🚀 Guia de Deploy - Servidor Miltec com Docker e Portainer

Data: 12 de novembro de 2025
Projeto: Sistema de Licenciamento Ambiental - Frontend
Origem: Bolt.new (desenvolvimento)
Destino: Servidor Miltec via Docker + Portainer

📋 Índice

🌐 Contexto: Migração do Bolt.new

O que é o Bolt.new?

O **Bolt.new** é uma plataforma de desenvolvimento rápido que permite criar e testar aplicações web diretamente no navegador. Ele é excelente para prototipagem e desenvolvimento, mas não é adequado para produção de longo prazo.

Por que migrar para servidor próprio?

- ☑ **Controle total:** Gerenciamento completo da infraestrutura
- ☑ **Performance:** Recursos dedicados ao invés de compartilhados
- ☑ **Segurança:** Dados dentro da infraestrutura da empresa
- ☑ **Custo:** Elimina custos recorrentes do Bolt.new
- ☑ **Escalabilidade:** Capacidade de crescer conforme necessário
- ☑ **Compliance:** Atende requisitos de segurança corporativa

Diferenças principais

Aspecto	Bolt.new	Servidor Miltec
Hospedagem	Cloud (Bolt)	On-premise/Cloud próprio
Deploy	Automático	Manual via Portainer
Variáveis de ambiente	Interface Bolt	Docker/Portainer

Aspecto	Bolt.new	Servidor Miltec
Domínio	*.bolt.new	Domínio próprio Miltec
SSL/HTTPS	Automático	Configuração manual
Backup	Limitado	Controle total

🔧 Pré-requisitos

No Servidor Miltec:

- ☑ Docker instalado (versão 20.10+)
- ☑ Portainer instalado e acessível
- ☑ Acesso SSH ao servidor (para verificações)
- ☑ Porta 80 ou outra porta disponível para o frontend
- ☑ Acesso à rede onde está o backend FastAPI (se houver)

Informações Necessárias:

- **URL do Supabase:** `https://seu-projeto.supabase.co`
- **Chave Anon do Supabase:** `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...`
- **URL da API Backend:** `http://ip-servidor:porta` ou `https://api.miltec.com`
- **Porta desejada para o frontend:** Ex: 80, 8080, 3000, etc.

📦 Arquivos Criados

Os seguintes arquivos foram criados na raiz do projeto:

```
github-dzabccvf/
├── Dockerfile              # Build multi-stage com Node + NGINX
├── docker-compose.yml      # Orquestração do container
├── .dockerignore           # Arquivos excluídos do build
├── nginx.conf              # Configuração do servidor web
├── docker-entrypoint.sh    # Script de inicialização
└── Docs/
    └── DEPLOY_MILTEC.md    # Este guia
```

🔀 Preparação: Exportar do Bolt.new

Opção 1: Baixar o Projeto do Bolt.new (Recomendado)

1. **No Bolt.new, acesse seu projeto**
2. **Baixe todos os arquivos:**
    - Clique no botão de menu ou opções

- Procure por "Download" ou "Export"
- Baixe como ZIP

3. **Extraia os arquivos:**

```
# No Windows PowerShell
Expand-Archive -Path "projeto-bolt.zip" -DestinationPath
"C:\projetos\licenciamento-frontend"
cd C:\projetos\licenciamento-frontend
```

Opção 2: Usar o Projeto Atual (Você já tem!)

Se você já tem o código localmente em `d:\code\python\github-dzabccvf`, você já está pronto! ☑

Opção 3: Clonar do GitHub (Se já commitou)

```
git clone https://github.com/wmiltecti/github-dzabccvf.git
cd github-dzabccvf
```

Verificar Estrutura do Projeto

Certifique-se de que você tem:

```
☑ package.json          # Dependências do projeto
☑ vite.config.ts        # Configuração do Vite
☑ src/                  # Código fonte
☑ public/               # Arquivos públicos
☑ index.html            # HTML principal
☑ .env.example          # Exemplo de variáveis de ambiente
```

Instalar Dependências (Primeira vez)

```
# Instalar Node.js (se não tiver)
# Baixe de: https://nodejs.org/

# Instalar dependências do projeto
npm install
```

Testar Localmente

```
# Criar arquivo .env com suas credenciais
Copy-Item .env.example .env
```

```
# Editar .env e adicionar suas credenciais
notepad .env

# Executar em modo desenvolvimento
npm run dev

# Acessar http://localhost:5173
```

Se funcionar localmente, está pronto para deploy! 🎉

🔨 Preparação: Build Local para Teste (Opcional)

Passo 1: Testar Build de Produção

Antes de fazer deploy no servidor, teste localmente:

```
# Build da imagem
docker build -t licenciamento-frontend:test .

# Executar container de teste
docker run -d \
  -p 8080:80 \
  -e VITE_SUPABASE_URL="sua_url_supabase" \
  -e VITE_SUPABASE_ANON_KEY="sua_chave" \
  -e VITE_API_BASE_URL="http://localhost:8000" \
  --name teste-frontend \
  licenciamento-frontend:test

# Acessar http://localhost:8080
# Parar e remover container de teste
docker stop teste-frontend
docker rm teste-frontend
```

Passo 2: Preparar Arquivos para Deploy

Você tem **3 opções** para transferir o projeto para o servidor Miltec:

Opção A: Via Git (Mais profissional - Recomendado)

```
# Se ainda não tem repositório Git
git init
git add .
git commit -m "Adiciona configuração Docker para deploy Miltec"

# Configurar remote no GitHub/GitLab
git remote add origin https://github.com/wmiltecti/github-dzabccvf.git
git push -u origin main
```

☑ **Vantagem:** Fácil atualização posterior
☑ **Ideal para:** Trabalho em equipe e versionamento

Opção B: Comprimir e transferir via SCP/FTP

```
# Remover pastas grandes desnecessárias
Remove-Item -Recurse -Force node_modules, dist -ErrorAction SilentlyContinue

# Criar arquivo ZIP
Compress-Archive -Path * -DestinationPath licenciamento-frontend.zip -Force

# Transferir via SCP (requer ferramentas SSH no Windows)
# scp licenciamento-frontend.zip usuario@servidor-miltec:/opt/
```

☑ **Vantagem:** Simples e direto
☑ **Ideal para:** Deploy único rápido

Opção C: Via Portainer Upload (Mais simples)

1. Crie o ZIP (sem node_modules e dist)
2. Acesse Portainer
3. Use a interface de upload de arquivos
4. Extraia no servidor

☑ **Vantagem:** Não precisa de SSH/FTP
☑ **Ideal para:** Quem tem apenas acesso ao Portainer

🌐 Deploy via Portainer

Opção A: Deploy via Stack (Recomendado)

1. **Acesse o Portainer**
   - URL: `https://portainer.miltec.com` (ou IP do servidor)
   - Faça login com suas credenciais
2. **Navegue até Stacks**
   - Menu lateral → **Stacks**
   - Clique em **"+ Add stack"**
3. **Configure a Stack**
   - **Name:** `licenciamento-ambiental`
   - **Build method:** Escolha uma das opções:

Opção A1: Via Web editor (Copiar/Colar)

Cole o conteúdo do `docker-compose.yml`:

```
version: '3.8'

services:
  frontend:
    container_name: licenciamento-ambiental-frontend
    build:
      context: .
      dockerfile: Dockerfile
    image: licenciamento-ambiental-frontend:latest
    ports:
      - "80:80"  # Ajuste a porta se necessário
    environment:
      - VITE_SUPABASE_URL=${VITE_SUPABASE_URL}
      - VITE_SUPABASE_ANON_KEY=${VITE_SUPABASE_ANON_KEY}
      - VITE_API_BASE_URL=${VITE_API_BASE_URL}
    restart: unless-stopped
    networks:
      - licenciamento-network

networks:
  licenciamento-network:
    driver: bridge
```

⚠ **IMPORTANTE:** Como o Portainer precisa fazer o build, você precisa enviar os arquivos do projeto para o servidor primeiro!

Opção A2: Via Git Repository (Melhor para CI/CD)

- **Repository URL:** `https://github.com/wmiltecti/github-dzabccvf`
- **Repository reference:** `refs/heads/main`
- **Compose path:** `docker-compose.yml`
- **Authentication:** Configure se o repositório for privado

4. **Configure Environment Variables**

Na seção **Environment variables**, clique em **"+ Add environment variable"** para cada:

Nome	Valor	Descrição
`VITE_SUPABASE_URL`	`https://xxxxx.supabase.co`	URL do projeto Supabase
`VITE_SUPABASE_ANON_KEY`	`eyJhbGciOiJI...`	Chave anônima do Supabase
`VITE_API_BASE_URL`	`http://IP:8000`	URL do backend FastAPI

5. **Deploy da Stack**
    - Clique em **"Deploy the stack"**
    - Aguarde o build e inicialização (pode levar 5-10 minutos)

Opção B: Deploy Manual via Dockerfile

Se você preferir fazer upload manual dos arquivos:

1. **Conecte ao servidor via SSH**

```
ssh usuario@servidor-miltec.com
```

2. **Crie diretório do projeto**

```
mkdir -p /opt/licenciamento-frontend
cd /opt/licenciamento-frontend
```

3. **Transfira os arquivos**

Use SCP, FTP ou Git:

```
# Via SCP (do seu computador Windows)
scp -r . usuario@servidor:/opt/licenciamento-frontend/

# OU via Git (no servidor)
git clone https://github.com/wmiltecti/github-dzabccvf.git .
```

4. **Build da imagem no servidor**

```
cd /opt/licenciamento-frontend
docker build -t licenciamento-frontend:latest .
```

5. **No Portainer, crie Container manualmente**
   - Menu: **Containers** → **+ Add container**
   - **Name:** `licenciamento-ambiental-frontend`
   - **Image:** `licenciamento-frontend:latest`
   - **Port mapping:** `80:80` (ou porta desejada)
   - **Environment variables:** Adicione as 3 variáveis acima
   - **Restart policy:** `Unless stopped`
   - Clique em **"Deploy the container"**

🔐 Configuração de Variáveis de Ambiente

Variáveis Obrigatórias

```

```

```
# Supabase
VITE_SUPABASE_URL=https://seu-projeto.supabase.co
VITE_SUPABASE_ANON_KEY=sua_chave_anon_publica

# Backend API
VITE_API_BASE_URL=http://ip-servidor-backend:8000
```

Onde Obter as Credenciais do Supabase

1. Acesse https://supabase.com/dashboard
2. Selecione seu projeto
3. Vá em **Settings** → **API**
4. Copie:
   - **Project URL** → `VITE_SUPABASE_URL`
   - **anon public** key → `VITE_SUPABASE_ANON_KEY`

Configuração da URL do Backend

- **Se backend estiver no mesmo servidor:**

  ```
  VITE_API_BASE_URL=http://localhost:8000
  ```

- **Se backend estiver em outro servidor:**

  ```
  VITE_API_BASE_URL=http://192.168.1.100:8000
  ```

- **Se backend tiver domínio:**

  ```
  VITE_API_BASE_URL=https://api.miltec.com
  ```

☑ Verificação e Testes

1. Verificar Status do Container

No Portainer ou via SSH:

```
# Listar containers
docker ps

# Ver logs do container
docker logs licenciamento-ambiental-frontend
```

```
# Ver logs em tempo real
docker logs -f licenciamento-ambiental-frontend
```

2. Acessar a Aplicação

Abra o navegador e acesse:

```
http://IP-DO-SERVIDOR
# OU
http://IP-DO-SERVIDOR:PORTA
```

3. Verificar APIs

Abra o DevTools do navegador (F12) e verifique:

- **Console:** Não deve ter erros de JavaScript
- **Network:** Requisições para Supabase e Backend devem retornar 200

4. Testar Funcionalidades

- ☐ Login/Autenticação funciona
- ☐ Dados carregam corretamente
- ☐ Formulários enviam dados
- ☐ Mapa geográfico renderiza (se aplicável)
- ☐ Upload de arquivos funciona

🔧 Troubleshooting

Problema 1: Container não inicia

Sintomas: Container aparece como "Exited" no Portainer

Solução:

```
# Ver logs de erro
docker logs licenciamento-ambiental-frontend

# Verificar se as portas estão disponíveis
sudo netstat -tulpn | grep :80

# Se porta 80 estiver em uso, mude no docker-compose.yml
ports:
  - "8080:80"  # Usa porta 8080 externa
```

Problema 2: Erro 404 ao acessar

Sintomas: "404 Not Found" ao acessar rotas da aplicação

Causa: Configuração incorreta do NGINX para SPA

Solução: Verificar se o `nginx.conf` tem:

```
location / {
    try_files $uri $uri/ /index.html;
}
```

Problema 3: Variáveis de ambiente não funcionam

Sintomas: Aplicação não conecta ao Supabase ou Backend

Solução:

```
# Verificar variáveis dentro do container
docker exec licenciamento-ambiental-frontend env | grep VITE

# Se não aparecerem, recrie o container com as variáveis
docker stop licenciamento-ambiental-frontend
docker rm licenciamento-ambiental-frontend

# No Portainer, edite o container e adicione as variáveis
# OU via comando:
docker run -d \
  --name licenciamento-ambiental-frontend \
  -p 80:80 \
  -e VITE_SUPABASE_URL="sua_url" \
  -e VITE_SUPABASE_ANON_KEY="sua_chave" \
  -e VITE_API_BASE_URL="http://backend:8000" \
  --restart unless-stopped \
  licenciamento-frontend:latest
```

Problema 4: Build falha no Portainer

Sintomas: Erro durante `npm ci` ou `npm run build`

Causas comuns:

- Falta de memória no servidor
- Arquivos corrompidos
- Dependências incompatíveis

Solução:

```
# Fazer build manual no servidor
cd /opt/licenciamento-frontend
```

```
docker build -t licenciamento-frontend:latest .

# Se falhar por falta de memória, aumente swap
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile

# Tente o build novamente
docker build -t licenciamento-frontend:latest .
```

Problema 5: Conexão com backend falha

Sintomas: Erros de CORS ou conexão recusada

Solução:

1. **Verificar se backend está acessível:**

```
curl http://IP-BACKEND:8000/api/health
```

2. **Configurar CORS no backend FastAPI:**

```
from fastapi.middleware.cors import CORSMiddleware

app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://IP-FRONTEND", "http://IP-FRONTEND:80"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

3. **Se backend e frontend estiverem na mesma rede Docker:**

```
# docker-compose.yml
services:
  frontend:
    # ...
    environment:
      - VITE_API_BASE_URL=http://backend:8000
    networks:
      - licenciamento-network

  backend:
    image: fastapi-backend:latest
```

```
    networks:
      - licenciamento-network
```

🔄 Manutenção e Atualização

Atualizar a Aplicação

1. **Via Git (se configurou repositório):**
   - No Portainer, vá em **Stacks**
   - Selecione `licenciamento-ambiental`
   - Clique em **Pull and redeploy**

2. **Via Build Manual:**

```
# No servidor
cd /opt/licenciamento-frontend
git pull origin main
docker build -t licenciamento-frontend:latest .
docker stop licenciamento-ambiental-frontend
docker rm licenciamento-ambiental-frontend

# No Portainer, recrie o container com a nova imagem
```

3. **Via Portainer (Upload de nova versão):**
   - Pare o container
   - Faça upload dos novos arquivos via SCP
   - Faça rebuild da imagem
   - Inicie o container

Backup

```
# Backup da imagem Docker
docker save licenciamento-frontend:latest | gzip > licenciamento-frontend-
backup.tar.gz

# Backup das configurações
cp docker-compose.yml docker-compose.yml.backup
```

Monitoramento

1. **Logs do NGINX:**

```
docker exec licenciamento-ambiental-frontend tail -f /var/log/nginx/access.log
docker exec licenciamento-ambiental-frontend tail -f /var/log/nginx/error.log
```

2. **Uso de recursos:**

```
docker stats licenciamento-ambiental-frontend
```

3. **Health Check:** Adicione ao `docker-compose.yml`:

```
services:
  frontend:
    healthcheck:
      test: ["CMD", "wget", "--quiet", "--tries=1", "--spider",
"http://localhost/"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s
```

📞 Suporte

Informações Úteis

- **Repositório:** https://github.com/wmiltecti/github-dzabccvf
- **Documentação adicional:** `/Docs`
- **Logs:** `/var/log/nginx/` (dentro do container)

Comandos Rápidos de Debug

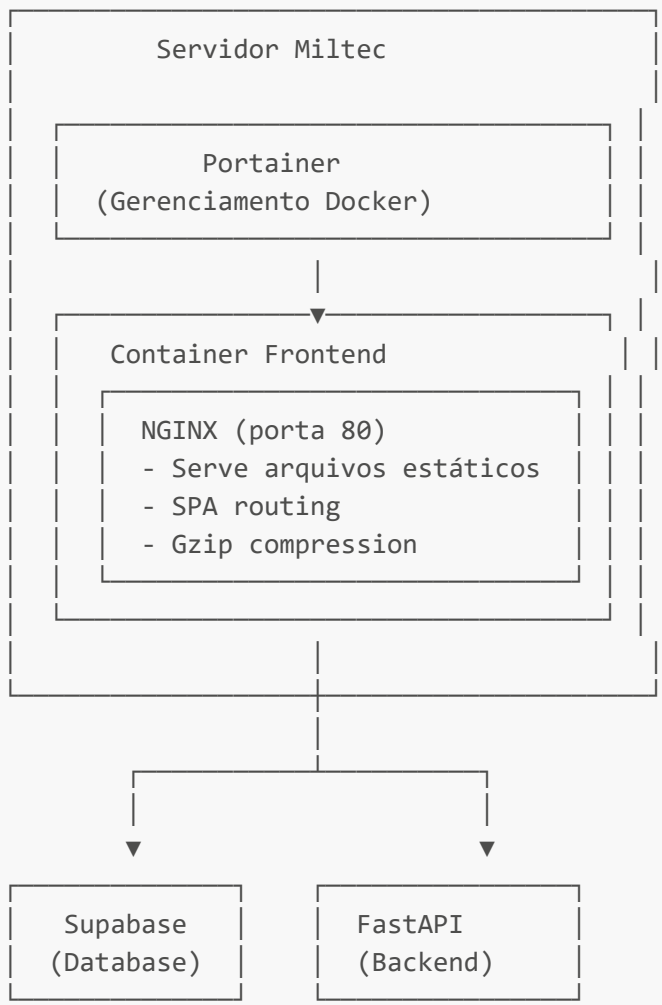
```
# Entrar no container
docker exec -it licenciamento-ambiental-frontend /bin/sh

# Ver arquivos compilados
ls -la /usr/share/nginx/html/

# Testar configuração do NGINX
docker exec licenciamento-ambiental-frontend nginx -t

# Reiniciar NGINX
docker exec licenciamento-ambiental-frontend nginx -s reload

# Ver processos rodando
docker exec licenciamento-ambiental-frontend ps aux
```

🎯 Checklist Final

Antes de considerar o deploy completo:

- [ ] Container está rodando (status "Up")
- [ ] Aplicação acessível via navegador
- [ ] Login funciona corretamente
- [ ] Dados carregam do Supabase
- [ ] API backend responde
- [ ] Não há erros no console do navegador
- [ ] Todas as rotas funcionam (SPA routing)
- [ ] Arquivos estáticos carregam (imagens, CSS, JS)
- [ ] Performance é aceitável
- [ ] Logs não mostram erros críticos
- [ ] Backup inicial foi feito
- [ ] Documentação foi atualizada

Arquitetura do Deploy



Criado em: 12/11/2025
Última atualização: 12/11/2025

Versão: 1.0.0