CareCove Deployment Guide

This guide provides comprehensive instructions for deploying CareCove in various environments, from local development to production servers.

Table of Contents

Local Development Setup

Prerequisites

- Python 3.8 or higher
- Git
- Node.js (optional, for frontend development)
- Database system (PostgreSQL recommended for production-like development)

Step-by-Step Setup

1. **Clone and Setup Project**

```
git clone <repository-url>
cd carecove-django
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
pip install -r requirements.txt
```

1. **Environment Configuration**

```
cp .env.example .env
# Edit .env file with your local settings
```

1. **Database Setup**

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

1. **Load Sample Data**

```
python manage.py seed_data
python manage.py seed_chatbot_data
```

1. **Run Development Server**

```
python manage.py runserver
```

Production Deployment

Server Requirements

Minimum Requirements:
- 2 CPU cores
- 4GB RAM
- 20GB SSD storage
- Ubuntu 20.04 LTS or CentOS 8

Recommended for Production:
- 4+ CPU cores
- 8GB+ RAM
- 50GB+ SSD storage
- Load balancer for high availability

Production Server Setup

1. **System Dependencies**

```
# Ubuntu/Debian
sudo apt update
sudo apt install python3 python3-pip python3-venv nginx postgresql postgresql-contrib r
edis-server

# CentOS/RHEL
sudo yum update
sudo yum install python3 python3-pip nginx postgresql postgresql-server redis
```

1. **Create Application User**

```
sudo useradd --system --home /opt/carecove --shell /bin/bash carecove
sudo mkdir -p /opt/carecove
sudo chown carecove:carecove /opt/carecove
```

1. **Deploy Application**

```
sudo -u carecove git clone <repository-url> /opt/carecove/app
cd /opt/carecove/app
sudo -u carecove python3 -m venv venv
sudo -u carecove venv/bin/pip install -r requirements.txt
```

1. **Production Environment**

```
sudo -u carecove cp .env.example .env
# Edit .env with production values
sudo -u carecove nano .env
```

1. **Database Setup**

```
# PostgreSQL setup
sudo -u postgres createuser carecove
sudo -u postgres createdb carecove -O carecove
sudo -u postgres psql -c "ALTER USER carecove PASSWORD 'secure_password';"
```

1. **Django Setup**

```
sudo -u carecove venv/bin/python manage.py collectstatic --noinput
sudo -u carecove venv/bin/python manage.py migrate
sudo -u carecove venv/bin/python manage.py createsuperuser
```

Web Server Configuration

Nginx Configuration (`/etc/nginx/sites-available/carecove`)

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;
    add_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'un
safe-inline'" always;

    location = /favicon.ico { access_log off; log_not_found off; }

    location /static/ {
        root /opt/carecove/app;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    location /media/ {
        root /opt/carecove/app;
        expires 1y;
        add_header Cache-Control "public";
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/opt/carecove/app/carecove.sock;
        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;
    }
}
```

Gunicorn Configuration (`/opt/carecove/app/gunicorn.conf.py`)

```
bind = "unix:/opt/carecove/app/carecove.sock"
workers = 3
worker_class = "sync"
worker_connections = 1000
max_requests = 1000
max_requests_jitter = 100
timeout = 30
keepalive = 2
user = "carecove"
group = "carecove"
```

Systemd Service

Gunicorn Service (`/etc/systemd/system/carecove.service`)

```
[Unit]
Description=CareCove Django App
After=network.target

[Service]
User=carecove
Group=carecove
WorkingDirectory=/opt/carecove/app
Environment="PATH=/opt/carecove/app/venv/bin"
ExecStart=/opt/carecove/app/venv/bin/gunicorn --config gunicorn.conf.py care-
cove.wsgi:application
ExecReload=/bin/kill -s HUP $MAINPID
Restart=on-failure

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

Enable and Start Services

```
sudo systemctl daemon-reload
sudo systemctl enable carecove
sudo systemctl start carecove
sudo systemctl enable nginx
sudo systemctl start nginx
```

Cloud Platform Deployments

Heroku Deployment

1. **Prerequisites**

```
pip install gunicorn dj-database-url
echo "web: gunicorn carecove.wsgi" > Procfile
```

1. **Heroku Setup**

```
heroku create your-app-name
heroku addons:create heroku-postgresql:hobby-dev
heroku addons:create heroku-redis:hobby-dev
```

1. **Environment Variables**

```
heroku config:set SECRET_KEY="your-secret-key"
heroku config:set DEBUG=False
heroku config:set ALLOWED_HOSTS="your-app-name.herokuapp.com"
heroku config:set ABACUSAI_API_KEY="your-api-key"
```

1. **Deploy**

```
git add .
git commit -m "Deploy to Heroku"
git push heroku main
heroku run python manage.py migrate
heroku run python manage.py createsuperuser
```

DigitalOcean App Platform

1. **Create** `app.yaml`

```yaml
name: carecove
services:
- name: web
  source_dir: /
  github:
    repo: your-username/carecove-django
    branch: main
  run_command: gunicorn --worker-tmp-dir /dev/shm carecove.wsgi
  environment_slug: python
  instance_count: 1
  instance_size_slug: basic-xxs
  envs:
  - key: DEBUG
    value: "False"
  - key: SECRET_KEY
    value: "your-secret-key"
    type: SECRET
databases:
- name: carecove-db
  engine: PG
  version: "13"
```

AWS Elastic Beanstalk

1. **Install EB CLI**

```
pip install awsebcli
```

1. **Initialize**

```
eb init -p python-3.8 carecove
eb create carecove-production
```

1. **Environment Configuration**

```
eb setenv SECRET_KEY="your-secret-key"
eb setenv DEBUG=False
eb setenv ALLOWED_HOSTS="your-domain.com"
```

Database Configuration

PostgreSQL Production Setup

1. **Installation & Configuration**

```
# Install PostgreSQL
sudo apt install postgresql postgresql-contrib

# Create database and user
sudo -u postgres psql
CREATE DATABASE carecove;
CREATE USER carecove WITH PASSWORD 'secure_password';
ALTER ROLE carecove SET client_encoding TO 'utf8';
ALTER ROLE carecove SET default_transaction_isolation TO 'read committed';
ALTER ROLE carecove SET timezone TO 'UTC';
GRANT ALL PRIVILEGES ON DATABASE carecove TO carecove;
\q
```

1. **Django Settings**

```
# Add to settings.py
import dj_database_url
DATABASES = {
    'default': dj_database_url.parse(os.environ.get('DATABASE_URL'))
}
```

1. **Database Migrations**

```
python manage.py makemigrations
python manage.py migrate
```

Database Backup & Restore

Backup

```
pg_dump -U carecove -h localhost carecove > backup_$(date +%Y%m%d_%H%M%S).sql
```

Restore

```
psql -U carecove -h localhost carecove < backup_file.sql
```

Static Files & Media

Local Static Files (Development)

```
# settings.py
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / 'static']
STATIC_ROOT = BASE_DIR / 'staticfiles'
```

Production Static Files

WhiteNoise (Simple)

```
# settings.py
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    # ... other middleware
]

STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'
```

AWS S3 (Scalable)

```
pip install django-storages boto3
```

```
# settings.py
DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'
STATICFILES_STORAGE = 'storages.backends.s3boto3.S3StaticStorage'

AWS_ACCESS_KEY_ID = os.environ.get('AWS_ACCESS_KEY_ID')
AWS_SECRET_ACCESS_KEY = os.environ.get('AWS_SECRET_ACCESS_KEY')
AWS_STORAGE_BUCKET_NAME = os.environ.get('AWS_STORAGE_BUCKET_NAME')
AWS_S3_REGION_NAME = os.environ.get('AWS_S3_REGION_NAME')
```

SSL/HTTPS Setup

Let’s Encrypt (Free SSL)

1. **Install Certbot**

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

1. **Obtain Certificate**

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

1. **Auto-renewal**

```
sudo crontab -e
# Add: 0 12 * * * /usr/bin/certbot renew --quiet
```

Nginx SSL Configuration

```
server {
    listen 443 ssl http2;
    server_name yourdomain.com www.yourdomain.com;

    ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;

    # SSL Security
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;

    # ... rest of configuration
}

# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    return 301 https://$server_name$request_uri;
}
```

Performance Optimization

Caching Configuration

Redis Setup

```
# Install Redis
sudo apt install redis-server

# Configure Django
pip install django-redis
```

```
# settings.py
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
        }
    }
}

SESSION_ENGINE = "django.contrib.sessions.backends.cache"
SESSION_CACHE_ALIAS = "default"
```

Database Optimization

1. **Database Indexes**

```
# Add to models.py
class Product(models.Model):
    name = models.CharField(max_length=200, db_index=True)
    category = models.ForeignKey(Category, on_delete=models.CASCADE, db_index=True)

    class Meta:
        indexes = [
            models.Index(fields=['name', 'category']),
            models.Index(fields=['created_at']),
        ]
```

1. **Query Optimization**

```
# Use select_related and prefetch_related
products = Product.objects.select_related('category').prefetch_related('images')
```

CDN Configuration

CloudFlare Setup

1. Sign up for CloudFlare
2. Add your domain
3. Update nameservers
4. Configure caching rules

Monitoring & Maintenance

Logging Configuration

```
# settings.py
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
            'format': '{levelname} {asctime} {module} {process:d} {thread:d}
{message}',
            'style': '{',
        },
    },
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': '/var/log/carecove/django.log',
            'formatter': 'verbose',
        },
    },
    'root': {
        'handlers': ['file'],
        'level': 'INFO',
    },
}
```

Health Checks

Basic Health Check (`/health/`)

```
# views.py
from django.http import JsonResponse
from django.db import connections

def health_check(request):
    try:
        db_conn = connections['default']
        db_conn.cursor()
        return JsonResponse({'status': 'healthy'})
    except Exception as e:
        return JsonResponse({'status': 'unhealthy', 'error': str(e)}, status=500)
```

Backup Strategy

Automated Database Backup

```
#!/bin/bash
# backup.sh
BACKUP_DIR="/opt/backups/carecove"
DATE=$(date +%Y%m%d_%H%M%S)
mkdir -p $BACKUP_DIR

# Database backup
pg_dump -U carecove carecove > $BACKUP_DIR/db_backup_$DATE.sql

# Media files backup
tar -czf $BACKUP_DIR/media_backup_$DATE.tar.gz /opt/carecove/app/media/

# Keep only last 7 days of backups
find $BACKUP_DIR -type f -mtime +7 -delete
```

Cron Job

```
sudo crontab -e
# Add: 0 2 * * * /opt/scripts/backup.sh
```

Troubleshooting

Common Issues

1. Static Files Not Loading

```
# Collect static files
python manage.py collectstatic --noinput

# Check nginx static configuration
sudo nginx -t
sudo systemctl reload nginx
```

2. Database Connection Issues

```
# Check PostgreSQL status
sudo systemctl status postgresql

# Test connection
psql -U carecove -h localhost -d carecove
```

3. Gunicorn Not Starting

```
# Check logs
sudo journalctl -u carecove -f

# Test gunicorn manually
cd /opt/carecove/app
venv/bin/gunicorn carecove.wsgi
```

4. Permission Issues

```
# Fix ownership
sudo chown -R carecove:carecove /opt/carecove/
sudo chmod -R 755 /opt/carecove/app/static/
```

Performance Issues

1. Slow Database Queries

```
# Enable query logging in development
DATABASES = {
    'default': {
        # ... other settings
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        }
    }
}

LOGGING = {
    # ... other settings
    'loggers': {
        'django.db.backends': {
            'level': 'DEBUG',
            'handlers': ['console'],
        }
    }
}
```

2. High Memory Usage

```
# Monitor memory usage
htop

# Adjust Gunicorn workers
# gunicorn.conf.py
workers = 2  # Reduce if memory is limited
```

Security Checklist

- [ ] DEBUG = False in production
- [ ] Strong SECRET_KEY
- [ ] HTTPS enabled
- [ ] Database passwords secured
- [ ] API keys in environment variables
- [ ] Regular security updates
- [ ] File upload restrictions
- [ ] Rate limiting configured
- [ ] CORS properly configured
- [ ] Security headers enabled

Maintenance Tasks

Weekly

- [ ] Check application logs
- [ ] Verify backup integrity
- [ ] Monitor disk space
- [ ] Check SSL certificate expiry

Monthly

- [ ] Update dependencies
- [ ] Review security logs
- [ ] Performance optimization
- [ ] Database maintenance

For additional support, refer to the main README.md or contact the development team.