Guia de Instalação em Produção do Sistema de Agendamento Online

Este documento consolida os guias de instalação e configuração para o sistema de agendamento online em ambientes de produção, tanto para Windows Server 2022 quanto para Linux (Ubuntu Server). Ele é projetado para usuários iniciantes, fornecendo instruções detalhadas e passo a passo para cada tecnologia.

Introdução Geral

O sistema de agendamento online é uma solução robusta para conectar profissionais e clientes, permitindo o gerenciamento eficiente de agendas, serviços e agendamentos. Para garantir que o sistema funcione de forma otimizada, segura e escalável em um ambiente de produção, é essencial seguir um processo de instalação e configuração cuidadoso. Este guia aborda todas as etapas necessárias, desde a instalação dos pré-requisitos até a configuração de serviços essenciais como banco de dados, servidores web e integrações de notificação.

Cada seção é dividida por sistema operacional para facilitar a navegação e a aplicação das instruções. Recomenda-se ler todo o guia antes de iniciar o processo para ter uma visão geral dos passos envolvidos.

Guia de Instalação em Produção - Windows Server 2022

Guia de Instalação em Produção - Windows Server 2022

Este guia detalha o processo de instalação e configuração das tecnologias necessárias para o sistema de agendamento online em um ambiente de produção no Windows Server 2022. Ele é destinado a usuários iniciantes, com instruções passo a passo.

1. Introdução

Para garantir a estabilidade, segurança e performance do sistema de agendamento em um ambiente de produção, é fundamental configurar corretamente o servidor. Este guia abordará a instalação de:

- **Python 3.11+:** Para o backend da aplicação Flask.
- **PostgreSQL:** Como banco de dados robusto e escalável.
- **Gunicorn:** Servidor WSGI para o Flask (execução em produção).
- **Nginx:** Servidor web reverso para servir o frontend e proxy reverso para o backend.
- **Node.js e pnpm:** Para o desenvolvimento e build do frontend React.
- **Frontend React:** A aplicação web que interage com os usuários.
- **n8n e Evolution API:** Para o envio de notificações via WhatsApp.

2. Pré-requisitos

Antes de iniciar a instalação, certifique-se de que seu Windows Server 2022 possui:

- **Acesso de Administrador:** Você precisará de permissões de administrador para instalar softwares e configurar serviços.
- **Conexão com a Internet:** Para baixar os pacotes e dependências necessárias.
- **Recursos Mínimos:** Recomenda-se pelo menos 4GB de RAM e 2 vCPUs para um ambiente de produção pequeno a médio. Para ambientes maiores, ajuste conforme a demanda.
- **Firewall:** Certifique-se de que as portas necessárias (80, 443 para HTTP/HTTPS, 5000 para o backend, 5432 para PostgreSQL, 5678 para n8n, 8080 para Evolution API) estejam abertas no firewall do Windows e em qualquer firewall de rede externo.

3. Instalação do Python e Ambiente Virtual

O Python é a linguagem de programação utilizada pelo backend da aplicação. É crucial instalar a versão correta e configurar um ambiente virtual para isolar as dependências do projeto.

3.1. Baixar e Instalar o Python

1. **Acesse o site oficial do Python:** Abra seu navegador e vá para https://www.python.org/downloads/windows/ [1].
2. **Baixe o instalador:** Procure pela versão mais recente do Python 3.11.x (por exemplo, Python 3.11.9) e baixe o instalador executável de 64 bits (`Windows installer (64-bit)`).
3. **Execute o instalador:** Localize o arquivo `.exe` baixado e clique duas vezes para executá-lo.
4. **Importante - Marque a opção "Add Python to PATH":** Na primeira tela do instalador, **certifique-se de marcar a caixa de seleção "Add Python.exe to PATH"**. Isso permitirá que você execute comandos Python de qualquer diretório no Prompt de Comando ou PowerShell. [2]
5. **Instalação Personalizada (Recomendado):** Escolha "Customize installation" para ter mais controle. Na próxima tela, mantenha todas as opções padrão

marcadas.
6. **Opções Avançadas:** Na tela de "Advanced Options", você pode escolher o diretório de instalação. O padrão (`C:\Program Files\Python311`) geralmente é adequado. Marque "Install for all users".
7. **Conclua a instalação:** Clique em "Install" e aguarde o processo ser concluído. Pode levar alguns minutos.

3.2. Verificar a Instalação do Python

1. **Abra o Prompt de Comando ou PowerShell:** Pressione `Win + R`, digite `cmd` ou `powershell` e pressione `Enter`.
2. **Verifique a versão do Python:** Digite o seguinte comando e pressione `Enter`:
   ```bash
   python --version
   ```
   Você deverá ver a versão do Python instalada (ex: `Python 3.11.9`). Se você vir um erro, verifique se a opção "Add Python to PATH" foi marcada durante a instalação ou adicione-o manualmente às variáveis de ambiente do sistema.
3. **Verifique a versão do pip:** O `pip` é o gerenciador de pacotes do Python. Digite:
   ```bash
   pip --version
   ```
   Você deverá ver a versão do pip instalada.

3.3. Criar e Ativar um Ambiente Virtual

Um ambiente virtual é uma ferramenta que permite isolar as dependências de um projeto Python de outros projetos. Isso evita conflitos de pacotes e mantém seu ambiente de desenvolvimento limpo.

1. **Navegue até o diretório do seu projeto:** Use o Prompt de Comando ou PowerShell para ir até a pasta onde você clonou o repositório do backend do sistema de agendamento (ex: `C:\agendamento-online`).
   ```bash
   cd C:\caminho\para\seu\projeto\agendamento-online
   ```
2. **Crie o ambiente virtual:** Digite o seguinte comando:
   ```bash
   python -m venv venv
   ```
   Isso criará uma pasta chamada `venv` dentro do seu diretório de projeto, contendo os arquivos do ambiente virtual.
3. **Ative o ambiente virtual:** Para começar a usar o ambiente virtual, execute:
   ```bash
   .\venv\Scripts\activate
   ```
   Você notará que o nome `(venv)` aparecerá no início da linha de comando, indicando que o ambiente virtual está ativo. Todas as instalações de pacotes `pip` agora serão feitas dentro deste ambiente.

4. **Instale as dependências do projeto:** Com o ambiente virtual ativo, navegue até a pasta do backend (`agendamento-online`) e instale as dependências listadas no arquivo `requirements.txt`:
   ```bash
   pip install -r requirements.txt
   ```
   Este comando instalará todas as bibliotecas Python necessárias para o funcionamento do backend (Flask, SQLAlchemy, Gunicorn, etc.).

4. Instalação e Configuração do PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional robusto e de código aberto, ideal para ambientes de produção.

4.1. Baixar e Instalar o PostgreSQL

1. **Acesse o site oficial do PostgreSQL:** Vá para https://www.postgresql.org/download/windows/ [3].
2. **Baixe o instalador:** Clique no link para baixar o instalador para Windows (geralmente o "EDB installer"). Escolha a versão mais recente estável (ex: PostgreSQL 16.x).
3. **Execute o instalador:** Clique duas vezes no arquivo `.exe` baixado.
4. **Siga o assistente de instalação:**
   - **Installation Directory:** Mantenha o padrão ou escolha um local de sua preferência (ex: `C:\Program Files\PostgreSQL\16`).
   - **Select Components:** Certifique-se de que `PostgreSQL Server`, `pgAdmin 4` (ferramenta gráfica para gerenciar o banco de dados) e `Command Line Tools` estejam selecionados.
   - **Data Directory:** Mantenha o padrão ou escolha um local para armazenar os dados do banco de dados.
   - **Password for postgres superuser:** **Defina uma senha forte** para o usuário `postgres`. **Anote esta senha**, pois você precisará dela para acessar o banco de dados.
   - **Port:** Mantenha a porta padrão `5432`.
   - **Advanced Options:** Mantenha o padrão.
5. **Conclua a instalação:** Clique em "Next" e "Finish" para completar o processo.

4.2. Configurar o Banco de Dados para o Projeto

Após a instalação, você precisará criar um banco de dados e um usuário específico para a sua aplicação.

```markdown

## Guia de Instalação em Produção - Linux (Ubuntu Server)

---

```markdown
# Guia de Instalação em Produção - Linux (Ubuntu Server)

Este guia detalha o processo de instalação e configuração das tecnologias necessárias para o sistema de agendamento online em um ambiente de produção no Linux (Ubuntu Server). Ele é destinado a usuários iniciantes, com instruções passo a passo.

## 1. Introdução

Para garantir a estabilidade, segurança e performance do sistema de agendamento em um ambiente de produção, é fundamental configurar corretamente o servidor. Este guia abordará a instalação de:

- **Python 3.11+:** Para o backend da aplicação Flask.
- **PostgreSQL:** Como banco de dados robusto e escalável.
- **Gunicorn:** Servidor WSGI para o Flask (execução em produção).

```
- **Nginx:** Servidor web reverso para servir o frontend e proxy reverso para o backend.
- **Node.js e pnpm:** Para o desenvolvimento e build do frontend React.
- **Frontend React:** A aplicação web que interage com os usuários.
- **n8n e Evolution API:** Para o envio de notificações via WhatsApp.

2. Pré-requisitos

Antes de iniciar a instalação, certifique-se de que seu Ubuntu Server possui:

- **Acesso SSH:** Você precisará de acesso SSH ao servidor para executar os comandos.
- **Usuário com privilégios sudo:** Para instalar softwares e configurar serviços.
- **Conexão com a Internet:** Para baixar os pacotes e dependências necessárias.
- **Recursos Mínimos:** Recomenda-se pelo menos 4GB de RAM e 2 vCPUs para um ambiente de produção pequeno a médio. Para ambientes maiores, ajuste conforme a demanda.
- **Firewall (UFW):** Certifique-se de que as portas necessárias (80, 443 para HTTP/HTTPS, 5000 para o backend, 5432 para PostgreSQL, 5678 para n8n, 8080 para Evolution API) estejam abertas no firewall do Ubuntu (UFW).

3. Instalação do Python e Ambiente Virtual

O Python é a linguagem de programação utilizada pelo backend da aplicação. É crucial instalar a versão correta e configurar um ambiente virtual para isolar as dependências do projeto.

3.1. Atualizar o Sistema e Instalar Dependências Essenciais

É sempre uma boa prática atualizar o sistema antes de instalar novos pacotes.

```bash
sudo apt update
sudo apt upgrade -y
sudo apt install -y software-properties-common
```

3.2. Instalar Python 3.11

O Ubuntu 22.04 já vem com Python 3.10. Para instalar o Python 3.11, você pode usar o PPA (Personal Package Archive) `deadsnakes`.

```
 sudo add-apt-repository ppa:deadsnakes/ppa -y
sudo apt update
sudo apt install -y python3.11 python3.11-venv python3.11-dev
```

3.3. Criar e Ativar um Ambiente Virtual

Um ambiente virtual é uma ferramenta que permite isolar as dependências de um projeto Python de outros projetos. Isso evita conflitos de pacotes e mantém seu ambiente de desenvolvimento limpo.
```

1. **Navegue até o diretório do seu projeto:** Use o terminal para ir até a pasta onde você clonou o repositório do backend do sistema de agendamento (ex: `/var/www/agendamento-online`). `bash sudo mkdir -p /var/www/agendamento-online sudo chown -R $`USER:`$USER /var/www/agendamento-online cd /var/www/agendamento-online # Clone seu repositório aqui # git clone <your-repo-url> .`
2. **Crie o ambiente virtual:** Digite o seguinte comando: `bash python3.11 -m venv venv` Isso criará uma pasta chamada `venv` dentro do seu diretório de projeto, contendo os arquivos do ambiente virtual.
3. **Ative o ambiente virtual:** Para começar a usar o ambiente virtual, execute: `bash source venv/bin/activate` Você notará que o nome `(venv)` aparecerá no início da linha de comando, indicando que o ambiente virtual está ativo. Todas as instalações de pacotes `pip` agora serão feitas dentro deste ambiente.
4. **Instale as dependências do projeto:** Com o ambiente virtual ativo, instale as dependências listadas no arquivo `requirements.txt`: `bash pip install -r requirements.txt` Este comando instalará todas as bibliotecas Python necessárias para o funcionamento do backend (Flask, SQLAlchemy, Gunicorn, etc.).

4. Instalação e Configuração do PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional robusto e de código aberto, ideal para ambientes de produção.

4.1. Instalar PostgreSQL

```
sudo apt install -y postgresql postgresql-contrib
```

4.2. Configurar o Banco de Dados para o Projeto

Após a instalação, você precisará criar um banco de dados e um usuário específico para a sua aplicação.

1. **Acesse o shell do PostgreSQL como usuário `postgres`:** `bash sudo -i -u postgres psql`

2. **Crie um novo usuário (role) para a aplicação:** `sql CREATE USER agendamento_user WITH PASSWORD 'sua_senha_forte';` **Mude `sua_senha_forte` para uma senha segura e única.**
3. **Crie um novo banco de dados e defina o proprietário:** `sql CREATE DATABASE agendamento_db OWNER agendamento_user;`
4. **Conceda privilégios ao usuário no banco de dados:** `sql GRANT ALL PRIVILEGES ON DATABASE agendamento_db TO agendamento_user;`
5. **Saia do shell do PostgreSQL:** `sql \q exit`

4.3. Configurar a Conexão no Backend

No arquivo `.env` do seu projeto backend (`agendamento-online`), atualize a variável `DATABASE_URL` para apontar para o PostgreSQL:

```
DATABASE_URL=postgresql://agendamento_user:sua_senha_forte@localhost:5432/agendam
```

Substitua `agendamento_user`, `sua_senha_forte` e `agendamento_db` pelos valores que você definiu.

5. Instalação e Configuração do Gunicorn e Nginx

Em produção, o Flask não deve ser executado diretamente. O Gunicorn é um servidor WSGI que gerencia a execução do Flask, e o Nginx atuará como um proxy reverso, servindo arquivos estáticos e encaminhando requisições para o Gunicorn.

5.1. Instalar Gunicorn

O Gunicorn é um pacote Python, então ele deve ser instalado dentro do seu ambiente virtual do backend:

1. **Ative o ambiente virtual** (se ainda não estiver ativo): `bash cd /var/www/agendamento-online source venv/bin/activate`
2. **Instale o Gunicorn:** `bash pip install gunicorn`

5.2. Instalar Nginx

```
sudo apt install -y nginx
```

5.3. Configurar Gunicorn como Serviço Systemd

Para garantir que o Gunicorn inicie automaticamente e seja gerenciado pelo sistema, crie um serviço Systemd.

1. **Crie o arquivo de serviço:** `bash sudo nano /etc/systemd/system/agendamento.service`
2. **Adicione o seguinte conteúdo:** ```ini [Unit] Description=Gunicorn instance for Agendamento Online After=network.target

[Service] User=www-data Group=www-data WorkingDirectory=/var/www/agendamento-online ExecStart=/var/www/agendamento-online/venv/bin/gunicorn --workers 3 --bind unix:/var/www/agendamento-online/agendamento.sock src.main:app Restart=always

[Install] WantedBy=multi-user.target `` **Atenção:** - User=www-data e Group=www-data: O Gunicorn será executado com o usuário www-data (usuário padrão do Nginx). Certifique-se de que este usuário tenha permissões de leitura e execução na pasta do seu projeto. - ExecStart: Caminho completo para o executável gunicorn dentro do seu ambiente virtual e o comando para iniciar a aplicação Flask. - --workers 3: Número de processos de worker do Gunicorn. Ajuste conforme a capacidade do seu servidor. - --bind unix:/var/www/agendamento-online/agendamento.sock`: O Gunicorn se comunicará com o Nginx através de um socket Unix, que é mais eficiente para comunicação local.

1. **Salve e feche o arquivo** (`Ctrl+X`, `Y`, `Enter`).
2. **Habilite e inicie o serviço Gunicorn:** `bash sudo systemctl daemon-reload sudo systemctl start agendamento sudo systemctl enable agendamento`
3. **Verifique o status do serviço:** `bash sudo systemctl status agendamento` Você deverá ver que o serviço está `active (running)`.

5.4. Configurar Nginx como Proxy Reverso

Você precisará configurar o Nginx para: - Servir os arquivos estáticos do frontend (React). - Encaminhar as requisições da API (`/api`) para o Gunicorn (que estará rodando o Flask).

1. **Crie um novo arquivo de configuração para o seu site:** `bash sudo nano /etc/nginx/sites-available/agendamento`
2. **Adicione o seguinte conteúdo:** ```nginx server { listen 80; server_name seu_dominio_ou_ip; # Substitua pelo seu domínio ou IP do servidor

   location / { root /var/www/agendamento-frontend/dist; # Caminho para a pasta 'dist' do build do React index index.html; try_files $`uri` $uri/ /index.html; }

   location /api/ { include proxy_params; proxy_pass http://unix:/var/www/agendamento-online/agendamento.sock; } } `` `**Atenção:** -` server_name seu_dominio_ou_ip; `: Substitua pelo seu nome de domínio (ex:` meuagendamento.com `) ou pelo endereço IP público do seu servidor. -` root /var/www/agendamento-frontend/dist; `: Altere para o caminho real da pasta` dist `do seu frontend React (que será gerada após o build). -` proxy_pass http://unix:/var/www/agendamento-online/agendamento.sock;`: Aponta para o socket Unix do Gunicorn.
3. **Salve e feche o arquivo** (`Ctrl+X`, `Y`, `Enter`).
4. **Crie um link simbólico para habilitar o site:** `bash sudo ln -s /etc/nginx/sites-available/agendamento /etc/nginx/sites-enabled`
5. **Remova a configuração padrão do Nginx:** `bash sudo rm /etc/nginx/sites-enabled/default`
6. **Teste a configuração do Nginx:** `bash sudo nginx -t` Você deverá ver `syntax is ok` e `test is successful`.
7. **Reinicie o Nginx para aplicar as mudanças:** `bash sudo systemctl restart nginx`

5.5. Configurar Firewall (UFW)

Permita o tráfego HTTP e HTTPS no firewall.

```
sudo ufw allow 'Nginx HTTP'
sudo ufw allow 'Nginx HTTPS' # Se você for configurar HTTPS
sudo ufw enable
sudo ufw status
```

6. Instalação do Node.js e pnpm

O Node.js é necessário para construir o frontend React, e o pnpm é um gerenciador de pacotes eficiente.

6.1. Instalar Node.js

Recomenda-se usar o NodeSource para instalar a versão LTS mais recente do Node.js.

```
curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash -
sudo apt install -y nodejs
```

6.2. Verificar a Instalação do Node.js e npm

```
node -v
npm -v
```

6.3. Instalar pnpm

```
sudo npm install -g pnpm
pnpm -v
```

7. Configuração e Build do Frontend (React)

Após instalar o Node.js e o pnpm, você precisará configurar e construir a aplicação React para produção.

1. **Navegue até o diretório do frontend:** `bash cd /var/www/agendamento-frontend # Clone seu repositório aqui # git clone <your-frontend-repo-url> .`
2. **Instale as dependências do frontend:** `bash pnpm install`

3. **Configure as variáveis de ambiente:** Crie um arquivo `.env` na raiz do projeto frontend (`agendamento-frontend`) e adicione a URL da sua API (que será servida pelo Nginx): `env VITE_API_URL=http://seu_dominio_ou_ip/api` Substitua `seu_dominio_ou_ip` pelo domínio ou endereço IP do seu servidor onde o Nginx está rodando.

4. **Construa a aplicação para produção:** `bash pnpm run build` Este comando criará uma pasta `dist` dentro do seu diretório `agendamento-frontend`. Esta pasta contém todos os arquivos estáticos otimizados para produção que o Nginx servirá.

5. **Atualize a configuração do Nginx:** Conforme mencionado na Seção 5.4, certifique-se de que a diretiva `root` no Nginx aponte para esta nova pasta `dist`. `nginx location / { root /var/www/agendamento-frontend/dist; # Caminho para a pasta 'dist' do build do React index index.html; try_files $\`uri \`$uri/ /index.html; }` Após atualizar o `nginx.conf`, lembre-se de recarregar o Nginx (`sudo systemctl reload nginx`).

8. Configuração do n8n e Evolution API para Notificações WhatsApp

Esta seção aborda a configuração do n8n e da Evolution API para o envio de notificações via WhatsApp. Recomenda-se a instalação via Docker para maior facilidade e isolamento.

8.1. Instalar Docker e Docker Compose

```
 sudo apt update
sudo apt install -y ca-certificates curl gnupg lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list >
/dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

# Adicionar seu usuário ao grupo docker para executar comandos docker sem sudo
sudo usermod -aG docker $USER
newgrp docker # Aplica as mudanças de grupo imediatamente
```

8.2. Configurar a Evolution API com Docker Compose

1. **Crie um diretório para a Evolution API:** `bash mkdir ~/evolution-api cd ~/evolution-api`
2. **Crie o arquivo `docker-compose.yml`:** Use um editor de texto e salve o seguinte conteúdo como `docker-compose.yml` dentro da pasta `~/evolution-api`: ```yaml version: '3.8'

services: evolution-api: image: evolutionapi/evolution-api:latest container_name: evolution-api restart: always ports: - 8080:8080 # Porta da API volumes: - ./data:/app/data # Persistência de dados environment: - SERVER_PORT=8080 - AUTHENTICATION_TYPE=apikey - AUTHENTICATION_API_KEY=your-evolution-api-key # **MUDE ESTA CHAVE!** - WEBHOOK_GLOBAL_ENABLED=true - WEBHOOK_GLOBAL_URL=http://seu_dominio_ou_ip:5678/webhook/whatsapp-status # Aponta para o n8n - QRCODE_LIMIT=30 - DATABASE_ENABLED=false # Pode ser true se usar MongoDB externo # - DATABASE_CONNECTION_URI=mongodb://mongo:27017/evolution # Se usar MongoDB

networks: default: name: evolution-network `` **Atenção:** - Altere your-evolution-api-key `para uma chave forte e segura. - http://seu_dominio_ou_ip:5678 deve ser o endereço IP ou domínio do seu servidor onde o n8n estará acessível. Se o n8n estiver no mesmo servidor, você pode usar o IP interno do Docker ou localhost se a rede Docker

`permitir. - Se você precisar de persistência de dados para a Evolution API (histórico de mensagens, etc.), descomente a seção` mongo `e configure` DATABASE_ENABLED=true `e` DATABASE_CONNECTION_URI `no serviço` evolution-api`.

1. **Inicie a Evolution API:**
2. Abra o terminal no diretório `~/evolution-api`.
3. Execute: `bash docker compose up -d`
4. Verifique se o contêiner está rodando: `bash docker ps`
5. **Crie uma instância do WhatsApp:**
6. Após a Evolution API iniciar, você precisará criar uma instância para conectar seu WhatsApp. Substitua `your-evolution-api-key` pela sua chave. `bash curl -X POST http://localhost:8080/instance/create \ -H "Content-Type: application/json" \ -H "apikey: your-evolution-api-key" \ -d '{"instanceName": "instance1", "qrcode": true, "webhook": {"url": "http://seu_dominio_ou_ip:5678/webhook/whatsapp-status", "events": ["messages", "connection"]}}'`
7. Você receberá um QR Code no console ou nos logs da Evolution API. Escaneie-o com seu celular para conectar o WhatsApp.

8.3. Configurar o n8n com Docker Compose

1. **Crie um diretório para o n8n:** `bash mkdir ~/n8n cd ~/n8n`
2. **Crie o arquivo `docker-compose.yml`:** Salve o seguinte conteúdo como `docker-compose.yml` dentro da pasta `~/n8n`: ```yaml version: '3.8'

services: n8n: image: n8n/n8n container_name: n8n restart: always ports: - 5678:5678 # Porta da interface web do n8n volumes: - ./data:/home/node/.n8n # Persistência de dados do n8n environment: - N8N_HOST=localhost - N8N_PORT=5678 - N8N_PROTOCOL=http - WEBHOOK_URL=http://seu_dominio_ou_ip:5678/webhook/ # Substitua pelo seu domínio ou IP - GENERIC_TIMEZONE=America/Sao_Paulo # Ou seu fuso horário ``` **Atenção:** - WEBHOOK_URL=http://seu_dominio_ou_ip:5678/webhook/ `é crucial para que o`

`n8n possa gerar URLs de webhook acessíveis. Substitua`seu_dominio_ou_ip` pelo endereço IP ou domínio do seu servidor.

1. **Inicie o n8n:**
2. Abra o terminal no diretório `~/n8n`.
3. Execute: `bash docker compose up -d`
4. Verifique se o contêiner está rodando: `bash docker ps`
5. **Acesse a interface web do n8n:** Abra seu navegador e vá para `http://seu_dominio_ou_ip:5678`.
6. **Crie o Workflow de WhatsApp no n8n:**
7. No n8n, crie um novo workflow.
8. Adicione um nó **Webhook** como trigger.
    - **Webhook URL:** Anote a URL gerada (ex: `http://seu_dominio_ou_ip:5678/webhook/sua-webhook-id`). Você usará isso no backend do Flask.
    - **HTTP Method:** `POST`
    - **Response Mode:** `Respond to Webhook`
9. Adicione um nó **HTTP Request** (para a Evolution API).
    - Conecte-o ao nó Webhook.
    - **Method:** `POST`
    - **URL:** `http://evolution-api:8080/message/sendText/instance1` (se ambos estiverem na mesma rede Docker Compose) ou `http://seu_dominio_ou_ip:8080/message/sendText/instance1`.
    - **Authentication:** `Generic Credential Type`
    - **Credential Type:** `HTTP Header Auth`
    - **Header Name:** `apikey`
    - **Header Value:** Sua `your-evolution-api-key` (a mesma que você definiu no `docker-compose.yml` da Evolution API).
    - **Body Parameters:**
    - `number`: `={{$json.phone}}`

- `text`: `={{$json.message}}`

10. Ative o workflow.

8.4. Configurar o Backend Flask para Notificações

No arquivo `.env` do seu projeto backend (`agendamento-online`), atualize as variáveis para apontar para o n8n e a Evolution API:

```
 # n8n Configuration
N8N_WEBHOOK_URL=http://seu_dominio_ou_ip:5678/webhook/sua-webhook-id # Use a
URL gerada pelo n8n

# Evolution API Configuration
EVOLUTION_API_URL=http://seu_dominio_ou_ip:8080
EVOLUTION_API_KEY=your-evolution-api-key
EVOLUTION_INSTANCE=instance1

# Notification Settings
NOTIFICATIONS_ENABLED=true
SEND_CONFIRMATION=true
SEND_REMINDERS=true
REMINDER_HOURS_BEFORE=24
```

Substitua `seu_dominio_ou_ip`, `sua-webhook-id` e `your-evolution-api-key` pelos valores corretos.

9. Teste Final e Considerações de Produção

Após todas as instalações e configurações, é hora de testar o sistema completo.

1. **Verifique o status dos serviços:** `bash sudo systemctl status agendamento sudo systemctl status nginx docker ps`
2. **Acesse o Frontend:** Abra seu navegador e vá para o endereço do seu servidor (ex: `http://seu_dominio_ou_ip`).
3. **Realize um agendamento:** Crie um profissional, configure serviços e horários, e então faça um agendamento como cliente.
4. **Verifique as notificações:** Confirme se as mensagens de WhatsApp estão sendo enviadas e recebidas.
5. **Monitore os logs:** Verifique os logs do Gunicorn (`sudo journalctl -u agendamento`), Nginx (`sudo tail -f /var/log/nginx/access.log`

`/var/log/nginx/error.log`), Evolution API e n8n (via interface web ou logs do Docker).

9.1. Considerações Adicionais para Produção

- **HTTPS:** Para segurança, é **altamente recomendado** configurar HTTPS no Nginx usando certificados SSL/TLS (ex: Let's Encrypt). Isso criptografará a comunicação entre o cliente e o servidor.
- **Firewall (UFW):** Mantenha o firewall configurado para permitir apenas o tráfego necessário (portas 80 e 443 para acesso público, e as portas internas para comunicação entre os serviços).
- **Backup:** Implemente uma rotina de backup regular para o banco de dados PostgreSQL e para os volumes de dados do n8n e Evolution API.
- **Monitoramento:** Utilize ferramentas de monitoramento para acompanhar a saúde do servidor, uso de recursos e performance da aplicação.
- **Atualizações:** Mantenha o sistema operacional, Python, Node.js e todas as dependências atualizadas para garantir segurança e performance.
- **Segurança do Docker:** Siga as melhores práticas de segurança para contêineres Docker.
- **Variáveis de Ambiente:** Não armazene informações sensíveis (chaves de API, senhas) diretamente no código. Use variáveis de ambiente ou um sistema de gerenciamento de segredos.

Este guia fornece uma base sólida para a implantação do seu sistema de agendamento em um ambiente de produção no Linux (Ubuntu Server). Lembre-se de adaptar os caminhos e configurações conforme a sua infraestrutura específica. ```

Referências

[1] Python.org - Python Releases for Windows: https://www.python.org/downloads/windows/ [2] Real Python - How to Install Python on Windows: https://realpython.com/installing-python/ [3] PostgreSQL.org - Download PostgreSQL for Windows: https://www.postgresql.org/download/windows/ [4] Nginx.org - Download Nginx: http://nginx.org/en/download.html [5] NSSM - the Non-Sucking Service Manager: https://nssm.cc/ [6] NSSM - Download:

https://nssm.cc/download [7] DigitalOcean - How To Install Nginx on Windows: https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-windows [8] Nodejs.org - Download Node.js: https://nodejs.org/en/download/ [9] Docker Docs - Install Docker Desktop on Windows: https://docs.docker.com/desktop/install/windows-install/ [10] Docker Docs - Install Docker Engine on Ubuntu: https://docs.docker.com/engine/install/ubuntu/ [11] Docker Docs - Install Docker Compose: https://docs.docker.com/compose/install/ [12] Evolution API - Documentation: https://docs.evolutionapi.com/ [13] n8n - Documentation: https://docs.n8n.io/