

Classifier

Table of Contents

Installation	1
Windows/Unix	1
Docker	1
Configuration	4
Server port	5
Knowledgebase	5
Monitor	5
Changing	6
Components	7
Engine	7
Monitor	8

Installation

Windows/Unix

Since it is a simple .jar file, which can run on any device featuring [Java](#), you can run it by simply doubleclicking.

TIP Be aware of the [application configuration](#).

The default port as in said configuration is **8080**

```
server_port: 8080
```

So if you want to access the webUI on your local machine navigate to <https://localhost:8080>

```
knowledge_base: build/knowledgebase
```

Docker

Dockerfile

If you can use a dockerfile, you can create one like this: Note to self: Need to test this!

```
FROM openjdk:15-jdk
MAINTAINER Cuupa
WORKDIR /opt/app/classificator
COPY knowlegebase/kb-1.0.0.db ./knowledgebase/kb-1.0.0.db
COPY *.jar ./app.jar



EXPOSE 8080

ENTRYPOINT ["java", "-jar", "./app.jar", "--server.port=8080", "--knowledge_base=./knowledgebase/kb-1.0.0.db"]
```

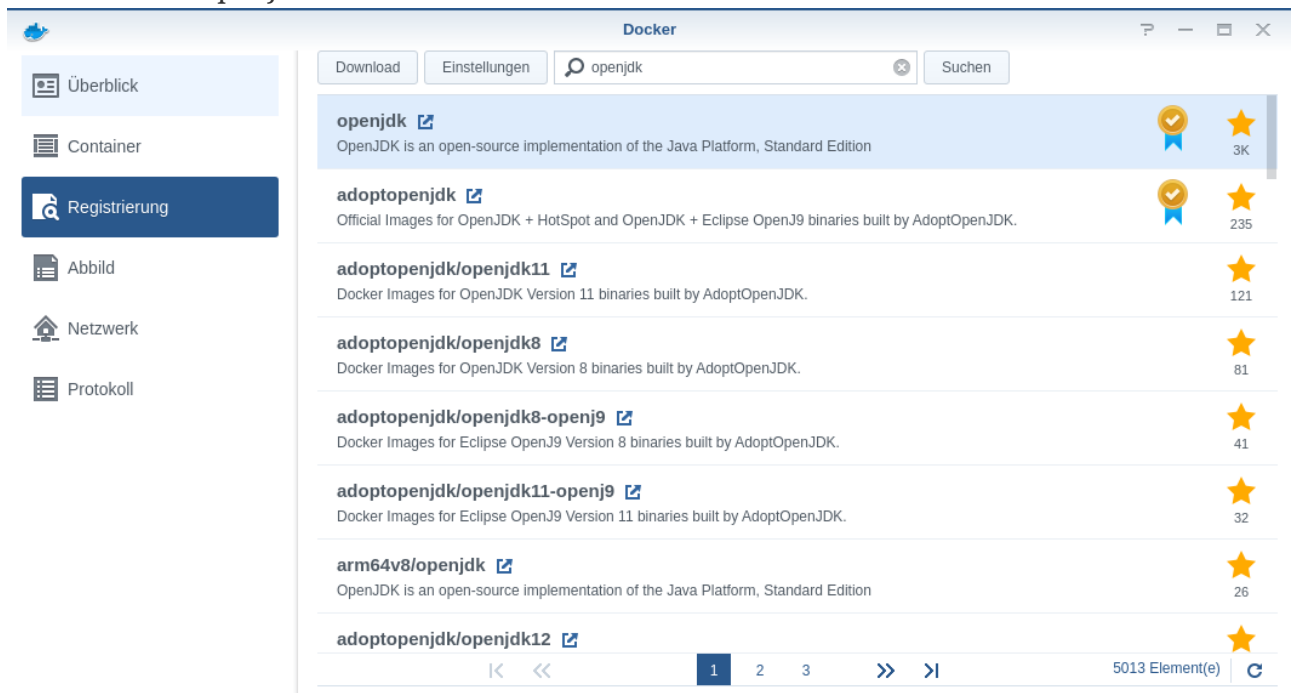
Please change the port of the application and change it accordingly.

Synology NAS

1. Create yourself a folder, where you upload the files to

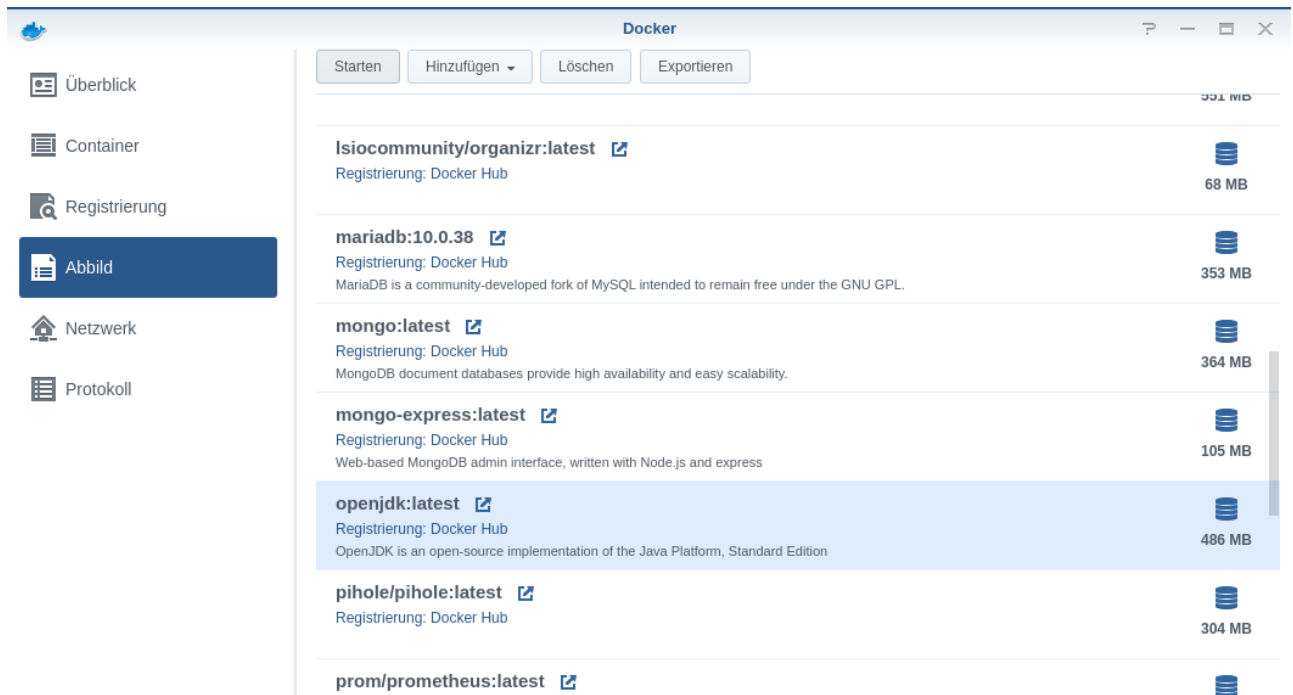
Name	Größe	Dateityp	Änderungsdatum	:
 knowledgebase		Ordner	03.02.2021 18:30:57	
 app-1.0.0.jar	30.4 MB	JAR Datei	05.02.2021 17:44:59	

2. Download the openJDK container

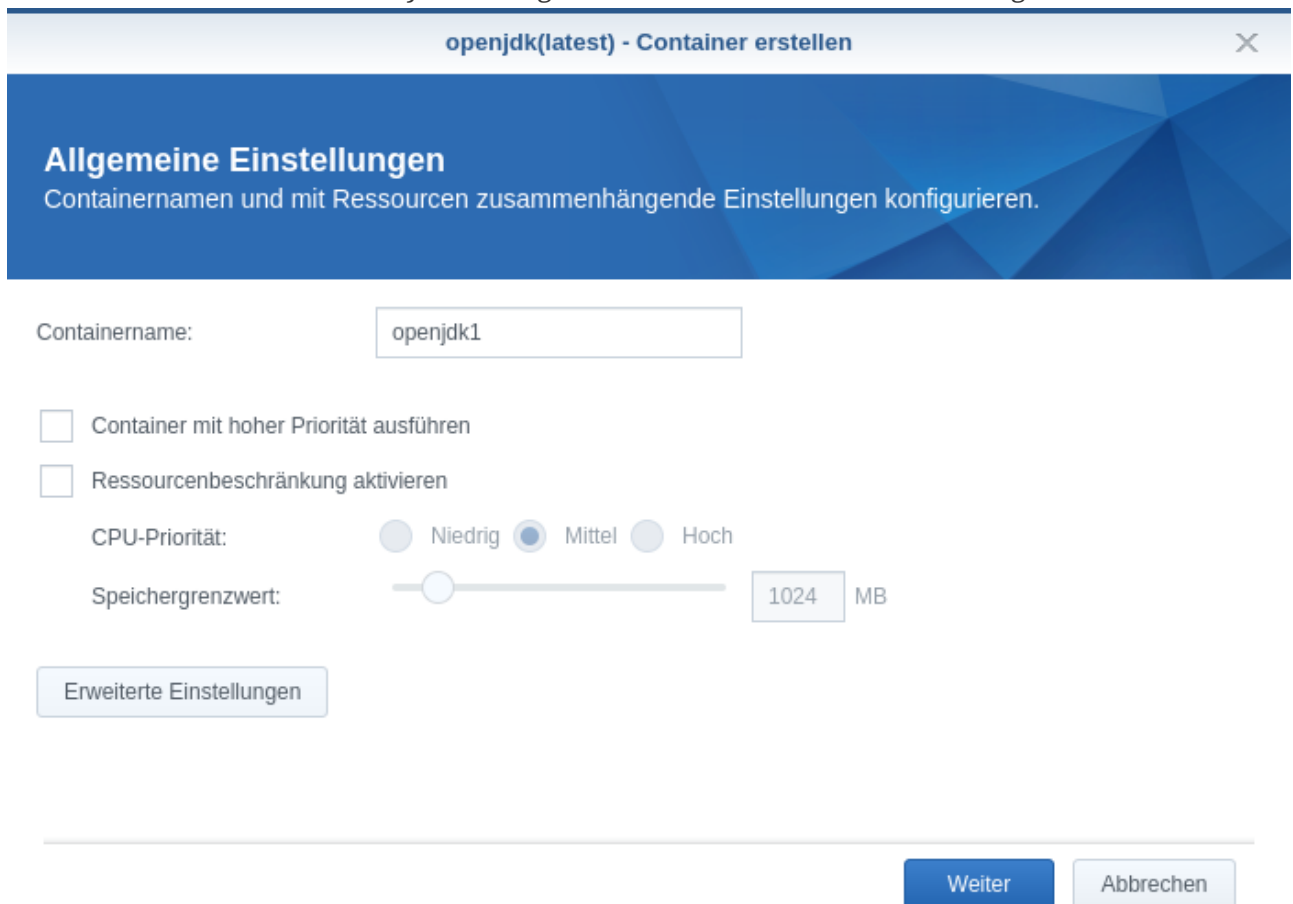


The screenshot shows the Docker Hub search results for 'openjdk'. The interface includes a sidebar with navigation links: Überblick, Container, Registrierung, Abbild, Netzwerk, and Protokoll. The main content area displays a list of Docker images. The first image is 'openjdk' by 'openjdk', described as 'OpenJDK is an open-source implementation of the Java Platform, Standard Edition', with 3K stars. Below it are several 'adoptopenjdk' images, including 'adoptopenjdk/openjdk11', 'adoptopenjdk/openjdk8', 'adoptopenjdk/openjdk8-openj9', 'adoptopenjdk/openjdk11-openj9', 'arm64v8/openjdk', and 'adoptopenjdk/openjdk12'. Each image entry shows the image name, a brief description, and the number of stars. At the bottom, there is a pagination bar showing '1' as the current page, with links for '2', '3', and '5013 Element(e)'.

3. Create a new container from this image



4. Give this container a name to your likings and click on the 'Advanced Settings' button



5. Mount your folders and files to the docker container. It is also a good idea to change this to "read only" mode, as this program does not change these files

Erweiterte Einstellungen

Volume

Netzwerk

Port-Einstellungen

Links

Umgebung

Datei hinzufügen

Ordner hinzufügen

Löschen

Datei/Ordner	Mount-Pfad	<input type="checkbox"/> Nur Lesen
docker/classificator/app-1.0.0.jar	/opt/classificator/app.jar	<input type="checkbox"/>
docker/classificator/knowledgebase	/opt/classificator/knowledgebase	<input type="checkbox"/>

6. Change the ports to your liking

Erweiterte Einstellungen

Volume

Netzwerk

Port-Einstellungen

Links

Umgebung

+

-

Lokaler Port	Container-Port	Typ
8081	8080	TCP

7. Go to the environment tab and paste this as command to run:

```
java -jar /opt/classificator/app.jar '--server.port=8081' '--knowledge_base=/opt/classificator/knowledgebase'
```

Finally, run this container. You can access the webui by going to <http://your-nas-ip:your-port> For example <http://192.168.0.3:8081>

Configuration

This is the default configuration of the application It sets the server port to 8080 and the knowledgebase-directory to **build/knowledgebase**

```
server_port: 8080
knowledge_base: build/knowledgebase

server.port: ${server_port}

logging.level:
  root: ERROR
  com.cuupa.classificator: WARN

classifier:
  kbfiles: ${knowledge_base}

monitor:
  enabled: true
  logText: true
  database-name: "monitor.db"
```

Server port

The default port as in said configuration is **8080** So if you want to access the webUI on your local machine navigate to <https://localhost:8080>

```
server_port: 8080
```

Knowledgebase

This entry means, that there has to be a folder named **build/knowledgebase** beside your jar file. You can specify which knowledgebase to use, by changing it via the command line arguments (How that works will be covered by me a [few lines down below](#)).

TIP

If this entry only contains a folder name without the specific knowledgebase, it will load the file with the highest version tag in that folder
For example if you have **kb-1.0.0.db** and **kb-1.0.1.db** it'll load the **kb-1.0.1.db**

```
knowledge_base: build/knowledgebase
```

Monitor

The monitor is enabled by default and uses the database **monitor.db**

```
monitor:
  enabled: true
  logText: true
  database-name: "monitor.db"
```

If **enabled** is set to **true**, all events will be logged. To turn it off, set it to **false**. If **logText** is enabled, the actual analyzed texts will be logged into the database. You might want to turn it off for privacy reasons

Database name

The database name is defined via **database-name**. You can change it to your likings. It uses relative paths by default. If you want to use absolute paths, change it to

```
database-name: "C:\Users\John Doe\monitor.db"
```

Changing

Via config

If you know about programming: Great! You can change it as you like for example

```
knowledge_base: knowledgebase/kb-1.0.0.db
```

or

```
server_port: 1234
```

Via command line arguments

If you don't, don't panic. You can run the application by typing

```
java -jar app.jar --server_port=8080 --knowledge_base=knowledgebase/kb-1.0.0.db
--classifier.monitor.logText=false
```

NOTE The first part simply runs the jar by the name "app.jar"

```
java -jar app.jar
```

NOTE This part sets the port to 8080 and overwrites the value of the default configuration

```
--server_port=8080
```

NOTE This part sets the location of the knowledgebase. You can use relative paths like

```
--knowledge_base=knowledgebase/kb-1.0.0.db
```

NOTE or absolute paths like

```
--knowledge_base="C:\Users\John Doe\knowledgebase\kb-1.0.0.db"
```

CAUTION Notice that, you need to quote the value as soon as you have spaces in a parameter

Components

Engine

The engine is the core component of this application. It classifies the text and extracts the metadata

Using the GUI

You can use the gui exposed at <http://address-of-your-server:port>

You can type in or paste the text to the left-hand textarea, which the engine shall analyze and hit the "Submit"-Button. The result will be presented in the right-hand area

Text

Hello World!

Senden

Result

Topic	Sender	Metadata
OTHER	UNKNOWN	NameValue

Using the REST-API

The engine exposes several methods for analyzing the input text. The most simple one receives the text as a string and returns a `List<SemantikResult>`

The endpoint-path is:

```
"/api/rest/1.0/classifyText"
```

If you want to analyze anything except plain text the method accepts any byte array and uses a

combination of **PDFBox** and **Apache Tika** to extract its contents for you.

```
"/api/rest/1.0/classify"
```

TIP

There's also a method for pinging the application. This method simply returns a HTTP/200

```
"/api/rest/1.0/ping"
```

Monitor