WhatSaaS Installation

Introduction:

WhatSaaS is a comprehensive SaaS project that leverages the robustness of Next.js alongside the Evolution API to deliver a powerful multi-tenant CRM focused on WhatsApp service management and automation. Designed for high performance and scalability, the system orchestrates a modern architecture where the interactive front-end communicates in real-time via Pusher , while the Evolution API manages WhatsApp Web/WABA instances in an isolated and stable manner. Featuring native integration with Stripe for subscriptions and Resend for transactional emails, WhatSaaS offers an "out-of-the-box" solution for entrepreneurs looking to launch their own white-label chat platform.

Important!
After updating to version 1.0.5, **payment** gateways must be configured through the Admin Panel (Admin > Payments). Stripe credentials previously set via environment variables will continue to work as a fallback, but we recommend migrating them to the admin interface.

System Requirements:

To ensure the **WhatSaaS** platform runs efficiently with high availability for real-time messaging, the following hardware and software requirements must be met.

1. Hardware Requirements (Minimum)

- **Processor (CPU):** 2 Cores (Recommended for handling multiple WhatsApp instances).
- **Memory (RAM):** 4 GB (Essential for running Docker containers, PostgreSQL, and Next.js build processes).
- **Storage (SSD):** 20 GB available space (To store logs, databases, and media files like audio/video).
- **Network:** Static IP address

2. Software Requirements

- **Operating System:** Ubuntu 20.04 LTS or 22.04 LTS (Recommended for stability and native Docker support).
- **Runtime Environment:** Node.js v20.x (LTS).
- **Package Manager:** `pnpm` (Used for fast and disk-efficient dependency management).
- **Process Manager:** `pm2` (To keep the Next.js application running in the background).
- **Web Server:** Nginx (Acts as a Reverse Proxy and serves static media files efficiently).
- **Virtualization:** Docker & Docker Compose (Required for isolating the Evolution API and Database services).

3. Third-Party Services

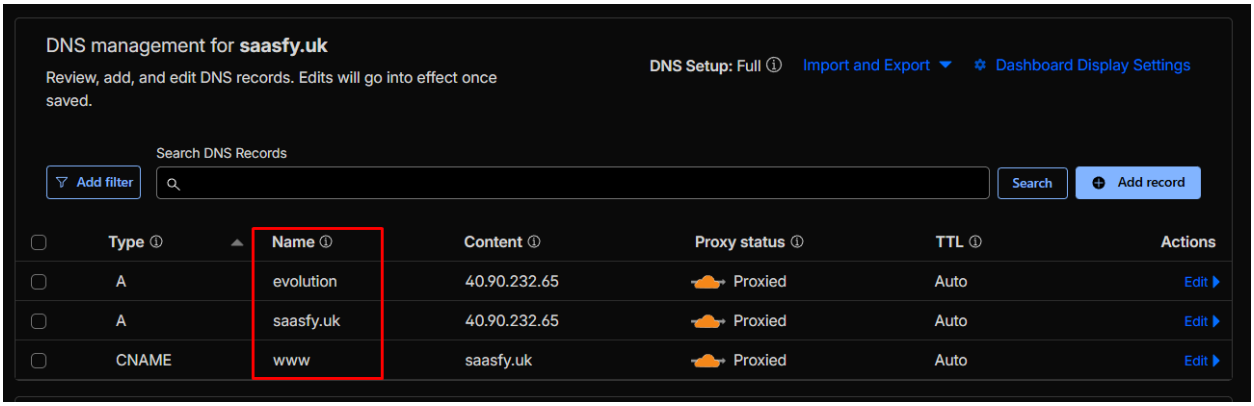
The project relies on specific external integrations to function as a complete SaaS:

- **DNS & Security:** Cloudflare (Required for SSL management and domain pointing).
- **Real-time Messaging:** Pusher (Required for instant bi-directional message synchronization).
- **Payments & Billing:** Stripe (Required for subscription management and webhooks).
- **Transactional Email:** Resend (Required for magic links and system notifications).
- **Messaging Backend:** Evolution API (Required for WhatsApp connectivity).

Installation:

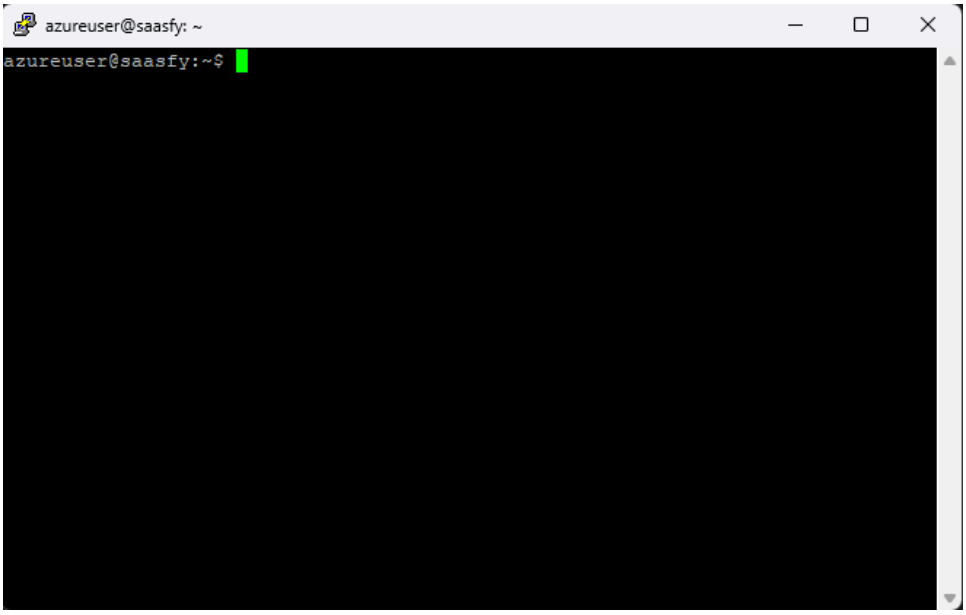
1️⃣ Step 1:

Make sure your domain is pointing to your VPS.



I'm using Cloudflare, but feel free to choose your own domain provider.

Step 1.1 Access your VPS via SSH:



During the step-by-step process, I'll be using PuTTY, but it's optional.

Step 1.2 Update System & Install Essentials:

Run the following commands:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y curl git unzip build-essential
```

Step 1.3: Install Node.js (v20 LTS)

```
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt install -y nodejs
```

📌 Verify installation

`node -v`

Step 1.4: Install Package Managers (pnpm & PM2)

```
sudo npm install -g pnpm
sudo npm install -g pm2
```

Step 1.5: Install Docker & Docker Compose

📌 Required to run the PostgreSQL databases and the Evolution API.

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker $USER
```

⚠️: Log out and log back in for group changes to take effect.

Step 1.6: Install Nginx & Certbot (SSL)

```
sudo apt install -y nginx certbot python3-certbot-nginx
```

2️⃣ Part I: Installing Evolution API

✅ We install the Evolution API first because the main SaaS project depends on it.

Step 2.1: Directory Setup

```
mkdir evolution-api
cd evolution-api
```

Step 2.2: Create Environment Files

```
nano .env
```

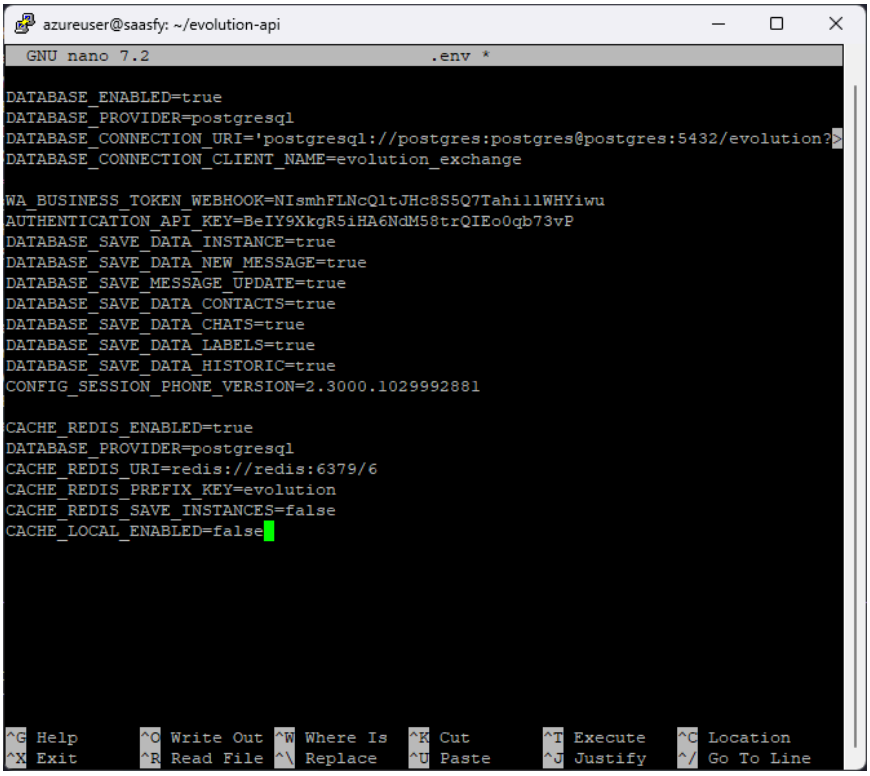
Paste these variables into your .env file:

```
DATABASE_ENABLED=true
DATABASE_PROVIDER=postgresql
DATABASE_CONNECTION_URI=postgresql://postgres:postgres@postgres:5432/evolution?schema=pub
lic
DATABASE_CONNECTION_CLIENT_NAME=evolution_exchange

WA_BUSINESS_TOKEN_WEBHOOK= //e.g. 9VXu3w... (generate at 'https://generate-random.org/api
-keys')
AUTHENTICATION_API_KEY= //e.g. 9VXu3w... (generate at 'https://generate-random.org/api-ke
ys')
DATABASE_SAVE_DATA_INSTANCE=true
DATABASE_SAVE_DATA_NEW_MESSAGE=true
DATABASE_SAVE_MESSAGE_UPDATE=true
DATABASE_SAVE_DATA_CONTACTS=true
DATABASE_SAVE_DATA_CHATS=true
DATABASE_SAVE_DATA_LABELS=true
DATABASE_SAVE_DATA_HISTORIC=true
CONFIG_SESSION_PHONE_VERSION=2.3000.1029992881

CACHE_REDIS_ENABLED=true
DATABASE_PROVIDER=postgresql
CACHE_REDIS_URI=redis://redis:6379/6
CACHE_REDIS_PREFIX_KEY=evolution
CACHE_REDIS_SAVE_INSTANCES=false
CACHE_LOCAL_ENABLED=false
```

Save this for use in the WhatSaaS setup.
`WA_BUSINESS_TOKEN_WEBHOOK`
`AUTHENTICATION_API_KEY`

📌 Although the Evolution API handles WhatsApp Web versions very well, in case of problems connecting a new instance, manually update the WhatsApp version by setting this variable:
`CONFIG_SESSION_PHONE_VERSION`

You should have something like this in your SSH:



Save (ctrl + X).

Create the `docker-compose.yaml` file:
`nano docker-compose.yaml`

paste into your file:

```
services:

  postgres:
```

```
    container_name: evolution_postgres
    image: postgres:15
    restart: always
    networks:
      - evolution-net
    command: ["postgres", "-c", "max_connections=1000"]
    ports:
      - "127.0.0.1:5432:5432"
    environment:
      - POSTGRES_PASSWORD=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_DB=evolution
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:latest
    container_name: evolution_redis
    restart: always
    networks:
      - evolution-net
    command: redis-server --port 6379 --appendonly yes
    ports:
      - "127.0.0.1:6379:6379"
    volumes:
      - evolution_redis:/data

  evolution-api:
    container_name: evolution_api
    image: evoapicloud/evolution-api:latest
    restart: always
    depends_on:
      - redis
      - postgres
    ports:
      - "127.0.0.1:8080:8080"
```

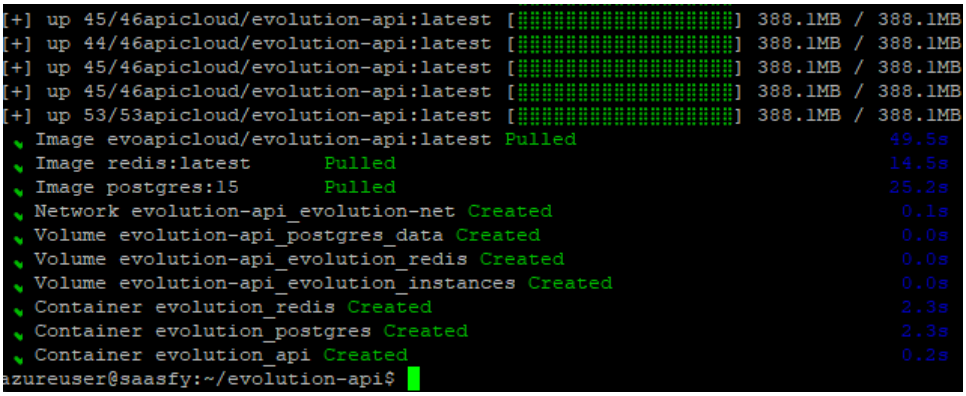
```
    volumes:
      - evolution_instances:/evolution/instances
    networks:
      - evolution-net
    env_file:
      - .env

volumes:
  postgres_data:
  evolution_redis:
  evolution_instances:

networks:
  evolution-net:
    driver: bridge
```

Save (ctrl + X).

Step 2.3: Start the API

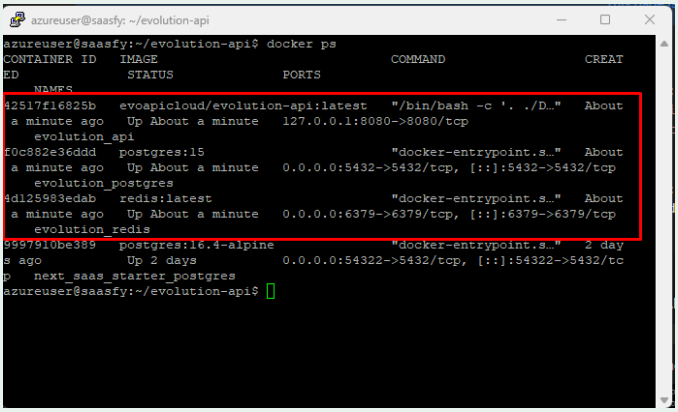
`docker compose up -d`

You should see something like:



📌 Verify containers are running

`docker ps`



Step 2.4: Configure Nginx for API

Create a reverse proxy to expose the API safely.

`sudo nano /etc/nginx/sites-available/evolution`

Past this:

```
server {
    server_name evolution.yourdomain.com; #⚠⚠⚠ Replace with your subdomain
    client_max_body_size 10M;
    location / {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;

        # Headers for API & Webhooks
        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;
    }
}
```

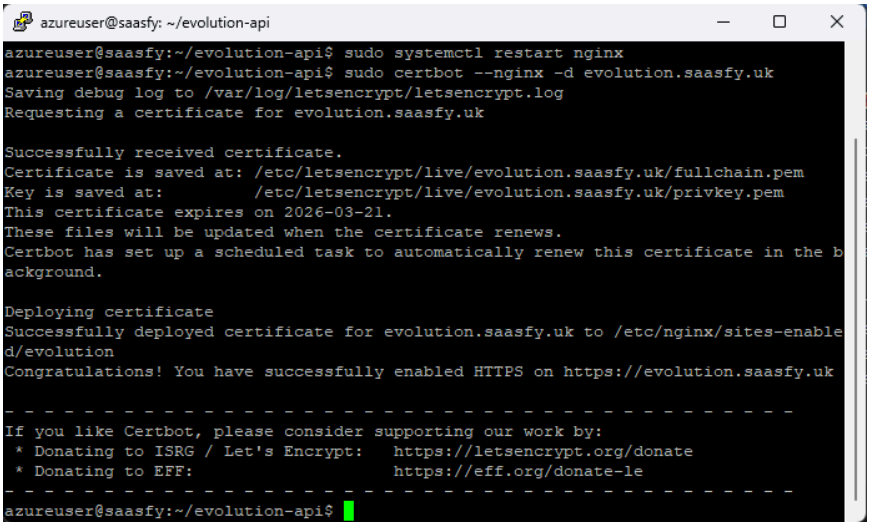
Save (ctrl + X).

Enable the site and generate SSL:

```
sudo ln -s /etc/nginx/sites-available/evolution /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

`sudo certbot --nginx -d evolution.yourdomain.com` #⚠️ Replace with your subdomain

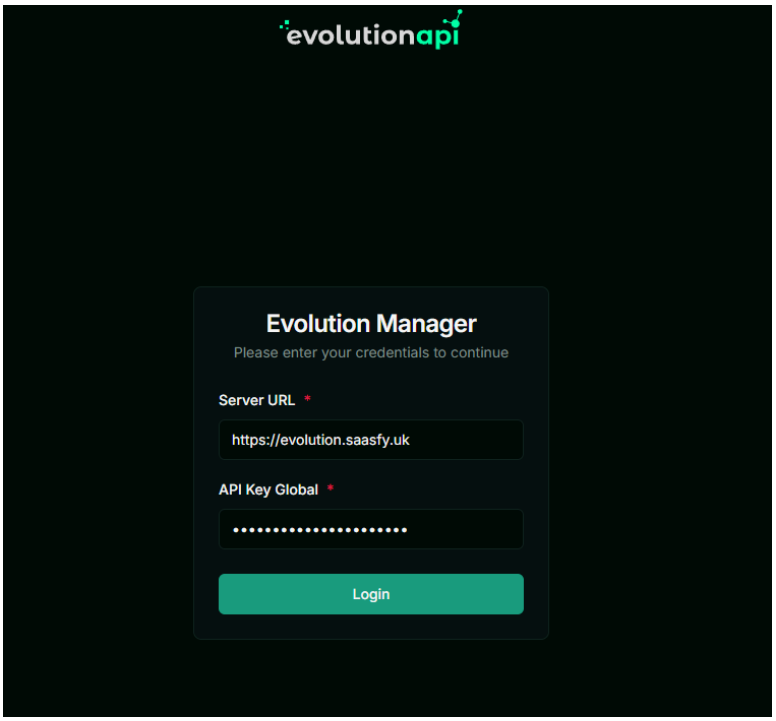
You should see something like:



Step 2.5: 🎉 You should be able to access the Evolution API at:
`https:/evolution.yourdomain.com`

You can:

`https:/evolution.yourdomain.com/manager`



Use the AUTHENTICATION_API_KEY to manage evolution.

🔥 No configuration will be required within the Evolution API as WhatSaaS will manage the instances.
However, if needed, you can use the Evolution API as a backup for your clients: you can retrieve lost messages, **configure Webhooks, WebSockets, or message queues via RabbitMQ and SQS, and manage advanced integrations with powerful tools such as n8n, Typebot, OpenAI, Dify, Flowise, and Chatwoot to enhance your automation and customer support workflows.**

3️⃣ Part II: Installing the SaaS Project

Step 3.1: Clone and Install

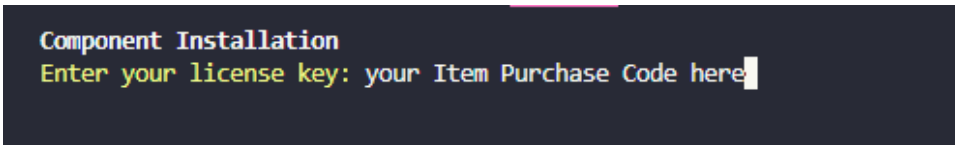
🔒 Add your project to GitHub; this is a good practice and will help you manage version control for future versions of your project

`cd ~`

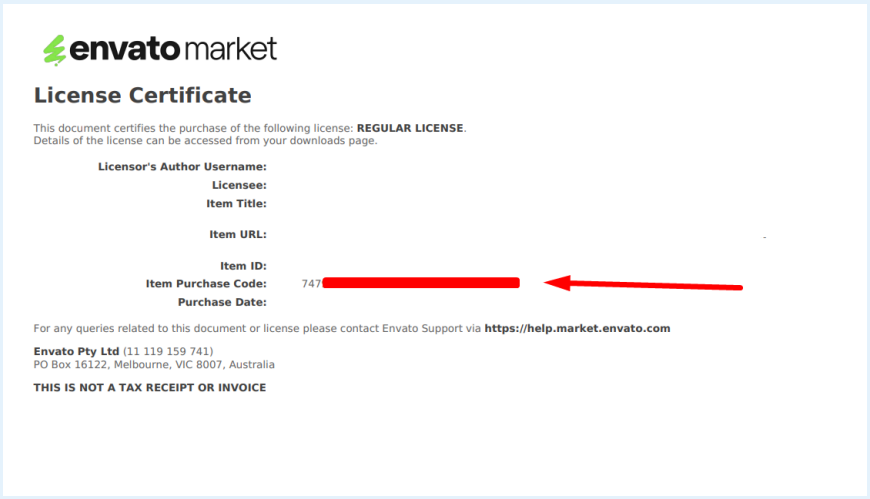
`git clone https://github.com/your-username/your-repo.git` ⚠️ Replace with your repository.

```
cd your-project-folder
pnpm install
```

`pnpm db:components` ⇒ Enter your Item Purchase Code



🔑 You can find your Item Purchase Code on your License Certificate.


envato market

License Certificate

This document certifies the purchase of the following license: **REGULAR LICENSE**.
Details of the license can be accessed from your downloads page.

Licensor's Author Username:
Licensee:
Item Title:
Item URL:
Item ID:
Item Purchase Code: 747[redacted]
Purchase Date:

For any queries related to this document or license please contact Envato Support via **https://help.market.envato.com**

Envato Pty Ltd (11 119 159 741)
PO Box 16122, Melbourne, VIC 8007, Australia

THIS IS NOT A TAX RECEIPT OR INVOICE

Step 3.2: Project Setup

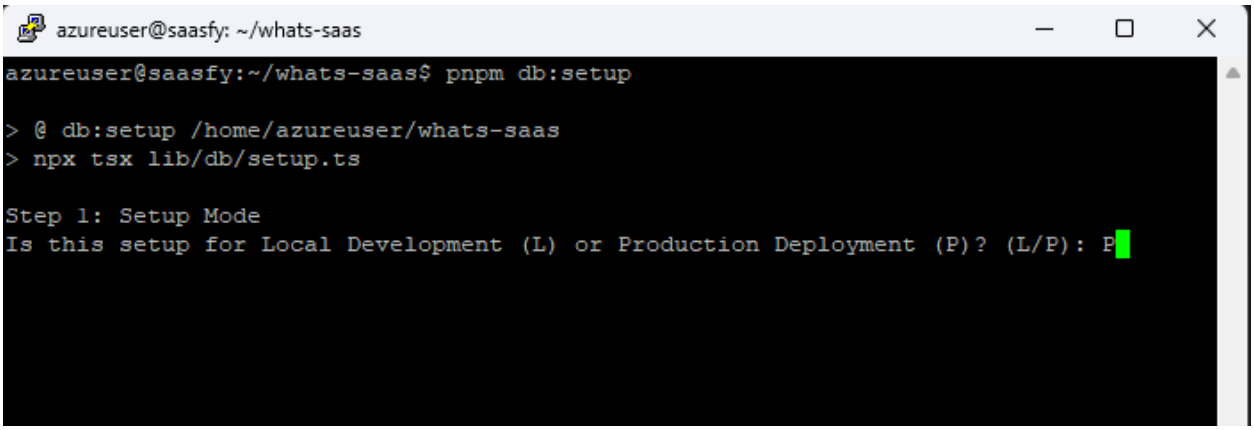
Run the interactive setup script. This script will generate your `.env` file automatically.

```
pnpm db:setup
```

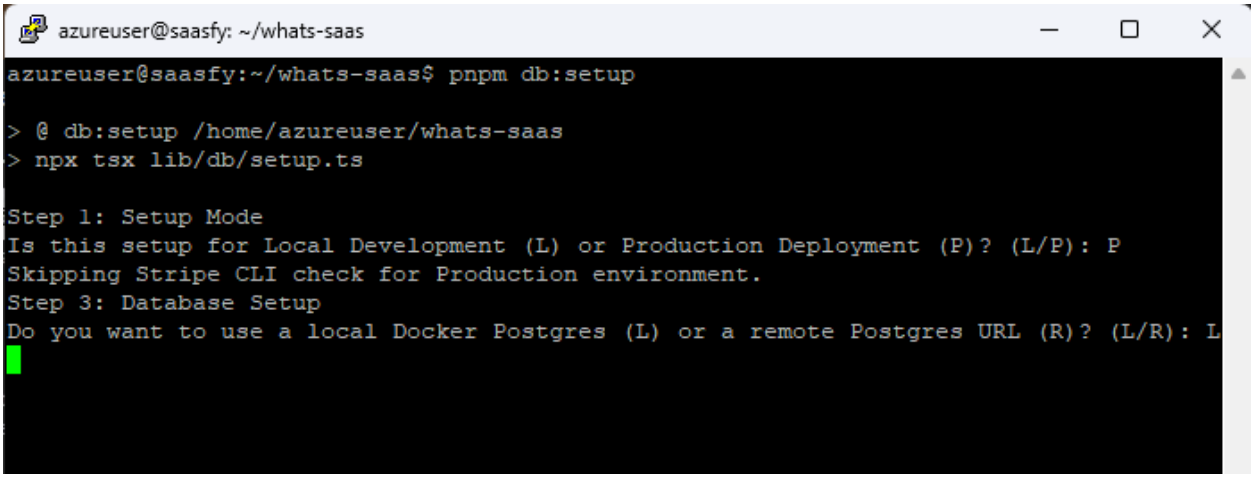
> ⚠️ This step requires a little more care, as it will be necessary to generate some keys to include in the .env file.
> (pusher, resend, stripe)

Setup steps:

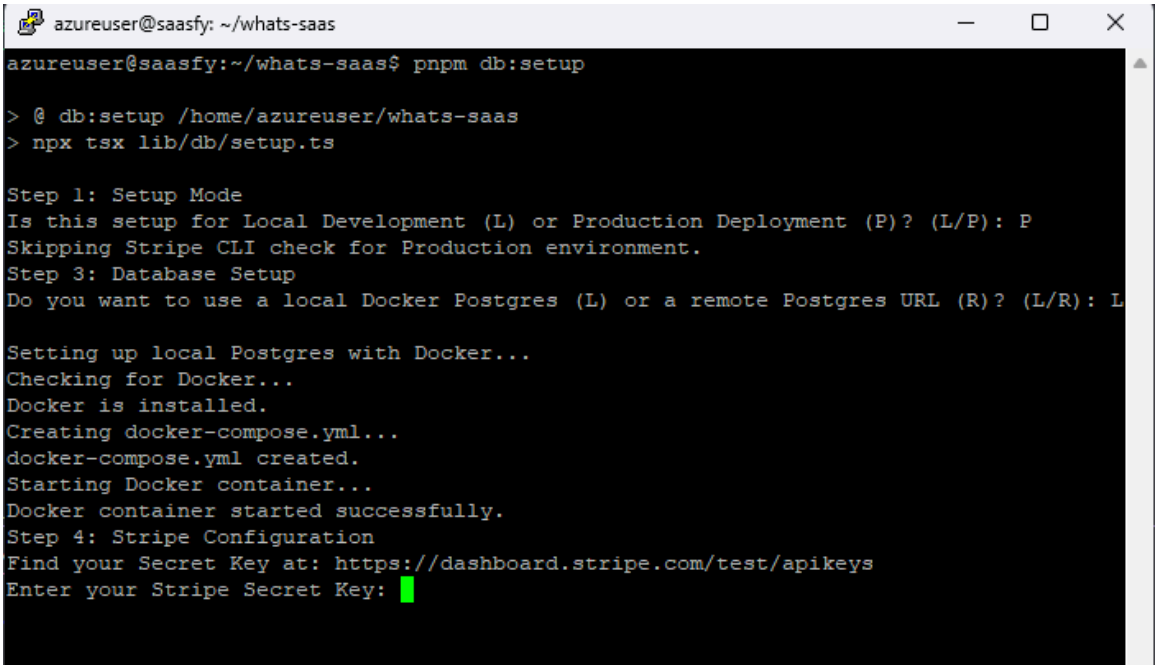
1. Type `P` for Production.



2. Type `L` for local db



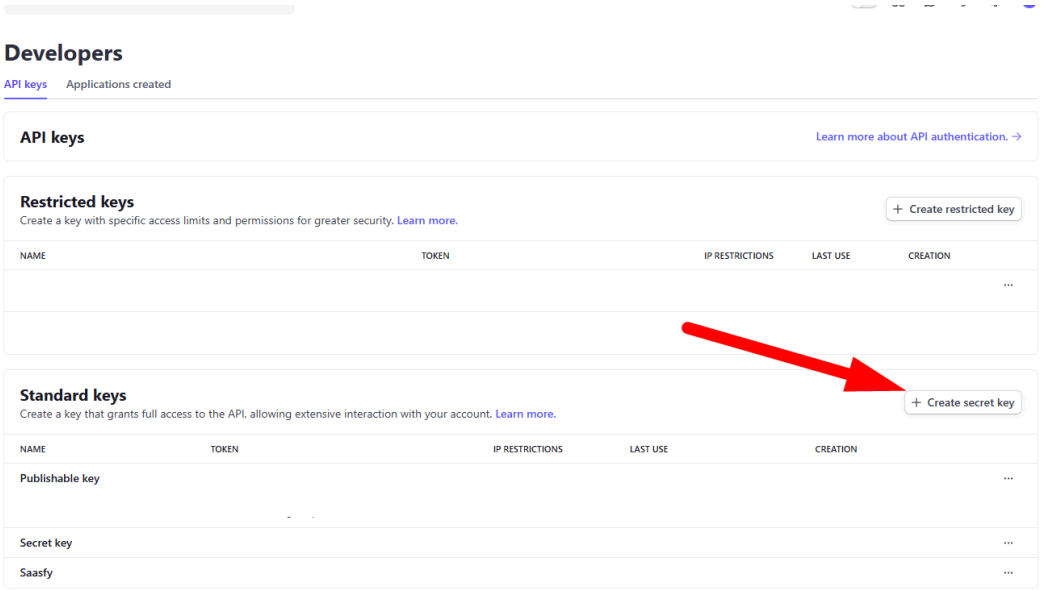
3. (Deprecated in v1.0.5) Enter your `Stripe Secret Key`

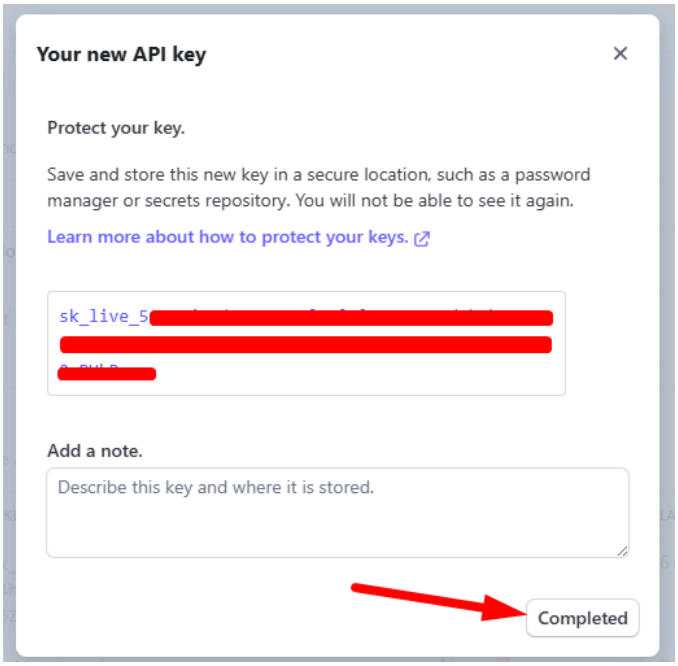


Get your Secret Key:

`https://dashboard.stripe.com/apikeys`

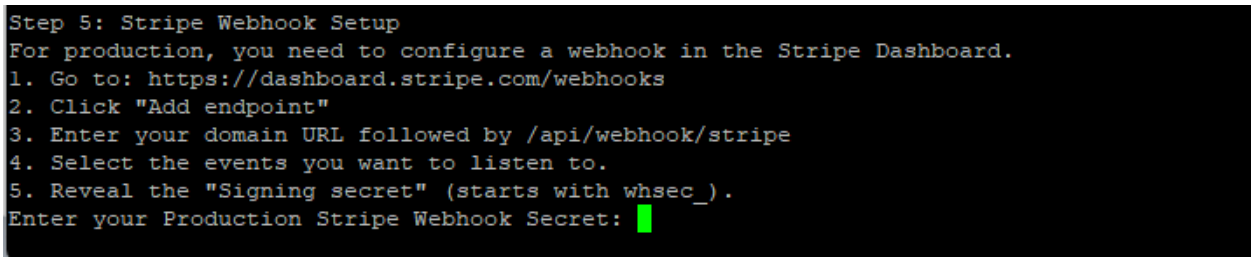
Locate the “Create secret key” button:





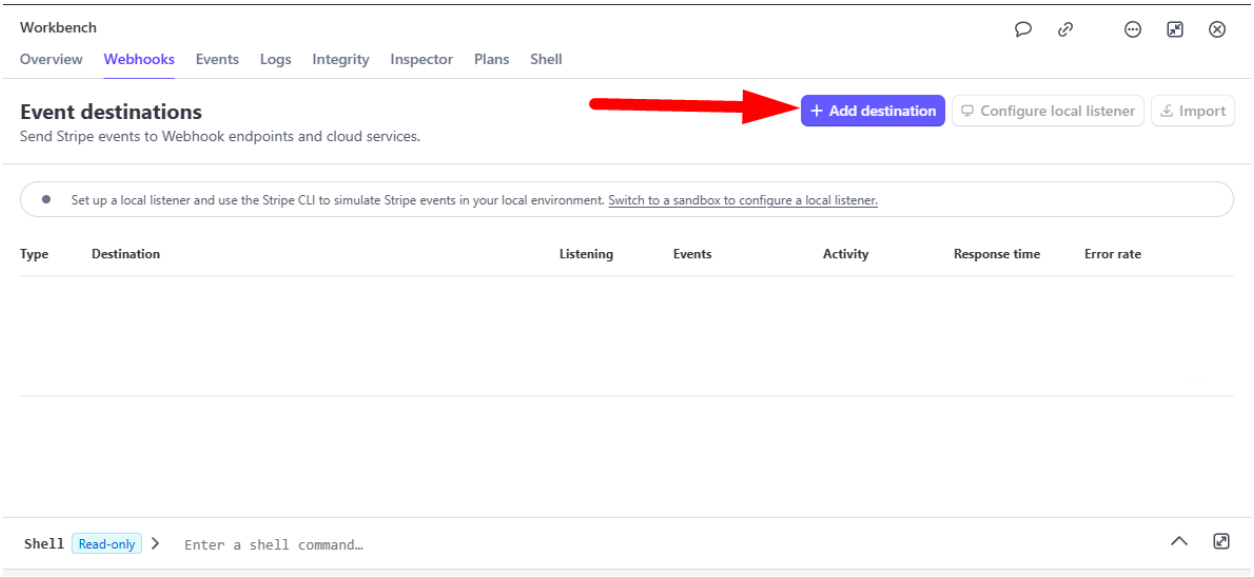
✅ Copy your key and paste it into the setup.

4. (Deprecated in v1.0.5) Enter your Production Stripe Webhook Secret:



Get your Production Stripe Webhook Secret:

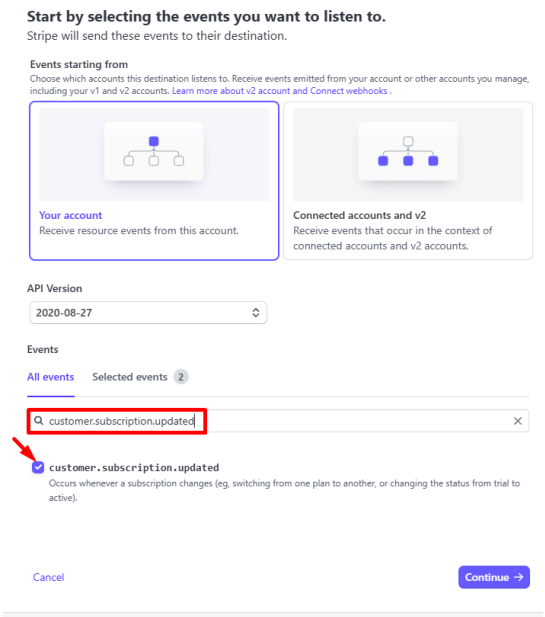
`https://dashboard.stripe.com/webhooks`

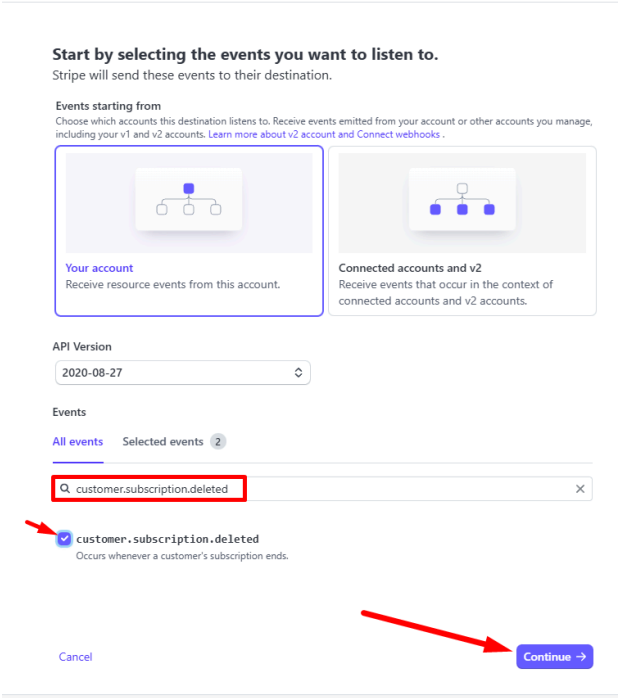


Search for:

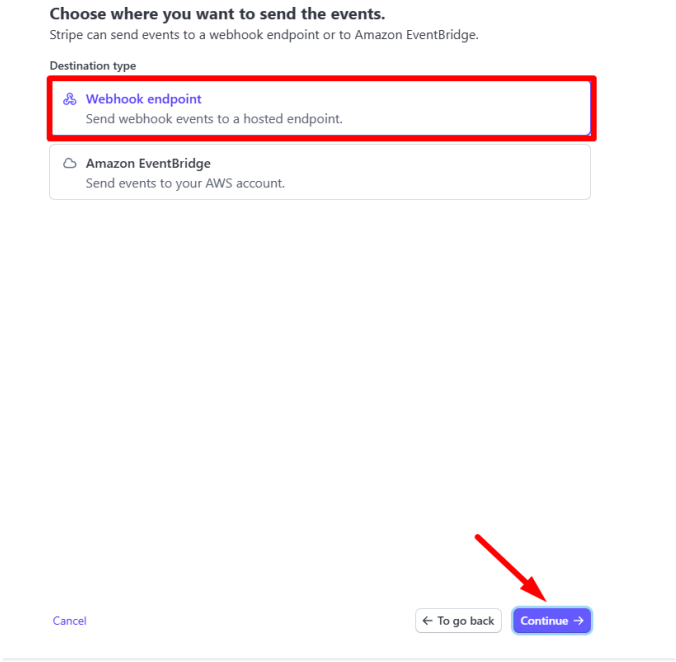
`customer.subscription.updated`

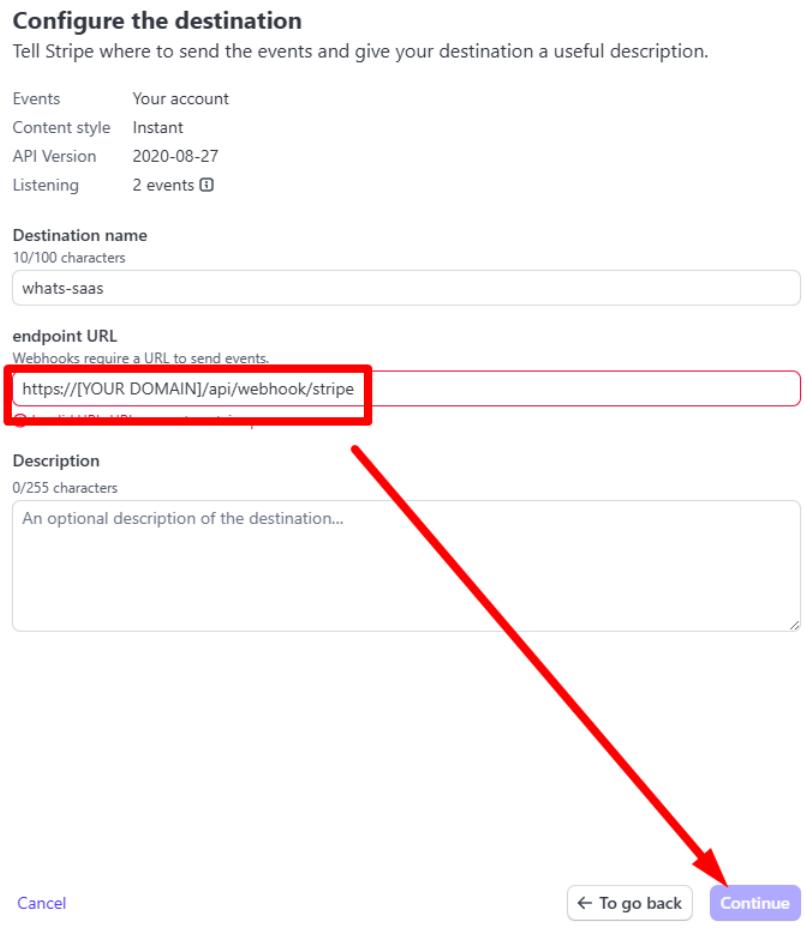
`customer.subscription.deleted`



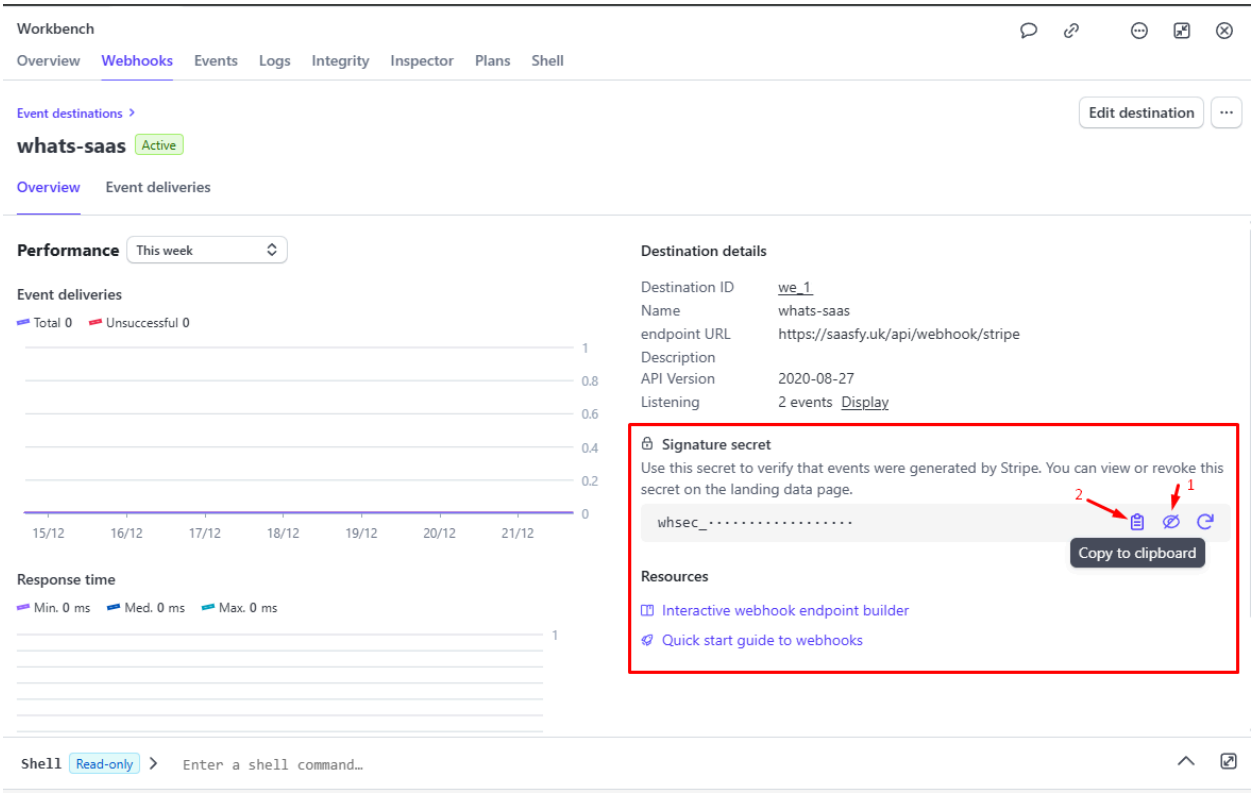


Configure the stripe webhook URL


Choose where you want to send the events.
Stripe can send events to a webhook endpoint or to Amazon EventBridge.
Destination type
Webhook endpoint
Send webhook events to a hosted endpoint.
Amazon EventBridge
Send events to your AWS account.
Cancel
← To go back
Continue →

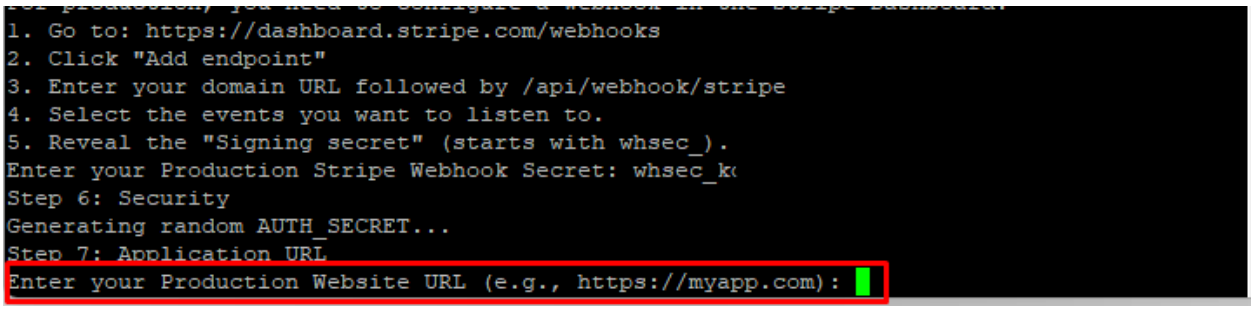


⚠️ Enter your domain. e.g.: https://saasfy.uk/api/webhook/stripe

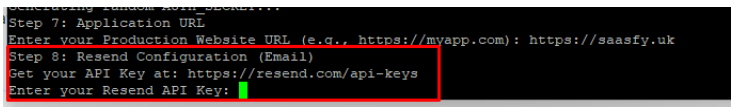


✅ Copy your Production Stripe Webhook Secret and paste it into the setup.

5. Enter the main URL of the SaaS. e.g., https://saasfy.uk

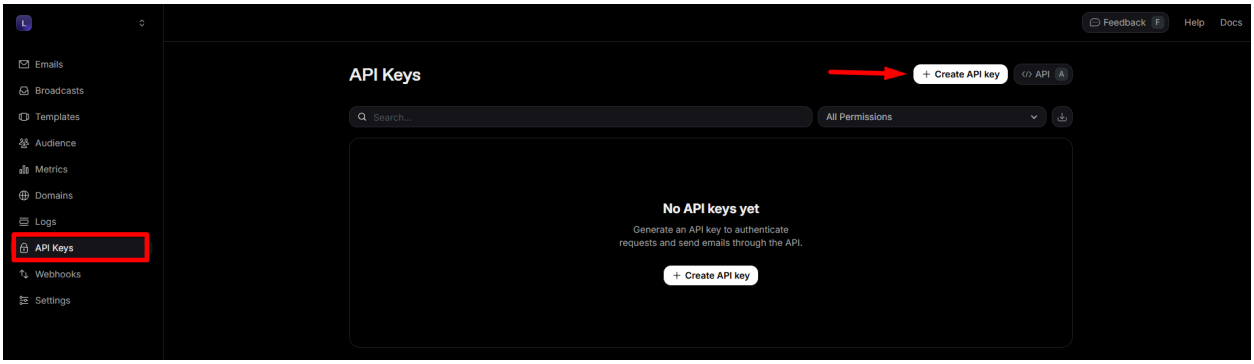


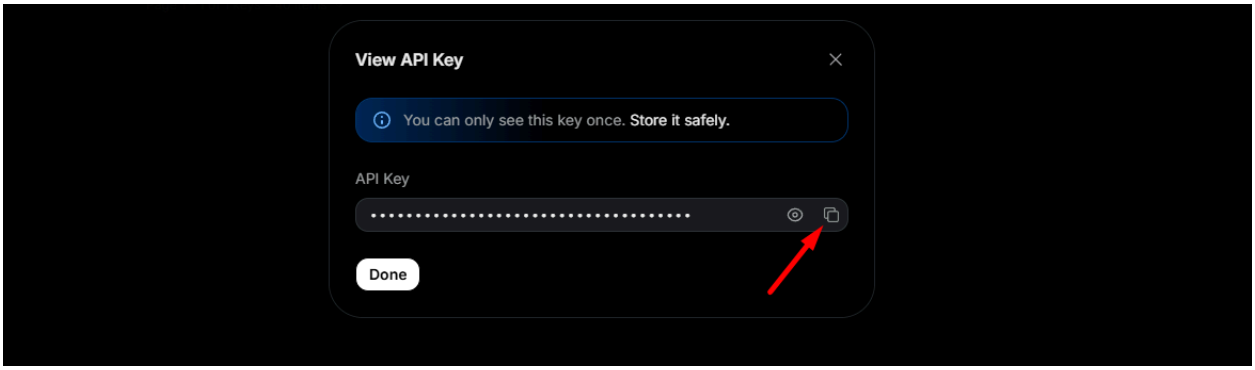
6. Enter your Resend API key.



- Create an account: https://resend.com/login

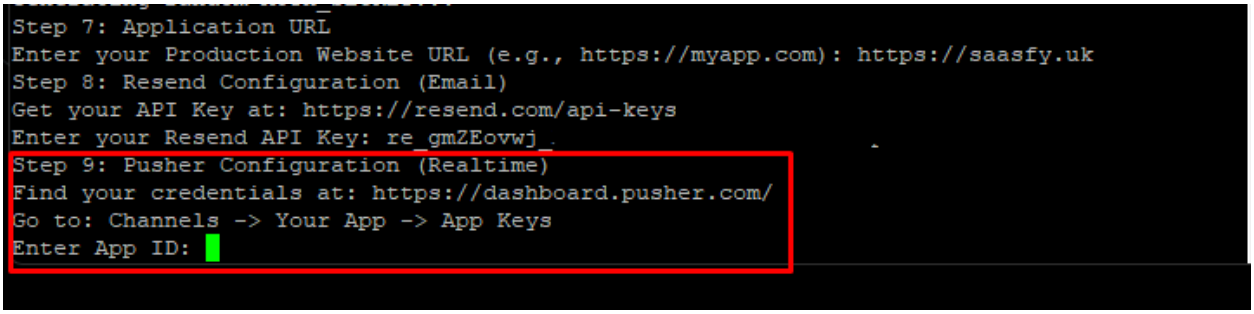
- Navigate to API Keys and create a new key.



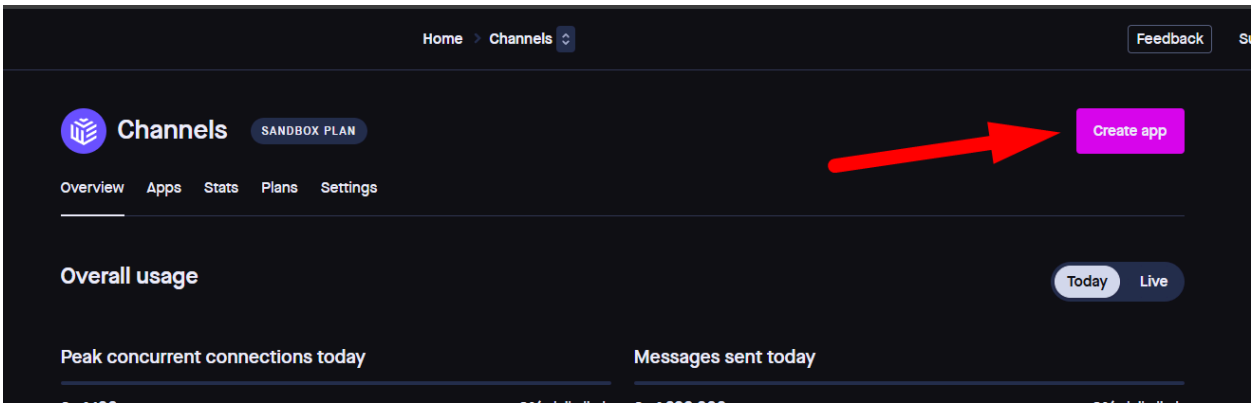


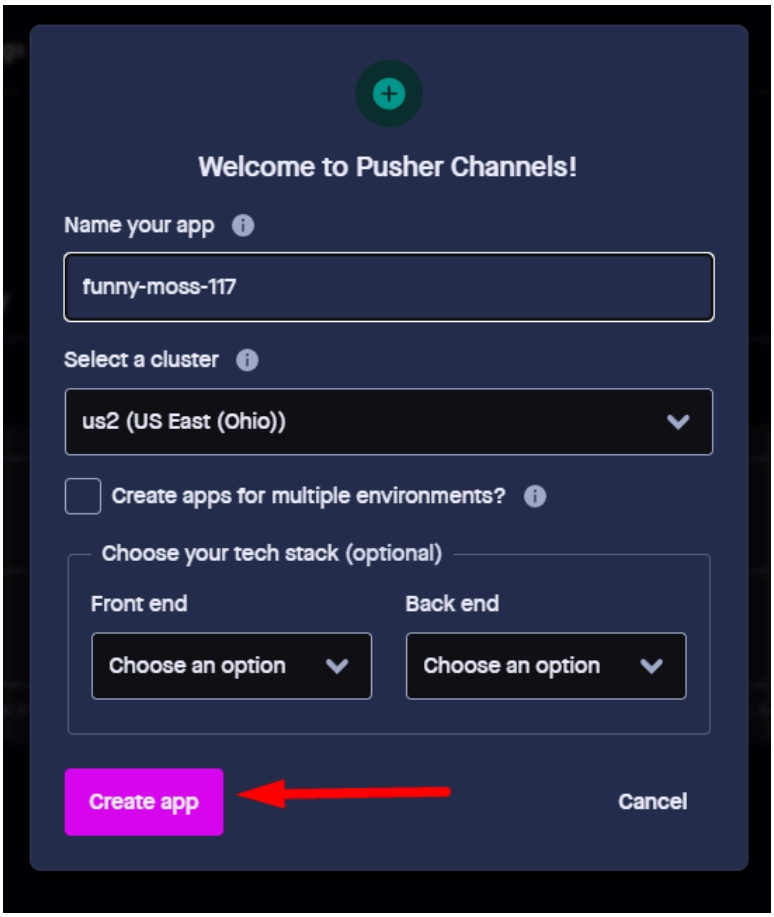
✅ Copy the key and paste it into the setup.

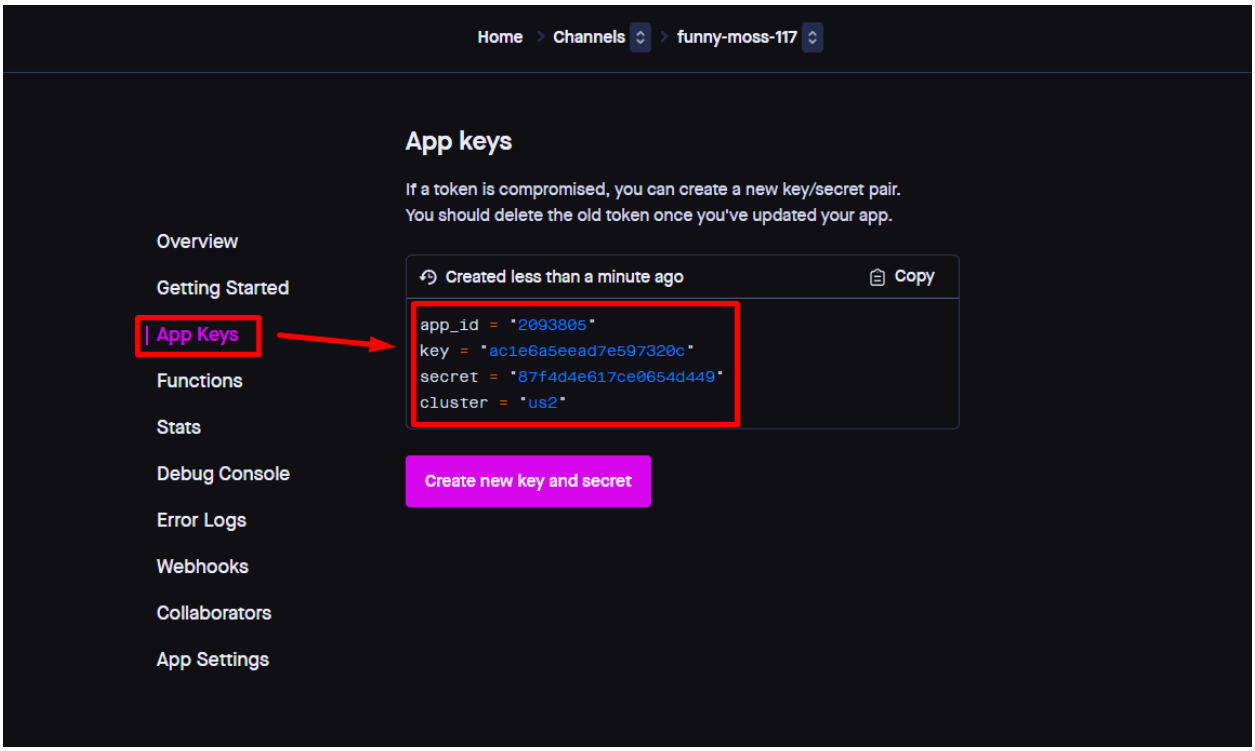
6. Pusher Configuration



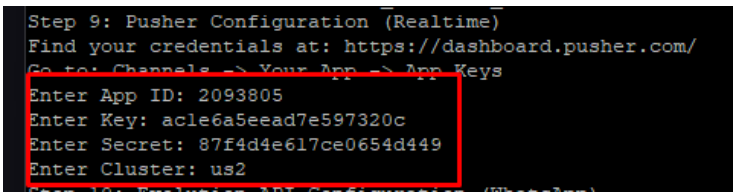
Find your credentials at: https://dashboard.pusher.com/channels


Home
Channels
Feedback
Channels
SANDBOX PLAN
Create app
Overview
Apps
Stats
Plans
Settings
Overall usage
Today
Live
Peak concurrent connections today
Messages sent today


Welcome to Pusher Channels!
Name your app
funny-moss-117
Select a cluster
us2 (US East (Ohio))
Create apps for multiple environments?
Choose your tech stack (optional)
Front end
Back end
Choose an option
Choose an option
Create app
Cancel

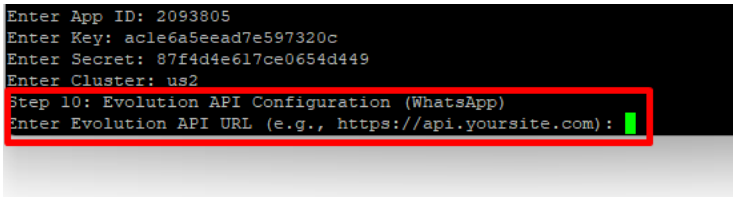


📌 Paste the credentials as requested by the setup:
app_id;
key;
secret;
cluster.



✅ Pusher Configuration is done.

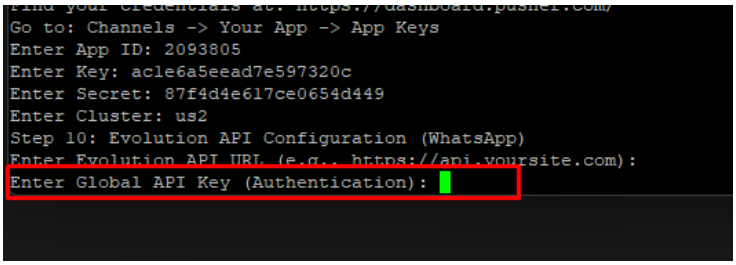
7. Enter Evolution API URL:



Enter the Evolution URL that we configured earlier:
e.g., https://evolution.saasfy.uk

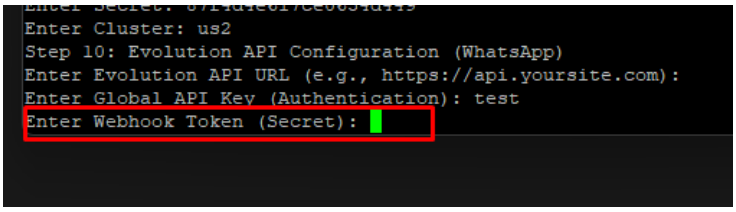
8. Enter `AUTHENTICATION_API_KEY`

> 📌 Remember that we saved a variable in the Evolution configuration.



> ⚠️ Do not type the variable name, but rather the value assigned to it.

9. Enter `WA_BUSINESS_TOKEN_WEBHOOK`



> ⚠️ Do not type the variable name, but rather the value assigned to it.

🎉 Setup completed successfully!

☕ grab a coffee and move on to part 4.

4️⃣ Database Migrations & Seeding

Step 4.1: Create tables

```
pnpm db:migrate
```

`pnpm drizzle-kit push` ⚠️ palliative

Seed the database (Creates Admin user: test@test.com / admin123)

```
pnpm db:seed
```

> 📌 *Note: The seeding process sets the user role to 'admin' and creates a default Team.*

5️⃣ Build and Start

Compile the Next.js application and start it with PM2.

```
pnpm build
pm2 start "npx next start" --name saasfy

pm2 startup
pm2 save
```

5.1: Configure cron to trigger campaigns asynchronously.

Read the token from the .env file:

```
grep CRON_SECRET /home/whats-saas/.env
```

Configure your VPS cron job to trigger campaigns asynchronously.

Use in crontab:

```
crontab -e
```

```
* * * * * curl -s -H "Authorization: Bearer YOUR CRON_SECRET HERE"
http://localhost:3000/api/campaigns/process > /dev/null 2>&1
```

6️⃣ Nginx Configuration & Static Files (Critical)

In production, Next.js does not serve files uploaded at runtime (to the `public` folder) efficiently. We must configure Nginx to serve these files directly from the disk.

Step 6.1: Configure Nginx

`sudo nano /etc/nginx/sites-available/saasfy`

Configuration:

```
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com; //⚠⚠⚠ replace to your domain

    # --- UPLOADS BLOCK (Serve images directly) ---
    location /uploads/ {
        # Absolute path to your project's public/uploads folder
        alias /home/your-user/your-project-folder/public/uploads/;

        # Security & Performance
        autoindex off;
        expires 30d;
        access_log off;
        add_header Access-Control-Allow-Origin *;
        add_header X-Content-Type-Options nosniff;

        # Block malicious scripts
        location ~ \.(php|pl|py|sh|cgi|html)$ {
            return 403;
        }
    }

    # --- MAIN APP BLOCK (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_cache_bypass $http_upgrade;

        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;
    }
}
```

Step 6.2: Activate and Secure

Enable the site and generate SSL for the main domain.

```
sudo ln -s /etc/nginx/sites-available/saasfy /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

`sudo certbot --nginx -d yourdomain.com -d www.yourdomain.com` ⚠️⚠️⚠️ replace to your domain

Step 6.3: Permission Fixes (Security)

Ensure Nginx can read the images, but secure your `.env` file from prying eyes.

Bash

1. Protect sensitive files (Only owner can read/write)

   ```
   chmod 600 /home/your-user/your-project-folder/.env
   ```

2. Allow system to traverse parent folders (Execute only)

   ```
   chmod 711 /home/your-user
   ```

   ```
   chmod 711 /home/your-user/your-project-folder
   ```

3. Allow Nginx to read public assets

   ```
   chmod 755 /home/your-user/your-project-folder/public
   chmod -R 755 /home/your-user/your-project-folder/public/uploads
   ```

🎉 INSTALLATION COMPLETED SUCCESSFULLY!

WhatSaaS has been successfully deployed and configured on the server. You should have something like this:

📍 Access Links:

- Main Application: https://saasfy.uk
- Evolution API: https://evolution.saasfy.uk

🔑 Admin Credentials (Seed):

- Email: test@test.com
- Password: admin123

🛠️ Next Steps:

1. Access the dashboard and log in.
2. Configure your plans in https://saasfy.uk/admin
3. Connect your first WhatsApp instance.

⚠️ Security: Remember to change the admin password immediately.