

MANUAL TÉCNICO

Instalação do Docker e Configuração do Módulo de Integração WhatsApp

Ambiente Linux (servidor) via Docker Engine (Evolution API)

TI Denis Ripol
Desenvolvimento de Sistemas — Cuiabá, Mato Grosso

Julho de 2026

1. Objetivo deste manual

Este manual orienta a instalação do Docker Engine (nativo do Linux, sem a camada extra do Docker Desktop) e a configuração do ambiente que hospeda a Evolution API — o serviço responsável pela integração de envio de mensagens via WhatsApp utilizada pelo sistema.

Importante

Este procedimento é executado uma única vez, no servidor Linux que ficará responsável por manter a integração de WhatsApp ativa. Depois de configurado, o ambiente roda em segundo plano e reinicia sozinho junto com o servidor.

2. Requisitos do ambiente

Antes de iniciar, confirme se o servidor atende aos requisitos abaixo:

Requisito	Especificação
Sistema operacional	Ubuntu Server 22.04 LTS ou 24.04 LTS (ou Debian 11/12 equivalente)
Acesso	Usuário com privilégio sudo (ou root)
Memória RAM	2 GB no mínimo (4 GB recomendado)
Espaço em disco	Aproximadamente 2 GB livres para as imagens e dados
Portas	8080 liberada (localmente ou no firewall, conforme o cenário - ver Passo 6 da Seção 4)
Conexão à internet	Necessária na instalação e para o WhatsApp Web manter a sessão ativa

3. Instalação do Docker Engine

No Linux, o Docker roda nativamente (usa o próprio kernel do sistema, sem máquina virtual), o que torna a instalação mais simples e mais leve que no Windows. Os comandos abaixo seguem o método oficial da Docker para Ubuntu/Debian.

Passo 1 — Remover versões antigas (se houver)

Evita conflito com pacotes antigos do repositório padrão da distribuição:

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

Passo 2 — Instalar dependências e adicionar o repositório oficial da Docker

Copie e execute o bloco completo abaixo (pode colar tudo de uma vez no terminal):

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
  sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Se o servidor for Debian em vez de Ubuntu, troque "ubuntu" por "debian" nas duas URLs acima.

Passo 3 — Instalar o Docker Engine e o plugin Compose

O "Docker Compose" no Linux vem como um plugin do próprio Docker (comando `docker compose`, sem hífen):

```
sudo apt-get update
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

Passo 4 — Habilitar o Docker para iniciar automaticamente com o servidor

Importante em servidor: garante que, se o servidor reiniciar (queda de energia, manutenção), o Docker (e a Evolution API dentro dele) voltem sozinhos:

```
sudo systemctl enable docker
sudo systemctl start docker
```

Passo 5 — Permitir rodar o Docker sem sudo (opcional, mas recomendado)

Adiciona seu usuário ao grupo `docker`, evitando digitar `sudo` em todo comando. É necessário sair e entrar novamente na sessão SSH (ou rodar `"newgrp docker"`) para o grupo ter efeito:

```
sudo usermod -aG docker $USER
newgrp docker
```

Passo 6 — Confirmar a instalação

Deve retornar o número da versão instalada, sem erro:

```
docker --version
docker compose version
```

4. Criação do ambiente da Evolution API

Com o Docker instalado, o próximo passo é o mesmo em qualquer sistema operacional: criar a pasta do projeto com o arquivo de configuração (`docker-compose.yml`) e subir os serviços.

Passo 1 — Criar a pasta do projeto

Crie uma pasta dedicada para o ambiente, por exemplo dentro do diretório `home` do usuário:

```
mkdir -p ~/evolution-api && cd ~/evolution-api
```

Passo 2 — Criar o arquivo `docker-compose.yml`

Crie o arquivo com um editor de texto do terminal (`nano` é o mais simples):

```
nano docker-compose.yml
```

Cole o conteúdo abaixo, salve (`Ctrl+O`, `Enter`) e saia (`Ctrl+X`):

```
services:
  postgres:
    container_name: evolution_postgres
    image: postgres:15
    restart: always
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_DB=evolution
    volumes:
      - postgres_data:/var/lib/postgresql/data

  evolution-api:
    container_name: evolution_api
    image: evoapicloud/evolution-api:v2.3.7
    restart: always
```

```
ports:
  - "8080:8080"
environment:
  - AUTHENTICATION_API_KEY=troque-por-uma-chave-sua
  - DATABASE_ENABLED=true
  - DATABASE_PROVIDER=postgresql
  - DATABASE_CONNECTION_URI=postgres:postgres@postgres:5432/evolution
  - DATABASE_SAVE_DATA_INSTANCE=true
  - CACHE_REDIS_ENABLED=false
  - CACHE_LOCAL_ENABLED=true
volumes:
  - evolution_instances:/evolution/instances
depends_on:
  - postgres

volumes:
  postgres_data:
  evolution_instances:
```

Importante

Troque o valor "troque-por-uma-chave-sua" por uma senha/token de sua escolha (sem espaços). Esse será o "Token global" usado depois na tela de configuração do sistema — guarde esse valor.

Passo 3 — Subir os containers

Dentro da pasta ~/evolution-api:

```
docker compose up -d
```

Esse comando baixa as imagens necessárias (pode levar alguns minutos na primeira vez) e inicia dois containers: o banco de dados (postgres) e a Evolution API.

Passo 4 — Verificar se os containers subiram

Aguarde de 15 a 20 segundos e confirme que ambos os containers estão em execução:

```
docker ps
```

Devem aparecer duas linhas — "evolution_postgres" e "evolution_api" — ambas com status "Up".

Passo 5 — Conferir os logs (opcional, mas recomendado na primeira vez)

Para acompanhar a inicialização e garantir que não há erros:

```
docker logs -f evolution_api
```

Quando aparecer uma mensagem semelhante a "Server running on port 8080", o ambiente está pronto. Pressione Ctrl+C para sair do modo de acompanhamento — os containers continuam rodando normalmente.

Passo 6 — Liberar a porta 8080 (se o sistema for acessar de outra máquina)

Se o sistema Delphi vai rodar na MESMA máquina/rede local do servidor, esse passo pode não ser necessário. Se for acessar de outro servidor/computador pela rede, libere a porta no firewall:

```
sudo ufw allow 8080/tcp
sudo ufw status
```

Segurança

Se o servidor tiver IP público (VPS na nuvem), NÃO deixe a porta 8080 aberta para a internet toda sem proteção — a Evolution API já exige o "Token global" (AUTHENTICATION_API_KEY) para qualquer chamada, o que dá uma proteção mínima, mas o ideal para produção é liberar o acesso apenas para o IP fixo de onde o sistema Delphi vai se conectar, ou colocar atrás de um reverse proxy com HTTPS (Nginx + Let's Encrypt, por exemplo). Se preferir, podemos configurar isso numa etapa futura.

Passo 7 — Testar

De dentro do próprio servidor, ou de outra máquina (se a porta estiver liberada), acesse pelo navegador ou via curl:

```
curl http://localhost:8080
```

Uma resposta em formato JSON de boas-vindas da API confirma que o ambiente está funcionando.

5. Configuração no sistema

Com o ambiente rodando, o último passo é conectar o sistema a ele:

Passo 1 — Abrir a tela de integração WhatsApp

No sistema, acesse a tela de configuração do módulo WhatsApp.

Passo 2 — Preencher os campos de conexão

Informe os seguintes valores:

URL da API:	http://IP_DO_SERVIDOR:8080 (ou http://localhost:8080 se o sistema roda na mesma máquina do Docker)
Token global:	(o valor que você definiu no docker-compose.yml)
Nome da instância:	(um nome de sua escolha, ex: principal)

Passo 3 — Conectar e escanear o QR Code

Clique em "Conectar". Um QR Code será exibido na tela. Abra o WhatsApp no celular que será usado para o envio das mensagens, vá em Configurações → Aparelhos conectados → Conectar um aparelho, e escaneie o código exibido.

Passo 4 — Confirmar a conexão

Após escanear, o status na tela deve mudar automaticamente para "Conectado" em poucos segundos.

6. Rotina do dia a dia

- Parar o ambiente: "docker compose down" (dentro da pasta ~/evolution-api)
- Iniciar novamente: "docker compose up -d"
- Ver se está rodando: "docker ps"
- Ver os logs em caso de problema: "docker logs -f evolution_api"
- Ver uso de recursos dos containers: "docker stats"

Nota

A sessão do WhatsApp conectado fica salva no banco de dados do ambiente (volume do Docker). Reiniciar o servidor ou os containers não desconecta o WhatsApp - graças ao "restart: always" e ao "systemctl enable docker" configurados, tudo volta sozinho depois de uma reinicialização.

7. Solução de problemas comuns

"Permission denied" ao rodar comandos docker

- Confirme se rodou o Passo 5 da Seção 3 (usermod + newgrp), ou use "sudo" na frente do comando como alternativa.

curl http://localhost:8080 não responde

- Rode "docker ps" e confira se os dois containers aparecem com status "Up".

- Rode "sudo systemctl status docker" para confirmar que o serviço do Docker está ativo.

QR Code não aparece na tela do sistema

- Confirme que a "URL da API" usa o IP correto do servidor (não localhost, se o sistema estiver rodando em outra máquina) e que a porta 8080 está liberada no firewall (Passo 6 da Seção 4).
- Rode "docker logs -f evolution_api" e procure por mensagens de erro.

Preciso migrar para outro servidor

- Copie a pasta ~/evolution-api inteira (com o docker-compose.yml) para o novo servidor.
- Instale o Docker no novo servidor seguindo a Seção 3 deste manual.
- Execute "docker compose up -d" dentro da pasta copiada - o histórico da conexão é preservado (desde que o volume de dados também tenha sido copiado).

TI Denis Ripol

Desenvolvimento de Sistemas — Cuiabá, Mato Grosso