# TWIST ERP - Complete Implementation Guide

### Deployment, Configuration & DevOps

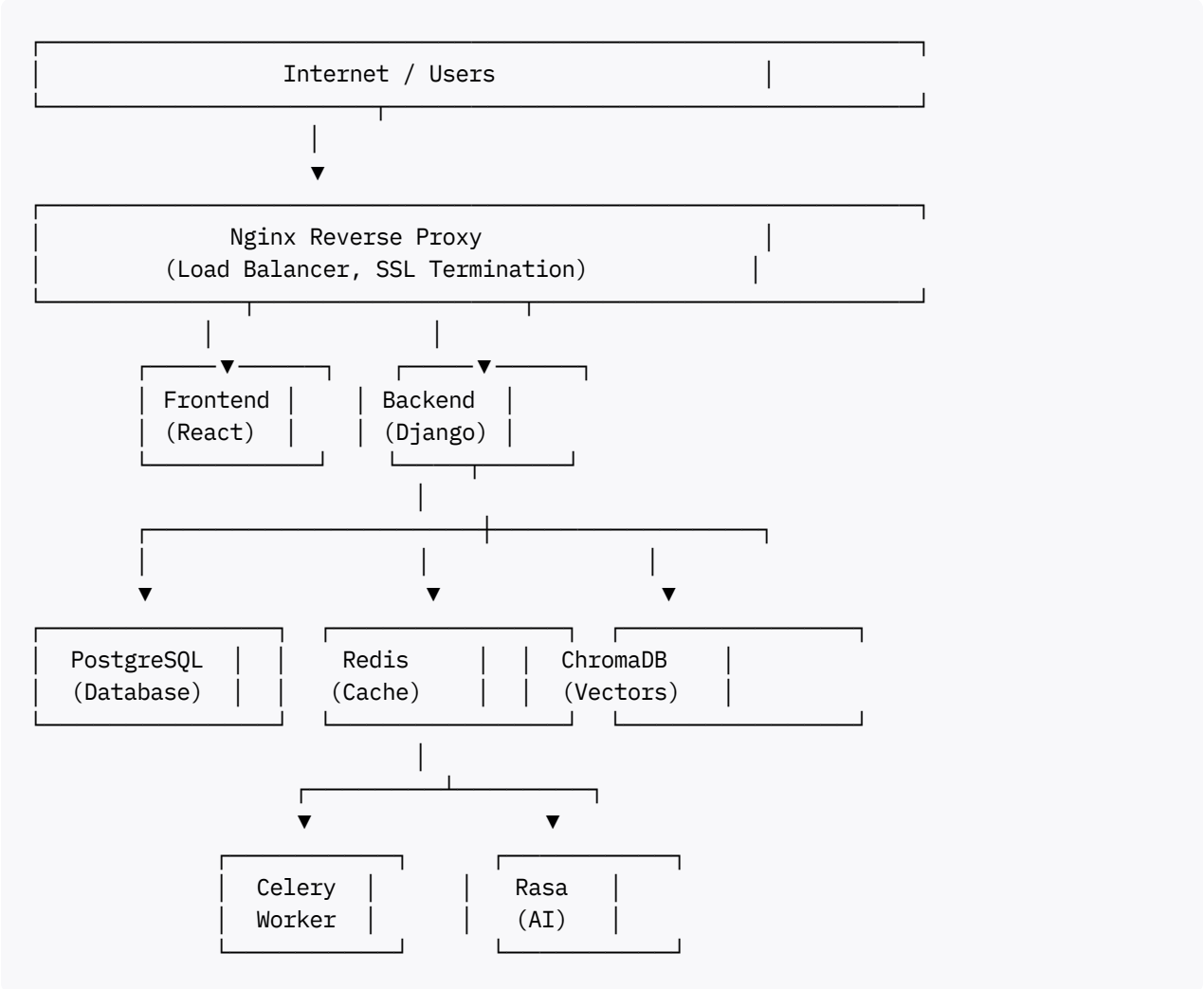
**Version:** 1.0
**Date:** October 2025
**Project:** TWIST ERP - Visual Drag-and-Drop Multi-Company ERP

## 1. Overview

This guide covers the complete deployment, configuration, and DevOps setup for TWIST ERP. It includes production deployment, monitoring, backup strategies, and operational procedures.

## 2. System Architecture

### 2.1 Production Architecture



## 3. Installation Methods

### 3.1 Docker Deployment (Recommended)

**Prerequisites:**

- Docker 24.0+
- Docker Compose 2.0+
- 8GB RAM minimum
- 50GB disk space

**Steps:**

```
# 1. Clone repository
git clone https://github.com/your-org/twist-erp.git
cd twist-erp

# 2. Configure environment
cp .env.example .env
nano .env  # Edit with your settings

# 3. Build and start services
chmod +x deploy.sh
./deploy.sh

# 4. Access the system
# Frontend: http://localhost:3000
# Backend API: http://localhost:8000
# Admin: http://localhost:8000/admin
```

### 3.2 Manual Installation

**For Ubuntu 22.04 LTS:**

```
# 1. Install system dependencies
sudo apt update
sudo apt install -y python3.11 python3.11-venv python3-pip \
    postgresql-15 redis-server nginx nodejs npm git

# 2. Create Python virtual environment
python3.11 -m venv venv
source venv/bin/activate

# 3. Install Python packages
pip install -r requirements.txt
pip install -r phase4-6-requirements.txt

# 4. Configure PostgreSQL
sudo -u postgres psql
CREATE DATABASE twist_erp_db;
CREATE USER twist_user WITH PASSWORD 'your_password';
```

```
GRANT ALL PRIVILEGES ON DATABASE twist_erp_db TO twist_user;
\\q

# 5. Configure environment
cp .env.example .env
# Edit .env with your settings

# 6. Run migrations
python manage.py migrate

# 7. Create superuser
python manage.py createsuperuser

# 8. Collect static files
python manage.py collectstatic

# 9. Start services
# Terminal 1: Django
gunicorn core.wsgi:application --bind 0.0.0.0:8000

# Terminal 2: Celery Worker
celery -A core worker -l info

# Terminal 3: Celery Beat
celery -A core beat -l info

# Terminal 4: Rasa (Phase 5)
cd apps/ai_companion/rasa
rasa run --enable-api
```

## 4. Configuration

### 4.1 Environment Variables

Create `.env` file in project root:

```
# Database
DB_PASSWORD=your_strong_password_here
DATABASE_URL=postgresql://postgres:password@db:5432/twist_erp_db

# Django
SECRET_KEY=your-50-character-random-secret-key
DEBUG=False
ALLOWED_HOSTS=yourdomain.com,www.yourdomain.com

# Redis
REDIS_URL=redis://redis:6379/0

# Email
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USE_TLS=True
EMAIL_HOST_USER=notifications@yourdomain.com
```

```
EMAIL_HOST_PASSWORD=your-app-password

# AI
AI_LLM_MODEL=mistralai/Mistral-7B-Instruct-v0.1
RASA_SERVER_URL=http://rasa:5005

# Security (Production)
SECURE_SSL_REDIRECT=True
SESSION_COOKIE_SECURE=True
CSRF_COOKIE_SECURE=True
```

### 4.2 Django Settings

**Key configurations in `backend/core/settings.py`:**

```
# Multi-Company Support
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'shared.middleware.company_context.CompanyContextMiddleware',  # Custom
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

# Installed Apps
INSTALLED_APPS = [
    # Django
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Third-party
    'rest_framework',
    'corsheaders',
    'django_filters',
    'drf_spectacular',
    'celery',

    # TWIST ERP Apps
    'apps.companies',
    'apps.users',
    'apps.permissions',
    'apps.authentication',

    # Phase 2
    'apps.finance',
    'apps.inventory',
    'apps.sales',
```

```
    'apps.procurement',

    # Phase 3
    'apps.data_migration',

    # Phase 4
    'apps.form_builder',
    'apps.workflows',

    # Phase 5
    'apps.ai_companion',

    # Phase 6
    'apps.assets',
    'apps.budgeting',
    'apps.hr',
    'apps.projects',
]

# Database
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('DB_NAME', 'twist_erp_db'),
        'USER': os.getenv('DB_USER', 'postgres'),
        'PASSWORD': os.getenv('DB_PASSWORD'),
        'HOST': os.getenv('DB_HOST', 'localhost'),
        'PORT': os.getenv('DB_PORT', '5432'),
        'CONN_MAX_AGE': 600,
        'OPTIONS': {
            'connect_timeout': 10,
        },
    }
}

# Celery Configuration
CELERY_BROKER_URL = os.getenv('REDIS_URL', 'redis://localhost:6379/0')
CELERY_RESULT_BACKEND = os.getenv('REDIS_URL', 'redis://localhost:6379/0')
CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
CELERY_RESULT_SERIALIZER = 'json'
CELERY_TIMEZONE = 'Asia/Dhaka'

# Celery Beat Schedule
CELERY_BEAT_SCHEDULE = {
    'run-anomaly-detection': {
        'task': 'apps.ai_companion.tasks.detect_anomalies',
        'schedule': crontab(hour=1, minute=0),
    },
    'calculate-depreciation': {
        'task': 'apps.assets.tasks.calculate_monthly_depreciation',
        'schedule': crontab(day_of_month=1, hour=2),
    },
    'process-payroll-reminders': {
        'task': 'apps.hr.tasks.send_payroll_reminders',
        'schedule': crontab(day_of_month=25, hour=9),
```

```
    },
}

# REST Framework
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 50,
    'DEFAULT_FILTER_BACKENDS': [
        'django_filters.rest_framework.DjangoFilterBackend',
    ],
    'DEFAULT_SCHEMA_CLASS': 'drf_spectacular.openapi.AutoSchema',
}
```

## 5. Security Configuration

### 5.1 SSL/TLS Setup

**Using Let's Encrypt (Free):**

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

# Obtain certificate
sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com

# Auto-renewal (already configured)
sudo certbot renew --dry-run
```

**Nginx SSL Configuration:**

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

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

    # Modern SSL configuration
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
    ssl_prefer_server_ciphers off;

    # HSTS
    add_header Strict-Transport-Security "max-age=31536000" always;
}
```

### 5.2 Firewall Configuration

```
# UFW (Ubuntu)
sudo ufw allow 22/tcp    # SSH
sudo ufw allow 80/tcp    # HTTP
sudo ufw allow 443/tcp   # HTTPS
sudo ufw enable

# Block direct access to database
sudo ufw deny 5432/tcp
```

### 5.3 Database Security

```
-- Create read-only user for reporting
CREATE USER twist_readonly WITH PASSWORD 'readonly_password';
GRANT CONNECT ON DATABASE twist_erp_db TO twist_readonly;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO twist_readonly;

-- Enable row-level security
ALTER TABLE companies ENABLE ROW LEVEL SECURITY;

CREATE POLICY company_isolation ON companies
    USING (id = current_setting('app.current_company_id')::integer);
```

## 6. Backup & Recovery

### 6.1 Database Backup

**Automated Backup Script:**

```
#!/bin/bash
# /opt/twist-erp/backup.sh

BACKUP_DIR="/backup/twist-erp"
DATE=$(date +%Y%m%d_%H%M%S)
DB_NAME="twist_erp_db"

# Create backup
pg_dump -U postgres $DB_NAME | gzip &gt; $BACKUP_DIR/db_$DATE.sql.gz

# Keep last 30 days
find $BACKUP_DIR -name "db_*.sql.gz" -mtime +30 -delete

# Upload to cloud (optional)
# aws s3 cp $BACKUP_DIR/db_$DATE.sql.gz s3://your-bucket/backups/
```

**Cron Schedule:**

```
# Daily backup at 2 AM
0 2 * * * /opt/twist-erp/backup.sh
```

### 6.2 Recovery Procedure

```
# Stop services
docker-compose down

# Restore database
gunzip &lt; /backup/twist-erp/db_20251024_020000.sql.gz | \
    docker-compose run --rm db psql -U postgres twist_erp_db

# Restart services
docker-compose up -d
```

## 7. Monitoring & Logging

### 7.1 Application Monitoring

**Prometheus + Grafana Setup:**

```
# docker-compose.monitoring.yml
version: '3.8'

services:
  prometheus:
    image: prom/prometheus:latest
    volumes:
      - ./monitoring/prometheus.yml:/etc/prometheus/prometheus.yml
      - prometheus_data:/prometheus
    ports:
      - "9090:9090"

  grafana:
    image: grafana/grafana:latest
    volumes:
      - grafana_data:/var/lib/grafana
    ports:
      - "3001:3000"
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=admin

volumes:
  prometheus_data:
  grafana_data:
```

### 7.2 Log Management

**Centralized Logging:**

```
# settings.py
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'json': {
            '()': 'pythonjsonlogger.jsonlogger.JsonFormatter',
        },
    },
    'handlers': {
        'file': {
            'level': 'INFO',
            'class': 'logging.handlers.RotatingFileHandler',
            'filename': '/var/log/twist-erp/django.log',
            'maxBytes': 1024 * 1024 * 15,
            'backupCount': 10,
            'formatter': 'json',
        },
    },
    'root': {
        'handlers': ['file'],
        'level': 'INFO',
    },
}
```

## 8. Performance Optimization

### 8.1 Database Optimization

```
-- Create indexes for common queries
CREATE INDEX idx_companies_active ON companies(id) WHERE is_active = true;
CREATE INDEX idx_invoices_company_status ON invoices(company_id, status);
CREATE INDEX idx_products_company_category ON products(company_id, category_id);

-- Vacuum and analyze
VACUUM ANALYZE;
```

### 8.2 Caching Strategy

```
# views.py
from django.views.decorators.cache import cache_page

@cache_page(60 * 15)  # Cache for 15 minutes
def dashboard_view(request):
```

```
    # Expensive query
    pass
```

### 8.3 Query Optimization

```
# Use select_related and prefetch_related
invoices = Invoice.objects.select_related(
    'company', 'customer'
).prefetch_related(
    'lines__product'
).filter(company=request.company)
```

## 9. Scaling Strategies

### 9.1 Horizontal Scaling

**Load Balancer Configuration:**

```
upstream backend_servers {
    least_conn;
    server backend1:8000;
    server backend2:8000;
    server backend3:8000;
}

server {
    location /api/ {
        proxy_pass http://backend_servers;
    }
}
```

### 9.2 Database Replication

```
# Primary-Replica setup for read scaling
# Primary: Write operations
# Replicas: Read operations (reports, analytics)
```

## 10. Maintenance Procedures

### 10.1 System Updates

```
# Monthly maintenance window

# 1. Backup
./backup.sh
```

```
# 2. Update code
git pull origin main

# 3. Update dependencies
pip install -r requirements.txt --upgrade

# 4. Run migrations
python manage.py migrate

# 5. Collect static
python manage.py collectstatic --noinput

# 6. Restart services
docker-compose restart
```

### 10.2 Health Checks

```
# Create health check endpoint
# /api/health/

{
  "status": "healthy",
  "database": "connected",
  "redis": "connected",
  "celery": "running",
  "disk_usage": "45%"
}
```

## 11. Troubleshooting Guide

### Common Issues

#### Issue: Database Connection Error

```
# Check database status
docker-compose ps db

# Check logs
docker-compose logs db

# Restart database
docker-compose restart db
```

#### Issue: Celery Not Processing Tasks

```
# Check Celery status
docker-compose logs celery_worker
```

```
# Restart Celery
docker-compose restart celery_worker
```

**Issue: High Memory Usage**

```
# Monitor resource usage
docker stats

# Check queries
python manage.py shell
from django.db import connection
print(connection.queries)
```

## 12. Production Checklist

### Pre-Launch Checklist

- ☐ Set DEBUG=False
- ☐ Configure strong SECRET_KEY
- ☐ Set up SSL/TLS certificates
- ☐ Configure firewall rules
- ☐ Set up automated backups
- ☐ Configure monitoring
- ☐ Set up log rotation
- ☐ Configure email notifications
- ☐ Test disaster recovery
- ☐ Document admin procedures
- ☐ Train support staff
- ☐ Perform security audit
- ☐ Load test with expected traffic
- ☐ Set up uptime monitoring

**Document Version:** 1.0
**Last Updated:** October 2025
**Next Review:** January 2026