生产环境虚拟环境部署指南

第一步：服务器准备

bash

```
# 1. 更新系统并安装必要软件
sudo apt update && sudo apt upgrade -y
sudo apt install python3 python3-pip python3-venv python3-dev -y

# 2. 安装系统级依赖（MySQL 客户端库等）
sudo apt install build-essential libssl-dev libffi-dev -y
sudo apt install default-libmysqlclient-dev pkg-config -y

# 3. 创建项目用户（推荐）
sudo adduser django_user
sudo usermod -aG sudo django_user

# 4. 切换到项目用户
su - django_user
```

第二步：创建项目目录结构

bash

```
# 创建标准目录结构
mkdir -p ~/projects/myproject
cd ~/projects/myproject

# 建议的目录结构
# ~/projects/myproject/
# ├── venv/              # 虚拟环境
# ├── src/               # 项目源码
# │   ├── config/        # Django 配置
# │   ├── api/           # 应用代码
# │   └── manage.py
# ├── logs/              # 日志文件
# ├── staticfiles/       # 收集的静态文件
# ├── media/             # 用户上传文件
# └── frontend/          # 前端构建文件
```

第三步：创建和配置虚拟环境

```bash
# 1. 创建虚拟环境
cd ~/projects/myproject
python3 -m venv venv

# 2. 激活虚拟环境
source venv/bin/activate

# 激活后，命令提示符会显示 (venv)
# (venv) django_user@server:~/projects/myproject$

# 3. 升级 pip
pip install --upgrade pip setuptools wheel

# 4. 验证虚拟环境
which python  # 应显示: /home/django_user/projects/myproject/venv/bin/python
which pip     # 应显示: /home/django_user/projects/myproject/venv/bin/pip
python --version
```

第四步：上传项目代码

方法 A: 使用 Git（推荐）

```bash
# 在项目目录下克隆代码
cd ~/projects/myproject
git clone https://github.com/yourusername/yourproject.git src

# 或者如果已有 src 目录
cd src
git init
git remote add origin https://github.com/yourusername/yourproject.git
git pull origin main
```

方法 B: 使用 SCP/SFTP

```bash
# 在本地电脑执行
scp -r /path/to/local/project django_user@your-server-ip:~/projects/myproject/src

# 或使用 rsync（更好的选择）
rsync -avz --exclude 'venv' --exclude '__pycache__' --exclude '*.pyc' \
  /path/to/local/project/ django_user@your-server-ip:~/projects/myproject/src/
```

第五步：安装项目依赖

```bash
# 1. 确保虚拟环境已激活
source ~/projects/myproject/venv/bin/activate

# 2. 安装依赖
cd ~/projects/myproject/src
pip install -r requirements.txt

# 3. 安装 Gunicorn
pip install gunicorn

# 4. 验证安装
pip list
pip freeze > requirements_frozen.txt  # 保存精确版本
```

如果遇到 mysqlclient 安装问题

```bash
# 先安装系统依赖
sudo apt install default-libmysqlclient-dev pkg-config -y

# 然后重新安装
pip install mysqlclient

# 或者使用 PyMySQL 替代
pip uninstall mysqlclient
pip install PyMySQL

# 如果使用 PyMySQL，在 Django 项目的 __init__.py 添加:
# import pymysql
# pymysql.install_as_MySQLdb()
```

第六步：配置环境变量

```bash
# 在项目根目录创建 .env 文件
cd ~/projects/myproject/src
nano .env
```

.env 文件内容：

bash

```
# Django 设置
DJANGO_SECRET_KEY=your-very-secure-secret-key-here
DJANGO_SETTINGS_MODULE=config.settings
DEBUG=False

# 数据库配置
DB_NAME=TH_Project
DB_USER=django_user
DB_PASSWORD=your-secure-db-password
DB_HOST=localhost
DB_PORT=3306

# Redis 配置
REDIS_URL=redis://127.0.0.1:6379/1
REDIS_PASSWORD=

# 应用配置
ALLOWED_HOSTS=your-domain.com,www.your-domain.com,your-server-ip
CORS_ALLOWED_ORIGINS=https://your-domain.com,https://www.your-domain.com

# Lark 配置
LARK_WEBHOOK_URL=your-lark-webhook-url
LARK_WEBHOOK_SECRET=your-lark-secret
LARK_ENABLE_NOTIFICATIONS=True

# 前端 URL
FRONTEND_URL=https://your-domain.com
```

bash

```
# 设置文件权限（重要！）
chmod 600 .env
```

第七步：创建其他必要目录

```bash
cd ~/projects/myproject

# 创建日志目录
mkdir -p logs
chmod 755 logs

# 创建静态文件目录
mkdir -p staticfiles
chmod 755 staticfiles

# 创建媒体文件目录
mkdir -p media
chmod 755 media

# 创建字体目录（如果需要）
mkdir -p static/fonts
```

第八步：初始化 Django 项目

```bash
# 激活虚拟环境
source ~/projects/myproject/venv/bin/activate
cd ~/projects/myproject/src

# 1. 运行数据库迁移
python manage.py makemigrations
python manage.py migrate

# 2. 收集静态文件
python manage.py collectstatic --noinput

# 3. 创建超级用户
python manage.py createsuperuser

# 4. 测试运行
python manage.py runserver 0.0.0.0:8000
# 如果能访问，按 Ctrl+C 停止
```

第九步：配置 Gunicorn 服务

1. 创建 Gunicorn 配置文件

bash

```
cd ~/projects/myproject/src
nano gunicorn_config.py
```

python

```
import multiprocessing
import os

# 绑定地址
bind = "127.0.0.1:8000"

# Worker 进程数
workers = multiprocessing.cpu_count() * 2 + 1
worker_class = "sync"
worker_connections = 1000

# 超时设置
timeout = 30
keepalive = 2
graceful_timeout = 30

# 日志配置
accesslog = "/home/django_user/projects/myproject/logs/gunicorn_access.log"
errorlog = "/home/django_user/projects/myproject/logs/gunicorn_error.log"
loglevel = "info"
access_log_format = '%(h)s %(l)s %(u)s %(t)s "%(r)s" %(s)s %(b)s "%(f)s" "%(a)s"'

# 进程命名
proc_name = "myproject_django"

# 安全
limit_request_line = 4094
limit_request_fields = 100
limit_request_field_size = 8190

# 服务器机制
daemon = False
pidfile = None
user = None
group = None
tmp_upload_dir = None

# SSL（如果需要）
# keyfile = "/path/to/key.pem"
# certfile = "/path/to/cert.pem"
```

2. 测试 Gunicorn

```bash
# 激活虚拟环境
source ~/projects/myproject/venv/bin/activate
cd ~/projects/myproject/src

# 测试运行
gunicorn --config gunicorn_config.py config.wsgi:application

# 如果正常运行，按 Ctrl+C 停止
```

第十步：创建 Systemd 服务

```bash
# 创建服务文件（需要 sudo）
sudo nano /etc/systemd/system/myproject.service
```

myproject.service 内容：

ini

```ini
[Unit]
Description=MyProject Django Application
After=network.target mysql.service redis.service
Requires=mysql.service

[Service]
Type=notify
User=django_user
Group=django_user

# 项目目录
WorkingDirectory=/home/django_user/projects/myproject/src

# 虚拟环境路径（重要！）
Environment="PATH=/home/django_user/projects/myproject/venv/bin"

# 加载环境变量
EnvironmentFile=/home/django_user/projects/myproject/src/.env

# 启动命令（使用虚拟环境中的 gunicorn）
ExecStart=/home/django_user/projects/myproject/venv/bin/gunicorn \
    --config /home/django_user/projects/myproject/src/gunicorn_config.py \
    config.wsgi:application

# 重启策略
Restart=always
RestartSec=10

# 安全设置
PrivateTmp=true
NoNewPrivileges=true

[Install]
WantedBy=multi-user.target
```

启动服务

```bash
# 1. 重新加载 systemd 配置
sudo systemctl daemon-reload

# 2. 启用开机自启
sudo systemctl enable myproject

# 3. 启动服务
sudo systemctl start myproject

# 4. 查看状态
sudo systemctl status myproject

# 5. 查看日志
sudo journalctl -u myproject -f

# 6. 如果有错误，查看详细日志
sudo journalctl -u myproject -n 50 --no-pager
```

第十一步：虚拟环境管理命令

日常维护

```bash
# 激活虚拟环境
source ~/projects/myproject/venv/bin/activate

# 查看已安装的包
pip list

# 更新某个包
pip install --upgrade package-name

# 安装新包
pip install new-package
pip freeze > requirements.txt  # 更新 requirements.txt

# 退出虚拟环境
deactivate
```

重新创建虚拟环境

```bash
# 1. 停止服务
sudo systemctl stop myproject

# 2. 备份旧环境
cd ~/projects/myproject
mv venv venv.backup

# 3. 创建新环境
python3 -m venv venv
source venv/bin/activate

# 4. 安装依赖
pip install --upgrade pip
cd src
pip install -r requirements.txt
pip install gunicorn

# 5. 重启服务
sudo systemctl start myproject
```

清理虚拟环境

```bash
# 清理 pip 缓存
pip cache purge

# 查看虚拟环境大小
du -sh ~/projects/myproject/venv

# 完全删除虚拟环境
rm -rf ~/projects/myproject/venv
```

第十二步：部署更新流程

bash

```bash
#!/bin/bash
# 创建更新脚本: ~/projects/myproject/deploy.sh

cd ~/projects/myproject/src

# 1. 拉取最新代码
git pull origin main

# 2. 激活虚拟环境
source ../venv/bin/activate

# 3. 安装/更新依赖
pip install -r requirements.txt

# 4. 运行迁移
python manage.py migrate

# 5. 收集静态文件
python manage.py collectstatic --noinput

# 6. 重启服务
sudo systemctl restart myproject

# 7. 查看状态
sudo systemctl status myproject

echo "部署完成！"
```

bash

```bash
# 使脚本可执行
chmod +x ~/projects/myproject/deploy.sh

# 运行更新
~/projects/myproject/deploy.sh
```

常见问题排查

1. 虚拟环境激活失败

bash

```
# 检查虚拟环境是否存在
ls -la ~/projects/myproject/venv/bin/activate

# 重新创建
python3 -m venv ~/projects/myproject/venv --clear
```

2. pip 安装包失败

bash

```
# 升级 pip
source venv/bin/activate
pip install --upgrade pip setuptools wheel

# 清理缓存
pip cache purge
```

3. Systemd 服务无法启动

bash

```
# 检查虚拟环境路径是否正确
cat /etc/systemd/system/myproject.service | grep Environment

# 手动测试启动命令
source ~/projects/myproject/venv/bin/activate
cd ~/projects/myproject/src
gunicorn config.wsgi:application

# 查看详细错误
sudo journalctl -u myproject -n 100 --no-pager
```

4. 权限问题

bash

```
# 确保项目目录所有者正确
sudo chown -R django_user:django_user ~/projects/myproject

# 检查日志目录权限
ls -la ~/projects/myproject/logs
chmod 755 ~/projects/myproject/logs
```

安全最佳实践

1. **虚拟环境权限**

```bash
chmod 755 ~/projects/myproject/venv
```

2. **环境变量文件权限**

```bash
chmod 600 ~/projects/myproject/src/.env
```

3. **定期更新依赖**

```bash
source venv/bin/activate
pip list --outdated
pip install --upgrade package-name
```

4. **备份虚拟环境配置**

```bash
pip freeze > requirements_$(date +%Y%m%d).txt
```

总结

✅ **必须使用虚拟环境的原因：**

- 依赖隔离和版本控制
- 安全性和可维护性
- 便于团队协作和部署
- 行业标准最佳实践

✅ **虚拟环境路径：**

- `/home/django_user/projects/myproject/venv`

✅ **激活命令：**

- `source ~/projects/myproject/venv/bin/activate`

✅ **Systemd 服务配置关键：**

- `Environment="PATH=/path/to/venv/bin"`
- `ExecStart=/path/to/venv/bin/gunicorn ...`