf /var/lib/apt/lists/****
*Purpose: Deletes the package index files downloaded by apt-get update.*
*Result: Since these files are no longer needed after the installation is complete, removing them significantly reduces the size of the final Docker image.*
-----
***COPY requirements.txt .***
*Action: Copies just your dependency list from your computer into the container's current working directory.*
*The "Why": Docker caches layers. By copying only the requirements.txt file first (before the rest of your code), Docker will reuse the "install" layer in future builds as long as your dependencies haven't changed. If you copied your entire project first, any small change to your code would force Docker to reinstall every single Python library from scratch.*
-----
***RUN pip install --no-cache-dir -r requirements.txt***
***-r requirements.txt**: Tells pip to install everything listed in that file.*
***--no-cache-dir**: Prevents pip from saving the downloaded .whl or .tar.gz files inside the image.*
*The "Why": Normally, pip keeps a copy of downloaded packages to speed up future installs. In a Docker image, you only need the packages installed, not the installer files themselves. Using this flag can shrink your final image size by hundreds of megabytes.*
-------
*# **gunicorn** = production server, clickmart_main.wsgi:application = Django entry point, --bind 0.0.0.0:8000 = external traffic. Reminaing: tuning options*
*# A worker is just one instance of your Django app running inside Gunicorn.*
------
*The **CMD** instruction defines the default executable for your container. This specific command launches Gunicorn, a production-grade Python WSGI HTTP Server, to serve your Django or Flask application.*
*Here is the breakdown of each part:*
*"**gunicorn**": The name of the executable being run.*
*"**clickmart_main.wsgi:application**": This tells Gunicorn where to find your app. It looks for a variable named application inside the file clickmart_main/wsgi.py.*
***"--bind", "0.0.0.0:8000":***
***0.0.0.0** tells Gunicorn to listen on all available network interfaces inside the container, not just localhost.*
***:8000** sets the port the server will run on.*
***"--workers", "3":                # instances of django***
*Tells Gunicorn to spawn 3 independent worker processes to handle incoming requests.*
*Pro Tip: The general recommendation for the number of workers is (2 * number of CPU cores) + 1.*
***"--timeout", "180":***
*Increases the worker timeout to 180 seconds (the default is only 30 seconds).*
*If a request takes longer than this, Gunicorn will kill and restart the worker. This is often increased for apps that handle long-running tasks like file uploads or complex database queries.*
***Why use this instead of python manage.py runserver?***
*The built-in Django development server is only for testing; it is not secure or efficient. Gunicorn is designed for production environments because it can handle multiple concurrent users and automatically restarts crashed workers.*

---

## *10 - Dockerfile for Frontend*

---

*## Create Dockerfile for frontend*
*Create a new file "Dockerfile" inside /frontend/ folder*
*>>> frontend > **Dockerfile***

---

*Frontend Docker Container*

---

```sh
# Stage 1: Build
FROM node:18 AS build

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

# Build arguments for environment variables             # frontend .env - url
ARG VITE_SERVER_BASE_URL                                 # contain backend api  url

# This line passes an environment variable into the Docker container so the React app knows the backend API URL.
ENV VITE_SERVER_BASE_URL=$VITE_SERVER_BASE_URL

RUN npm run build
---------------------------
# Stage 2: Nginx, alpine means the lighter version of Nginx
FROM nginx:alpine

# Copy build output to Nginx html directory
COPY --from=build /app/dist /usr/share/nginx/html

EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]              # run fore ground not background
```sh
~~~

---

*11 – docker-compose.yml File Dockerizing the Project*

---

*## On the **root** directory, create a file "docker-compose.yml"*

```sh
services:
 db:                                        *** DB service
   image: postgres:16-alpine
   env_file:
     - ./backend-drf/.env.production
                                # file name create in backend-drf folder
   volumes:
     - postgres_data:/var/lib/postgresql/data

 backend:                                   *** Backend service
   build: ./backend-drf
   ports:
     - "8000:8000"
   env_file:
     - ./backend-drf/.env.docker
                                # file name create in backend-drf folder
   depends_on:
     - db
   volumes:
     - ./backend-drf/static:/app/static
     - ./backend-drf/media:/app/media
   command: >
     sh -c "python manage.py collectstatic --noinput &&
          python manage.py migrate &&
          python manage.py runserver 0.0.0.0:8000"

 frontend:                                  *** Frontend service
   build:
     context: ./frontend
     args:
       VITE_SERVER_BASE_URL: "http://127.0.0.1:8000/api/v1"
   ports:
     - "5173:80"
   depends_on:
     - backend

 volumes:
   postgres_data:
# This creates a named Docker volume to permanently store
PostgreSQL data.
# Without this:
  # Database data is stored inside the container
  # If container is deleted → data is lost
# With this:
  # Data is stored in a Docker-managed volume
  # Data persists even if container stops or restarts
```

---

*>>> backend-drf > .env.production*

---

```sh
POSTGRES_DB=clickmart_local_db
POSTGRES_USER=postgres
POSTGRES_PASSWORD=root
```

*Make sure to create a copy of ```.env``` and name it as ```.env.docker```*

---

*>>> Backend-drf > .env.docker*

---

```sh
DEBUG=True
SECRET_KEY="8k(cww2yxuz1lx)he7u=0kw#)mhi0@6k7qx!-ul)s31)=!5j"
# Database Settings
DB_NAME=clickmart_local_db
DB_USER=postgres
DB_PASSWORD=root
DB_HOST=db                    # db comes from docker compose db service
DB_PORT=5432
# Email Configuration
EMAIL_HOST_USER=developer.manojbhandarkar@gmail.com
EMAIL_HOST_PASSWORD=dkfgequakgacohqh
# USE APP PASSWORD IF YOU ARE USING GMAIL
```

---

*Run this command to Dockerize your project:*

```sh
docker compose up --build
```

*Your project is now Dockerized ✅*
*If any any error occured and then resolved please **restart***

```sh
docker compose down -v
```

*See the docker container **health**:*

```sh
docker compose ps
```

---

*12 - Creating Superuser Inside Docker Container*

---

*You can try creating **superuser inside Docker** container.*
*- Open **new terminal***

```sh
docker compose exec backend python manage.py createsuperuser
```

---

*5 - Linode Server Setup*

---

*13 - Creating a Linode Account*

---

*## Create Linode Server & SSH Key*
*☞ [Create a Linode account](https://rathank.com/linode/)*

---

*14 - Creating a Linode Server with SSH Key*

---

*Steps*
*- After creating account click on create button on right side.*
*- click on LINODE.*
*OS:*
*- region : mumbai  (ap-west)*
*- choose an OS : Ubuntu 24.0.4 **LTS***
*- Linode Plans :*
        *Shared CPU - Nanode 1GB    $5      100users*
*- Details : linode label – clickmart_auto_deploy*
*- Security :*
        *root password – linux server password*
        *- SSH Keys - Add an SSH keys*
                *- ssh public key*

---

*##### Create SSH Key In windows*

---

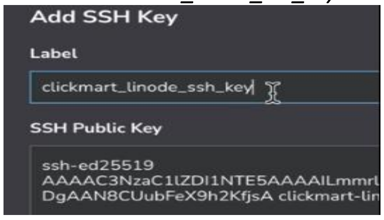
```
 (env) PS C:\Users\Manoj> mkdir ~/.ssh
C:\Users\Manoj> cd ~/.ssh
C:\Users\Manoj>.ssh> ssh-keygen -t ed25519 -C "clickmart-linode"
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\Manoj/.ssh/id_ed25519):
linode_clickmart
Enter passphrase (empty for no passphrase): NA
Enter same passphrase again: NA
Your identification has been saved in linode_clickmart
Your public key has been saved in linode_clickmart.pub
The key fingerprint is:
SHA256:ydmdThyQ1FiKZHW4aE717FZ1JihJMhz8izLCCYfUtvU clickmart-linode
----
>>> C:\Users\Manoj\.ssh> ls
```

```
Mode            Time        Length Name
----            -------------        ------ ----
-a----       09-02-2026          411 linode_clickmart              - set lock
-a----       09-02-2026           99 linode_clickmart.pub          - password
```

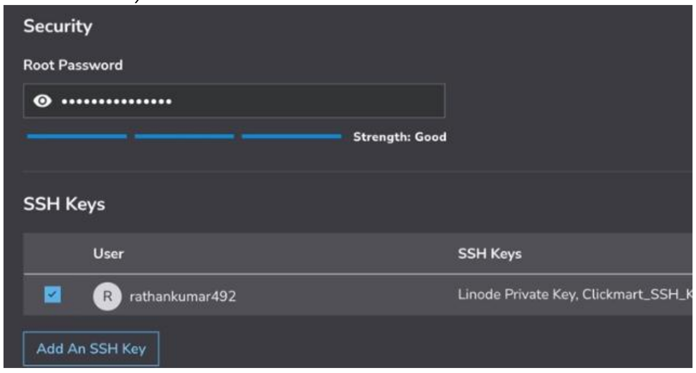
*Copy the public key and add it to Linode UI:*

```ssh
(env) PS C:\Users\Manoj\.ssh> cat ~/.ssh/linode_clickmart.pub
" ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIBifzYA8M2HXgUtilsCyFPljTAXsvEc33zuE5gJ/LLbe
clickmart-linode "
```

*- copy as it is and paste into linode secury page - ssh public key*
*- **label** it **"clickmart_linode_ssh_key"***


Add SSH Key
Label
clickmart_linode_ssh_key
SSH Public Key
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAILmmrl
DgAAN8CUubFeX9h2KfjsA clickmart-lin

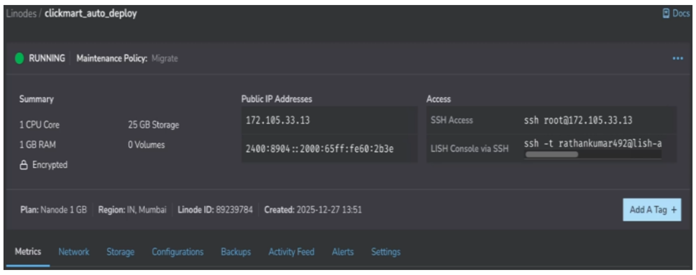
*- add key*
*- select ssh key*


Security
Root Password
•••••••••••••••
Strength: Good
SSH Keys

| User | SSH Keys |
|---|---|
| ✓ rathankumar492 | Linode Private Key, Clickmart_SSH_K |

Add An SSH Key

*- Create linode*


Linodes / clickmart_auto_deploy
RUNNING | Maintenance Policy: Migrate

Summary
1 CPU Core    25 GB Storage
1 GB RAM    0 Volumes
Encrypted

Public IP Addresses
172.105.33.13
2400:8904::2000:65ff:fe60:2b3e

Access
SSH Access    ssh root@172.105.33.13
LISH Console via SSH    ssh -t rathankumar492@lish-a

Plan: Nanode 1 GB | Region: IN, Mumbai | Linode ID: 89239784 | Created: 2025-12-27 13:51

---

*## SSH into Linode (Passwordless)          #new terminal*

```sh
>>> django_clickmart> ssh root@172.105.33.13
- Now you are in linode server
>>> root@localhost:~#
```

---

*15 - Installing Docker and Git on Production Server*

---

***Update the server:***

```sh
>>> root@localhost:~#  apt update && apt upgrade -y
package config
               keep the local ver locally installed          - Yes
```

***## Install Required Software***

---

***Install Docker:***

```sh
>>> root@localhost:~#  curl -fsSL https://get.docker.com | sh
docker --version
```

***Install Docker Compose:***

```sh
>>> root@localhost:~#  apt install docker-compose-plugin -y
```

***Install Git:***

```sh
>>> root@localhost:~#  apt install git -y
git --version
```

*✅Docker, Docker Compose, and Git installed successfully.*

---

*16 - Clone Project into Server*

---

*## Clone Project into /opt*
*Reconnect to SSH (if disconnected):*

```sh
>>> root@localhost:~#  cd /
>>> root@localhost:~#  ls
>>> root@localhost:~#  cd /opt
>>> root@localhost:~#  mkdir clickmart
>>> root@localhost:~#  cd clickmart
>>> root@localhost:~#  git clone https://github.com/Manoj-Bhandarkar/django_auto_deploy.git
```

*Repo is now cloned inside /opt/clickmart*

---

*17 - Update Frontend Environment Variables*

---

*## Update Frontend Environment Variable*
***In local docker-compose.yml change ip of linode server:***

```
 frontend:                                  # replace with linode server ip
       VITE_SERVER_BASE_URL: "http://127.0.0.1:8000/api/v1"
```

```sh
VITE_SERVER_BASE_URL="http://172.105.33.13:8000/api/v1"
```

*Push changes:*

```sh
>>>C:/> django_clickmart > git push origin main
```

---

*18 - Create Environment Files on Server*

---

***## Create Environment Files on Linode Server***

```sh
>> root@localhost:/opt/clickmart/backend-drf#  nano .env.production
POSTGRES_DB=clickmart_deploy_live_db
POSTGRES_USER=postgres
POSTGRES_PASSWORD=StrongPassword@2026
```

*ctrl+x*
*yes – save*

```
>> root@localhost:/opt/clickmart/backend-drf#  nano .env.docker
DEBUG=True
SECRET_KEY="8k(cww2yxuz1lx)he7u=0kw#)mhi0@6k7qx!-ul)s31)=!5j"

# Database Settings
DB_NAME=clickmart_deploy_live_db
DB_USER=postgres
DB_PASSWORD= StrongPassword@2026
DB_HOST=db
DB_PORT=5432

# Email Configuration
EMAIL_HOST_USER=developer.manojbhandarkar@gmail.com
EMAIL_HOST_PASSWORD=dkfgequakgacohqh
```

*ctrl+x*
*yes – save*

---------------------------------------------------------------------------------------------

*19 - Open Firewall Ports on Server*

---------------------------------------------------------------------------------------------

***- test project on port and then remove***

*Add required environment variables inside it.*

*## Open Firewall Ports on Linode*

*⚠️ If ports are not opened, the app will run but won't be accessible.*

***Required Ports (Initial Setup)***

```sh
SSH: 22
Django Backend: 8000
React Frontend: 5173
```

```sh
Inbound Rules:
Allow TCP 22
Allow TCP 8000
Allow TCP 5173
```

***Setup firewall***

*https://cloud.linode.com/firewalls*

***click on create firewall***

*label : clickmart_live_firewall*
*inbound policy – drop*
*outbound policy – accept*
*linodes – clickmart_auto_deploy*
*create firewall*

-----------

***SSH: 22***

----------

*open same firewall - **add inbound rule***

*Preset – SSH*
*Lebel – accept-inbound-SSH*
*Protocol – TCP*
*Ports – SSH(22)*
*Sources – All IPv4, all IPv6*
*Action – accept*
*Click on add rule*

------------------------------

***Django Backend: 8000***

------------------------------

*- **add inbound rule***

*Preset –*
*Lebel – backend*
*Protocol – TCP*
*Ports - Custom custom range - 8000*
*Sources – All IPv4, all IPv6*
*Action – accept*
*Click on add rule*

-----------------------------

***React Frontend: 5173***

----------------------------

*- **add inbound rule***

*Preset –*
*Lebel – frontend*
*Protocol – TCP*
*Prots – Custom custom range - 5173*
*Sources – All IPv4, all IPv6*
*Action – accept*
*Click on add rule*

*>>> Save Changes*

---------------------------------------------------------------------------------------------

*20 - Build the Containers and Run the Project*

---------------------------------------------------------------------------------------------

*## Build & Run Docker Containers*

*>>> connect server*

*>>> ssh root@172.105.33.13*

```sh
>> root@localhost:~# cd opt
>> root@localhost:/opt# cd clickmart
>> root@localhost:/opt/clickmart# git pull
   - Now pull docker-compose.yml file in server
>> root@localhost:/opt/clickmart# docker compose up --build –d
>> root@localhost:/opt/clickmart# docker compose ps
```

*Test in browser:*

*Backend: http://<LINODE_IP> http://172.105.33.13:8000/*

*Frontend: http://<LINODE_IP> http://172.105.33.13:5173/*

*- Disaalowed host error – from backend*

***## Fix Django ALLOWED_HOSTS***

***- don't goto server project setting and add allowed host code it is bad practice***

***- go through git push/pull***

***- only env modify in linode server not other file***

--------------------------

*In **local settings.py**:*

```sh
Import os
ALLOWED_HOSTS = os.getenv("ALLOWED_HOSTS", "").split(",")

In last
CORS_ALLOWED_ORIGINS = [
   'http://localhost:5173',
   'http://172.105.33.13:5173'
]
```

---------------------------

*In **local .env.docker**: # add after secret key*

```sh
ALLOWED_HOSTS=<LINODE_IP>,localhost,127.0.0.1
```

***In linode .env.docker:***

```sh
>> root@localhost:/opt/clickmart/backend-drf# nano .env.docker
Past after secrete_key add line
ALLOWED_HOSTS=172.105.33.13,localhost,127.0.0.1
```

***In docker-compose.yml:***

```sh
VITE_SERVER_BASE_URL: "http://<LINODE_IP>/api/v1"
```

***Push to GitHub: new terminal***

```sh
(env) C://> django_clickmart> git add .
(env) git commit -m "Allowed host & environments added"
(env) git push origin main
```

*This will push the changes to GitHub.*

---------------------------------------------------------------------------------------------

*### 🎯 Goal - Whenever I push code to GitHub, my Linode server should automatically update.*

*But first...*

***### Manually pull the code from GitHub to Linode.***

*While logged-in to Linode:*

***Linode server pull the code***

```sh
>> root@localhost:/opt/clickmart# git pull
Or git pull origin main
```

*Rebuild containers:*

```sh
>> root@localhost:/opt/clickmart#  docker compose down        -v delete database
>> root@localhost:/opt/clickmart#  docker compose up --build -d
```

http://172.105.33.13:8000/admin
http://172.105.33.13:5173

>> root@localhost:/opt/clickmart#  **docker compose exec backend python manage.py createsuperuser**

==========================================================

## 6 - CICD Pipeline Setup GitHub Actions

==========================================================

### 21 - The Goal

------------------------------------------------------------------------------------------

## Rule Before Automation

❗**Never automate something you haven't done manually.**

------------------------------------------------------------------------------------------

### 22 - Setup the Automation CICD Pipeline

------------------------------------------------------------------------------------------

**## Setup CI/CD (GitHub Actions)**

In **local project**:

Create new folder **.github** inside new folder **workflows** Create a new file: **automate.yml**

>> c:/>django_clickmart>.github>workflows>automate.yml

---------------------

**automate.yml**

---------------------

```sh
name: Auto Deploy to Linode

on:
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Deploy via SSH
        uses: appleboy/ssh-action@v1.0.3
        with:
          host: ${{ secrets.LINODE_HOST }}
          username: ${{ secrets.LINODE_USER }}
          key: ${{ secrets.LINODE_SSH_KEY }}
          script: |
            cd /opt/clickmart
            git pull origin main
            docker compose up --build -d
```

------------------------------------------------------------------------------------------

**Add GitHub Secrets:**

----------------------------

**GitHub → Your Repository → Settings → Secrets and variables → Actions → New repository secret**

```
LINODE_HOST → <LINODE_IP>
LINODE_USER → root
LINODE_SSH_KEY → Private SSH Key
```

----------------------

**Private SSH Key**

---------------------

**PS C:\users\manoj\.ssh> cat linode_clickmart**

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACAYn82APDNh14FLYpbAshT5Y0wF7LxHN987hOYCfyy23gAAAJhQC+O7UAvj
uwAAAAtzc2gtZWQyNTUxOQAAACAYn82APDNh14FLYpbAshT5Y0wF7LxHN987hOYCfyy23g
AAAEBxRNSe0r9XG03PplkB80xZulFTT2PwET9Pfx+IlNo7dBifzYA8M2HXgUtilsCyFPlj
TAXsvEc33zuE5gJ/LLbeAAAAEGNsaWNrbWFydC1saW5vZGUBAgMEBQ==
-----END OPENSSH PRIVATE KEY-----

------------------------------------------------------------------------------------------

**## Push automation file:**

```sh
git add .
git commit -m "CI/CD Setup"
git push origin main
```

Check GitHub **Actions** tab.

Make a small frontend change and **confirm auto-deploy**.

✅Auto deploy successful.

----------------------------------

Now project running on

Frontend - http://172.105.33.13:5173

Backend - http://172.105.33.13:8000

- We don't want access same application with different port.
- http://172.105.33.13 run only on this whole application.

==========================================================

## 7 - Nginx Gunicorn Configuration

==========================================================

### 23 - Nginx Configuration

------------------------------------------------------------------------------------------

## Nginx Config

From local project, create file:

```sh
>> c:/>django_clickmart>nginx>default.conf
```

----------------

**default.conf**

----------------

```
server {
    listen 80;

    # Frontend (React)
    location / {
        proxy_pass http://frontend:80;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # Backend (Django)
    location /api/ {
        proxy_pass http://backend:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }

    # Django admin & static
    location /admin/ {
        proxy_pass http://backend:8000;
    }

    location /static/ {
        proxy_pass http://backend:8000;
    }

    location /media/ {
        proxy_pass http://backend:8000;
    }
}
```

------------------------------------------------------------------------------------------------

### Docker Compose Changes
- Add nginx service
- **Remove ports** from backend & frontend
- Update frontend API URL: ``` **VITE_SERVER_BASE_URL="/api/v1"** ```

-----------------------------

Docker-compose.yml

-----------------------------

```sh
services:
  db:                                          *** DB service
    image: postgres:16-alpine
    env_file:
      - ./backend-drf/.env.production
                                    # file name create in backend-drf folder
    volumes:
      - postgres_data:/var/lib/postgresql/data

  backend:                                     *** Backend service
    build: ./backend-drf
    env_file:                                  # port is remove
      - ./backend-drf/.env.docker
                                    # file name create in backend-drf folder
    depends_on:
      - db
    volumes:
      - ./backend-drf/static:/app/static
      - ./backend-drf/media:/app/media
    command: >
      sh -c "python manage.py collectstatic --noinput &&
             python manage.py migrate &&
             python manage.py runserver 0.0.0.0:8000"

  frontend:                                    *** Frontend service
    build:
      context: ./frontend
      args:
        VITE_SERVER_BASE_URL: "/api/v1"
    depends_on:                                # port is remove
      - backend

  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
    volumes:
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - frontend
      - backend
```

------------------------------------------------------------------------------------------------

Git Push changes:

```sh
>> c:/>django_clickmart> git add .
>> c:/>django_clickmart> git commit -m "Nginx Setup"
>> c:/>django_clickmart> git push origin main
```

------------------------------------------------------------------------------------------------

24 - Update Firewall

------------------------------------------------------------------------------------------------

## Update Firewall (Production) in linode setup
**Keep:**
- ```**22**``` (SSH)
- Add new 80 port
    - ```**80**``` (HTTP) - tcp

**Remove inbound rule:**
- ```8000``` (Backend)
- ```5173``` (Frontend)
- save changes

## Final Test
http://172.105.33.13/ done
http://172.105.33.13/admin disallowed host

-------------------

If you get error: Add ```backend``` to allowed host in linode server manually.

```
>> root@localhost:/opt/clickmart/backend-drf# nano .env.docker
ALLOWED_HOSTS=172.105.33.13,localhost,127.0.0.1,backend
```

----------------------

Restart docker:

```sh
>> root@localhost:/opt/clickmart/backend-drf#  docker compose down
>> root@localhost:/opt/clickmart/backend-drf#  docker compose up --build -d
```

------------------------------------------------------------------------------------------------

25 - Gunicorn Setup (Production WSGI Server)

------------------------------------------------------------------------------------------------

## Gunicorn Setup (Production WSGI Server)
### 1. Add Gunicorn Dependency
Add `**gunicorn**` inside `**requirements.txt**`:

#### Update Backend Dockerfile
No special change is required other than ensuring requirements.txt is installed. Gunicorn will be installed automatically via dependencies.

#### Update docker-compose.yml
Replace the Django run command with Gunicorn:

```
    command: >
      sh -c "python manage.py collectstatic --noinput &&
             python manage.py migrate &&
             python manage.py runserver 0.0.0.0:8000"
replace
    command: >
      sh -c "python manage.py collectstatic --noinput &&
             python manage.py migrate &&
             gunicorn clickmart_main.wsgi:application --bind 0.0.0.0:8000 --workers 3"

```

- clickmart_main.wsgi:application → **Django entry point**
- --bind 0.0.0.0:8000 → **Listen on all interfaces**
- --workers 3 → **Run 3 Python worker processes**

```

>> c:/>django_clickmart> git add .
>> c:/>django_clickmart> git commit -m "Deploy Gunicorn"
>> c:/>django_clickmart> git push origin main
```

#### Important Note
✅We did not change the application code.
✅We only changed how Python code is executed in production.

### Verify Gunicorn Is Running        or check in git action
SSH into the Linode server:

```

>> root@localhost:/opt/clickmart#  ssh root@<LINODE_IP>
>> root@localhost:/opt/clickmart#  cd /opt/clickmart
>> root@localhost:/opt/clickmart#  docker compose logs backend
- It show error static files not found
- runserver not running application now gunicorn run application
```

Now admin not run properly because static files issue – resolve after

==========================================================
## 8 - Custom Domain Implementation
==========================================================
### 26 - Purchasing a Custom Domain
---------------------------------------------------------------------------------------------

*## Purchase a Domain*
*[www.namecheap.com](www.namecheap.com) and [www.godaddy.com](www.godaddy.com)*
*Purchase a domain from any provider (GoDaddy, Namecheap, etc.).*

---------------------------------------------------------------------------------------------
### 27 - DNS Setup for Domain
---------------------------------------------------------------------------------------------

*## Connect Domain to Linode (DNS)*
***Goto – godaddy/domain/DNS – add new record***
*Add the following A records in your domain DNS:*

| Type | Name | Value | TTL |
|---|---|---|---|
| **A** | @ | `<YOUR_LINODE_IP>` | 1 hour |
| **CNAME** | www | `<YOUR_LINODE_IP>` | 1 hour |

*Wait for DNS propagation (usually a few minutes to a few hours).*

---------------------------------------------------------------------------------------------
### 28 - Add Domain to Nginx Configuration
---------------------------------------------------------------------------------------------

*### Nginx Config as Server-Managed File*
***- Certbot modifies the Nginx config directly on the server,***
***- so we must remove it from Git tracking.***

```
>> c:/>django_clickmart> git rm --cached nginx/default.conf
```

***- Removes the file from Git***
*>> c:/>django_clickmart>* ***Add to .gitignore:***

```
backend-drf/.env.docker
backend-drf/.env.production
nginx/default.conf
```

*#### Commit and Push*

```
>> c:/>django_clickmart> git add .
>> c:/>django_clickmart> git commit -m "Make nginx config server-managed"
>> c:/>django_clickmart> git push origin main
```

---------------------------------------------------------------------------------------------
***- Add domain to this file:***

```
server_name example.com www.example.com;
```

*#### SSH into Linode server*
*- now ngnix folder is not there in server*
*- Create `nginx/default.conf` file*

```
>> root@localhost:/opt/clickmart# mkdir nginx
>> root@localhost:/opt/clickmart# cd ngnix
>> root@localhost:/opt/clickmart/ngnix# nano default.conf
```

---------------------
***default.conf***
---------------------

```
server {
   listen 80;
   server_name djangoclickmart.store www. djangoclickmart.store;
   # Frontend (React)
   location / {
      proxy_pass http://frontend:80;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
   }
   # Backend (Django)
   location /api/ {
      proxy_pass http://backend:8000;
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
   }
   # Django admin & static
   location /admin/ {
      proxy_pass http://backend:8000;
   }
   location /static/ {
      proxy_pass http://backend:8000;
   }
   location /media/ {
      proxy_pass http://backend:8000;
   }
}
```

---------------------------------------------------------------------------------------------
***Verify linode***

```
>> root@localhost:/opt/clickmart/ngnix  cat default.conf
```

***Restart nginx:***

```
>> root@localhost:/opt/clickmart/ngnix  docker compose restart nginx
```

---------------------------------------------------------------------------------------------
***#### Update Django ALLOWED_HOSTS - `.env.docker`***
*Add your domain into `.env.docker`*

```
>> root@localhost:/opt/clickmart/django-drf#  nano .env.docker
Secret_key=...
ALLOWED_HOSTS=172.105.33.13,localhost,127.0.0.1,backend,djangoclickmart.store
```

***### Restart backend:***

```
>> root@localhost:/opt/clickmart/django-drf#  docker compose restart backend
```

***## Check all container running***

```
>> root@localhost:/opt/clickmart/django-drf#  docker compose ps
```

***### Test Domain (HTTP only)  - some time it takes (time 1-2 hour)***
*[http://djangoclickmart.store](http://djangoclickmart.store)*

---------------------------------------------------------------------------------------------
### 29 - Encountered Issues with Nginx
---------------------------------------------------------------------------------------------

*[http://djangoclickmart.store](http://djangoclickmart.store)/*
*local* ***settings.py***

```
CORS_ALLOWED_ORIGINS = [
            'http://localhost:5173',
            'http://172.105.33.13:5173',
            'http://djangoclickmart.store',
            'https://djangoclickmart.store',
]
```

*## git push code*

```sh
>>c:/>django_clickmart> git add .
>>c:/>django_clickmart> git commit -m "domain name allowed in cors"
>>c:/>django_clickmart> git push origin main
```

---------------------------------------------------------------------------------------------
### 30 - Fix Static Files Issues on Production
---------------------------------------------------------------------------------------------

*[http://djangoclickmart.store/admin](http://djangoclickmart.store/admin)      fix below*
*Replace static directory root in default.conf file*

```
>> root@localhost:/opt/clickmart#  nano ngnix/default.conf
            # django admin & static
            location /static/ {
                        alias /static/;
            }
```

*In Local* ***docker-compose.yml add static folder in ngnix backend***

```
Ngnix:
            Volumes:
              -  ./ngnix/default.conf:/etc/ngnix/conf.d/default.conf
              -  ./backend-drf/static:/static
```

```
>>c:/>django_clickmart> git add .
>>c:/>django_clickmart> git commit -m "static files to be loaded from ngnix"
>>c:/>django_clickmart> git push origin main

>> root@localhost:/opt/clickmart#  docker compose down
>> root@localhost:/opt/clickmart#  docker compose –build –d
```

***Now reload***
*[http://djangoclickmart.store/admin](http://djangoclickmart.store/admin)      working*

==========================================================

9 - Install SSL Certificate

==========================================================

31 - Installing SSL Includes Errors Troubleshooting

------------------------------------------------------------------------------------------

## Install SSL (Let's Encrypt)

*In the server root directory, create folders:*

```
>> root@localhost:/opt/clickmart#  mkdir -p certbot/www
>> root@localhost:/opt/clickmart#  mkdir -p certbot/conf
>> root@localhost:/opt/clickmart/certbot#  ls
conf        www
```

### Update docker-compose.yml (Nginx service)                #local-vscode

*Edit docker-compose.yml locally (**nginx** service):*

```
Ngnix:
            ports:
              - "80:80"
              - "443:443"          # for ssl cerfificate
            Volumes:
              -   ./ngnix/default.conf:/etc/ngnix/conf.d/default.conf
              -   ./backend-drf/static:/static
              -   ./certbot/www:/var/www/certbot
              -   ./certbot/conf:/etc/letsencrypt
```

#### Commit and Push

```
>> c:/>django_clickmart> git add .
>> c:/>django_clickmart> git commit -m "ssl installation started"
>> c:/>django_clickmart> git push origin main
```

### Update nginx/default.conf

*Edit `nginx/default.conf`*

```
>> root@localhost:/opt/clickmart#  nano ngnix/default.conf
```

*Add this block:*

```
server {
   listen 80;
   server_name djangoclickmart.store www.djangoclickmart.store;
   location /.well-known/acme-challenge/ {
         root /var/www/certbot;
   }
```

## Restart Nginx container:

```
>> root@localhost:/opt/clickmart#  docker compose restart nginx
>> root@localhost:/opt/clickmart#  docker compose ps
```

**Make sure the site with HTTP still works at this point:**

http://djangoclickmart.store

------------------------------------------------------------------------------------------

### Install Certbot

```
>> root@localhost:/opt/clickmart#  apt update
>> root@localhost:/opt/clickmart#  apt install certbot -y
```

------------------------------------------------------------------------------------------

### Get SSL Certificate (WEBROOT METHOD)

```
>> root@localhost:/opt/clickmart#  certbot certonly \
  --webroot \
  -w /opt/clickmart/certbot/www \
  -d djangoclickmart.store \
  -d www.djangoclickmart.store
```

------------------------------------------------------------------------------------------

### Enable HTTPS in Nginx

**Remove ngnix/default.conf file**

```
>> root@localhost:/opt/clickmart/ngnix#  rm default.conf
```

*Edit `nginx/default.conf` again:*

*Replace with FINAL CONFIG:*

```
>> root@localhost:/opt/clickmart/ngnix#  nano default.conf
```

```
server {
   listen 80;
   server_name djangoclickmart.store www.djangoclickmart.store;
   return 301 https://$host$request_uri;
}

server {
   listen 443 ssl;
   server_name djangoclickmart.store www.djangoclickmart.store;

   ssl_certificate
/etc/letsencrypt/live/djangoclickmart.store/fullchain.pem;
   ssl_certificate_key
/etc/letsencrypt/live/djangoclickmart.store/privkey.pem;

   location / {
      proxy_pass http://frontend:80;
   }

   location /api/ {
      proxy_pass http://backend:8000;
   }

   location /admin/ {
      proxy_pass http://backend:8000;
   }

   location /static/ {
      alias /static/;
   }
}
```

------------------------------------------------------------------------------------------

#### Restart Nginx

```
>> root@localhost:/opt/clickmart#  docker compose restart nginx
>> root@localhost:/opt/clickmart/ngnix#  docker compose ps
```

## If ngnix down (if 31. title port not update then error occured)

```
>> root@localhost:/opt/clickmart/ngnix#  docker compose logs ngnix
>> root@localhost:/opt/clickmart/ngnix#  docker compose down
```

**Docker-compose.yml update ssl 443 port in ngnix**

```
  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
      - "443:443"        # for ssl cerfificate
```

```
>> root@localhost:/opt/clickmart/ngnix#  docker compose up –d - -
force-recreate ngnix
```

-------------------## If ngnix down----------------------------------------------------

### Update docker-compose.yml (Nginx service)              #local-vscode

*Edit docker-compose.yml locally (nginx service):*

```
Ngnix:
            Volumes:
              -   ./ngnix/default.conf:/etc/ngnix/conf.d/default.conf
              -   ./backend-drf/static:/static
              -   ./certbot/www:/var/www/certbot
              -   /etc/letsencrypt/:/etc/letsencrypt:ro
```

#### Commit and Push

```
>> c:/>django_clickmart> git add .
>> c:/>django_clickmart> git commit -m "ssl installation started"
>> c:/>django_clickmart> git push origin main
```

-----------------------------------------------------------------------------------------

*Goto Linode firewall – add new rule for ssl 443 certificate*

-----------------------------------------------------------------------------------------

*Preset : HTTPS*
*Protocol : TCP*
*Add rule*
*Open cmd/powershell terminal*

```
Curl –Iv https://djangoclickmart.store/
```

*Also check ngnix listing 443 port or not - if blank mean not leasting*

```
>> root@localhost:/opt/clickmart# ss –tulnp | grep 443
>> root@localhost:/opt/clickmart# docker compose ps
```

-----------------------------------------------------------------------------------------

*#### Test HTTPS 🎉*
*https://djangoclickmart.store* *Congratulations 🎉 You did it.*

-----------------------------------------------------------------------------------------

```
>> root@localhost:/opt/clickmart# docker compose exec backend python manage.py createsuperuser
```

*https://djangoclickmart.store/admin* *login*

-----------------------------------------------------------------------------------------

*## CSRF error set in setting and .env.docker*
*Clickmart_main - Settings.py*

--------------------------------------

```
CSRF_TRUSTED_ORIGINS=os.getenv("CSRF_TRUSTED_ORIGINS", "").split(",")
```

-----------------------------------------------------------------------------------------

*Backend-drf - .env.docker*

----------------------------------

```
CSRF_TRUSTED_ORIGINS=https://djangoclickmart.store,https://www.djangoclickmart.store
```

*#### Commit and Push*

```
>> c:/>django_clickmart> git add .
>> c:/>django_clickmart> git commit -m "added CSRF_TRUSTED_ORIGINS"
>> c:/>django_clickmart> git push origin main
```

*Add in nginix server also Backend-drf - .env.docker*

-----------------------------------------------------------------

```
>> root@localhost:/opt/clickmart# nano backend-drf/.env.docker
CSRF_TRUSTED_ORIGINS=https://djangoclickmart.store,https://www.djangoclickmart.store

>> root@localhost:/opt/clickmart# docker compose down
>> root@localhost:/opt/clickmart# docker compose --build -d
```

-----------------------------------------------------------------------------------------

*All Done*

-----------------------------------------------------------------------------------------

*Extra used in media files above code - 30 - Fix Static Files Issues on Production*
*# Fixing Media Files in Production (Docker + Nginx + Django)*
*This guide explains how to fix issues where **media files (uploaded images)** are not loading correctly in production.*
*---*
*### Step 1: Update Nginx Configuration (Server)*
*1. Login to your production server.*
*2. Open the Nginx config file:*

```bash
nano nginx/default.conf
```

*3. Add the following block inside the HTTPS server block:*

```
location /media/ {
    alias /media/;
}
```

*This tells Nginx to serve uploaded media files directly.*
*4. Restart nginx container:*

```
docker compose restart nginx
```

*### Step 2: Mount Media Folder in Docker (Local Project)*
*1. Open `docker-compose.yml` - in your local project*
*2. Inside the nginx service, add the media volume mapping:*

```
nginx:
  volumes:
    - ./backend-drf/media:/media
```

*This allows the Nginx container to access uploaded media files created by Django.*
*3. Commit and push the changes:*

```
git add .
git commit -m "Serve media files using nginx"
git push origin main
```

*### Step 3: Verify Media Files*
*Try opening a media file directly in the browser:*

```
https://your-domain.com/media/example.jpg
```

*If the image loads, media serving is working correctly.*
*### Step 4 (Fallback): Fix Serializer Image URL*
*If media files load directly but still do not appear on the webpage, update the serializer to return a relative media path.*
*1. Open `products/serializers.py`*
*3. Update `ProductSerializer` - or whatever serializer the image is coming from.*
*Refer to below code:*

```
from rest_framework import serializers
class ProductSerializer(serializers.ModelSerializer):
  image = serializers.SerializerMethodField()

  class Meta:
    model = Product
    fields = "__all__"

  def get_image(self, obj):
    return obj.image.url if obj.image else None
```

*This ensures the API returns: `/media/products/image.jpg` instead of Docker-internal URLs like `backend:8000`*

*4. Commit and push again:*

```
git add .
git commit -m "Fix media image URL in serializer"
git push origin main
```

*5. Test again.*