

README

Source:

Supabase: <https://supabase.com/>

RapidAPI: <https://rapidapi.com/>

Telegram @BotFather: <https://t.me/BotFather>

Google Cloud Platform: <https://console.cloud.google.com/>

AssemblyAI: <https://www.assemblyai.com/>

Author:

Petrinin Platon · GitHub: <https://github.com/petrinin-platon>

A Note from the Author: Important Operational Nuances

Hello! I'm glad you've decided to use this automation system. I've invested a lot of time and effort into making it as powerful and flexible as possible. Before you begin, I want to share a few key points that will help you get the most out of this project and avoid common issues.

1. Local vs. Cloud: A Hosting Recommendation

This workflow was primarily designed and tested for **local use** (on your home PC or server). The reason is simple: YouTube actively combats automated content downloading from server IP addresses.

- **The Problem:** When running on cloud servers (VDS/VPS), you will almost certainly encounter a **403 error** over time, which means the server's IP address has been temporarily or permanently blocked by YouTube.
- **The Solution:** Running the workflow locally generally avoids this issue. If you absolutely must use a cloud server, be prepared for potential blocks and the need for more complex solutions, such as residential proxies.

2. Cost and System Resources

This automation performs resource-intensive tasks, such as downloading and processing videos with `ffmpeg`.

- **The Advantage of Local Hosting:** By using your own computer, you only pay for what's essential—the API calls to the AI models (AssemblyAI, OpenRouter, etc.) for transcription and summary generation. All the "heavy lifting" is done by your hardware for free.

- **Cost in the Cloud:** On cloud servers, you are charged for CPU and disk space usage, and these operations can significantly increase your monthly expenses.
- **Execution Time:** Remember that processing time is directly dependent on two factors: the **quality** and **duration** of the video. The higher the resolution you choose and the longer the video, the longer the download, transcription, and analysis process will take. Be sure to select AI models with a large enough "context window" to handle the full text of long videos.

3. Language Support

In the current version, the section headers inside the final note (e.g., "📄 Structured Summary," "🔑 Key Takeaways") are translated into **Russian** and **English**.

- **How to Add Your Own Language:** You can easily add another language. To do this, open the workflow, find the `Build Final Note` node, and look for the `languageMaps` and `textMaps` dictionaries inside its code. Carefully add your language by following the existing structure, and the system will begin using it.

I hope these tips help you! Happy automating!

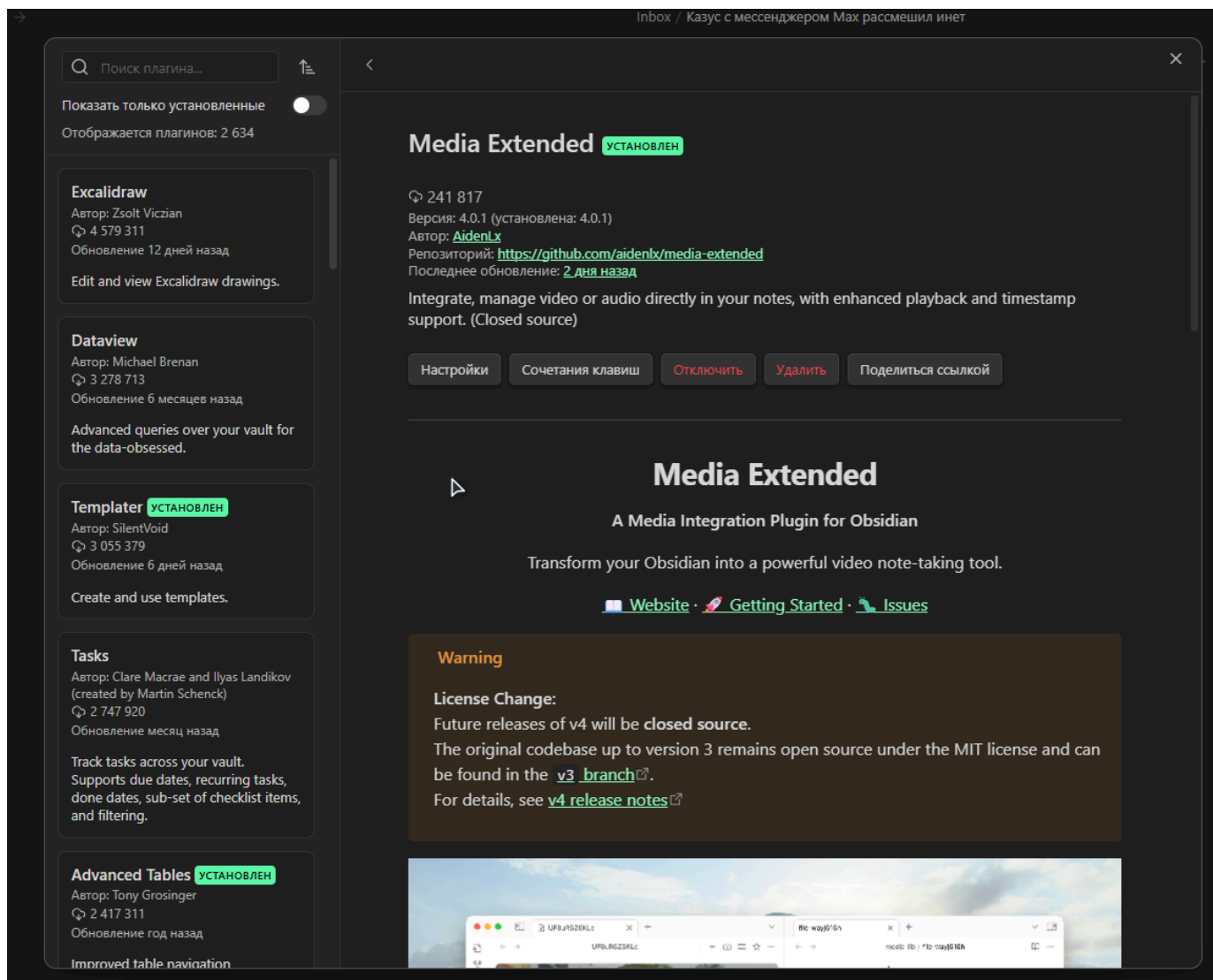
Installing the Obsidian Media Extended Plugin

What is it? This is a special community plugin for Obsidian that enhances its capabilities for working with video and audio.

Why is it needed? This is a critical step. Without this plugin, the clickable timestamp links (e.g., `[00:05:10]`) that our workflow creates **will not work**. It is this specific plugin that allows you to control the media player directly from the note, jumping to the correct moment in the video or audio.

- **How to install:**
 1. Open your Obsidian application.
 2. Go to **Settings** by clicking the gear icon in the bottom-left corner.
 3. In the left-hand menu, select **"Community plugins"**.
 4. If "Safe Mode" is on, you will need to turn it off by clicking the button.
 5. Click the **"Browse"** button to open the community plugins store.
 6. In the search bar, type: `Media Extended`.
 7. Once you've found the plugin, click the **"Install"** button.
 8. After it has installed, return to the previous page and **enable** the plugin by flipping the toggle switch next to its name.

✅ Done! Your Obsidian is now ready to correctly handle timestamped links.



Part 1: Prerequisites — Installing Everything You Need

Hello! 🙌 This guide will help you step-by-step to install and configure a powerful engine for processing information from YouTube. Before we begin, let's make sure you have all the necessary tools on your computer.

Step 1.1: Installing WSL2 (For Windows Users Only)

What is it? WSL2 is a special component from Microsoft that allows you to run a genuine Linux environment directly inside Windows.

Why is it needed? Docker Desktop (the program we'll need next) runs much faster and more stably on Windows using WSL2.

- **How to install:**

1. Click the **Start** button.
2. Type `PowerShell`.

3. "Windows PowerShell" will appear in the search results. Right-click on it and select **"Run as administrator"**.
4. In the blue window that opens, type or copy the following command and press `Enter` :

PowerShell

```
wsl --install
```

5. Wait for the installation to finish. Afterward, the computer will ask for a restart. Be sure to restart it.

✅ Done! Your Windows is now ready for Docker.

Step 1.2: Installing Docker Desktop

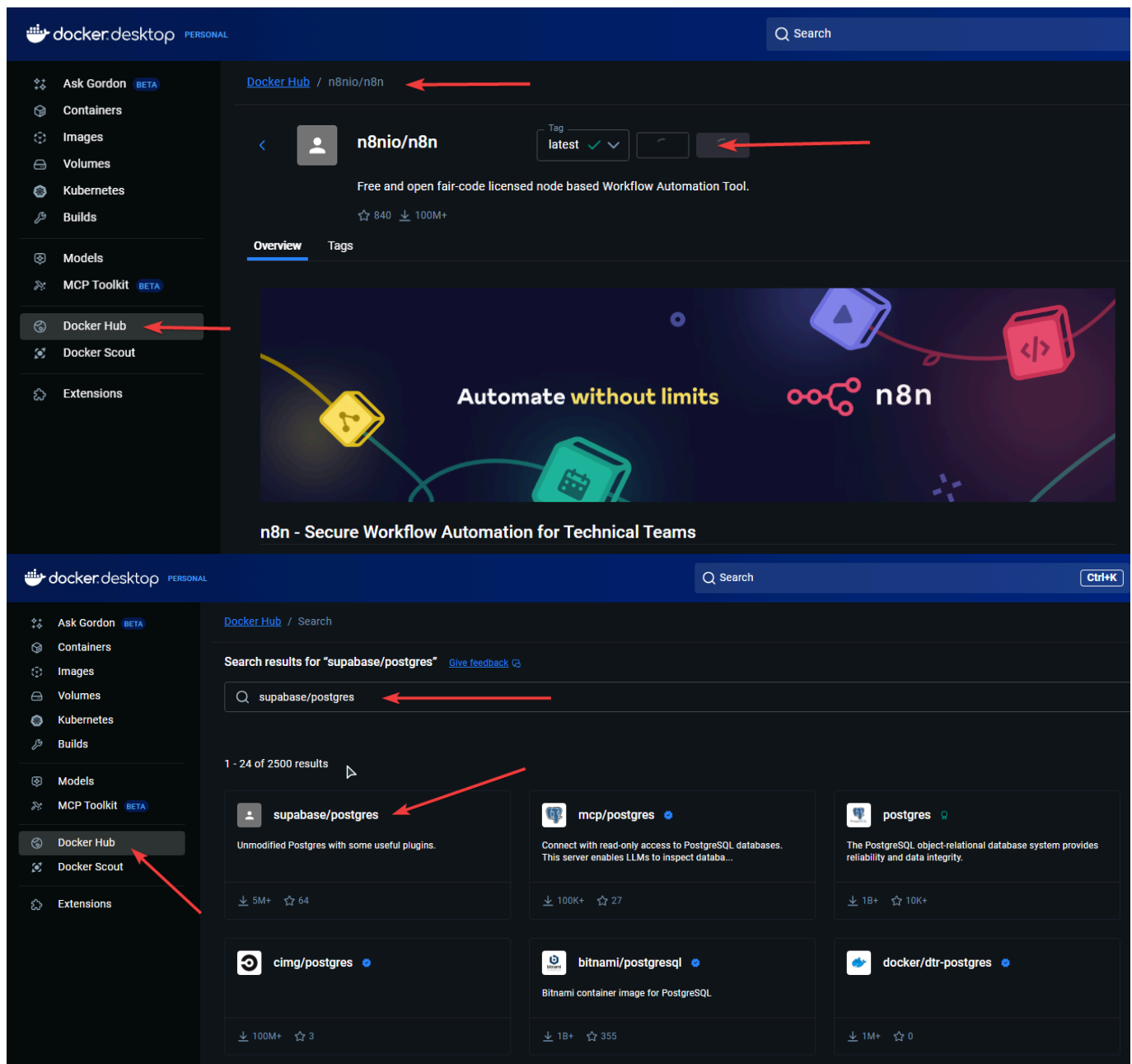
What is it? Imagine a "container" — it's like a fully equipped portable kitchen that has all the necessary ingredients and tools. In our case, n8n will live and work in this "container". Docker is the program that manages such containers.

Why is it needed? Thanks to Docker, our n8n with all its settings will work exactly the same on any computer, and you won't have to install dozens of dependencies manually.

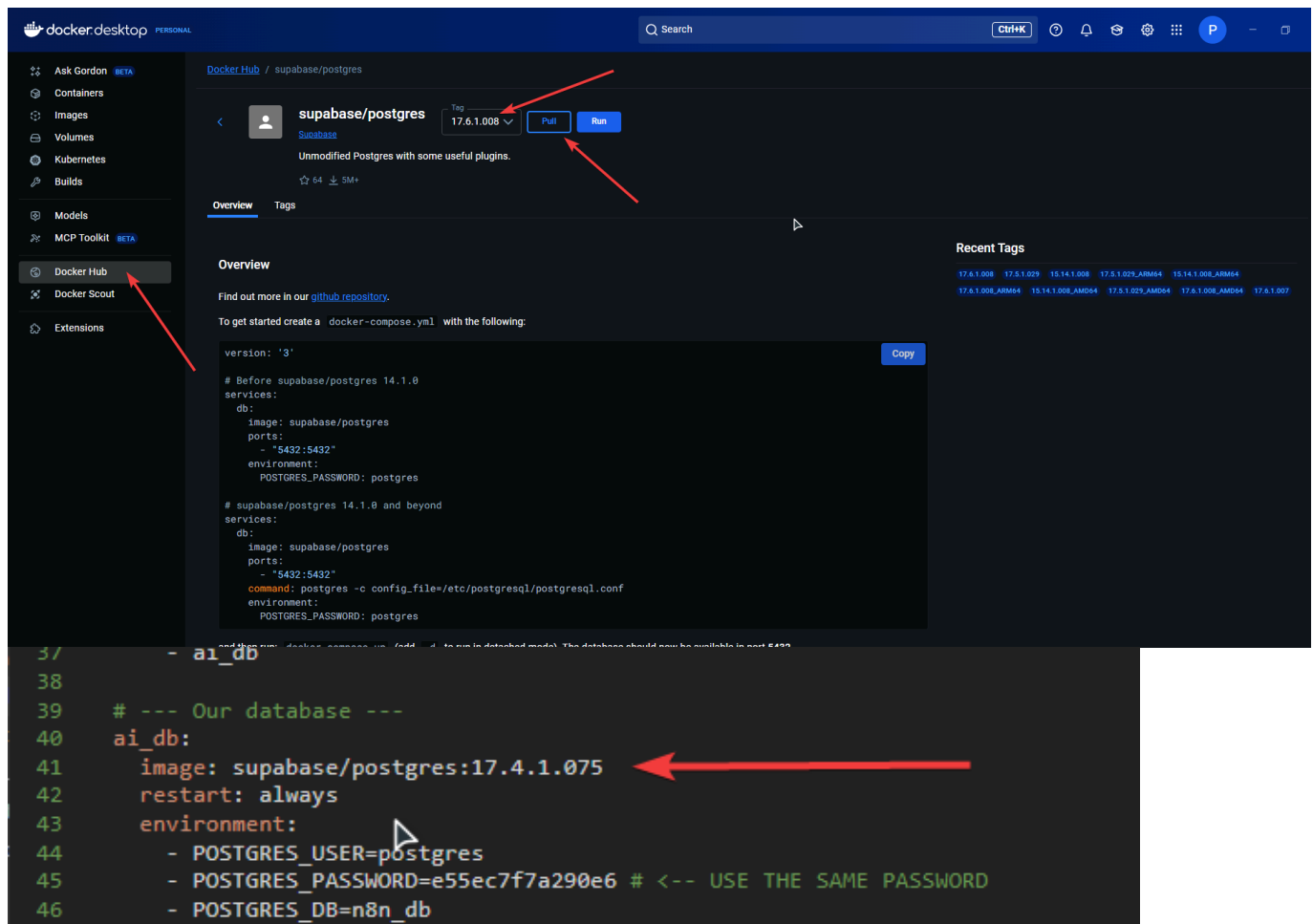
- **How to install:**

1. Go to the official website: [Docker Desktop](#).
2. Click the download button for your operating system (Windows or Mac).
3. Run the downloaded installer and follow the on-screen instructions. Just click "Next", leaving all the default settings.
4. After installation, Docker might ask for another restart.

✅ Done! You now have the system to run our project on your computer.



If you are installing Supabase, be sure to select the appropriate image and specify the image name and its tag in the `docker-compose.yml` file before running it.



Step 1.3: Text Editor (Recommended)

What is it? A program for conveniently editing text configuration files.

Why is it needed? Although you can use a basic "Notepad", a good code editor highlights syntax, shows line numbers, and generally makes working with files like `docker-compose.yml` much easier.

- **How to install:**

1. Go to the official website: [Visual Studio Code](#).
2. Download and install the program. It's fast and free.

✅ Done! You now have a handy tool for editing files.

Step 1.4: Project Files

What is it? These are the "blueprints" and "instructions" for our engine: the `n8n` template files and the Docker configuration files.

- **How to prepare:**

1. Create a folder for the entire project in a convenient location on your computer. Name it, for example, `n8n-project`.
2. Download all the files I will provide you (`.json` , `Dockerfile` , `docker-compose.yml`) and

place them inside this folder.

✓ Done! The foundation is laid.

Part 2: Project Structure and Docker Setup

On this step, we will prepare the "foundation" for our project — we will create the correct folder structure and configure the two main files: `Dockerfile` and `docker-compose.yml`.

Step 2.1: Final Preparation of Folders and Files

Before we begin, let's make sure all the files are in their proper places. Based on your information, here is the final structure we should have:

- `E:\n8n-project\` — this is your main project folder. It should contain the following files:
 - `Dockerfile`
 - `docker-compose.yml`
 - `start.bat`
 - `stop.bat`
 - `rebuild.bat`
- `E:\n8n-project\Obsidian\n8n-project\` — this is your Obsidian vault. Inside it, you already have:
 - `Inbox\` — the folder for finished `.md` notes.
 - `00_Files\` — the folder for downloaded videos and audio.

We will also need two more folders for Docker's technical data. **You do not need to create them manually; Docker will do it for you on the first launch!**

- `E:\n8n-project\n8n_data\` — this is where n8n itself will store its data (workflows, credentials).
- `E:\n8n-project\supabase_data\` — this is where the database will be stored.

Step 2.2: `Dockerfile` — The "Recipe" for Our n8n

A `Dockerfile` is a text file that contains a step-by-step instruction for Docker on how to build the n8n image we need. In our case, we are taking the official n8n image and adding the `yt-dlp` and `ffmpeg` utilities to it.

- **What to do:**
 1. Make sure the `Dockerfile` is located in your `E:\n8n-project` folder.
 2. Open it in a text editor (like VS Code) and ensure its content looks like this:

`Dockerfile`

```
# 1. Use the official n8n image as a base
FROM n8nio/n8n:latest

# 2. Switch to the root user to get permissions to install packages
USER root

# 3. Update the package list and install ffmpeg and yt-dlp
RUN apk update && apk add --no-cache \
    ffmpeg \
    yt-dlp

# 4. Switch back to the standard user for security
USER node
```

Step 2.3: docker-compose.yml — The Main Configuration File

This is the most important file. It tells Docker how to launch our project, what passwords to use, and most importantly, how to link the folders inside the "container" to your folders on the E: drive.

- **What to do:**

1. Open the docker-compose.yml file in your E:\n8n-project folder.
2. **Important Step: Replace Passwords.** Your file specifies passwords. For security, it is best to replace them with your own. You can use any password generator. You will need to generate **ONE** strong password and insert it in **TWO** places.
3. Replace the file's content with the following, inserting **YOUR PASSWORD** in the marked spots:

YAML

```
# Docker Compose syntax version
version: '3.8'

# Definition of all services (containers)
services:
  # --- Our n8n service ---
  n8n:
    # Tell Docker to build an image from the Dockerfile in the current folder
    build: .
    image: custom-n8n-engine:latest # The name of our custom image
    restart: always
    ports:
      - "5678:5678" # Forward the port so we can access n8n via a browser
    environment:
      - NODE_OPTIONS=--max-old-space-size=8192
      - N8N_DEFAULT_NODE_OPTIONS=--max-old-space-size=8192
      - N8N_PAYLOAD_SIZE_MAX=134217728
      - EXECUTIONS_DATA_PRUNE=true
```



```

- EXECUTIONS_DATA_MAX_AGE=168
- N8N_REQUEST_TIMEOUT=300000
- DB_TYPE=postgresdb
- DB_POSTGRESDB_HOST=ai_db
- DB_POSTGRESDB_PORT=5432
- DB_POSTGRESDB_USER=postgres
- DB_POSTGRESDB_PASSWORD=YOUR_SUPER_STRONG_PASSWORD # <-- REPLACE THIS
- DB_POSTGRESDB_DATABASE=n8n_db
- GENERIC_TIMEZONE=Europe/Minsk
volumes:
  # --- THE MOST IMPORTANT SECTION: FOLDER MAPPING (VOLUMES) ---
  # Format: 'PATH_ON_YOUR_COMPUTER': 'PATH_INSIDE_THE_CONTAINER'
  - './n8n_data:/home/node/.n8n' # n8n data
  - './Obsidian/n8n-project/Inbox:/inbox' # Folder for .md notes
  - './Obsidian/n8n-project/00_Files:/media' # Folder for video and audio
  - './temp:/temp' # Folder for temporary files
  - './cookies:/cookies'
depends_on:
  - ai_db

# --- Our database ---
ai_db:
  image: supabase/postgres:17.4.1.075
  restart: always
  environment:
    - POSTGRES_USER=postgres
    - POSTGRES_PASSWORD=YOUR_SUPER_STRONG_PASSWORD # <-- USE THE SAME
    PASSWORD
    - POSTGRES_DB=n8n_db
  volumes:
    - './supabase_data:/var/lib/postgresql/data' # Database data
  healthcheck:
    test: ["CMD-SHELL", "pg_isready -U postgres"]
    interval: 10s
    timeout: 5s
    retries: 5

```

Breaking Down the `volumes` Section (The Most Important Part!):

This section is the "bridge" between your computer and the isolated world of Docker. Let's break down what we wrote:

- `'./n8n_data:/home/node/.n8n'`
 - **What it means:** The `n8n_data` folder in your project (`E:\n8n-project\n8n_data`) becomes the `/home/node/.n8n` folder inside the container. This is where n8n stores all of its data.

- `'./Obsidian/n8n-project/Inbox:/inbox'`
 - **What it means:** Your folder for notes (`E:\n8n-project\Obsidian\n8n-project\Inbox`) becomes the `/inbox` folder inside the container. When n8n saves a note to `/inbox` , it will appear on your `E:` drive.
- `'./Obsidian/n8n-project/00_Files:/media'`
 - **What it means:** Your folder for media (`E:\n8n-project\Obsidian\n8n-project\00_Files`) becomes the `/media` folder inside the container.
- `'./temp:/temp'`
 - **What it means:** The `temp` folder in your project (`E:\n8n-project\temp`) becomes the `/temp` folder inside the container.

Now, when we configure the workflow in the Google Sheet, we will use the **paths inside the container:** `/inbox` , `/media` , and `/temp` .

Part 3: First Launch and n8n Setup

Congratulations! All the preparatory work is complete. Your folders and configuration files are ready. Now comes the most exciting moment—the first launch. At this stage, we will bring n8n to life and upload our templates into it.

Step 3.1: Launching the Project

For convenience, the project includes special scripts (`.bat` files) that execute all the necessary commands for you.

- **What to do:**
 1. Make sure the **Docker Desktop** application is running. You should see the whale icon in your taskbar.
 2. Open your main project folder: `E:\n8n-project` .
 3. Find the `rebuild.bat` file and **run it by double-clicking**.
- **What will happen?**
 - A black terminal window will appear. Don't worry, this is normal!
 - The **very first time**, Docker will start "building" your custom n8n image according to the instructions in the `Dockerfile` . It will download the official n8n image and install `yt-dlp` and `ffmpeg` inside it. This process can take **5-15 minutes** depending on your internet speed. Be patient.
 - After the build is complete, Docker will launch two containers: n8n itself and its database `ai_db` .
 - When everything is ready, you will see a success message, and the window can be closed.

Note: In the future, for a normal start (without rebuilding), use the `start.bat` file. To stop it, use `stop.bat`.

✓ Done! Your custom n8n is up and running in the background.

Step 3.2: Logging into n8n and Creating an Administrator

Now that n8n is running, you need to access its web interface and create the main user.

- **What to do:**

1. Open your favorite web browser (Chrome, Firefox, etc.).
2. In the address bar, type: `http://localhost:5678` and press `Enter`.
3. You will be greeted by the n8n new user setup page. Create and enter your details (name, email, password) to set up the owner account.
4. **Important:** Be sure to save this password in a safe place.

✓ Done! You are now logged into your freshly installed n8n.

Step 3.3: Importing the Workflow Templates

Right now, your n8n is empty. Let's upload our powerful templates into it.

- **What to do:**

1. In the left menu of the n8n interface, find and click on the **"Workflows"** section.
2. In the top right corner, you will see an **"Import"** button. Click it.
3. Select the **"Import from File"** option.
4. A file selection window will open. Find the first template on your computer, for example, `YT based YT-DLP NEW!!! (English version).json`, and select it.
5. n8n will load and open the workflow on the canvas. **Be sure to click the "Save" floppy disk icon in the top right corner** to save it.
6. Repeat steps 2-5 for the second template (`YT based Rapid NEW!!! (English version).json`).

✓ Done! Both templates are now loaded into your n8n and ready for final configuration.

4. Configure Credentials

Sheets OP

Google Sheets OAuth2 API


Saved

Connection

Sharing

Details

Account connected

Reconnect:  Sign in with Google

Need help filling out these fields? [Open docs](#)

OAuth Redirect URL

http://localhost:5678/rest/oauth2-credential/callback

In Google Sheets, use the URL above when prompted to enter an OAuth callback or redirect URL

Client ID *

2280

Client Secret *

Allowed HTTP Request Domains

All

Make sure you enabled the following APIs & Services in the Google Cloud Console: Google Drive API, Google Sheets API. [More info.](#)

https://www.youtube.com/watch?v=X0W72n-Wg-M&list=PLsimB70v3O87vw8GkbkinOmJxxSZ1mJJM

FR

YouTube

Playlist ID

анпок

Supabase PPLaton

Supabase API

Connection

Sharing

Details

Connection tested successfully

Retry

Need help filling out these fields? [Open docs](#)

Host

https:// supabase.co

Service Role Secret

.....

Allowed HTTP Request Domains

All

Enterprise plan users can pull in credentials from external vaults. [More info](#)

petrunin-platon's Free

petrunin-platon's Project

main Production

Connect

Feedback

Settings

PROJECT SETTINGS

General

Compute and Disk

Infrastructure

Integrations

Log Drains

Data API

API Keys NEW

JWT Keys NEW

Add Ons

Vault ALPHA

CONFIGURATION

Database

Authentication

Storage

Edge Functions

BILLING

Subscription

Usage

API Settings

Project URL

Source Primary Database

URL https:// supabase.co

Copy

RESTful endpoint for querying and managing your database

Data API Settings

Docs Harden Data API

Enable Data API

When enabled you will be able to use any Supabase client library and PostgREST endpoints with any schema configured below.

Exposed schemas

public graphql_public

Select schemas for Data API...

The schemas to expose in your API. Tables, views and stored procedures in these schemas will get API endpoints.

Extra search path

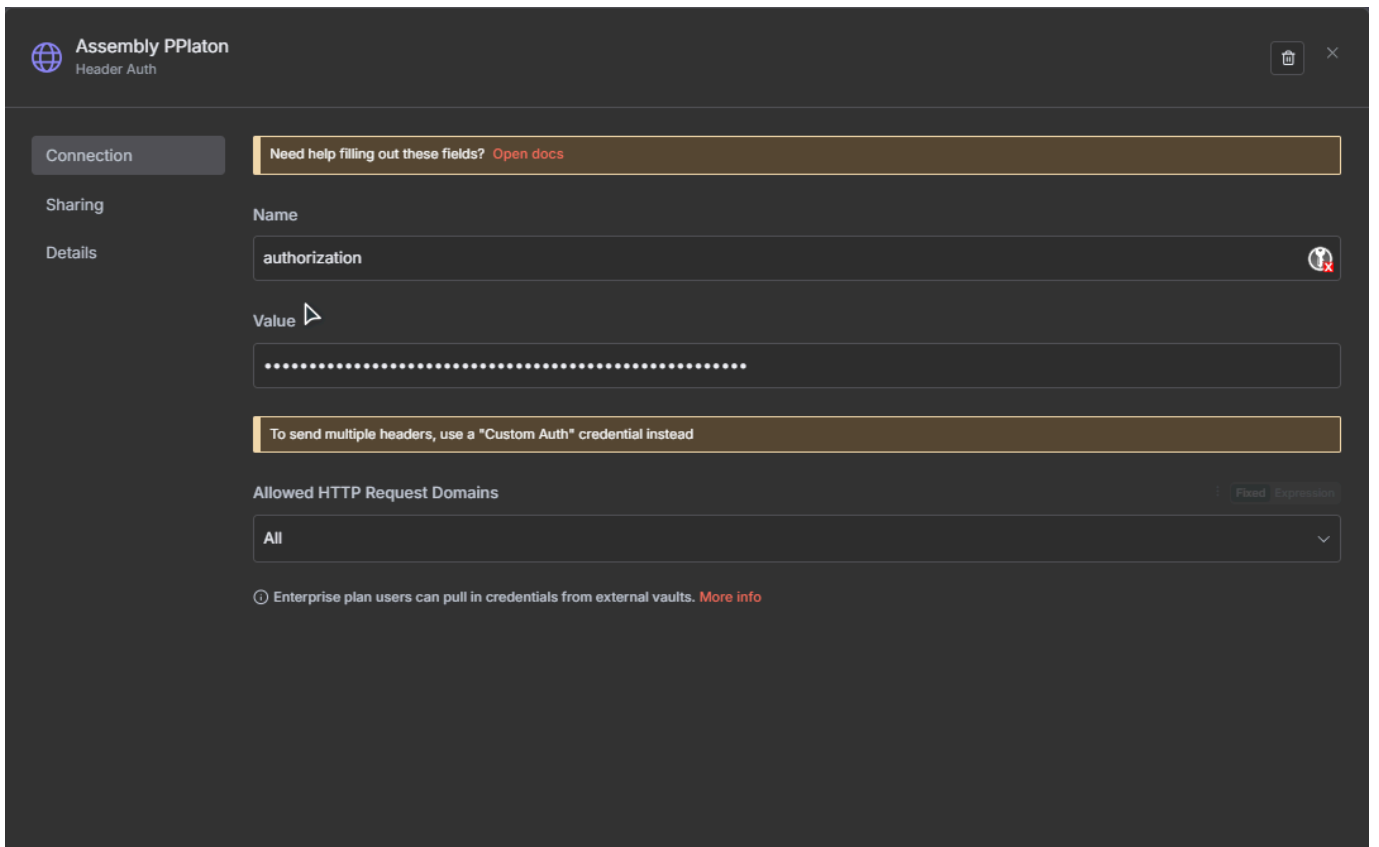
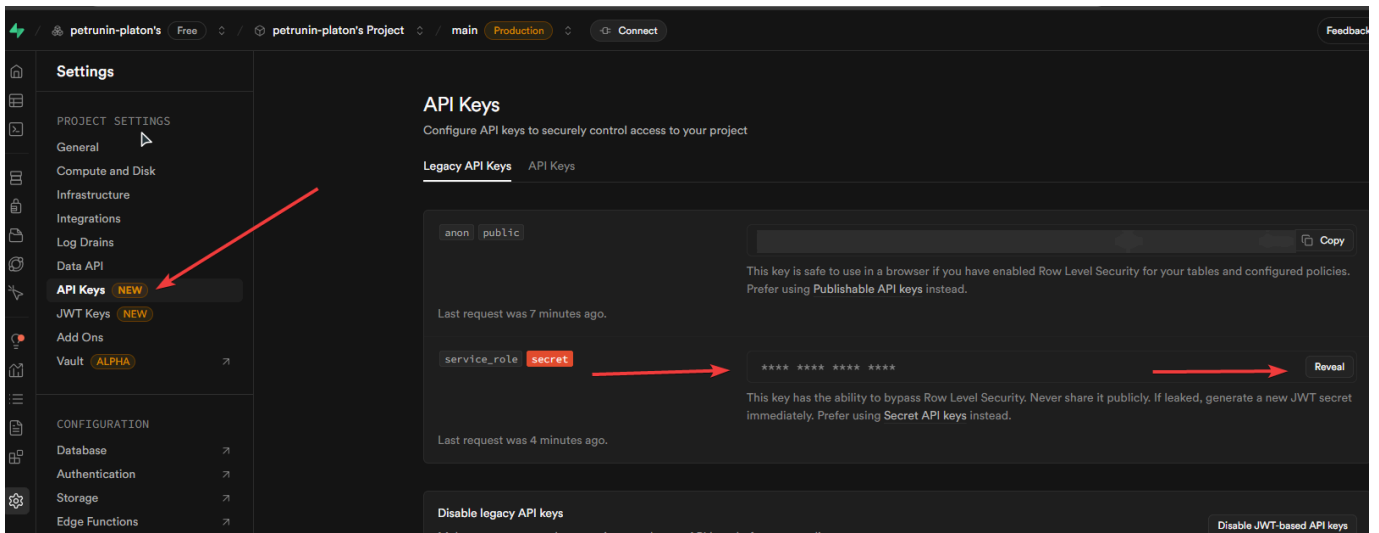
public, extensions

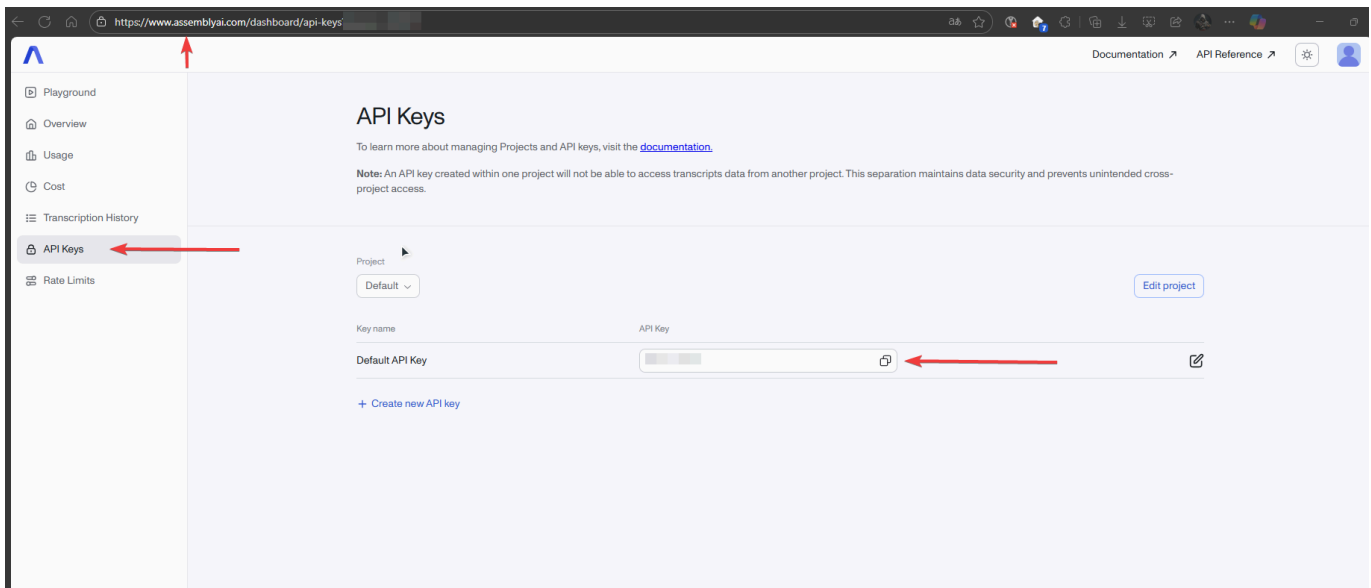
Extra schemas to add to the search path of every request. Multiple schemas must be comma-separated.

Max rows

1000

The maximum number of rows returned from a view, table, or stored procedure. Limits payload size for accidental or malicious requests.





Creating a Telegram Bot and Getting Keys

To send notifications, our workflow needs its own Telegram bot. Let's create one now.

Phase 1: Creating the Bot and Getting the API Token

1. Open the Telegram application.
2. In the search bar, find the official bot for creating other bots: `@BotFather` (it has a blue verification checkmark).
3. Click on it and, in the chat that opens, click **"Start"** (or send the `/start` command).
4. Send it the command to create a new bot:

```
/newbot
```

5. BotFather will ask you to choose a **name** for your bot (e.g., `My n8n Assistant`). Send it the name.
6. Next, BotFather will ask you to choose a **unique username** for the bot. It must be in English and must end in `bot` (e.g., `my_n8n_engine_bot`). Send it the username.
7. If successful, BotFather will send you a congratulatory message containing your **API Token**. It looks like a long string of numbers and letters, for example: `1234567890:ABCDEFGHIJKL...`
8. **Copy this token.** It will be needed to create the credential in n8n.

Phase 2: Getting Your Chat ID

Now we need to find the unique ID of the chat where the bot will send messages.

9. Find your newly created bot in the Telegram search using its **username** (e.g., `@my_n8n_engine_bot`) and start it by clicking **"Start"**.
10. Send it any message (e.g., `hello`). This "activates" the chat.

11. Now, open a web browser and paste the following link into the address bar, replacing `TOKEN` with the API token you copied in step 8:

```
https://api.telegram.org/botTOKEN/getUpdates
```

12. Press `Enter`. You will see a page with text in JSON format. Find the block of text that looks like this:

JSON

```
... "chat": { "id": -1234567890, "title": ...
```

13. The `id` from this block (`-1234567890`) is your **Chat ID**. Copy it (including the minus sign, if there is one). It will be needed for the configuration spreadsheet.

☒ Done! You now have both the API Token for the n8n credential and the Chat ID for the Google Sheet.



BotFather

3 248 444 пользователей в месяц



BotFather



BotFather

BotFather is the one bot to rule them all. Use it to create new bot accounts and manage your existing bots. [Learn more >](#)



Search

My bots



Create a New Bot



@BotFather

Open



Сообщение...





BotFather

3 248 444 пользователей в месяц



BotFather



New bot

Enter a name, description and username to create a new bot.

Чт

Bo

ас

Ab

hti

Bo

hti

Co

testin_engine_or

About (Optional)

t.me/testin_engine_bot

testin_engine_bot is available.

Choose a username for your bot. It must end in "bot".
Like this, for example: TetrisBot or tetris_bot.

Create Bot

Open



Сообщение...





BotFather

3 248 444 пользователей в месяц



BotFather



testin_engine_or

@testin_engine_bot



8174462031:AAFjwNQyLflLwJm62esZiyYg7iHH2pYnq8s

Copy

Revoke

Access the API using this token. Keep your token secure and store it safely, anyone can use it to control your bot. [Read more >](#)

Settings



Edit Info



Commands



Mini Apps



Bot Settings



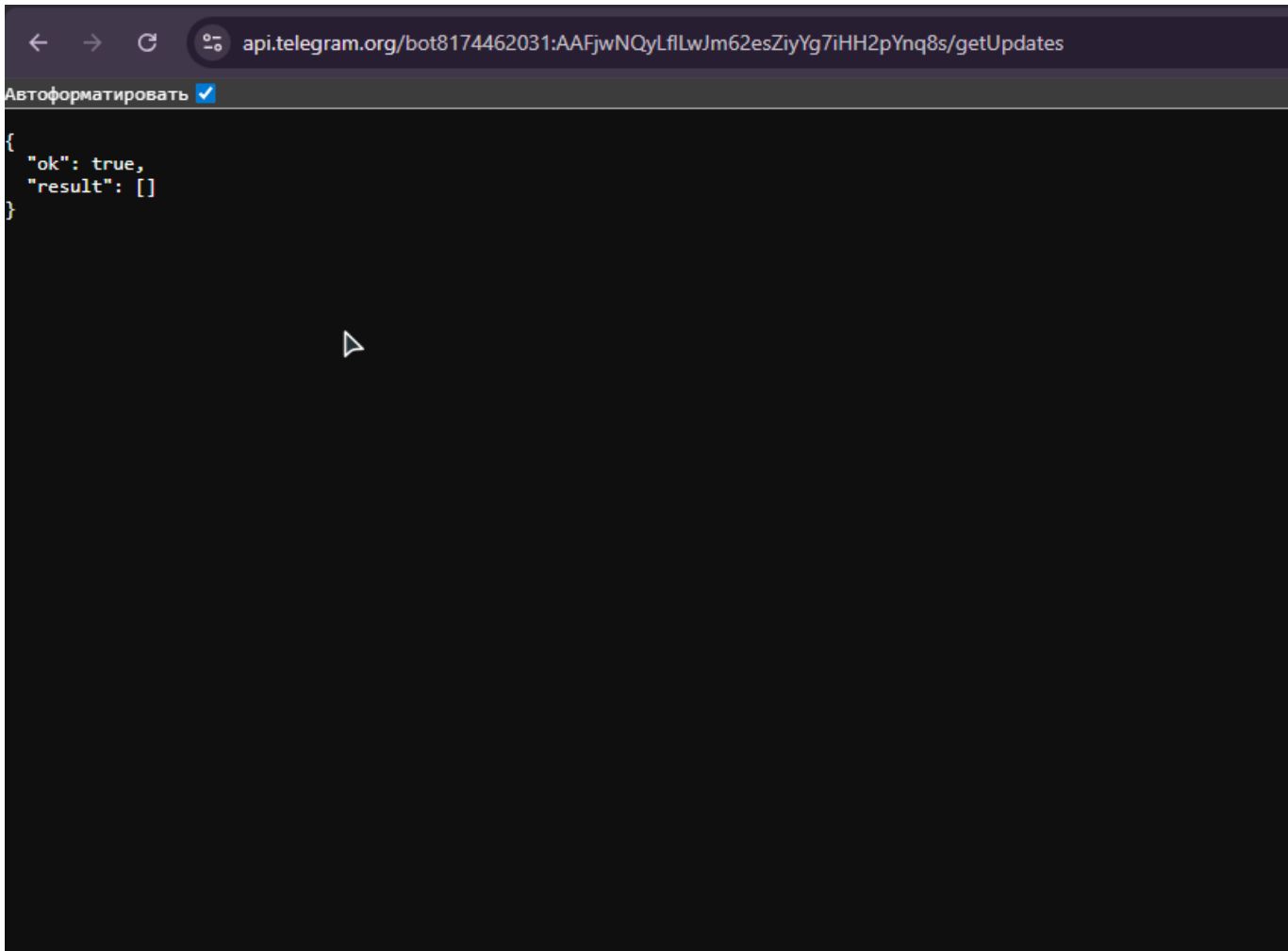
Games

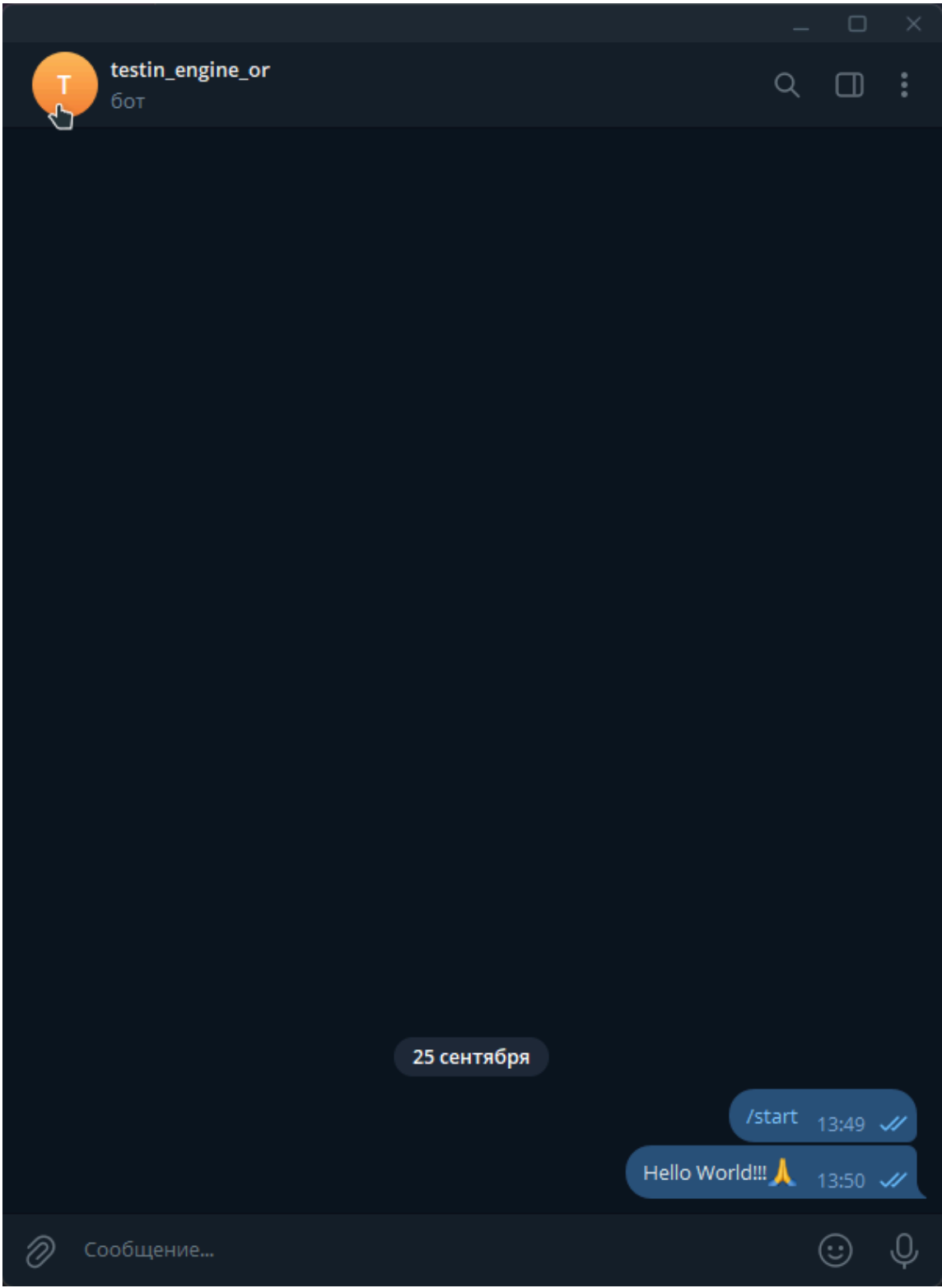


Open

@BotFather







Автоформатировать ☒

```
{
  "ok": true,
  "result": [
    {
      "update_id": 443255716,
      "message": {
        "message_id": 1,
        "from": {
          "id": 5628138361,
          "is_bot": false,
          "first_name": " ",
          "last_name": " ",
          "username": " ",
          "language_code": "ru"
        },
        "chat": {
          "id": 5628138361,
          "first_name": " ",
          "last_name": " ",
          "username": " ",
          "type": "private"
        },
        "date": 1758797379,
        "text": "/start",
        "entities": [
          {
            "offset": 0,
            "length": 6,
            "type": "bot_command"
          }
        ]
      }
    },
    {
      "update_id": 443255717,
      "message": {
        "message_id": 2,
        "from": {
          "id": 5628138361,
          "is_bot": false,
          "first_name": " ",
          "last_name": " ",
          "username": " ",
          "language_code": "ru"
        },
        "chat": {
          "id": 5628138361,
          "first_name": " ",
          "last_name": " ",
          "username": " ",
          "type": "private"
        },
        "date": 1758797413,
        "text": "Hello World!!! 🙋"
      }
    }
  ]
}
```

Header Auth account
Header Auth

Connection

Sharing

Details

Need help filling out these fields? [Open docs](#)

Name

X-RapidAPI-Key

Value

Fixed Expression

To send multiple headers, use a "Custom Auth" credential instead

Allowed HTTP Request Domains

All

Enterprise plan users can pull in credentials from external vaults. [More info](#)

Guide: Creating a Supabase Table via SQL

This guide will walk you through setting up a project in Supabase and creating a table using a SQL script. This table is designed to log the processing status of videos.

Step 1: Sign Up and Create a Supabase Project

First, you'll need a Supabase account and a project to work with.

1. **Navigate to the official Supabase website** at supabase.com.
2. Click on **"Start your project"**. You can sign up with a GitHub account, Google, or an email address.
3. After logging in, you will be prompted to create a new **Organization**. An organization can hold multiple projects. Give it a name relevant to your work.
4. Next, create a new project by clicking **"New Project"**.
 - **Name**: Assign a clear name to your project (e.g., `video-processing-log`).
 - **Database Password**: Create a strong password and **be sure to save it in a secure place**, like a password manager. You will need it to connect to your database later.
 - **Region**: Choose the server region that is geographically closest to you or your users to ensure lower latency.

5. Click **"Create new project"** and wait a few moments for the project infrastructure to be set up.

Step 2: Create the Table with a SQL Script

Now that the project is ready, it's time to create the table that will store the data.

1. In the left sidebar of your Supabase dashboard, find and click the **SQL Editor** icon.
2. Click the **"+ New query"** button to open a new window for your script.
3. Copy the SQL script below and paste it into the query window:

SQL

```
-- Create the 'processed_videos' table with the exact structure
CREATE TABLE public.processed_videos (
  -- id (integer, primary key)
  id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

  -- video_id (text, for the YouTube video ID)
  video_id character varying NULL,

  -- video_title (long text for the video title)
  video_title text NULL,

  -- channel_title (text, for the channel name)
  channel_title character varying NULL,

  -- status (text, for 'processing', 'completed', 'error' statuses)
  status character varying NULL,

  -- started_at (timestamp with time zone)
  started_at timestamp with time zone NULL,

  -- finished_at (timestamp with time zone)
  finished_at timestamp with time zone NULL,

  -- error_message (long text for error messages)
  error_message text NULL,

  -- Specify that the 'id' column is the primary key of the table
  CONSTRAINT processed_videos_pkey PRIMARY KEY (id)
);

-- Disable Row Level Security (RLS) for this table to allow n8n full access
ALTER TABLE public.processed_videos DISABLE ROW LEVEL SECURITY;
```


4. Click the green **"RUN"** button to execute the query.

That's it! Your `processed_videos` table has now been created and is ready to receive data.

Understanding the SQL Script

Here is a breakdown of what each line in the script does:

- `CREATE TABLE public.processed_videos (...)` This command instructs the database to create a new table named `processed_videos` inside the `public` schema. The columns of the table are defined within the parentheses.
 - `id bigint GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,`
 - `id`: The name of the column.
 - `bigint`: A data type for storing large whole numbers.
 - `GENERATED BY DEFAULT AS IDENTITY`: This automatically assigns a unique, sequential number (1, 2, 3, ...) to each new row.
 - `PRIMARY KEY`: Designates this column as the table's primary key, which is a unique identifier for each row.
 - `video_id text, , video_title text, , channel_title text,`
 - These columns use the `text` data type, which can store text strings of any length.
 - `status varchar(255),`
 - The `varchar(255)` type is used for text with a maximum length of 255 characters, which is efficient for short status messages like "completed" or "failed".
 - `started_at timestamp with time zone, and finished_at timestamp with time zone,`
 - The `timestamp with time zone` data type is the standard for storing precise date and time information, including the time zone. It is ideal for logging exact moments.
 - `error_message text,`
 - A `text` column to store detailed error messages if a process fails.
 - `created_at timestamp with time zone DEFAULT now()`
 - This is a common and highly recommended practice. The `DEFAULT now()` clause automatically sets the column's value to the exact timestamp when the row was created. You do not need to provide this value; the database handles it automatically.
-

Step 5.1: Fill out the configuration table

Google Sheets: [Engine local and yt-dlp - Google Sheets](#)

Google Sheets: [Engine Cloud and Rapid - Google Sheets](#)

Download the spreadsheets for yourself, and set them up in Google Sheets by following the links above.

Values to insert into your Google Sheet:

- **For the Obsidian Inbox Path setting:**

- Paste this value into the "Value" column:

```
/inbox/
```

- **For the Media Download Path (Container) setting:**

- Paste this value into the "Value" column:

```
/media/
```

- **For the Temporary Files Path setting:**

- Paste this value into the "Value" column:

```
/temp/
```

- **For the Obsidian Link Path setting:**

- Paste this value into the "Value" column (this is the path to the media files folder relative to the Obsidian vault root):

```
00_Files/
```

Part 6: Testing and Launch

At this stage, we will conduct a "test flight" to ensure that all system components (Docker, n8n, Google Sheets, Telegram) are communicating correctly with each other. After a successful test, we will activate the workflow for automatic operation.

Step 6.1: Preparing for the Test — Adding a Video

To give the workflow something to process, we need to manually add a video to your working playlist.

- **What to do:**


1. Open YouTube.
2. Find any short video for the test.
3. Add it to the playlist whose ID you specified in the `Youtubelist ID` setting in your Google Sheet.

✅ Done! The "bait" for our engine is set.

Step 6.2: Manual Workflow Execution

Now we will run the workflow manually, not on a schedule, to see how it works in real-time.


- **What to do:**

1. Open the n8n interface in your browser (`http://localhost:5678`).
2. Go to the **"Workflows"** section.
3. Select one of your templates (e.g., `YT based YT-DLP...`) and click on it to open it on the canvas.
4. In the top right corner, you will see the **"Execute Workflow"** button (or a "Play" icon ). Click it.

Step 6.3: Observing the Process

You have started the workflow! Now the magic begins.

- **What you should see:**


- **In n8n:** The nodes on the canvas will start executing one by one. A successfully executed node will be highlighted with a **green border**. If a node encounters an error, it will turn **red**.
- **In Telegram:** You should receive the first notification "  **New Process...**". As the workflow progresses, this message will be **edited**, showing the current stage (downloading, transcribing, etc.).

This is the most important moment for diagnostics. If any node turns red, you can click on it and check the "Output" tab to see what specific error occurred.

Step 6.4: Verifying the Result

If the workflow completed to the end and all nodes turned green, congratulations! Let's check the result.

- **Checklist for successful execution:**

1. **In Telegram**, the final message "  **Process Finished...**" has arrived.
2. **In the notes folder** (`E:\n8n-project\Obsidian\n8n-project\Inbox`), a new `.md` file with the video's title has appeared.
3. **(If download_type was video or audio) In the media folder** (`E:\n8n-project\Obsidian\n8n-project\00_Files`), a new `.mp4` or `.mp3` file has appeared.
4. **On YouTube**, the test video has been **deleted** from your working playlist.
5. **(Optional)** The **temporary files folder** (`E:\n8n-project\temp`) should be empty, as the workflow has cleaned up after itself.

Step 6.5: Activating Automatic Mode

After you have confirmed that a manual run was successful, you can "turn on the autopilot."

- **What to do:**

1. In the n8n interface, while inside your workflow.

2. In the top left corner, find the **"Active"** toggle switch.
3. **Click on it** to make it active (it usually turns blue or green).
4. **Save the workflow** by clicking the "Save" floppy disk icon.

✅ **Done!** Your workflow is now fully active and will automatically check the playlist every 15 minutes, as specified in the very first `Schedule Trigger` node.

Important Note: External Services and Troubleshooting Error 403

To download media files, this workflow relies on external tools. It's important to understand how they operate and what potential issues you may encounter.

1. The YouTube Media Downloader Service on RapidAPI

This workflow utilizes a specialized service called **YouTube Media Downloader**, which is available on the **RapidAPI** developer platform.

- **Service Link:** <https://rapidapi.com/DataFanatic/api/youtube-media-downloader>

Its purpose is to provide direct links for downloading audio and video tracks from YouTube. To use it, you must register on RapidAPI, subscribe to this service (it has a free tier), and obtain your personal **API key**. This is the key you will then save in your n8n credentials for authentication.

2. The Most Common Problem: Error 403 (Forbidden)

When working with any service for downloading from YouTube (whether it's RapidAPI, yt-dlp, or others), the most common error you will encounter is **Error 403**.

- **What does it mean?** In simple terms, the YouTube server sees your download request and understands it, but **refuses to grant access**.
- **Why does this happen?** Most often, the reason is the **IP address** from which the request originates. YouTube actively combats bots and automated systems. If too many download requests come from a single IP address, YouTube will blacklist it and start blocking requests. The servers of RapidAPI, or your own server running n8n, can be affected by this.

3. The Solution: Using a Quality VPN

The most reliable way to bypass a 403 error is to send the request from a "clean" IP address that doesn't arouse YouTube's suspicion. To achieve this, you need a good VPN service.

Important: It is crucial to use a **paid VPN**.

- **Why not a free one?** Free VPN services provide a very small pool of IP addresses to thousands of users. These IPs are almost always on YouTube's blacklists because they are constantly used for automation.
 - **Benefits of a Paid VPN:** Paid services offer a vast selection of IP addresses that have not been flagged for suspicious activity. They are faster and more reliable. By running a VPN on the server where n8n is hosted, you mask your actual IP with a "clean" one, and YouTube is likely to approve the request.
-

Downloading Media with yt-dlp (Advanced Method)

In this version of the workflow, we are moving from an external API to the direct use of **yt-dlp**, a powerful and flexible program that runs directly on your server. This approach provides maximum control over the download process, but for stable and reliable operation, it requires a small additional setup: using **cookies** from your web browser.

Why Are Cookies Necessary?

Imagine that `yt-dlp` is a robot visiting YouTube to download a video. By default, it is an anonymous guest. YouTube sees that this is an automated request and may block its access (resulting in an **Error 403**), especially if the content is:

- Age-restricted (18+).
- Part of a private or unlisted playlist.
- Available only to YouTube Premium subscribers.

When we provide `yt-dlp` with a `cookies.txt` file, we are essentially giving it a digital "pass" from our browser. The robot now visits YouTube not as a guest, but as you—a logged-in user. YouTube recognizes a familiar user and readily grants access to all the content that is available to you personally.

Conclusion: Using cookies is the most reliable way to bypass blocks and download any content accessible to your account.

Step-by-Step Setup Guide

The setup process is straightforward and is done once, only requiring occasional updates.

Step 1: Install the Browser Extension


We will use a simple and secure browser extension called **Get cookies.txt LOCALLY**. It works in Google Chrome and other Chromium-based browsers (Microsoft Edge, Brave, Opera, etc.).

1. Follow the link to the Chrome Web Store:

Get cookies.txt LOCALLY

2. Click the **"Add to Chrome"** button.

Step 2: Export the `cookies.txt` File

1. In your browser, navigate to [youtube.com](https://chrome.google.com/webstore/detail/get-cookies-txt-locally). **Critically important:** ensure that you are **logged into your Google account**.
2. Click the puzzle piece icon () in the top-right corner of your browser to open the extensions menu.
3. Find and click on **Get cookies.txt LOCALLY** in the list.
4. A new tab will open displaying the text content of your cookies for YouTube. Find and click the **"Export as TXT"** button.
5. The browser will download a file, which will likely be named `youtube.com_cookies.txt`.
6. **Crucial Step:** Rename the downloaded file to exactly `cookies.txt`. The name must be precisely this, in lowercase and without any extra characters.

Step 3: Place the `cookies.txt` File for n8n

Now that you have the file, you need to place it in a folder that n8n can "see". Your `docker-compose.yml` file is already configured for this.

In the `volumes` section of your `docker-compose.yml` file, there is a line:

YAML

```
- './cookies:/cookies'
```

Let's break down what this means:

- `./cookies` : This is a **folder on your host computer**. The `.` signifies that this folder must be located **in the same directory as your `docker-compose.yml` file**.
- `/cookies` : This is the **path inside the n8n container**. The workflow is configured to look for the file there.

To get everything working, follow these three simple steps:

1. Locate your `docker-compose.yml` file.
2. In that same directory, create a new folder and name it exactly `cookie` (singular, as specified in the `docker-compose.yml` file).
3. Place your renamed `cookies.txt` file inside this newly created `cookie` folder.

That's it! Now, when you run n8n with the `docker-compose up -d` command, Docker will automatically link these two folders, and n8n will find and use your cookies file for all `yt-dlp`

requests.

Your folder structure should look like this:

```
/Your_Main_Project_Folder/  
├─ docker-compose.yml  
├─ cookies/  
│   └─ cookies.txt  
└─ ... (other project files and folders)
```

Troubleshooting and Maintenance

Problem: Error 403 Even with Cookies

Occasionally, even with correctly configured cookies, YouTube might temporarily block your server's **IP address** if it detects a high volume of automated requests. In this situation, you will again see an **Error 403**.

In this case, as previously discussed, the best solution is to use a **high-quality, paid VPN**.

- **How it works:** A VPN running on your server (or on the router your server uses) will change its public IP address. Instead of the blocked address, YouTube will see a new, "clean" IP, and the download will proceed successfully.
- **Why paid?** Paid VPN services provide reliable IP addresses that are not on blacklists. The IPs used by free alternatives are almost always blocked.

Maintenance: Updating Your Cookies

Remember that cookies are not permanent. They typically expire after a few weeks or months. If you notice that downloads are failing again, the first thing you should do is update your `cookies.txt` file.

Simply repeat **Step 2** and **Step 3** from the guide: export a fresh file from YouTube and use it to replace the old one in your `cookie` folder.

Setting Up the Database in Supabase

We use Supabase as a "journal" for our workflow to track which videos have already been processed, which are currently in progress, and which have failed with an error.

Step 1: Register and Create a Project

1. Go to supabase.com and create an account.

2. In your dashboard, click **"New Project"**.
3. Choose a name for your project, generate a strong database password (and **be sure to save it!**), and select a region. Creating the project will take a couple of minutes.

Step 2: Create the Table with an SQL Query (the fast way)

To avoid creating all the columns manually, we will use the SQL Editor. This is the fastest and most accurate method.

1. In your Supabase project menu on the left, find the icon labeled **"SQL Editor"** (it looks like a page with SQL symbols).
2. Click on it, and then click the **" + New query "** button.
3. In the large text window that appears, **copy and paste the entire SQL code provided below:**

SQL

```
-- Create the 'processed_videos' table with the exact structure
CREATE TABLE public.processed_videos (
  -- id (integer, primary key)
  id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,

  -- video_id (text, for the YouTube video ID)
  video_id character varying NULL,

  -- video_title (long text for the video title)
  video_title text NULL,

  -- channel_title (text, for the channel name)
  channel_title character varying NULL,

  -- status (text, for 'processing', 'completed', 'error' statuses)
  status character varying NULL,

  -- started_at (timestamp with time zone)
  started_at timestamp with time zone NULL,

  -- finished_at (timestamp with time zone)
  finished_at timestamp with time zone NULL,

  -- error_message (long text for error messages)
  error_message text NULL,

  -- Specify that the 'id' column is the primary key of the table
  CONSTRAINT processed_videos_pkey PRIMARY KEY (id)
);
```



```
-- Disable Row Level Security (RLS) for this table to allow n8n full access  
ALTER TABLE public.processed_videos DISABLE ROW LEVEL SECURITY;
```

4. Click the green **"RUN"** button to execute the code.

✅ **Done!** After a moment, you will see a "Success. No rows returned" message at the bottom. This means your table `processed_videos` has been successfully created with the exact required structure. You can go to the **"Table Editor"** to confirm this.

Now you can proceed to `Project Settings -> API` to copy the keys needed for creating the credential in n8n.
