Deployment Guide

Production Deployment

Prerequisites

- Python 3.11+
- PostgreSQL 13+
- Redis 6+
- Nginx (recommended)
- SSL certificate

Environment Setup

1. **Create production environment file**:

```
cp .env.example .env.production
```

1. **Configure production settings**:

```
DEBUG=False
SECRET_KEY=your-production-secret-key
ALLOWED_HOSTS=yourdomain.com,www.yourdomain.com
DATABASE_URL=postgresql://user:password@localhost:5432/academic_papers
REDIS_URL=redis://localhost:6379/0
OPENAI_API_KEY=your-openai-api-key
```

Database Setup

1. **Create PostgreSQL database**:

```
CREATE DATABASE academic_papers;
CREATE USER academic_user WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE academic_papers TO academic_user;
```

1. **Run migrations**:

```
 python manage.py migrate
python manage.py collectstatic --noinput
python manage.py init_paper_data
```

Nginx Configuration

```
 server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    return 301 https://$server_name$request_uri;
}

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

    ssl_certificate /path/to/certificate.crt;
    ssl_certificate_key /path/to/private.key;

    location / {
        proxy_pass http://127.0.0.1:8000;
        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;
    }

    location /static/ {
        alias /path/to/staticfiles/;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    location /media/ {
        alias /path/to/media/;
        expires 1y;
        add_header Cache-Control "public";
    }
}
```

Systemd Service

Create `/etc/systemd/system/academic-papers.service`:

```
 [Unit]
Description=Academic Papers Django Application
After=network.target
```

```
[Service]
Type=notify
User=www-data
Group=www-data
WorkingDirectory=/path/to/restructured_backend
Environment=DJANGO_SETTINGS_MODULE=config.settings
EnvironmentFile=/path/to/.env.production
ExecStart=/path/to/venv/bin/gunicorn config.wsgi:application --bind 127.0.0.1:8000
--workers 3
ExecReload=/bin/kill -s HUP $MAINPID
Restart=on-failure

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

Enable and start the service:

```
 sudo systemctl enable academic-papers
sudo systemctl start academic-papers
```

Docker Deployment

Dockerfile

```
 FROM python:3.11-slim

WORKDIR /app

# Install system dependencies
RUN apt-get update && apt-get install -y \
    postgresql-client \
    && rm -rf /var/lib/apt/lists/*

# Install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application code
COPY . .

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

# Create non-root user
RUN useradd --create-home --shell /bin/bash app
USER app
```

```
EXPOSE 8000

CMD ["gunicorn", "config.wsgi:application", "--bind", "0.0.0.0:8000"]
```

Docker Compose

```
 version: '3.8'

services:
  web:
    build: .
    ports:
      - "8000:8000"
    environment:
      - DEBUG=False
      - DATABASE_URL=postgresql://postgres:password@db:5432/academic_papers
      - REDIS_URL=redis://redis:6379/0
    depends_on:
      - db
      - redis
    volumes:
      - static_volume:/app/staticfiles
      - media_volume:/app/media

  db:
    image: postgres:13
    environment:
      - POSTGRES_DB=academic_papers
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=password
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:6-alpine
    volumes:
      - redis_data:/data

  nginx:
    image: nginx:alpine
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - static_volume:/app/staticfiles
      - media_volume:/app/media
    depends_on:
      - web
```

```
volumes:
  postgres_data:
  redis_data:
  static_volume:
  media_volume:
```

Monitoring Setup

Health Check Endpoint

The application includes a health check endpoint at `/health/` that returns:

```
{
  "status": "healthy",
  "database": "connected",
  "redis": "connected",
  "timestamp": "2024-01-01T00:00:00Z"
}
```

Logging Configuration

Configure centralized logging in production:

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.FileHandler',
            'filename': '/var/log/academic-papers/django.log',
        },
        'sentry': {
            'level': 'ERROR',
            'class': 'sentry_sdk.integrations.logging.SentryHandler',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file', 'sentry'],
            'level': 'INFO',
            'propagate': True,
        },
```

```
    },
}
```

Security Checklist

- [ ] Set `DEBUG=False` in production
- [ ] Use strong `SECRET_KEY`
- [ ] Configure proper `ALLOWED_HOSTS`
- [ ] Enable HTTPS with valid SSL certificate
- [ ] Set up database connection encryption
- [ ] Configure Redis authentication
- [ ] Set up firewall rules
- [ ] Enable rate limiting
- [ ] Configure CORS properly
- [ ] Set up monitoring and alerting
- [ ] Regular security updates
- [ ] Backup strategy in place

Backup Strategy

Database Backup

```
 # Daily backup script
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
pg_dump academic_papers > /backups/db_backup_$DATE.sql
find /backups -name "db_backup_*.sql" -mtime +7 -delete
```

Media Files Backup

```
 # Sync media files to S3
aws s3 sync /path/to/media/ s3://your-bucket/media/ --delete
```

Performance Optimization

Database Optimization

- Enable connection pooling
- Set up read replicas for read-heavy operations

- Regular VACUUM and ANALYZE operations
- Monitor slow queries

Caching Strategy

- Redis for session storage
- Cache frequently accessed data
- Use Django's cache framework
- CDN for static assets

Application Optimization

- Use gunicorn with multiple workers
- Enable gzip compression
- Optimize database queries
- Monitor memory usage

Troubleshooting

Common Issues

1. **Database Connection Errors**
2. Check DATABASE_URL configuration
3. Verify PostgreSQL service is running
4. Check firewall settings
5. **Static Files Not Loading**
6. Run `python manage.py collectstatic`
7. Check STATIC_ROOT configuration
8. Verify Nginx static file serving
9. **API Errors**
10. Check application logs
11. Verify environment variables
12. Test API endpoints individually
13. **Performance Issues**
14. Monitor database query performance

15. Check Redis connection
16. Review application metrics

Log Locations

- Application logs: `/var/log/academic-papers/`
- Nginx logs: `/var/log/nginx/`
- PostgreSQL logs: `/var/log/postgresql/`
- System logs: `/var/log/syslog`

Maintenance

Regular Tasks

- Monitor system resources
- Update dependencies
- Review security logs
- Backup verification
- Performance monitoring
- Database maintenance

Update Procedure

1. Test updates in staging environment
2. Create database backup
3. Deploy new code
4. Run migrations if needed
5. Restart application services
6. Verify functionality
7. Monitor for issues

This deployment guide provides comprehensive instructions for setting up the Academic Paper Generator backend in production environments with proper security, monitoring, and maintenance procedures.