AWS EC2 Django Deployment (Ubuntu 22.04)

This is a **clean, repeatable checklist** distilled from your notes. Use it **next time without guessing**.

Step 1: Launch EC2 Instance

AMI: Ubuntu 22.04 LTS
Instance: t2.micro (1GB RAM)
Storage: 30GB gp2

Security Group

- SSH: 22 (restrict to your IP)
- HTTP: 80
- HTTPS: 443 (optional)
- Custom TCP: 8000 (temporary testing)

Connect

```
chmod 400 emart.pem
ssh -i emart.pem ubuntu@<PUBLIC_IP>
```

Base Packages

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y git curl wget build-essential
```

Step 2: Add Swap (Critical for 1GB RAM)

```
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
free -h
```

Step 3: Firewall & Basic Security

```
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https
sudo ufw enable
```

(Optional) Change SSH port:

```
sudo nano /etc/ssh/sshd_config
# Port 2222
sudo systemctl restart ssh
```

Step 4: Install Python 3.12 (Recommended: PPA)

```
sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install -y python3.12 python3.12-venv python3.12-dev python3-pip
curl -sS https://bootstrap.pypa.io/get-pip.py | python3.12
```

Verify:

```
python3.12 --version
python3.12 -m pip --version
```

Step 5: Install PostgreSQL 16

```
sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -
cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
wget -qO - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key
add -
sudo apt update
sudo apt install -y postgresql-16 postgresql-contrib-16
sudo systemctl enable --now postgresql
```

Optimize PostgreSQL for 1GB RAM

```
sudo nano /etc/postgresql/16/main/postgresql.conf
```

```
shared_buffers = 128MB
effective_cache_size = 256MB
work_mem = 4MB
maintenance_work_mem = 64MB
```

```
sudo systemctl restart postgresql
```

Step 6: PostgreSQL DB & User

```
sudo -i -u postgres
createuser --interactive --pwprompt
# username: shornamart
# password: strong password
createdb shornamartdb --owner=shornamart
exit
```

Edit access:

```
sudo nano /etc/postgresql/16/main/pg_hba.conf
```

```
host  shornamartdb  shornamart  127.0.0.1/32  scram-sha-256
```

```
sudo systemctl restart postgresql
```

Step 7: GitHub SSH Key

```
ssh-keygen -t ed25519 -C "your-email@example.com"
cat ~/.ssh/id_ed25519.pub
```

Add key to GitHub → SSH Keys.

Step 8: Clone Project

```
sudo mkdir -p /home/ubuntu/emart
sudo chown ubuntu:ubuntu /home/ubuntu/emart
cd /home/ubuntu

git clone --branch release --single-branch git@github.com:cseshahriar/e-mart.git emart
cd emart
```

Step 9: Virtual Environment & Dependencies

```
python3.12 -m venv venv
source venv/bin/activate
pip install --upgrade pip
pip install django gunicorn psycopg2-binary whitenoise[brotli] python-dotenv
```

Check Django:

```
python manage.py check
```

Step 10: Gunicorn (systemd)

```
sudo nano /etc/systemd/system/gunicorn.service
```

```
[Unit]
Description=Gunicorn for E-Mart
After=network.target postgresql.service

[Service]
User=ubuntu
Group=www-data
WorkingDirectory=/home/ubuntu/emart
ExecStart=/home/ubuntu/emart/venv/bin/gunicorn
  --workers 2
  --bind unix:/home/ubuntu/emart/emart.sock
  config.wsgi:application
Restart=on-failure
```

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

```
sudo systemctl daemon-reload
sudo systemctl enable --now gunicorn
sudo systemctl status gunicorn
```

Step 11: Test Gunicorn Manually

```
source venv/bin/activate
gunicorn --bind 0.0.0.0:8000 config.wsgi:application
```

Visit `http://IP:8000`

Step 12: Install & Configure Nginx

```
sudo apt install -y nginx
sudo rm -f /etc/nginx/sites-enabled/default
sudo nano /etc/nginx/sites-available/emart
```

```
upstream emart_server {
    server unix:/home/ubuntu/emart/emart.sock;
}

server {
    listen 80;
    server_name _;

    client_max_body_size 25M;

    location /static/ {
        alias /home/ubuntu/emart/static/;
    }

    location /media/ {
        alias /home/ubuntu/emart/media/;
    }
```

```
    location / {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass http://emart_server;
    }
}
```

```
sudo ln -s /etc/nginx/sites-available/emart /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

Step 13: Permissions Fix (502 Errors)

```
sudo chmod 755 /home/ubuntu
sudo chmod 755 /home/ubuntu/emart
sudo chown -R ubuntu:www-data /home/ubuntu/emart
```

Step 14: Debug Checklist

```
sudo systemctl status gunicorn
ls -la /home/ubuntu/emart/emart.sock
sudo journalctl -u gunicorn -n 50
sudo tail -f /var/log/nginx/error.log
```

Step 15: Enable HTTPS (Let's Encrypt – HTTPS Setup)

> ⚠️ **Requirement**: You MUST have a domain name pointed to this EC2 IP. HTTPS **does NOT work on public IP only**.

15.1 Point Domain to EC2

In your domain DNS (Namecheap / Cloudflare / etc):

Type	Host	Value	TTL
A	@	YOUR_EC2_PUBLIC_IP	Auto
A	www	YOUR_EC2_PUBLIC_IP	Auto

Wait 1–5 minutes, then verify:

```
ping yourdomain.com
```

15.2 Update Nginx `server_name`

Edit config:

```
sudo nano /etc/nginx/sites-available/emart
```

Change:

```
server_name _;
```

To:

```
server_name yourdomain.com www.yourdomain.com;
```

Test & reload:

```
sudo nginx -t
sudo systemctl reload nginx
```

15.3 Install Certbot

```
sudo apt update
sudo apt install -y certbot python3-certbot-nginx
```

15.4 Issue SSL Certificate (AUTO CONFIG)

```
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com
```

During prompts: - Email → enter valid email - Agree ToS → **Y** - Redirect HTTP → HTTPS → **2 (Recommended)**

Certbot will: - Install SSL certificate - Modify Nginx config - Enable HTTPS automatically

---

15.5 Verify HTTPS

Open browser:

```
https://yourdomain.com
```

Check certificate:

```
sudo certbot certificates
```

15.6 Auto-Renew (IMPORTANT)

Certbot installs auto-renew by default.

Test renewal:

```
sudo certbot renew --dry-run
```

System timer:

```
systemctl list-timers | grep certbot
```

15.7 Firewall Check

```
sudo ufw allow 443
sudo ufw reload
```

✅HTTPS SUCCESS

- 📶HTTPS enabled
- 🔁Auto-renew every 90 days
- 🚫Production-ready SSL

✅Final Result

- Django → Gunicorn (Unix socket)
- Nginx → Reverse proxy
- PostgreSQL 16 optimized for 1GB RAM
- Production-ready baseline

Tip: Save this file and reuse it for every EC2 Django deployment.