WhiteListChecker Installation Guide

Docker & Podman Setup for Linux and Windows

Version: 2.0.0 **Last Updated:** February 8, 2026

Table of Contents

Overview

WhiteListChecker is a WebAPI application for verifying entities against the Polish Ministry of Finance VAT taxpayer whitelist (Wykaz podatników VAT). The application automatically downloads and processes the whitelist data file published daily by the Ministry of Finance at plikplaski.mf.gov.pl.

Key Information:

- Application runs on internal port **8080**
- Default external port mapping: **5000**
- Health check endpoint: `/api/version`
- Swagger UI: `/swagger`
- Data source: plikplaski.mf.gov.pl (daily flat file)

Prerequisites

Hardware Requirements

Resource	Minimum
CPU	2-core processor
RAM	4 GB (the whitelist data file is large)
Disk	2 GB free space (for data files and archives)

Software Requirements

- **Docker** Engine 20.10+ or **Podman** 4.0+
- **Docker Compose** or **Podman Compose** (optional, recommended)
- Internet access to plikplaski.mf.gov.pl (for daily data downloads)

For building from source:

- .NET SDK 10.0

GitHub Container Registry Authentication

WhiteListChecker image is hosted on GitHub Container Registry (ghcr.io). You need to authenticate before pulling the image.

Creating a GitHub Personal Access Token

1. Go to GitHub: https://github.com/settings/tokens
2. Click **"Generate new token"** → **"Generate new token (classic)"**
3. Configure the token:
   - **Note:** `WhiteListChecker Docker Access`
   - **Expiration:** Choose appropriate duration (e.g., 90 days, 1 year, or no expiration)
   - **Scopes:** Select `read:packages` (required to pull images)
4. Click **"Generate token"**
5. **IMPORTANT:** Copy the token immediately - you won't be able to see it again!

Get access to package

After creating account please send request to grzegorz.giewon@uptime-erp.com with username and email of your account.

Authentication Methods

Method 1: Docker Login (Recommended)

Linux:

```
# Login to GitHub Container Registry
echo "YOUR_GITHUB_TOKEN" | docker login ghcr.io -u YOUR_GITHUB_USERNAME --p

# Verify login
docker login ghcr.io
```

Windows PowerShell:

```
# Set variables
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_USERNAME = "YOUR_GITHUB_USERNAME"

# Login
$env:GITHUB_TOKEN | docker login ghcr.io -u $env:GITHUB_USERNAME --password

# Clear token from environment for security
Remove-Item Env:\GITHUB_TOKEN
```

Success message:

```
Login Succeeded
```

Method 2: Podman Login

Linux:

```
# Login to GitHub Container Registry
echo "YOUR_GITHUB_TOKEN" | podman login ghcr.io -u YOUR_GITHUB_USERNAME --p
```

```
# Verify login
podman login ghcr.io
```

Windows PowerShell:

```
# Login
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_TOKEN | podman login ghcr.io -u YOUR_GITHUB_USERNAME --password
Remove-Item Env:\GITHUB_TOKEN
```

Troubleshooting Authentication

Error: "unauthorized: authentication required"

Solution:

```
# Verify you're logged in
docker login ghcr.io
# or
podman login ghcr.io

# If not logged in, authenticate again with your token
echo "YOUR_TOKEN" | docker login ghcr.io -u YOUR_USERNAME --password-stdin
```

Error: "denied: permission_denied"

Possible causes:

1. Token doesn't have `read:packages` permission
2. You don't have access to the repository
3. Repository is private and you're not a collaborator

Solution:

- Generate a new token with correct permissions
- Contact repository owner for access

Verify Authentication Status

```
# Docker
cat ~/.docker/config.json | grep ghcr.io

# Podman (Linux)
cat ~/.config/containers/auth.json | grep ghcr.io

# Windows PowerShell (Docker)
Get-Content "$env:USERPROFILE\.docker\config.json" | Select-String "ghcr.io
```

Security Best Practices

1. **Never commit tokens to Git**

   ```
   echo ".env" >> .gitignore
   ```

2. **Use environment variables**

   ```
   # Linux - add to ~/.bashrc or ~/.profile
   export GITHUB_TOKEN="your_token_here"
   ```

3. **Rotate tokens regularly**

   - Set expiration dates on tokens
   - Generate new tokens every 90 days
   - Revoke old tokens after rotation

4. **Use minimal permissions**

   - Only grant `read:packages` for pulling images

Alternative: Using Image as .tar File

If you cannot authenticate or prefer offline distribution:

1. **On machine with access:**

```
docker pull ghcr.io/ggiewon/whitelistchecker-webapi:develop
docker save ghcr.io/ggiewon/whitelistchecker-webapi:develop -o whitelis
gzip whitelistchecker-webapi.tar
```

2. **Transfer file to target machine** (USB, SCP, etc.)

3. **On target machine:**

```
docker load -i whitelistchecker-webapi.tar.gz
```

Linux Installation

Prerequisites

Update your system first:

```
sudo apt-get update
sudo apt-get upgrade -y
```

Linux Docker

Step 1: Install Docker and Docker Compose

```
# Install Docker
sudo apt-get install -y docker.io docker-compose

# Start Docker service
sudo systemctl start docker
sudo systemctl enable docker

# Add current user to docker group (to avoid using sudo)
sudo usermod -aG docker $USER
```

```
# IMPORTANT: Log out and log back in for group changes to take effect
```

Step 2: Verify Installation

After logging back in:

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

Step 3: Create Working Directory

```
mkdir -p ~/WhiteListChecker
cd ~/WhiteListChecker
```

Step 4: Create docker-compose.yml File

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

Paste the following content:

```
services:
  whitelistchecker-api:
    image: ghcr.io/ggiewon/whitelistchecker-webapi:develop
    container_name: whitelistchecker-api
    ports:
      - "5000:8080"
    volumes:
      - ./DataFiles:/app/DataFiles
      - ./Downloads:/app/Downloads
      - ./logs:/app/logs
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
```

```
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/api/version"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
    networks:
      - whitelistchecker-network

networks:
  whitelistchecker-network:
    driver: bridge
```

Save and exit (Ctrl+X, then Y, then Enter).

Step 5: Create Data Directories

```
mkdir -p DataFiles Downloads logs
chmod 755 DataFiles Downloads logs
```

Step 6: Verify Directory Structure

```
ls -la
```

Expected structure:

```
WhiteListChecker/
├── docker-compose.yml
├── DataFiles/
├── Downloads/
└── logs/
```

Step 7: Load Docker Image

Option A: Pull from GitHub Container Registry (requires authentication):

```
# First, authenticate (if not already done)
echo "YOUR_GITHUB_TOKEN" | docker login ghcr.io -u YOUR_GITHUB_USERNAME --p

# Pull the image
docker-compose pull
```

Option B: Load from .tar file (offline method):

```
docker load -i whitelistchecker-webapi.tar.gz
docker images | grep whitelistchecker
```

Step 8: Start the Application

```
docker-compose up -d
```

Step 9: Verify Application is Running

```
# Check container status
docker-compose ps

# Check logs
docker-compose logs -f whitelistchecker-api
```

Step 10: Test the Application

```
# Application version
curl http://localhost:5000/api/version

# Swagger UI (open in browser)
# http://localhost:5000/swagger
```

Linux Podman

Step 1: Install Podman

```
# Ubuntu 20.10 and newer
sudo apt-get install -y podman

# Install podman-compose
sudo apt-get install -y python3-pip
pip3 install podman-compose
```

Step 2: Verify Installation

```
podman --version
podman-compose --version
```

Step 3: Create Working Directory

```
mkdir -p ~/WhiteListChecker
cd ~/WhiteListChecker
```

Step 4: Create podman-compose.yml File

```
nano podman-compose.yml
```

Paste the following content (same format as docker-compose.yml):

```
services:
  whitelistchecker-api:
    image: ghcr.io/ggiewon/whitelistchecker-webapi:develop
    container_name: whitelistchecker-api
    ports:
      - "5000:8080"
```

```
    volumes:
      - ./DataFiles:/app/DataFiles
      - ./Downloads:/app/Downloads
      - ./logs:/app/logs
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/api/version"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
```

Step 5: Create Data Directories

```
mkdir -p DataFiles Downloads logs
```

Step 6: Load Image and Start

```
# Authenticate
echo "YOUR_GITHUB_TOKEN" | podman login ghcr.io -u YOUR_GITHUB_USERNAME --p

# Pull and start
podman-compose pull
podman-compose up -d
```

Step 7: Verify and Test

```
podman-compose ps
podman-compose logs -f whitelistchecker-api
curl http://localhost:5000/api/version
```

Windows Installation

Windows Docker

> **IMPORTANT LICENSING NOTICE**
>
> **Docker Desktop is FREE for:**
>
> - Personal use
> - Small businesses (fewer than 250 employees AND less than $10 million in annual revenue)
> - Education and non-commercial open source projects
>
> **Docker Desktop requires a PAID SUBSCRIPTION for:**
>
> - Large businesses (250+ employees OR $10 million+ annual revenue)
>
> **Alternative:** Consider **Podman Desktop** (see Windows Podman section) which is completely free and open source.

Step 1: Install Docker Desktop

1. Download Docker Desktop from: https://www.docker.com/products/docker-desktop
2. Run the installer
3. Restart your computer when prompted
4. Launch Docker Desktop from Start menu
5. Wait for Docker to start

Step 2: Verify Installation

Open PowerShell and run:

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

Step 3: Create Working Directory

```
New-Item -Path "C:\WhiteListChecker" -ItemType Directory
Set-Location -Path "C:\WhiteListChecker"
```

Step 4: Create docker-compose.yml File

```
notepad docker-compose.yml
```

Paste the following content:

```
services:
  whitelistchecker-api:
    image: ghcr.io/ggiewon/whitelistchecker-webapi:develop
    container_name: whitelistchecker-api
    ports:
      - "5000:8080"
    volumes:
      - ./DataFiles:/app/DataFiles
      - ./Downloads:/app/Downloads
      - ./logs:/app/logs
    environment:
      - ASPNETCORE_ENVIRONMENT=Production
      - ASPNETCORE_URLS=http://+:8080
    restart: unless-stopped
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/api/version"]
      interval: 30s
      timeout: 3s
      retries: 3
      start_period: 10s
    networks:
      - whitelistchecker-network

networks:
  whitelistchecker-network:
    driver: bridge
```

Save and close Notepad.

Step 5: Create Data Directories

```
New-Item -Path ".\DataFiles" -ItemType Directory
New-Item -Path ".\Downloads" -ItemType Directory
New-Item -Path ".\logs" -ItemType Directory
```

Step 6: Load Docker Image

```
# Authenticate
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_TOKEN | docker login ghcr.io -u YOUR_GITHUB_USERNAME --password
Remove-Item Env:\GITHUB_TOKEN

# Pull the image
docker-compose pull
```

Step 7: Start the Application

```
docker-compose up -d
```

Step 8: Verify and Test

```
docker-compose ps
docker-compose logs whitelistchecker-api

# Test endpoints
Invoke-WebRequest -Uri http://localhost:5000/api/version
```

In web browser:

- Swagger UI: http://localhost:5000/swagger

Windows Podman

> **FREE AND OPEN SOURCE**
>
> Podman Desktop is completely **free** for all users and use cases:
>
> - No licensing restrictions
> - Open source (Apache 2.0 license)
> - Suitable for personal, educational, and commercial use

Step 1: Install Podman Desktop

1. Download Podman Desktop from: https://podman-desktop.io/downloads
2. Run the installer
3. Launch Podman Desktop from Start menu
4. Complete the initial setup wizard (Initialize Podman Machine)

Step 2: Start Podman Machine

```
podman machine init
podman machine start
podman --version
```

Step 3: Create Working Directory and Compose File

```
New-Item -Path "C:\WhiteListChecker" -ItemType Directory
Set-Location -Path "C:\WhiteListChecker"
notepad docker-compose.yml
```

Use the same docker-compose.yml content as in the Docker Desktop section above.

Step 4: Create Data Directories

```
New-Item -Path ".\DataFiles" -ItemType Directory
New-Item -Path ".\Downloads" -ItemType Directory
New-Item -Path ".\logs" -ItemType Directory
```

Step 5: Load Image and Start

```
$env:GITHUB_TOKEN = "YOUR_GITHUB_TOKEN"
$env:GITHUB_TOKEN | podman login ghcr.io -u YOUR_GITHUB_USERNAME --password
Remove-Item Env:\GITHUB_TOKEN

podman-compose pull
podman-compose up -d
```

Step 6: Verify and Test

```
podman ps
podman logs whitelistchecker-api
Invoke-WebRequest -Uri http://localhost:5000/api/version
```

Common Operations

Viewing Logs

Docker:

```
docker-compose logs -f whitelistchecker-api
docker-compose logs --tail=100 whitelistchecker-api
docker-compose logs --since 1h whitelistchecker-api
```

Podman:

```
podman logs -f whitelistchecker-api
podman logs --tail 100 whitelistchecker-api
```

Application log files are also available in the `logs/` volume.

Stopping the Application

```
# Stop without removing container
docker-compose stop

# Stop and remove container (data in volumes remains)
docker-compose down
```

Restarting the Application

```
docker-compose restart

# Or recreate container
docker-compose up -d --force-recreate
```

Updating the Application

```
# 1. Stop and remove old container
docker-compose down

# 2. Pull new image
docker-compose pull

# 3. Start new version
docker-compose up -d

# 4. Check logs
docker-compose logs -f whitelistchecker-api
```

Checking Resource Usage

```
docker stats whitelistchecker-api
```

Configuration

Configuration Files

WhiteListChecker uses two configuration files:

1. `appsettings.json` - Main configuration file (built into the Docker image)
2. `appsettings.additional.json` - Optional override file (can be mounted as a volume)

Configuration is also overridable via **environment variables**.

HttpClient Settings

Controls the HTTP client timeout for downloading whitelist data files.

```
"HttpClient": {
  "TimeoutMinutes": 30
}
```

Parameter	Description	Default
`TimeoutMinutes`	HTTP request timeout in minutes for file downloads	`30`

Environment variable: `HttpClient__TimeoutMinutes=30`

Download Settings

Controls the automatic download behavior of the background `DownloadService`.

```
"Download": {
  "AlertAfterHour": 10
}
```

Parameter	Description	Default

`AlertAfterHour`	Hour after which a notification is sent if the daily data file has not been downloaded	`10`

The `DownloadService` runs every **10 minutes** and attempts to download the current day's data file from plikplaski.mf.gov.pl. If the file is already present, the download is skipped. Failed downloads are retried up to 3 times with exponential backoff.

Environment variable: `Download__AlertAfterHour=10`

Maintenance Settings

Controls automatic cleanup of old data files.

```
"Maintenance": {
  "RetentionDays": 4
}
```

Parameter	Description	Default
`RetentionDays`	Number of days to retain data files before automatic deletion	`4`

The `MaintenanceService` runs every **12 hours** and deletes data folders older than the configured retention period.

Environment variable: `Maintenance__RetentionDays=4`

Notifications - Email

Configures email notifications for download events and errors.

```
"Notifications": {
  "Email": {
    "Enabled": false,
    "SmtpHost": "",
    "SmtpPort": 587,
    "UseSsl": true,
    "Username": "",
    "Password": "",
```

```
    "FromAddress": "",
    "FromName": "WhiteListChecker",
    "Recipients": []
  }
}
```

Parameter	Description	Default
`Enabled`	Enable/disable email notifications	`false`
`SmtpHost`	SMTP server hostname	`""`
`SmtpPort`	SMTP server port	`587`
`UseSsl`	Use SSL/TLS for SMTP connection	`true`
`Username`	SMTP authentication username	`""`
`Password`	SMTP authentication password	`""`
`FromAddress`	Sender email address	`""`
`FromName`	Sender display name	`"WhiteListChecker"`
`Recipients`	Array of recipient email addresses	`[]`

Example with environment variables:

```
environment:
  - Notifications__Email__Enabled=true
  - Notifications__Email__SmtpHost=smtp.example.com
  - Notifications__Email__SmtpPort=587
  - Notifications__Email__Username=user@example.com
  - Notifications__Email__Password=secret
  - Notifications__Email__FromAddress=whitelistchecker@example.com
  - Notifications__Email__Recipients__0=admin@example.com
```

Notifications - Teams

Configures Microsoft Teams notifications via webhook.

```
"Notifications": {
  "Teams": {
    "Enabled": false,
    "WebhookUrl": ""
  }
}
```

Parameter	Description	Default
`Enabled`	Enable/disable Teams notifications	`false`
`WebhookUrl`	Microsoft Teams Incoming Webhook URL	`""`

Environment variable: `Notifications__Teams__WebhookUrl=https://...`

Logging

Controls application logging levels.

```
"Logging": {
  "LogLevel": {
    "Default": "Trace",
    "Microsoft": "Information"
  }
}
```

The application uses **NLog** for logging. Log files are written to the `logs/` directory.

Overriding Configuration

There are three ways to override configuration:

1. Using `appsettings.additional.json` file

Create the file in the application directory (or mount it via Docker volume):

```
{
  "Download": {
```

```
    "AlertAfterHour": 12
  },
  "Notifications": {
    "Email": {
      "Enabled": true,
      "SmtpHost": "smtp.example.com",
      "SmtpPort": 587,
      "Username": "user@example.com",
      "Password": "secret",
      "FromAddress": "whitelistchecker@example.com",
      "Recipients": ["admin@example.com"]
    }
  }
}
```

Docker Compose volume mount example:

```
volumes:
  - ./appsettings.additional.json:/app/appsettings.additional.json:ro
  - ./DataFiles:/app/DataFiles
  - ./Downloads:/app/Downloads
  - ./logs:/app/logs
```

2. Using environment variables

Use double underscore (`__`) as separator for nested keys:

```
environment:
  - Notifications__Email__Enabled=true
  - Maintenance__RetentionDays=7
```

3. Combining both methods

Environment variables take precedence over `appsettings.additional.json`, which takes precedence over `appsettings.json`.

Volumes

The application uses three persistent storage directories:

Volume	Container Path	Description
DataFiles	`/app/DataFiles`	Extracted whitelist data files (one subfolder per date in `yyyyMMdd` format)
Downloads	`/app/Downloads`	Temporary storage for downloaded 7z archive files
logs	`/app/logs`	Application log files (NLog)

Docker Compose example:

```
volumes:
  - ./DataFiles:/app/DataFiles
  - ./Downloads:/app/Downloads
  - ./logs:/app/logs
```

> **NOTE:** The `DataFiles` directory grows daily with new whitelist files. Old files are automatically cleaned up by the `MaintenanceService` based on the `RetentionDays` setting (default: 4 days).

Troubleshooting

Problem: Port 5000 Already in Use

Linux:

```
sudo netstat -tlnp | grep 5000
```

Solution: Change port in docker-compose.yml:

```
ports:
```

```
    - "5001:8080"  # Changed from 5000 to 5001
```

Then restart:

```
docker-compose down
docker-compose up -d
```

Problem: Container Won't Start

```
docker-compose logs whitelistchecker-api
```

Problem: Data File Not Downloaded

Check logs for download errors:

```
docker-compose logs whitelistchecker-api | grep -i "download"
```

Common causes:

- No internet access to plikplaski.mf.gov.pl
- HTTP timeout (increase `HttpClient:TimeoutMinutes`)
- Disk full (check available space)

Manual download trigger:

```
# Trigger manual download for today
curl http://localhost:5000/api/download

# Trigger manual download for specific date
curl "http://localhost:5000/api/download?date=20260208"
```

```
# Force re-download even if file exists
curl "http://localhost:5000/api/download?date=20260208&force=true"
```

Problem: Permission Denied (Linux)

```
# Check if user is in docker group
groups

# If "docker" is not in the list:
sudo usermod -aG docker $USER
# Log out and log back in
```

Problem: Application Not Responding

```
# Check if container is running
docker-compose ps

# Check health status
docker inspect --format='{{.State.Health.Status}}' whitelistchecker-api

# Test from inside container
docker exec whitelistchecker-api curl http://localhost:8080/api/version
```

Problem: Container Keeps Restarting

```
# View recent logs
docker-compose logs --tail=100 whitelistchecker-api

# Run interactively to see errors
docker-compose down
docker-compose up
# (without -d flag to see logs in real-time)
```

Production Considerations

Security Best Practices

```
# Configure firewall - allow only local network
sudo ufw allow from 192.168.1.0/24 to any port 5000
sudo ufw enable
```

Using Reverse Proxy (Linux - Nginx)

```
sudo apt-get install nginx
sudo nano /etc/nginx/sites-available/whitelistchecker
```

Add the following configuration:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:5000;
        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;
    }
}
```

Enable and restart:

```
sudo ln -s /etc/nginx/sites-available/whitelistchecker /etc/nginx/sites-ena
sudo nginx -t
sudo systemctl restart nginx
```

SSL/TLS Configuration

```
sudo apt-get install certbot python3-certbot-nginx
sudo certbot --nginx -d your-domain.com
```

Backup and Restore

Backup

Linux:

```
cd ~/WhiteListChecker
tar -czf whitelistchecker-backup-$(date +%Y%m%d).tar.gz DataFiles/ Download
```

Windows:

```
Compress-Archive -Path "C:\WhiteListChecker\DataFiles","C:\WhiteListChecker
    -DestinationPath "C:\Backups\whitelistchecker-$(Get-Date -Format 'yyyyM
```

Restore

```
docker-compose down
tar -xzf whitelistchecker-backup-20260208.tar.gz
docker-compose up -d
```

Monitoring

Set up Log Rotation (Linux - Docker)

```
sudo nano /etc/docker/daemon.json
```

Add:

```
{
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "10m",
    "max-file": "3"
  }
}
```

Restart Docker:

```
sudo systemctl restart docker
```

Continuous Health Monitoring

```
#!/bin/bash
while true; do
    if ! curl -f -s http://localhost:5000/api/version > /dev/null; then
        echo "$(date): WhiteListChecker is down, restarting..."
        cd ~/WhiteListChecker
        docker-compose restart
    fi
    sleep 60
done
```

Auto-start on System Boot

Docker

Docker is already set to auto-start. The container will automatically restart due to `restart: unless-stopped` policy.

Podman with Systemd

```
cd ~/WhiteListChecker
podman generate systemd --name whitelistchecker-api --files --new

mkdir -p ~/.config/systemd/user/
mv container-whitelistchecker-api.service ~/.config/systemd/user/

systemctl --user enable container-whitelistchecker-api.service
systemctl --user start container-whitelistchecker-api.service
loginctl enable-linger $USER
```

Resource Limits

Add resource limits to docker-compose.yml:

```
services:
  whitelistchecker-api:
    # ... existing configuration ...
    deploy:
      resources:
        limits:
          cpus: '2'
          memory: 4G
        reservations:
          cpus: '1'
          memory: 2G
```

Building from Source

Prerequisites

- .NET SDK 10.0
- Git

Steps

```
# Clone repository
git clone https://github.com/ggiewon/WhiteListChecker.git
cd WhiteListChecker/Source

# Restore dependencies
dotnet restore

# Build
dotnet build --configuration Release

# Run
dotnet run --project WhiteListChecker.WebAPI
```

The application will start on `http://localhost:5000` by default.

Building Docker Image

```
cd WhiteListChecker
docker build -f Docker/Dockerfile -t whitelistchecker-webapi:local .
```

Quick Reference

Docker Commands Cheatsheet

```
docker-compose up -d          # Start
docker-compose stop           # Stop
docker-compose down           # Remove
docker-compose restart        # Restart
docker-compose logs -f        # Logs
docker-compose ps             # Status
docker stats whitelistchecker-api  # Resources
```

Testing Endpoints

```
# Version / Health check
curl http://localhost:5000/api/version

# Check NIP+account against whitelist
curl -X POST http://localhost:5000/api/check \
  -F "Date=20260208" \
  -F "NipAccount=1234567890|PL12345678901234567890123456"

# Manual download
curl http://localhost:5000/api/download

# Clear old data
curl http://localhost:5000/api/clear

# Swagger UI (browser)
# http://localhost:5000/swagger
```

Support and Documentation

- **User Manual (PL):** WhiteListChecker_Instrukcja.pdf
- **User Manual (EN):** WhiteListChecker_Manual.pdf
- **Swagger Documentation:** http://localhost:5000/swagger

Document Version: 1.0 **Last Updated:** February 8, 2026 **Compatible with WhiteListChecker:** 2.0.0+