# Start and Stop the System

## Step 1 – Go to the project folder

Open a terminal and go to the folder where the application is located:

**Run:**

```
cd /path/to/project
```

```
/home/Sorin/Pricing
$
```

**Run:**

```
docker compose down
```

```
$ docker compose down
[+] Running 8/8
 ✔ Container pricing-traefik-1              Removed                  2.7s
 ✔ Container pricing-orchestrator-1         Removed                 10.3s
 ✔ Container pricing-scheduler-1            Removed                 10.2s
 ✔ Container pricing-api-1                  Removed                  0.3s
 ✔ Container pricing-metabase-1             Removed                  2.8s
 ✔ Container pricing-postgres-1             Removed                  0.2s
 ✔ Container pricing-metabase-postgres-1    Removed                  0.2s
 ✔ Network pricing_traefik-net              Removed                  0.2s
```

**This will stop and remove the containers and network.**

## Step 2 – Start all services

**Run:**

```
docker compose up -d
```

```
$ docker compose up -d
[+] Running 8/8
 ✔ Network pricing_traefik-net              Created                  0.1s
 ✔ Container pricing-traefik-1              Started                  1.5s
 ✔ Container pricing-metabase-postgres-1    Started                  1.4s
 ✔ Container pricing-orchestrator-1         Started                  2.0s
 ✔ Container pricing-postgres-1             Started                  1.8s
 ✔ Container pricing-scheduler-1            Started                  2.2s
 ✔ Container pricing-api-1                  Started                  2.6s
 ✔ Container pricing-metabase-1             Started                  2.3s
```

**This will create and start all containers in the background.**

These are the basic commands to start and stop the system. Always make sure you are inside the correct project folder before running them.

# Check Running Containers

## Step 1 – List all containers

### Run the command:

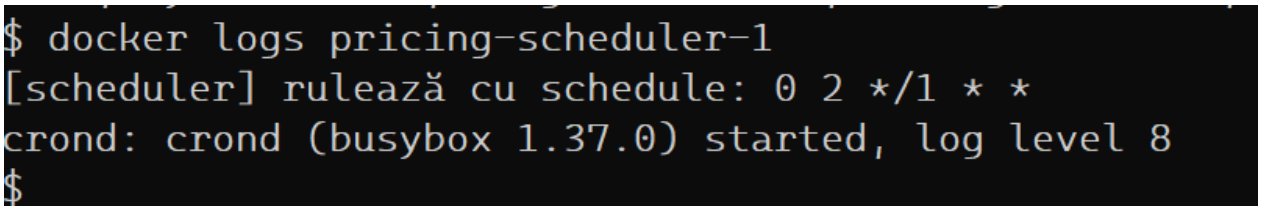
```
docker ps
```


You will see a table with information about each container:
**CONTAINER ID** – unique identifier of the container
**IMAGE** – the Docker image used
**STATUS** – should show *Up* if the container is running
**PORTS** – which ports are exposed
**NAMES** – the name of the container (example: `pricing-api-1`)

## View Logs for Containers

### Run:

```
docker logs pricing-api-1
docker logs pricing-scheduler-1
docker logs pricing-metabase-1
docker logs pricing-orchestrator-1
docker logs pricing-traefik-1
docker logs  pricing-postgres-1
docker logs  pricing-metabase-postgres-1
```



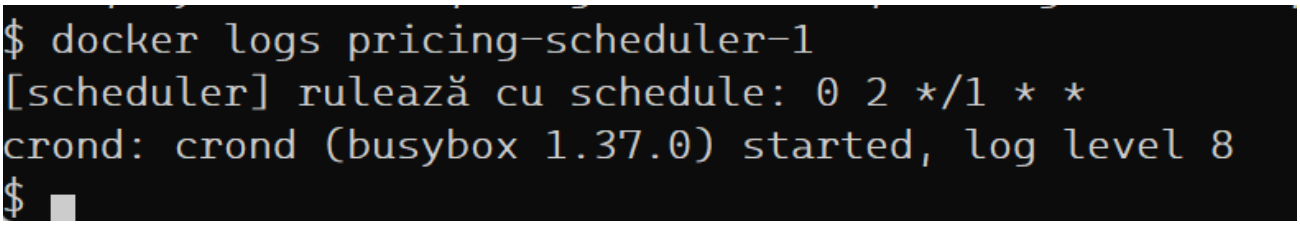

Use logs to check if containers are working correctly or to troubleshoot errors.
Note
This command **does not affect the system**.
It only displays the logs generated by the container.
You can use it for any container: `pricing-api-1`, `pricing-scheduler-1`, `pricing-metabase-1`, etc.

### Check and Modify the Scheduler

Step 1 – Verify current schedule

Run:

```
docker logs pricing-scheduler-1
```




# Understand the Scheduler Expression

Example from the logs:

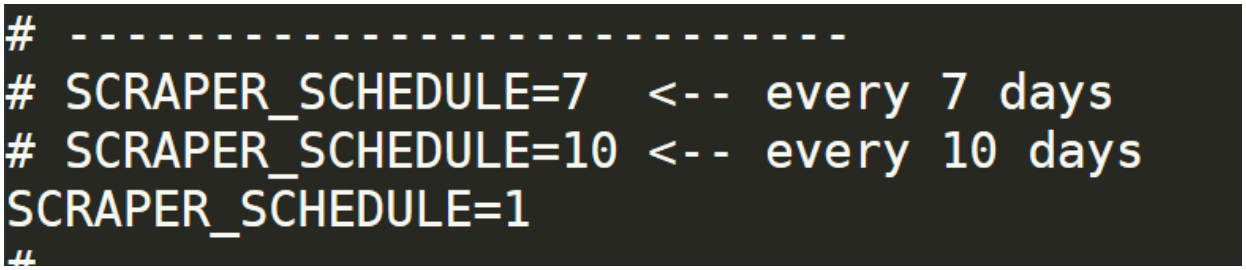
```
[scheduler] running with schedule: 0 2 */1 * *
```

This expression is a **cron schedule** with 5 fields:

```
┌ minute (0 - 59)
│ ┌ hour (0 - 23)
│ │ ┌ day of month (1 - 31)
│ │ │ ┌ month (1 - 12)
│ │ │ │ ┌ day of week (0 - 6) (Sunday=0)
│ │ │ │ │
0  2  */1 * *
```

Explanation:
**0** → minute = at minute 0
**2** → hour = at 02:00 in the morning
***/1** → every 1 day (which means daily)
*** → every month
*** → every day of the week
Together: **The scheduler runs every day at 02:00 AM.**

Env -file

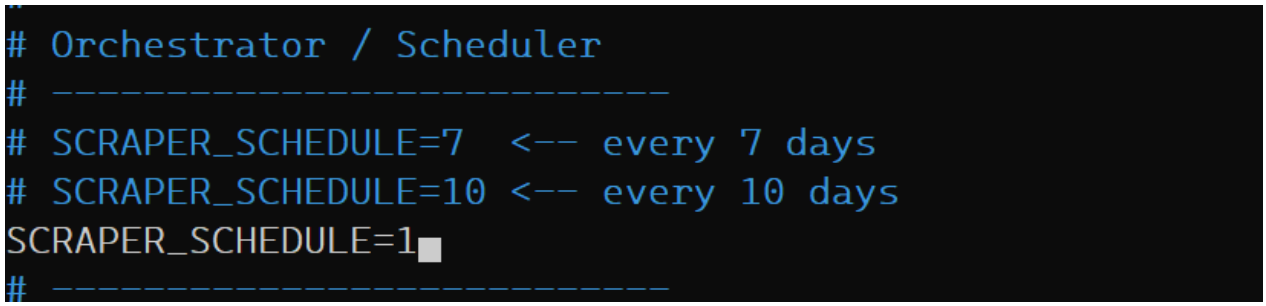



**Important Note**
Always check the `.env` file first to confirm the schedule.
In our case, `SCRAPER_SCHEDULE=1` → the program is correctly set to run **once per day**.
**Change the Scheduler Interval**

## Step 1 – Open the `.env` file

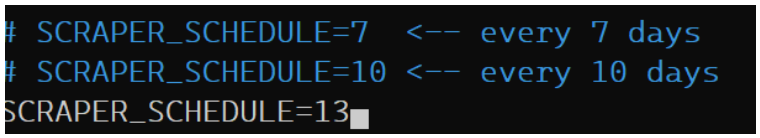
In the project folder, open the `.env` file with a text editor (for example `nano` or `vim`):

```
nano .env
```

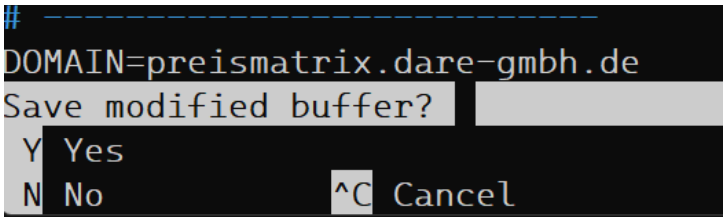



1.Find the line for schedule

2.Change `1` → run every **1 day** ,`7` → run every **7 days** ,`30` → run every **30 days** etc



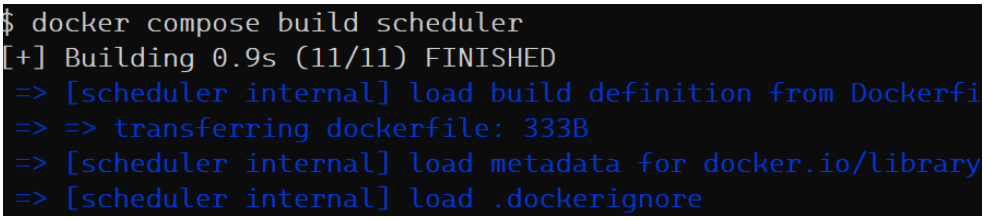

**Save and exit in nano:** Press **CTRL + X**



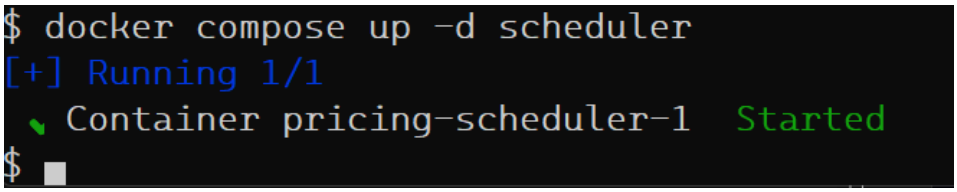

Confirm with **Y** Press **ENTER**

Rebuild and restart scheduler:

```
docker compose build scheduler
```




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




Verify the New Schedule

Run:

```
docker logs pricing-scheduler-1
```

```
$ docker logs pricing-scheduler-1
[scheduler] rulează cu schedule: 0 2 */13 * *
crond: crond (busybox 1.37.0) started, log level 8
$
```

This shows the program is now scheduled to run **every 13 days at 02:00 AM**.

The modification was applied correctly.

Sumary

1. Go to the project folder & Open the `.env` file & Change the value (number of days) &Save and exit the file
2. Rebuild the scheduler run `docker compose build scheduler`
3. Start the scheduler run `docker compose up -d scheduler`
4. Check the scheduler logs run `docker logs pricing-scheduler-1`

## Change Traefik Password – Steps

Go to the project folder

Generate a new password hash:

`htpasswd -nbB user newpassword`

`Replace <user> with any username you want`

Replace `<newpassword>` with your new password

*exemple : admin MyPassword4445##$#@#$*

`htpasswd -nbB` *admin MyPassword4445##$#@#$*

```
$ htpasswd -nbB admin MyPassword4445##$#@#$
admin:$2y$05$.XMRA4Ko0qJBi8nvrxrDneQXAkMgIlq2mOiu2NOqs3SMvcOs7kC9S
```

Open the Traefik configuration file (`traefik_dynamic.yml`)

```
$ cd traefik
$ pwd
/home/Sorin/Pricing/traefik
$
```

Replace the old hash with the new one

```
 GNU nano 6.2                                                 trae
http:
  middlewares:
    api-auth:
      basicAuth:
        users:
          - "admin:$apr1$x856PDmm$2urvGvU1aKjkBbJYqCgH2."
```

New password change it to new generate one:

```
http:
  middlewares:
    api-auth:
      basicAuth:
        users:
          - "admin:$2y$05$.XMRA4Ko0qJBi8nvrxrDneQXAkMgIlq2mOiu2NOqs3SMvcOs7kC9S"
```

Save and exit the file

Restart Traefik:

```
docker compose restart traefik
```

Test login with the new password

⚠ Note
If the new password does **not** work after restart, then rebuild the service:

```
docker compose build traefik
docker compose up -d traefik
```

After that, test the login again.

## Run the Orchestrator Manually – Important Note

The extractor has been tested and runs automatically according to the schedule defined in the .env file (variable SCRAPER_SCHEDULE).

However, in some cases (e.g. after configuration changes or debugging), it is useful to start the extractor manually to verify its behavior.

Data consistency behavior:
The extractor is designed to store only one set of prices per day in the database.
If it is executed multiple times in the same day:
Identical data will be skipped (not duplicated).
If there are updates, existing entries will be updated accordingly.

Recommended practice:
Once the extractor is started (manually or by the scheduler), allow it to complete the entire cycle.
During this cycle the orchestrator will:
Clean temporary log files from previous runs.
Extract and process the new price files.
Format products into type → category → subcategory.
Write data into the database (or update if needed).

Archive the result files and automatically delete archives older than 7 days.

👉 For transparency and debugging, the orchestrator can be started manually with:

👉 For transparency and debugging, the orchestrator can be started manually with:
Run:

*docker compose run –rm orchestrator python runner.py*

Starting orchestrator... scraper: cdz

```
$ docker compose run --rm orchestrator python runner.py

 Now Running: 1cdz-scraper.py
[SAVED] Product: Bauschutt / Boden siebfähig | Size: 3 | Price: 321,30 € | Title: Bauschutt / Boden siebfähig 3m³
[SAVED] Product: Bauschutt / Boden siebfähig | Size: 5 | Price: 392,00 € | Title: Bauschutt / Boden siebfähig 5m³
[SAVED] Product: Bauschutt | Size: 6 | Price: 471,00 € | Title: Bauschutt 6m³
[SAVED] Product: Bauschutt | Size: 7 | Price: 549,00 € | Title: Bauschutt 7m³
[SAVED] Product: Bauschutt | Size: 12 | Price: 942,00 € | Title: Bauschutt 12m³
[SAVED] Product: Baumischabfall | Size: 3 | Price: 446,25 € | Title: Baumischabfall 3 m³
```

Running scraper: dino

```
Full scraping. File saved in: results_cdz-berlin/cdz_container_prices_20251014_021545.csv
 Successful completion: 1cdz-scraper.py

 Now Running: 2dino-scraper.py
☑ 3 m³ | Sperrmüll | 381,17 €
☑ 3 m³ | Bauschutt | 370,16 €
☑ 7 m³ | Baumischabfall | 871,52 €
☑ 7 m³ | Beton | 322,59 €
```

Running scraper: Klebs

```
 Now Running: 3klebs-scraper.py
☑Accepted cookies
🔗 Waste card links: 11
📦 7 containers found in: https://www.klebs.info/abfaelle/altholz/
[✎] ['unsere 5,5 cbm (kubikmeter), 7 cbm und 10 cbm absetzcontainer halten wir
artcontainer (din-container) gibt es wahlweise mit deckel und klappe, oder in o
ps://www.klebs.info/containerdienst/absetzcontainer/']
```

Running scraper:cleargo

```
 Now Running: 4main_clearago.py
[🚀] Starting Clearago Scraper (debug mode: False)
[1] Opening site and submitting postcode...
[→] Navigating to https://www.clearago.de/
[✓] Page loaded
[✓] Cookiebot 'Allow all' clicked
```

Running scraper: entsorgo

```
[▩] Finished Clearago scraping and CSV export.
 Successful completion: 4main_clearago.py

 Now Running: 5main_entsorgo.py
🚀 I start the entsorgho scraper...

📦 Container: Baustellenmischabfall ohne Boden, Bauschutt & Steine
☑ Baustellenmischabfall ohne Boden, Bauschutt & Steine Container | 3 | 399,76 €
☑ Baustellenmischabfall ohne Boden, Bauschutt & Steine Container | 5 | 550,73 €
☑ Baustellenmischabfall ohne Boden, Bauschutt & Steine Container | 7 | 771,02 €
```

Scrapers Finished
You will see `Final JSON saved` and `Final CSV saved`.

```
☑ Final JSON saved: output_data.json
 Done: json_maker.py

 Finalizing: cvs_maker.py
☑ Final CSV saved: output_data.csv
☑ Final CSV saved: output_data.csv
[DELETED] Folder deleted: results_cdz-berlin
[DELETED] Folder deleted: results_clearago
[DELETED] Folder deleted: results_dino_container
[DELETED] Folder deleted: results_ensorgo
[DELETED] Folder deleted: results_klebs
[MOVED] output_data.csv → /app/results_data_14_10_2025/output_data.csv
[MOVED] output_data.json → /app/results_data_14_10_2025/output_data.json
 Done: cvs_maker.py
```

Category Parsing
Messages like Running category parser: create_type.py.

```
Running category parser: create_type.py
Category script completed: create_type.py

Running category parser: create_category.py
Category script completed: create_category.py
```

Data is categorized into type / category / subcategory.

**Database Import**

⚠ **Note for Client:**

**At the end of every successful run, the log should show:**

```
Database populated successfully.
Cleaner finished successfully.
```

```
 Importing data into PostgreSQL...
☑ Found 1 CSV file(s).
☑ Total rows before filtering: 375
☑ Connected to PostgreSQL.
☑ Table, column scraped_date, and unique index ensured.
☑ Deleted existing data for date: 2025-10-14
☑ Data inserted successfully (duplicates skipped by constraint).
 Database populated successfully.
```

```
 Running cleaner script...
[MOVED] results_data_14_10_2025 → /app/archive/results_data_14_10_2025
 Cleaner finished successfully.
```

At the **end of every successful run**, the log should show:

```
Database populated successfully.
Cleaner finished successfully.
```

Reset Databases and Containers

Sometimes you may want to restart the system **from zero**.
Be careful: this will erase existing data.
1. Stop All Containers

```
docker compose down
```

2. Remove All Data Volumes (⚠ deletes everything: both PostgreSQL and Metabase databases)

```
docker compose down -v
```

This will remove:
`pricing_pgdata` → Scraper database (extracted prices)
`pricing_metabase_pgdata` → Metabase internal database (dashboards, users, configs)

**Warning: All data will be lost. Only use this if you want a clean system.**

Remove Only One Database
If you only want to reset **one database**, you can remove only its volume.
Remove Scraper Database (keep Metabase intact)

```
docker volume rm pricing_pgdata
```

Remove Metabase Database (keep Scraper data intact)

```
docker volume rm pricing_metabase_pgdata
```

After removing, recreate the containers:

```
docker compose up -d
```

Notes
Removing a volume = permanent data loss.
If you only need to restart services **without deleting data**, use:
**Removing a volume = permanent data loss.**
The database will be rebuilt empty (fresh state).
After removing volumes, the databases will be recreated from scratch using the values in your `.env` file (DB name, user, password, port).

**Important:** Do not change `.env` database credentials while the system is running.
The scrapers and API read the connection details from `.env`.
If the credentials in `.env` don't match the running PostgreSQL instance, the system will fail to insert data.

**Further Commands**

This guide covers only the **most relevant commands** for running and maintaining the Pricing Data Platform.
For more advanced Docker commands and options, please refer to the official Docker documentation:

https://docs.docker.com/reference/cli/docker/