Guia de Deploy - Voz do Cliente

Visão Geral

Este guia apresenta diferentes estratégias para deploy da aplicação Voz do Cliente em ambiente de produção, incluindo deploy tradicional com systemd, containerização com Docker e configuração de proxy reverso com Nginx.

Deploy Tradicional (Systemd + Nginx)

1. Preparação do Servidor

Requisitos do Sistema

```
# Atualizar sistema
sudo apt update && sudo apt upgrade -y

# Instalar dependências essenciais
sudo apt install -y curl wget git build-essential

# Instalar Node.js 18+
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt-get install -y nodejs

# Instalar PostgreSQL
sudo apt install -y postgresql postgresql-contrib

# Instalar Nginx
sudo apt install -y nginx

# Instalar PM2 globalmente
sudo npm install -g pm2
```

2. Configurar Usuário da Aplicação

```
# Criar usuário dedicado
sudo adduser --system --group --home /opt/voz-cliente voc-app

# Criar diretórios necessários
sudo mkdir -p /opt/voz-cliente/{app,logs,backup}
sudo chown -R voc-app:voc-app /opt/voz-cliente
```

3. Deploy da Aplicação

```
# Clonar repositório
sudo -u voc-app git clone <URL_REPOSITORIO> /opt/voz-cliente/app
cd /opt/voz-cliente/app/app

# Instalar dependências
sudo -u voc-app npm ci --only=production

# Configurar variáveis de ambiente
sudo -u voc-app cp .env.example .env
sudo -u voc-app nano .env
```

Configurar .env para produção:

```
DATABASE_URL="postgresql://voc_user:senha_segura@localhost:5432/voz_cliente_prod"
NEXTAUTH_URL="https://voz-cliente.empresa.com"
NEXTAUTH_SECRET="chave_secreta_muito_longa_e_aleatoria"
NODE_ENV="production"
TWITTER_BEARER_TOKEN="seu_bearer_token"
REDDIT_CLIENT_ID="seu_client_id"
REDDIT_CLIENT_SECRET="seu_client_secret"
```

```
# Gerar cliente Prisma e executar migrações
sudo -u voc-app npx prisma generate
sudo -u voc-app npx prisma db push

# Compilar aplicação
sudo -u voc-app npm run build

# Testar aplicação
sudo -u voc-app npm start
```

4. Configurar Systemd Service

```
# Criar arquivo de serviço
sudo nano /etc/systemd/system/voz-cliente.service
```

```
[Unit]
Description=Voz do Cliente - Sistema de Análise de Sentimento
After=network.target postgresql.service

[Service]
Type=simple
User=voc-app
Group=voc-app
WorkingDirectory=/opt/voz-cliente/app/app
ExecStart=/usr/bin/npm start
Restart=always
RestartSec=10
Environment=NODE_ENV=production
Environment=PATH=/usr/bin:/usr/local/bin
EnvironmentFile=/opt/voz-cliente/app/app/.env

# Logs
StandardOutput=append:/opt/voz-cliente/logs/app.log
StandardError=append:/opt/voz-cliente/logs/error.log

# Security
NoNewPrivileges=true
PrivateTmp=true
ProtectSystem=strict
ProtectHome=true
ReadWritePaths=/opt/voz-cliente

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

```
# Habilitar e iniciar serviço
sudo systemctl daemon-reload
sudo systemctl enable voz-cliente
sudo systemctl start voz-cliente

# Verificar status
sudo systemctl status voz-cliente
```

5. Configurar Nginx

```
# Criar configuração do site
sudo nano /etc/nginx/sites-available/voz-cliente
```

```
# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name voz-cliente.empresa.com;
    return 301 https://$host$request_uri;
}

# HTTPS Configuration
server {
    listen 443 ssl http2;
    server_name voz-cliente.empresa.com;

    # SSL Configuration
    ssl_certificate /etc/letsencrypt/live/voz-cliente.empresa.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/voz-cliente.empresa.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-
GCM-SHA384:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # Security Headers
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

    # Gzip Compression
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_types text/plain text/css text/xml text/javascript application/javascript ap-
plication/xml+rss application/json;

    # Proxy to Next.js
    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        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_cache_bypass $http_upgrade;
        proxy_read_timeout 86400;
    }

    # Static files caching
    location /_next/static/ {
        proxy_pass http://localhost:3000;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Favicon and robots
    location = /favicon.ico {
```

```
        proxy_pass http://localhost:3000;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    location = /robots.txt {
        proxy_pass http://localhost:3000;
        expires 1d;
    }

    # Security - Block access to sensitive files
    location ~ /\. {
        deny all;
    }

    location ~ /(\.env|\.git|package\.json|yarn\.lock) {
        deny all;
    }
}
```

```
# Habilitar site
sudo ln -s /etc/nginx/sites-available/voz-cliente /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

6. Configurar SSL com Let's Encrypt

```
# Instalar Certbot
sudo apt install -y certbot python3-certbot-nginx

# Obter certificado SSL
sudo certbot --nginx -d voz-cliente.empresa.com

# Configurar renovação automática
sudo crontab -e
```

```
# Renovar certificados SSL automaticamente
0 12 * * * /usr/bin/certbot renew --quiet
```

Deploy com Docker

1. Criar Dockerfile

```
# Criar Dockerfile na raiz do projeto app/
nano /opt/voz-cliente/app/app/Dockerfile
```

```
# Multi-stage build
FROM node:18-alpine AS base

# Install dependencies only when needed
FROM base AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app

# Install dependencies based on the preferred package manager
COPY package.json package-lock.json* ./
RUN npm ci --only=production

# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

# Generate Prisma client
RUN npx prisma generate

# Build application
RUN npm run build

# Production image, copy all the files and run next
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

# Copy built application
COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static

# Copy Prisma files
COPY --from=builder /app/prisma ./prisma
COPY --from=builder /app/node_modules/.prisma ./node_modules/.prisma

USER nextjs

EXPOSE 3000

ENV PORT 3000
ENV HOSTNAME "0.0.0.0"

CMD ["node", "server.js"]
```

2. Criar docker-compose.yml

```yaml
version: '3.8'

services:
  app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: voz-cliente-app
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=postgresql://voc_user:senha_segura@db:5432/voz_cliente_prod
      - NEXTAUTH_URL=https://voz-cliente.empresa.com
      - NEXTAUTH_SECRET=${NEXTAUTH_SECRET}
      - TWITTER_BEARER_TOKEN=${TWITTER_BEARER_TOKEN}
      - REDDIT_CLIENT_ID=${REDDIT_CLIENT_ID}
      - REDDIT_CLIENT_SECRET=${REDDIT_CLIENT_SECRET}
      - NODE_ENV=production
    depends_on:
      - db
    networks:
      - voz-cliente-network
    volumes:
      - app-logs:/app/logs

  db:
    image: postgres:15-alpine
    container_name: voz-cliente-db
    restart: unless-stopped
    environment:
      - POSTGRES_DB=voz_cliente_prod
      - POSTGRES_USER=voc_user
      - POSTGRES_PASSWORD=senha_segura
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./backup:/backup
    ports:
      - "5432:5432"
    networks:
      - voz-cliente-network

  nginx:
    image: nginx:alpine
    container_name: voz-cliente-nginx
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
      - ./ssl:/etc/ssl/certs:ro
      - /etc/letsencrypt:/etc/letsencrypt:ro
    depends_on:
      - app
    networks:
      - voz-cliente-network
```

```
volumes:
  postgres_data:
  app-logs:

networks:
  voz-cliente-network:
    driver: bridge
```

3. Deploy com Docker

```
# Criar arquivo de variáveis de ambiente
nano .env.docker
```

```
NEXTAUTH_SECRET=sua_chave_secreta_muito_longa
TWITTER_BEARER_TOKEN=seu_bearer_token
REDDIT_CLIENT_ID=seu_client_id
REDDIT_CLIENT_SECRET=seu_client_secret
```

```
# Executar deploy
docker-compose --env-file .env.docker up -d

# Verificar logs
docker-compose logs -f app

# Executar migrações
docker-compose exec app npx prisma db push
```

Configuração de Monitoramento

1. Configurar Logs

```
# Criar configuração de logrotate
sudo nano /etc/logrotate.d/voz-cliente
```

```
/opt/voz-cliente/logs/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 voc-app voc-app
    postrotate
        systemctl reload voz-cliente
    endscript
}
```

2. Monitoramento com PM2 (Alternativa ao Systemd)

```
# Instalar PM2
sudo npm install -g pm2

# Criar arquivo de configuração
sudo -u voc-app nano /opt/voz-cliente/app/app/ecosystem.config.js
```

```
module.exports = {
  apps: [{
    name: 'voz-cliente',
    script: 'npm',
    args: 'start',
    cwd: '/opt/voz-cliente/app/app',
    instances: 'max',
    exec_mode: 'cluster',
    env: {
      NODE_ENV: 'production',
      PORT: 3000
    },
    error_file: '/opt/voz-cliente/logs/error.log',
    out_file: '/opt/voz-cliente/logs/app.log',
    log_file: '/opt/voz-cliente/logs/combined.log',
    time: true
  }]
};
```

```
# Iniciar com PM2
sudo -u voc-app pm2 start ecosystem.config.js
sudo -u voc-app pm2 save
sudo pm2 startup
```

Backup e Recuperação

1. Script de Backup Automatizado

```
sudo nano /usr/local/bin/backup-voz-cliente.sh
```

```
#!/bin/bash

# Configurações
BACKUP_DIR="/opt/voz-cliente/backup"
DATE=$(date +%Y%m%d_%H%M%S)
DB_NAME="voz_cliente_prod"
DB_USER="voc_user"
APP_DIR="/opt/voz-cliente/app"

# Criar diretório de backup
mkdir -p $BACKUP_DIR

# Backup do banco de dados
echo "Iniciando backup do banco de dados..."
pg_dump -h localhost -U $DB_USER -d $DB_NAME | gzip > $BACKUP_DIR/db_backup_$DATE.sql.g
z

# Backup dos arquivos da aplicação
echo "Iniciando backup dos arquivos..."
tar -czf $BACKUP_DIR/app_backup_$DATE.tar.gz -C $APP_DIR .

# Backup das configurações do sistema
tar -czf $BACKUP_DIR/config_backup_$DATE.tar.gz \
    /etc/nginx/sites-available/voz-cliente \
    /etc/systemd/system/voz-cliente.service \
    /etc/logrotate.d/voz-cliente

# Limpar backups antigos (manter últimos 7 dias)
find $BACKUP_DIR -name "*.gz" -mtime +7 -delete

echo "Backup concluído: $DATE"
```

```
# Tornar executável e configurar cron
sudo chmod +x /usr/local/bin/backup-voz-cliente.sh
sudo crontab -e
```

```
# Backup diário às 2:00 AM
0 2 * * * /usr/local/bin/backup-voz-cliente.sh >> /var/log/backup-voz-cliente.log 2>&1
```

Atualizações e Manutenção

1. Script de Deploy Automatizado

```
sudo nano /usr/local/bin/deploy-voz-cliente.sh
```

```
#!/bin/bash

APP_DIR="/opt/voz-cliente/app"
BACKUP_DIR="/opt/voz-cliente/backup"
DATE=$(date +%Y%m%d_%H%M%S)

echo "Iniciando deploy..."

# Backup antes da atualização
echo "Criando backup..."
/usr/local/bin/backup-voz-cliente.sh

# Parar aplicação
echo "Parando aplicação..."
sudo systemctl stop voz-cliente

# Atualizar código
echo "Atualizando código..."
cd $APP_DIR
sudo -u voc-app git pull origin main

# Instalar dependências
echo "Instalando dependências..."
cd $APP_DIR/app
sudo -u voc-app npm ci --only=production

# Executar migrações
echo "Executando migrações..."
sudo -u voc-app npx prisma generate
sudo -u voc-app npx prisma db push

# Compilar aplicação
echo "Compilando aplicação..."
sudo -u voc-app npm run build

# Iniciar aplicação
echo "Iniciando aplicação..."
sudo systemctl start voz-cliente

# Verificar status
sleep 5
sudo systemctl status voz-cliente

echo "Deploy concluído!"
```

Verificação do Deploy

1. Testes de Funcionalidade

```
# Verificar se a aplicação está respondendo
curl -I https://voz-cliente.empresa.com

# Testar API
curl -X POST https://voz-cliente.empresa.com/api/analyze \
  -H "Content-Type: application/json" \
  -d '{"companyName": "Teste"}'

# Verificar logs
sudo journalctl -u voz-cliente -f
```

2. Monitoramento Contínuo

```
# Verificar status dos serviços
sudo systemctl status voz-cliente
sudo systemctl status nginx
sudo systemctl status postgresql

# Verificar uso de recursos
htop
df -h
free -h
```

Próximos Passos

Após o deploy bem-sucedido:

1. Configure monitoramento adicional (Prometheus, Grafana)
2. Implemente alertas para falhas
3. Configure backup offsite
4. Documente procedimentos de rollback
5. Treine a equipe nos procedimentos de manutenção

Suporte

Para problemas de deploy:

1. Consulte Troubleshooting (./TROUBLESHOOTING.md)
2. Verifique logs da aplicação e sistema
3. Entre em contato com a equipe de desenvolvimento