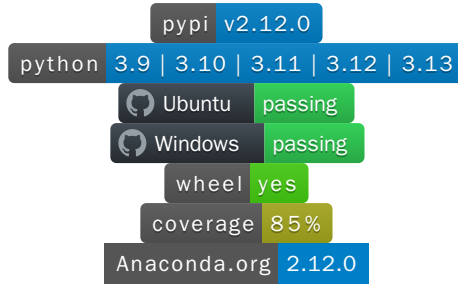


Table of Contents

- Home
 - [Overview of Scrapy.](#)
- Guide & Usage
 - [Installation and Deployment](#)
 - [Creating a Configuration](#)
 - [Running a Spider](#)
- Monitoring
 - [Spider Management](#)

Scrapy General Engine Documentation



Overview

Scrapy adalah framework **web scraping** open-source yang digunakan untuk mengekstrak data dari website. **Scrapy General Engine** bertanggung jawab untuk mengatur aliran data dari permintaan HTTP hingga hasil data yang diekstrak secara terstruktur.

Scrapy menggunakan **event-driven architecture** yang efisien dan scalable untuk menangani banyak permintaan dalam waktu singkat.

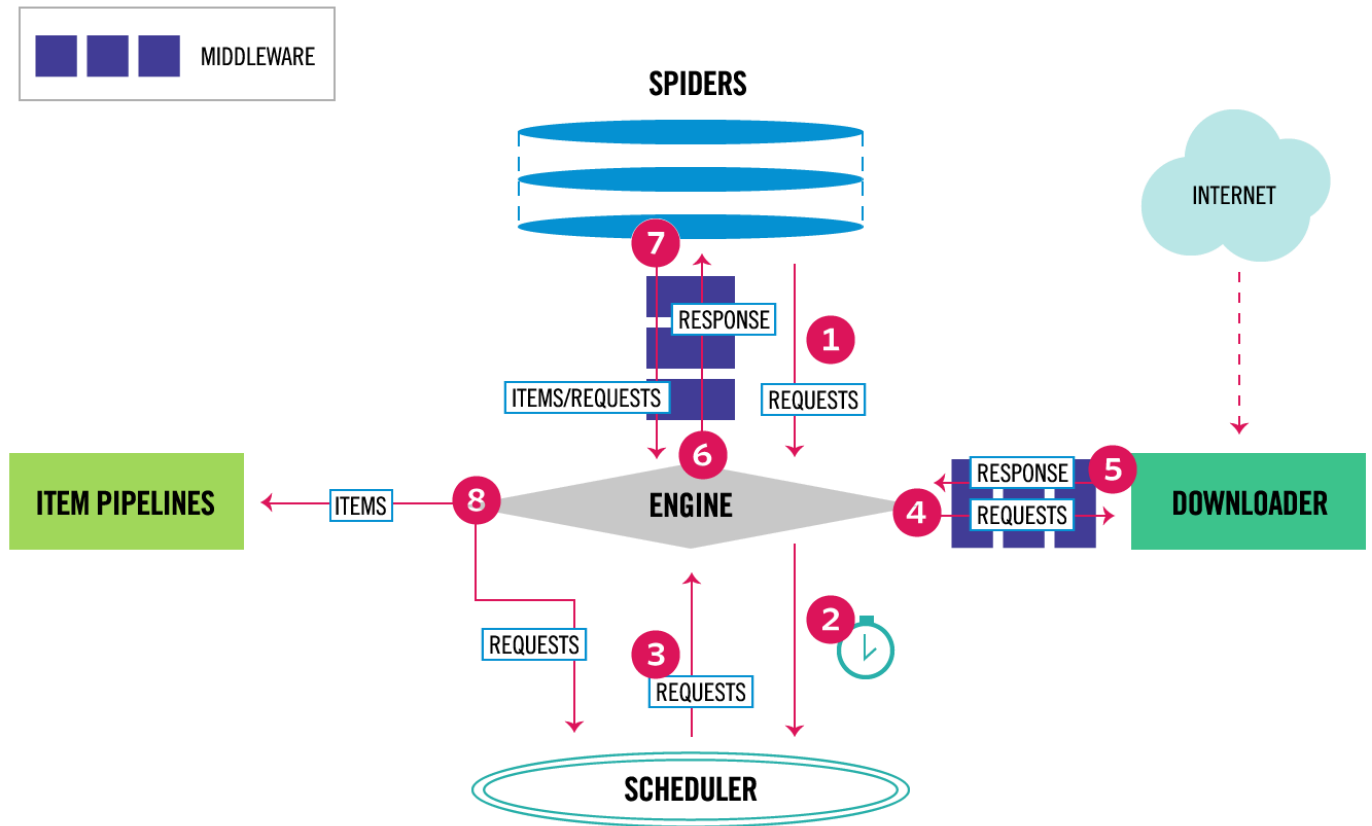
Note: Dokumentasi lengkap tentang **Framework Scrapy** kunjungi [Scrapy Documentation](#).

Architecture Scrapy

Diagram berikut menunjukkan **gambaran umum arsitektur Scrapy** dengan komponen-komponennya serta garis besar aliran data yang terjadi di dalam sistem (ditunjukkan oleh **panah merah**).

Penjelasan singkat mengenai komponen-komponen tersebut disertakan di bawah ini dengan tautan untuk informasi yang lebih rinci mengenai komponen tersebut.

Aliran data juga dijelaskan di bagian berikut.



Aliran Data

Aliran data di Scrapy dikendalikan oleh **Scrapy Engine** dan berjalan seperti ini:

1. **Mesin** (Engine) mendapatkan **Permintaan awal** (*Initial Requests*) dari **Spider**.
2. **Mesin** menjadwalkan Permintaan di **Scheduler** dan meminta Permintaan berikutnya untuk diproses.
3. **Scheduler** mengembalikan **Permintaan berikutnya** ke Engine.
4. **Mesin** mengirimkan Permintaan ke **Downloader**, melalui **Downloader Middleware** (`process_request()`).
5. Setelah halaman berhasil diunduh:
 - **Downloader** menghasilkan **Response** (halaman yang diunduh).
 - **Response** dikirim kembali ke **Engine** melalui **Downloader Middleware** (`process_response()`).
6. **Engine** menerima **Response** dari Downloader dan mengirimkannya ke **Spider**, melalui **Spider Middleware** (`process_spider_input()`).
7. **Spider** memproses **Response** dan menghasilkan:
 - **Item** yang telah di-*scrape* (data yang diekstrak).
 - **Permintaan baru** (*New Requests*) untuk diproses.

Hasil ini dikirim kembali ke **Engine** melalui **Spider Middleware** (`process_spider_output()`).
8. **Engine** melakukan dua hal:

- Mengirimkan **Item** yang telah diproses ke **Item Pipelines**.
- Mengirimkan **Permintaan baru** ke **Scheduler** untuk dijadwalkan kembali.

9. **Engine** meminta **Permintaan berikutnya** ke Scheduler dan proses ini berulang dari langkah 3.

Proses ini terus berjalan hingga tidak ada lagi permintaan yang tersedia di **Scheduler**.

Scrapy Engine Workflow

Langkah-Langkah Utama dalam Workflow Scrapy Engine:

1. Scheduler

Scheduler menerima **permintaan (request)** dari Spider dan menyimpan request dalam antrean untuk dieksekusi.

2. Downloader

Downloader mengambil request yang dijadwalkan oleh Scheduler dan mengeksekusi permintaan tersebut untuk mengambil **HTTP Response** dari website.

3. Spider

Spider menerima **respon (response)** dari Downloader, lalu:

- Mengekstrak data yang diinginkan menggunakan XPath atau CSS Selectors.
- Menghasilkan **Item** (data yang diekstrak) dan permintaan tambahan.

4. Item Pipeline

Data yang diekstrak (Items) dikirim ke **Item Pipeline** untuk diproses, disimpan, atau diformat sesuai kebutuhan (CSV, JSON, Database, dll).

5. Stats Collector

Stats Collector mencatat statistik scraping seperti jumlah request, response berhasil, dan error.

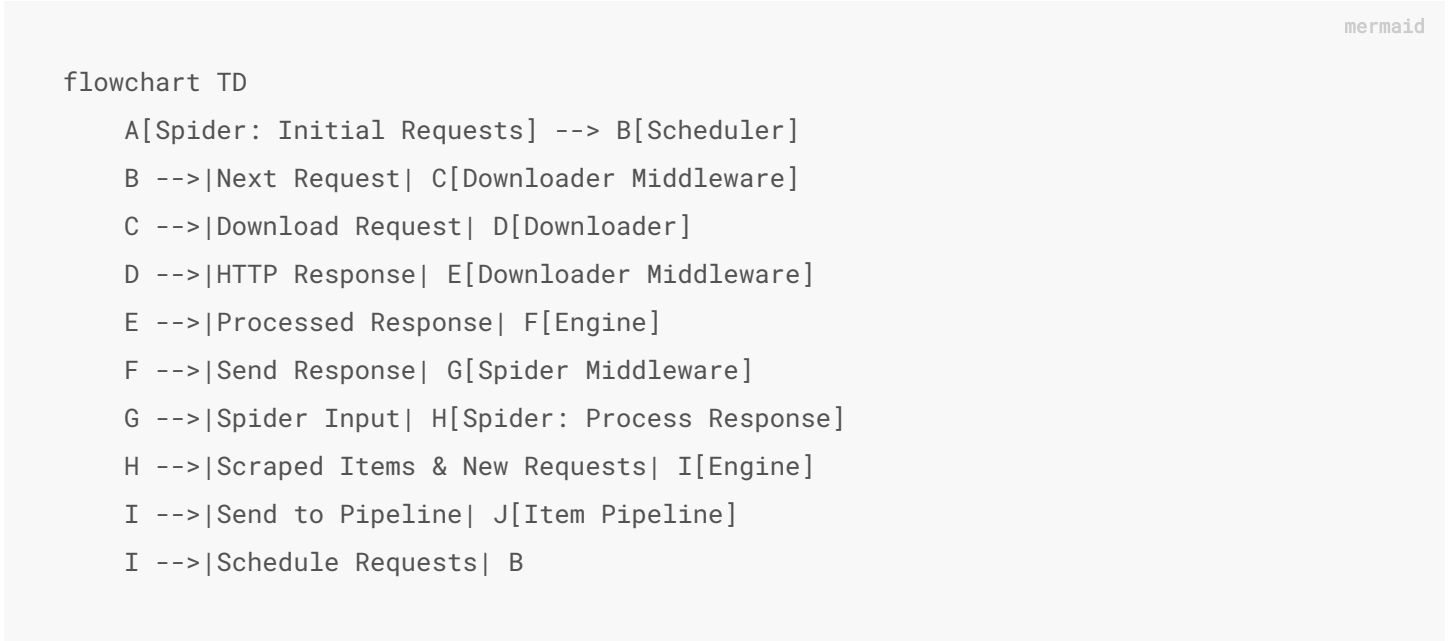
Scrapy Engine Components

Komponen	Deskripsi
Engine	Komponen inti Scrapy yang mengatur aliran permintaan dan respon.
Scheduler	Bertanggung jawab menyimpan dan menjadwalkan request untuk dieksekusi.
Downloader	Menangani permintaan HTTP dan mengembalikan respon ke Spider.

Spider	Mengekstrak data dari respon dan menghasilkan request tambahan.
Item Pipeline	Memproses, memvalidasi, dan menyimpan data yang diekstrak.
Stats Collector	Mengumpulkan statistik scraping seperti request dan error.

Flowchart Scrapy

Untuk memahami lebih dalam alur kerja Scrapy, berikut langkah-langkah yang dijelaskan secara visual:



Panduan Instalasi untuk Menjalankan Aplikasi dengan Docker

Panduan ini menjelaskan langkah-langkah untuk membangun dan menjalankan aplikasi menggunakan Docker dan Docker Compose. Anda perlu membangun image Docker terlebih dahulu dan kemudian menjalankannya menggunakan Docker Compose.

Langkah 1: Membangun Docker Image

Pastikan Anda memiliki Docker yang terinstal di sistem Anda. Jika belum, Anda dapat mengunduh dan menginstal Docker dari [situs resmi Docker](#).

1.1 Clone Dashboard Repo

```
git clone https://github.com/fossyy/general_spider_dashboard.git
```

bash

1.2 Clone General Spider Repo

```
git clone https://github.com/fossyy/general_spider.git
```

bash

1.3 Buat Dockerfile Untuk Dashboard

Pastikan Anda memiliki file `Dockerfile` di direktori aplikasi Anda. Berikut adalah contoh Dockerfile:

```
FROM python:3.10-slim as python_build

RUN apt-get update && apt-get install -y \
    build-essential && \
    pip install --no-cache-dir scrapyd-client

COPY /general_spider /src
WORKDIR /src
```

dockerfile

```
RUN scrapyd-deploy --build-egg general.egg

FROM node:current-alpine3.20 AS node_builder

COPY /general_spider_dashboard /src

WORKDIR /src

RUN npm install -g tailwindcss
RUN npm install -g clean-css-cli
RUN npx tailwindcss -i ./public/input.css -o ./tmp/output.css
RUN cleancss -o ./public/output.css ./tmp/output.css

FROM golang:1.23.4-alpine3.20 AS go_builder

RUN apk update && apk upgrade && apk add --no-cache ca-certificates tzdata

COPY /general_spider_dashboard /src
COPY --from=node_builder /src/public /src/public
COPY --from=python_build /src/general.egg /src/app/general.egg

WORKDIR /src

RUN update-ca-certificates
RUN go install github.com/a-h/templ/cmd/templ@latest
RUN templ generate
RUN go build -o ./tmp/main

FROM scratch

WORKDIR /general

COPY --from=go_builder /usr/share/zoneinfo /usr/share/zoneinfo
COPY --from=go_builder /etc/ssl/certs/ca-certificates.crt /etc/ssl/certs/
COPY --from=go_builder /src/public /general/public
COPY --from=go_builder /src/tmp/main /general

ENV TZ Asia/Jakarta

ENTRYPOINT [ "./main" ]
```

1.4 Buat Dockerfile Untuk Scrapyd (custom)

Pastikan Anda memiliki file `Dockerfile.scrapyd` di direktori aplikasi Anda. Berikut adalah contoh Dockerfile:

dockerfile

```
FROM python:3.11-slim

WORKDIR /app

RUN apt-get update && apt-get install -y git build-essential && apt-get clean

RUN git clone https://github.com/fossyy/scrapyd.git .
RUN cd scrapyd
RUN pip install --upgrade pip
RUN pip install setuptools wheel
RUN pip install -r requirements.txt
RUN pip install .

EXPOSE 6800

ENTRYPOINT ["scrapyd", "--pidfile="]
```

1.5 Bangun Docker Image Untuk Dashboard

Sekarang, untuk membangun image Docker, jalankan perintah berikut di terminal Anda:

bash

```
docker build --network host --no-cache -t dashboard .
```

1.6 Bangun Docker Image Untuk Scrapy (custom)

Sekarang, untuk membangun image Docker, jalankan perintah berikut di terminal Anda:

bash

```
docker build --network host -f Dockerfile.scrapyd --no-cache -t scrapyd .
```


Langkah 2: Menjalankan Aplikasi dengan Docker Compose

Setelah image berhasil dibangun, Anda dapat menjalankan aplikasi menggunakan Docker Compose. Pastikan Anda memiliki file `docker-compose.yml` di direktori proyek Anda.

2.1 Membuat File `docker-compose.yml`

Berikut adalah contoh file `docker-compose.yml` untuk menjalankan aplikasi menggunakan Docker Compose:

```
version: '3.8'

services:
  postgres:
    image: postgres:16.0
    restart: on-failure
    environment:
      - POSTGRES_PASSWORD=VerySecretPassword
      - POSTGRES_DB=general_spider
    volumes:
      - postgres:/var/lib/postgresql/data
    networks:
      - scrapyd
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -U user -d mydatabase" ]
      interval: 30s
      retries: 5
      timeout: 20s
      start_period: 10s

  scrapyd:
    image: scrapyd
    restart: on-failure
    depends_on:
      - postgres
    volumes:
      - /opt/general_engine:/scrapyd:Z
      - /opt/general_engine/logs:/scrapyd/logs
    environment:
      - DB_HOST=postgres
      - DB_PORT=5432
```

yaml

```

- DB_USERNAME=postgres
- DB_PASSWORD=VerySecretPassword
- DB_NAME=general_spider
- DASHBOARD_ADDRESS=http://general_engine_dashboard:8080
networks:
- scrapyd
healthcheck:
  test: [ "CMD-SHELL", "curl --silent --fail http://localhost:6800/ --max-time 5
|| exit 1" ]
  interval: 30s
  retries: 5
  timeout: 10s
  start_period: 10s

general_engine_dashboard:
  image: dashboard
  restart: on-failure
  depends_on:
    - scrapyd
    - postgres
  links:
    - scrapyd
  ports:
    - "8080:8080"
  volumes:
    - /opt/general_engine/logs:/general/logs
  environment:
    - SERVER_HOST=0.0.0.0
    - SERVER_PORT=8080
    - DB_HOST=postgres
    - DB_PORT=5432
    - DB_USERNAME=postgres
    - DB_PASSWORD=VerySecretPassword
    - DB_NAME=general_spider
    - SCRAPYD_URL=http://scrapyd:6800
  networks:
    - scrapyd

torproxy:
  image: dperson/torproxy
  environment:
    - BW=100

```

```
ports:
  - "8118:8118"
restart: always
networks:
  - scrapyd

volumes:
  scrapyd-volume:
  postgres:

networks:
  scrapyd:
```



Penting: torproxy ini mencakup **1 Browser TOR** dan **1 Reverse Tor Privoxy** , karena Spider tidak mendukung SOCKS5 secara langsung dan membutuhkan reverse proxy untuk digunakan.

2.2 Menjalankan Docker Compose

Setelah file docker-compose.yaml siap, jalankan perintah berikut untuk membangun dan menjalankan container:

```
docker compose up -d
```

bash

How To Use Config Maker

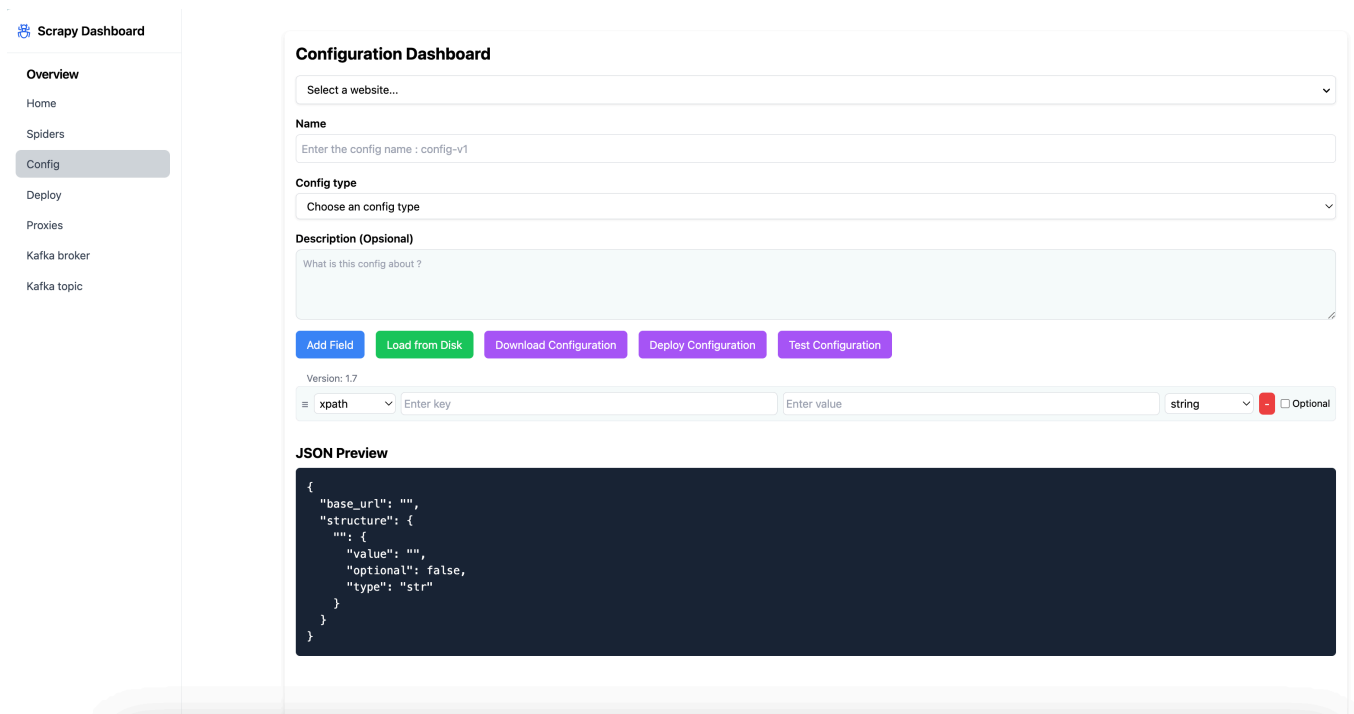
User Interface Config Maker

Initial Steps

Gunakan antarmuka aplikasi untuk membuat konfigurasi XPath yang akan digunakan untuk Scrapping. Setiap konfigurasi memungkinkan Anda mendefinisikan elemen-elemen spesifik yang akan diekstraksi dari halaman web.

Penting: Scrapy General Engine saat ini hanya mendukung **XPath 1.0** kunjungi [Dokumentasi XPath 1.0](#).

Berikut adalah tampilan antarmuka untuk membuat konfigurasi pertama Anda:



The screenshot shows the 'Configuration Dashboard' of the Scrapy Dashboard. On the left is a sidebar with navigation links: Overview, Home, Spiders, Config (highlighted), Deploy, Proxies, Kafka broker, and Kafka topic. The main area contains the configuration form. At the top is a 'Select a website...' dropdown. Below it is a 'Name' field with the value 'config-v1'. The 'Config type' is set to 'Choose an config type'. There is a 'Description (Optional)' text area. Below these are five buttons: 'Add Field', 'Load from Disk', 'Download Configuration', 'Deploy Configuration', and 'Test Configuration'. A 'Version: 1.7' label is present. Below that is a field for 'xpath' with a dropdown menu, followed by 'Enter key' and 'Enter value' fields. The 'string' type is selected, and there is an 'Optional' checkbox. At the bottom is a 'JSON Preview' section showing a JSON structure:

```
{  "base_url": "",  "structure": {    "": {      "value": "",      "optional": false,      "type": "str"    }  }}
```

Button Dashboard

Pada dashboard **Config Maker**, terdapat beberapa tombol (button) yang dapat digunakan untuk melakukan berbagai fungsi terkait pengelolaan konfigurasi. Berikut adalah penjelasan dari setiap tombol dan fungsinya:

[Add Field](#)[Load from Disk](#)[Load from Database](#)[Download Configuration](#)[Deploy Configuration](#)[Test Configuration](#)

add field

Tombol **Add Field** digunakan untuk menambahkan field baru ke dalam konfigurasi XPath. Ini memungkinkan pengguna untuk menyesuaikan konfigurasi sesuai kebutuhan Scrapping.

Ilustrasi:



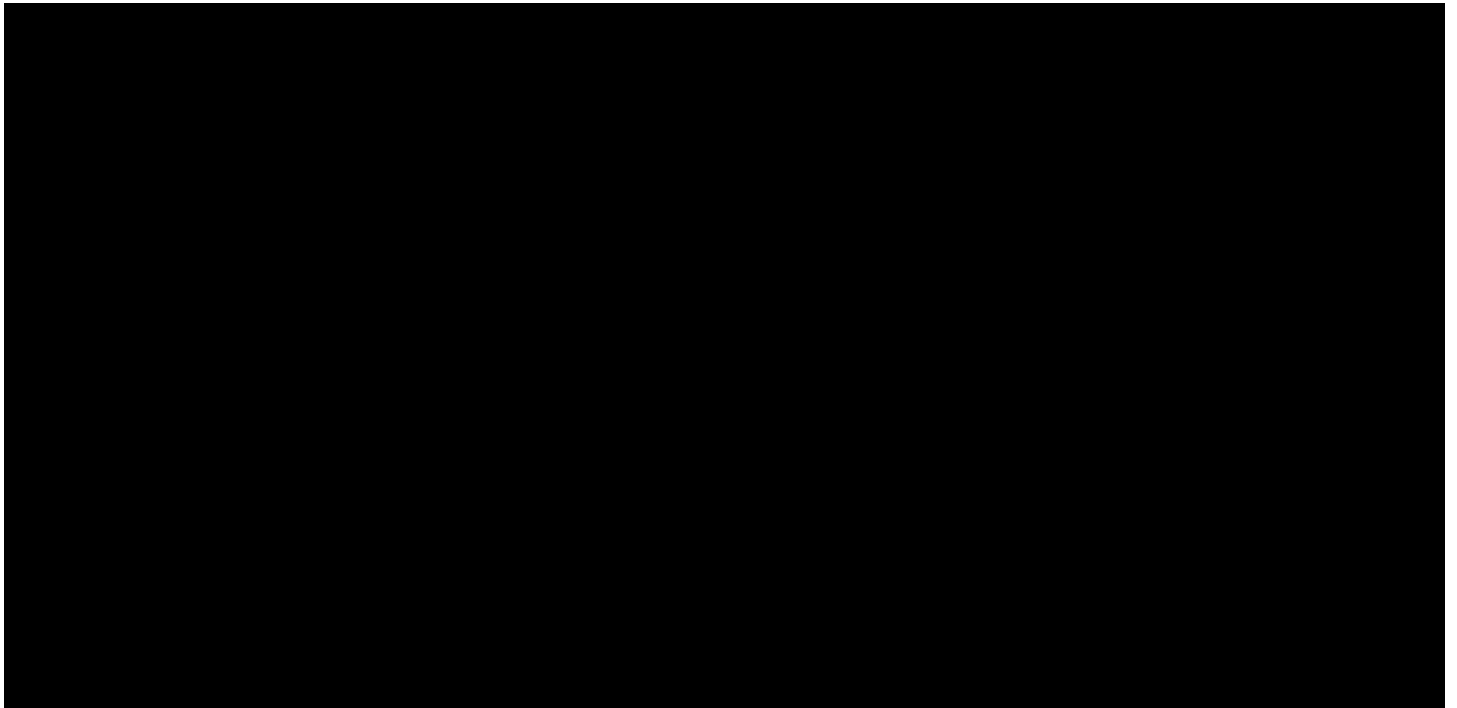
Fungsi Utama:

- Menambahkan elemen atau atribut baru ke dalam struktur konfigurasi.
- Contoh penggunaan: Menambahkan XPath baru untuk mengambil data dari elemen yang baru ditemukan pada halaman web.

load from disk

Tombol **Load from Disk** memungkinkan pengguna untuk mengimpor konfigurasi yang sudah ada dari file lokal. Dengan fitur ini, pengguna tidak perlu membuat konfigurasi dari awal setiap kali ingin menggunakan konfigurasi lama.

Ilustrasi:



Fungsi Utama:

- Mengunggah file konfigurasi dalam format JSON yang telah disimpan sebelumnya.
 - Membantu pengguna mengelola dan memuat konfigurasi dengan cepat tanpa perlu input manual.
-

load from database

Tombol **Load from Database** memungkinkan pengguna untuk mengimpor konfigurasi yang sudah ada dari database. Dengan fitur ini, pengguna tidak perlu membuat konfigurasi dari awal setiap kali ingin menggunakan konfigurasi lama.

Ilustrasi:



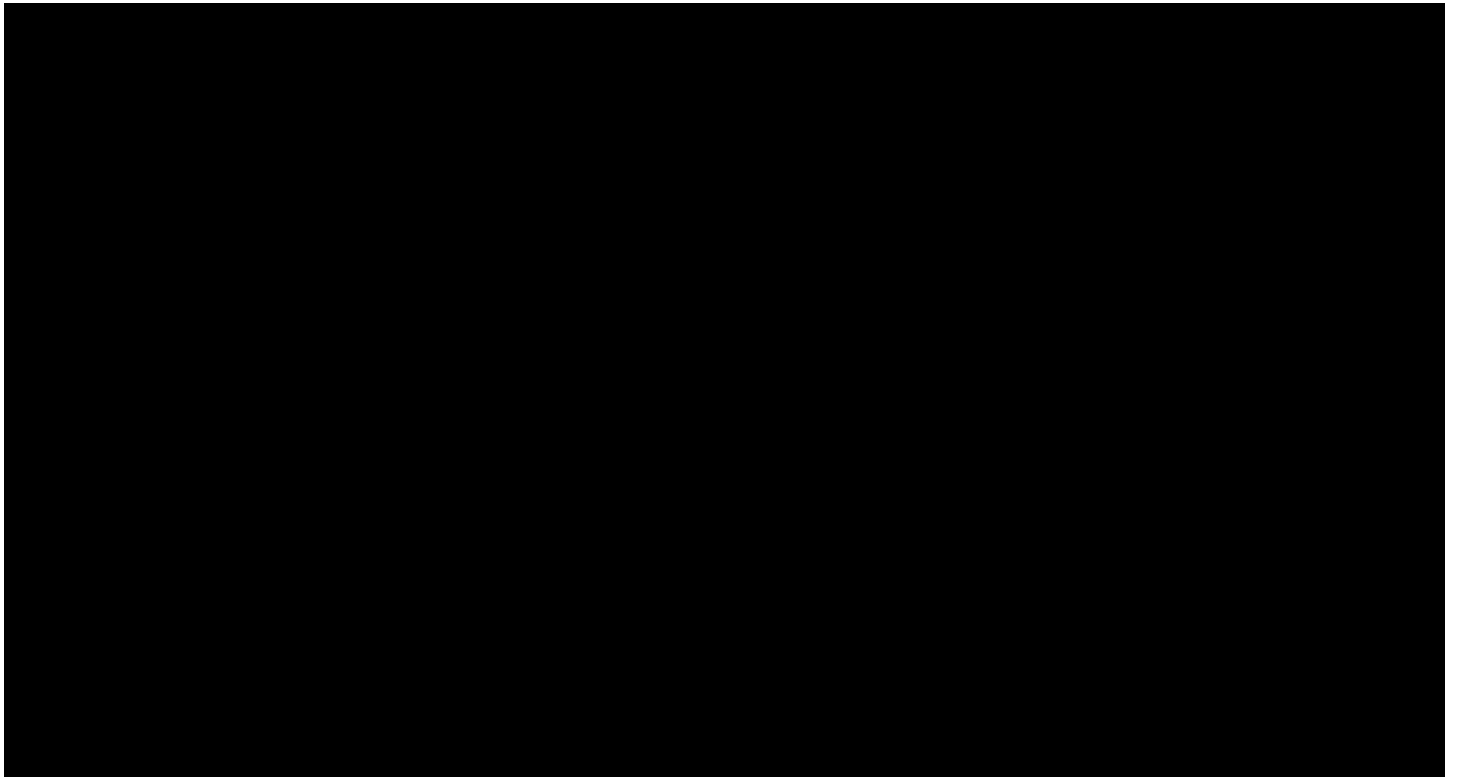
Fungsi Utama:

- Mengunggah file konfigurasi dalam format JSON yang telah disimpan sebelumnya.
- Membantu pengguna mengelola dan memuat konfigurasi dengan cepat tanpa perlu input manual.

download configuration

Tombol ini digunakan untuk mengunduh konfigurasi yang sudah dibuat atau dimodifikasi ke dalam file lokal. Dengan fitur ini, pengguna dapat menyimpan hasil konfigurasi untuk digunakan di lain waktu atau sebagai cadangan.

Ilustrasi:



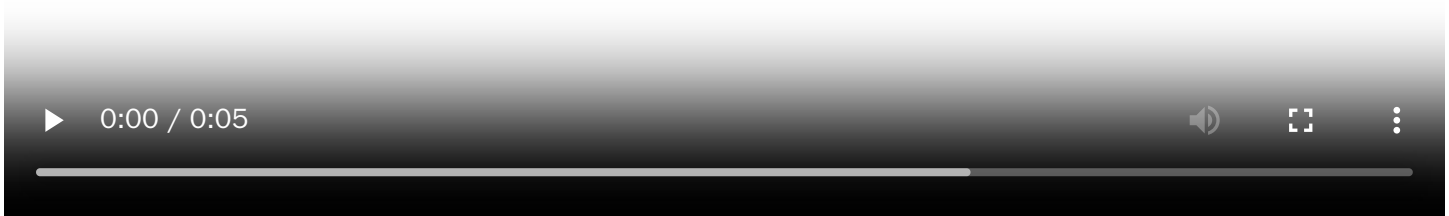
Fungsi Utama:

- Menyimpan konfigurasi dalam format JSON ke perangkat pengguna.
 - Memudahkan berbagi konfigurasi dengan tim lain.
-

deploy configuration

Tombol **Deploy Configuration** berfungsi untuk menerapkan konfigurasi yang telah dibuat ke dalam aplikasi atau sistem yang menggunakan konfigurasi tersebut. Fitur ini memastikan bahwa konfigurasi siap digunakan untuk proses Scrapping.

Ilustrasi:



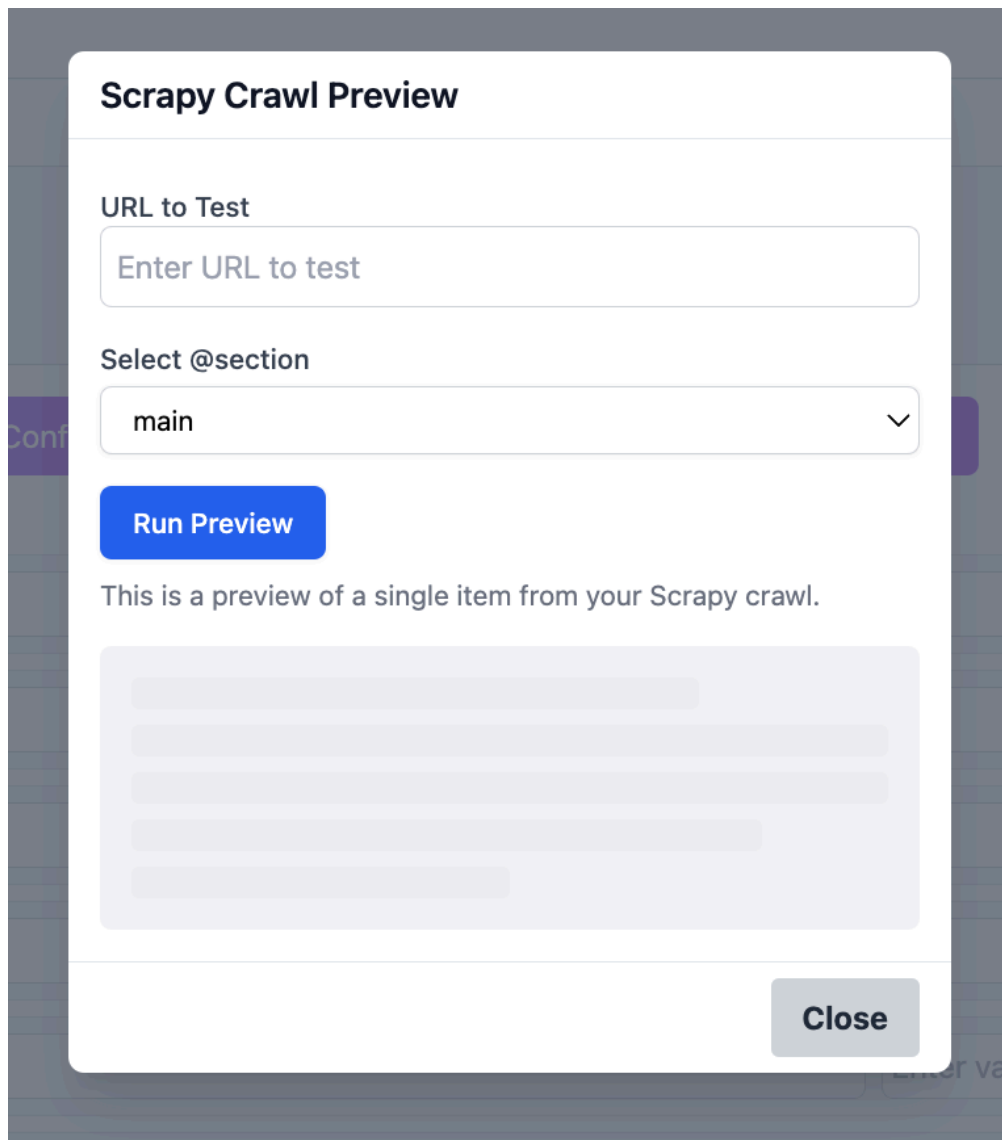
Fungsi Utama:

- Mengintegrasikan konfigurasi dengan aplikasi Scrapping.
- Memastikan konfigurasi diterapkan secara langsung di lingkungan produksi atau pengujian.

test configuration

Tombol Test Configuration memungkinkan pengguna untuk menguji konfigurasi XPath yang telah dibuat. Dengan fitur ini, pengguna dapat memastikan bahwa konfigurasi bekerja sesuai harapan sebelum diterapkan.

Ilustrasi:

A screenshot of a 'Scrapy Crawl Preview' dialog box. The dialog has a title bar 'Scrapy Crawl Preview'. Below the title, there is a section 'URL to Test' with a text input field containing the placeholder 'Enter URL to test'. Below that is a section 'Select @section' with a dropdown menu showing 'main' and a downward arrow. A blue button labeled 'Run Preview' is positioned below the dropdown. Underneath the button, there is a text line: 'This is a preview of a single item from your Scrapy crawl.' Below this text is a light gray rectangular area containing five horizontal bars of varying lengths, representing a preview of data. In the bottom right corner of the dialog, there is a gray button labeled 'Close'.

Fungsi Utama:

- Mengecek apakah XPath yang ditentukan benar-benar dapat mengambil data yang diinginkan dari halaman web.
- Memberikan umpan balik kepada pengguna jika terdapat kesalahan pada konfigurasi.

Tips: Gunakan struktur folder yang konsisten untuk mengelola file konfigurasi XPath Anda.

Template XPath JSON

Version: 1.1

≡

_list

▼

Enter value

+ -

≡

_pagination

▼

Enter value

-

≡

xpath

▼

Enter key

Enter value

string

▼

-

☐ Optional

≡

@section

▼

Select tag

▼

@

+ -

≡

constraint

▼

Enter key

Enter value

-

≡

_loop

▼

Select tag

▼

Enter key

Enter value

+ -

JSON Preview

```

{
  "base_url": "",
  "structure": {
    "_list": {
      "_element": ""
    },
    "_pagination": "",
    "": {
      "value": "",
      "type": "constraint"
    },
    "@": {
      "_tag": ""
    },
    "_loop": {
      "_element": "",
      "_key": "",
      "_tag": ""
    }
  }
}

```

Tips: Konfigurasi Scrapping didefinisikan dalam format JSON. Template ini membantu menentukan elemen-elemen yang ingin diambil.

Structure Configuration

Setiap konfigurasi Scrapping didefinisikan dalam format JSON seperti berikut:

```

terminal
{
  "base_url": "http://example.com",
  "structure": {
    "_list": {
      "_element": "XPath untuk list elemen"
    },
    "_pagination": "XPath untuk tombol next page",
    "key_XPath_constraint": {
      "value": "XPath untuk mengambil nilai",
      "type": "string",
      "optional": false
    },
  },
}

```

```

"@hanya_place_holder_untuk_element_lain": {
  "_tag": ""
},
"_loop": {
  "_element": "XPath untuk elemen yang akan di-loop",
  "_key": "kunci untuk elemen",
  "_tag": "tag untuk elemen"
}
}
}

```



Penting: Scrapy General Engine saat ini hanya mendukung **XPath 1.0** kunjungi [Dokumentasi XPath 1.0](#).

Details Fields Structure

base url

Digunakan untuk URL dasar dari situs web yang akan diScrapping.

terminal

```
"base_url": "http://bzzzzsvqcrqtki6uym6itiixfhni37ybt7mkbjyxn2pgllzxf2qgyd.onion"
```

list

Mendefinisikan XPath untuk elemen daftar, seperti daftar produk.

Expect -> ["<https://example-url.com/product>", "<https://example-url.com/product2>", "<https://example-url.com/product3>"]

terminal

```
"_list": {
```

```
"_element": "//div[@class='product-list']/a/@href"
}
```

Penting: Scrapy General Engine saat ini hanya mendukung XPath 1.0 kunjungi [Dokumentasi XPath 1.0](#).

pagination

XPath untuk tombol next page yang mengarah ke halaman berikutnya.

terminal

```
"_pagination": "//a[@class='next-page']/@href"
```

Penting: Scrapy General Engine saat ini hanya mendukung XPath 1.0 kunjungi [Dokumentasi XPath 1.0](#).

constraint

Field untuk menentukan nilai tertentu menggunakan XPath dengan opsi tipe dan opsional.

terminal

```
"source": {
  "value": "deepweb",
  "type": "constraint"
},
```

Penting: Scrapy General Engine saat ini hanya mendukung XPath 1.0 kunjungi [Dokumentasi XPath 1.0](#).

placeholder

Section placeholder untuk elemen lain yang tidak termasuk dalam kategori utama dengan menggunakan simbol `@`.

terminal

```
"@thread": {  
  "_tag": "global"  
}
```

Penting: Scrapy General Engine saat ini hanya mendukung `XPath 1.0` kunjungi [Dokumentasi XPath 1.0](#).

loop

XPath untuk elemen yang perlu diulang, seperti komentar atau ulasan.

terminal

```
"_loop": {  
  "_element": "//div[@class='example-class']/div",  
  "_key": "comment",  
  "_tag": "root"  
}
```

Penting: Scrapy General Engine saat ini hanya mendukung `XPath 1.0` kunjungi [Dokumentasi XPath 1.0](#).

key

XPath untuk menentukan `key/kunci` dari sebuah `value/nilai` yang ingin di Scrapping.

```
"_key": "example-key",
```

Penting: Scrapy General Engine saat ini hanya mendukung **XPath 1.0** kunjungi [Dokumentasi XPath 1.0](#).

element

XPath untuk element yang perlu diulang didalam **_loop and _list** , seperti komentar atau ulasan.

```
"_element": "//div[@class='example-class']/div",
```

Penting: Scrapy General Engine saat ini hanya mendukung **XPath 1.0** kunjungi [Dokumentasi XPath 1.0](#).

value

XPath untuk mengambil **value/nilai** dari sebuah **XPath** yang ingin di scrapping.

```
"value": "example-path-to-get-value",
```

Penting: Scrapy General Engine saat ini hanya mendukung **XPath 1.0** kunjungi [Dokumentasi XPath 1.0](#).

type

Tipe data mendefinisikan nilai yang diambil oleh XPath.

terminal

```
"type": "[string, integer, list, timestamp]"
```

Note: Type memiliki 4 value yaitu: `string` , `integer` , `list` dan `timestamp`

optional

Menentukan apakah sebuah field bersifat opsional.

terminal

```
"optional": [true, false]
```

Note: Optional memiliki 2 value boolean yaitu: `true` dan `false`

tag

Tag ini akan digunakan untuk menentukan data yang akan dimapping kedalam tag tersebut

terminal

```
"_tag": "[root, global, parent]"
```

Note: Tag memiliki 3 value yaitu: `root` , `global` dan `parent`

JSON Preview

Bagian ini memberikan tampilan hasil konfigurasi dalam format JSON. Format ini memudahkan integrasi dengan aplikasi.

```
json

{
  "base_url": "http://example.com",
  "structure": {
    "_list": {
      "_element": "//div[@class='product-list']/a/@href"
    },
    "_pagination": "//a[@class='next-page']/@href",
    "source": {
      "value": "deepweb",
      "type": "string",
      "optional": false
    },
    "@": {
      "_tag": "parent"
    },
    "_loop": {
      "_element": "//div[@class='comments']/div",
      "_key": "comment",
      "_tag": "root"
    }
  }
}
```

sample config

```
json

{
  "base_url": "http://bbzzzsvqcrqtki6umym6itiixfhni37ybtt7mkbjyxn2pgllzxf2qgyd.onion",
  "structure": {
    "@landing_page": {
      "_tag": "",
      "_list": {
        "_element": "//h3[@class='node-title']/a/@href",

```

```

"@thread_topic": {
    "_tag": "",
    "_pagination": "(//a[contains(@class, 'pageNav-jump pageNav-jump--next')]/@href)[1]",
    "_list": {
        "_element": "//div[contains(@class, 'structItem--thread')]/div[@class='structItem-title']/a/@href",
        "_pagination": "(//a[contains(@class, 'pageNav-jump pageNav-jump--next')]/@href)[1]",
        "@thread": {
            "_tag": "global",
            "source": {
                "value": "deepweb",
                "type": "constraint"
            },
            "name": {
                "value": "//h1[@class='p-title-value']/text()",
                "optional": false,
                "type": "str"
            },
            "timestamp": {
                "value": "//div[@class='p-body-header']/time/@title",
                "optional": false,
                "type": "timestamp"
            },
            "content": {
                "value": "string(//article[contains(@class, 'js-post ')]
[1]//div[@class='bbWrapper'])",
                "optional": false,
                "type": "str"
            },
            "username": {
                "value": "string(//article[contains(@class, 'js-post ')]
[1]//a[@class='username '])",
                "optional": true,
                "type": "str"
            },
            "media_url": {
                "value": "(//article[contains(@class, 'js-post ')]
[1]//div[@class='bbWrapper']/img/@src | (//article[contains(@class, 'js-post ')]
[1]//div[@class='bbWrapper']/video/source/@src ",
                "optional": false,

```

```

        "type": "list"
    },
    "_loop": {
        "_element": "(//article[not(ancestor::div[contains(@class, 'js-tp
rReplyMessageContainer')])] and not(contains(@class, 'message-body')))[
position()>1]",
        "_key": "thread_reply",
        "_tag": "parent",
        "id": {
            "value": "(./@data-content)[1]",
            "optional": false,
            "type": "str"
        },
        "username": {
            "value": "(./a[contains(@class, 'username')]/text() |
./a[contains(@class, 'username')]/span/text())[1]",
            "optional": false,
            "type": "str"
        },
        "content": {
            "value": "string(./div[@class='bbWrapper'])[1]",
            "optional": false,
            "type": "str"
        },
        "timestamp": {
            "value": "(./div[@class='meta-date']/@title)[1]",
            "optional": false,
            "type": "timestamp"
        },
        "media_url": {
            "value": "(./img/@src | ./video/source/@src)[1]",
            "optional": true,
            "type": "list"
        },
        "_loop": {
            "_element": ".//div[@class='message-
inner']//article[contains(@class, 'message message--post')]",
            "_key": "thread_comment_reply",
            "_tag": "root",
            "id": {
                "value": ".@data-content",
                "optional": false,

```

```
"type": "str"
},
"username": {
    "value": "(./a[contains(@class, 'username')]/text() | ./a[contains(@class, 'username')]/span/text())[1]",
    "optional": false,
    "type": "str"
},
"content": {
    "value": "string(./div[@class='bbWrapper'])",
    "optional": false,
    "type": "str"
},
"timestamp": {
    "value": ".//div[@class='meta-date']/@title",
    "optional": false,
    "type": "timestamp"
},
"media_url": {
    "value": ".//img/@src | .//video/source/@src",
    "optional": true,
    "type": "list"
}
}
}
}
}
}
}
}
```

Tips: Jika ingin melihat `sample-config.json` lebih lanjut kunjungi [sample-config-repository](#).

How To Run Spider General Engine 🚀

User Interface Deploy Spider

Scrapy Dashboard

Overview

Home

Spiders

Config

Deploy

Proxies

Kafka broker

Deploy New Spider

Select Website Domain

Choose a domain

Select Configuration

Choose a configuration

Select Proxies

No proxies available at the moment.
Please check the [proxy status page](#) for available proxy information.

Cookie JSON (Optional)

```
{  
  "cookie_1": "value_1",  
  "cookie_2": "value_2",  
  "cookie_3": "value_3",  
  "cookie_4": "value_4",  
}
```

Parse JSON

Parse Header

Output Destination

Choose an output destination

Deployment Time

Choose when to deploy the spider

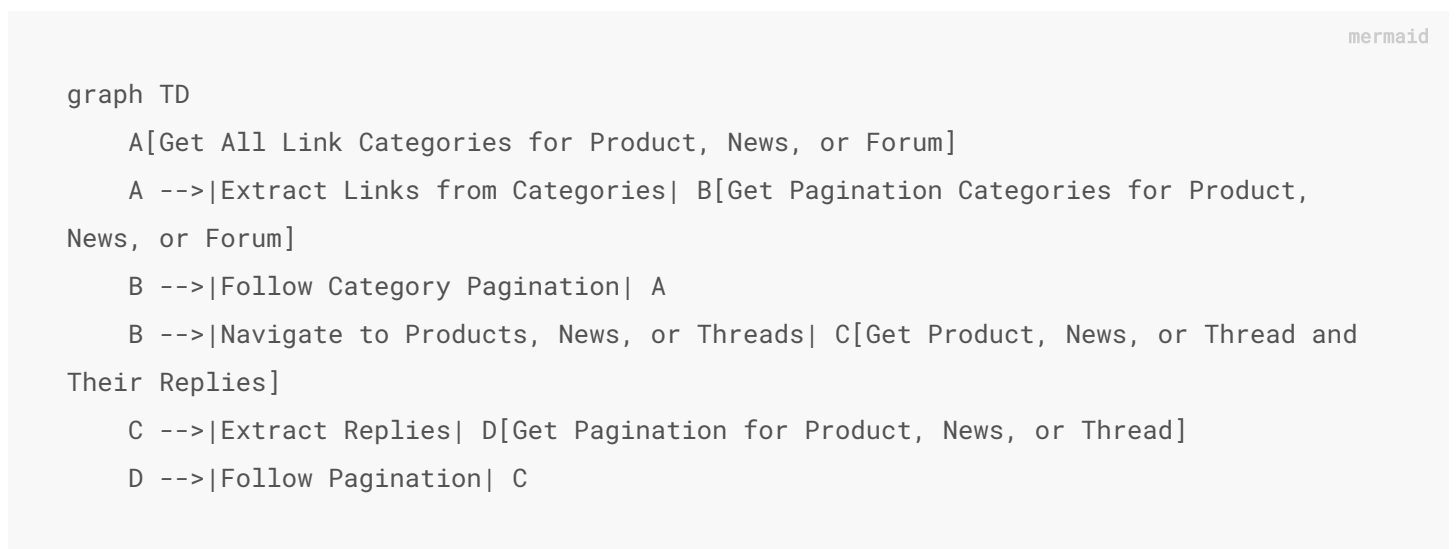
☒ Run Now ☐ Schedule Run

Additional Spider Custom Settings (JSON)

```
{"DOWNLOAD_DELAY": 1, "CONCURRENT_REQUESTS": 16}
```

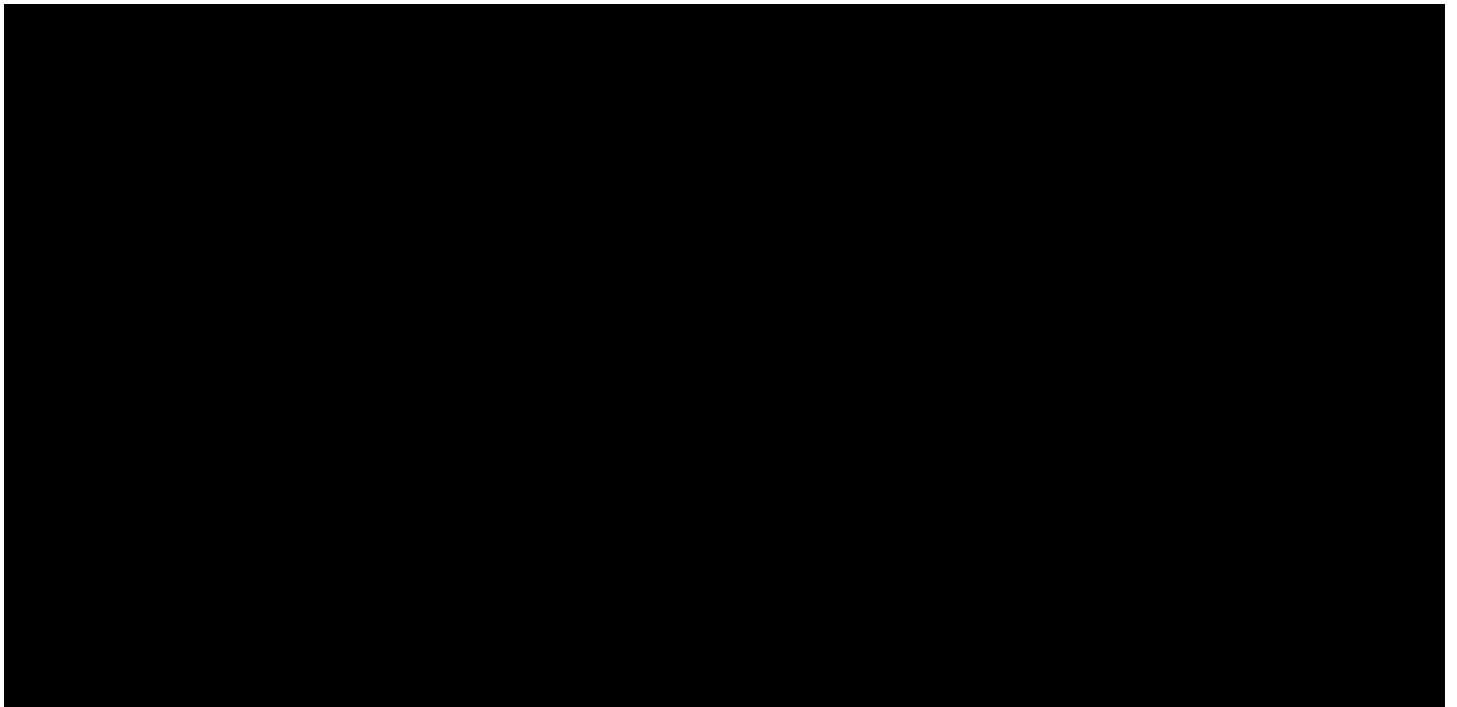
How it Works

Simple flow of scrapping process



Deploy New Spider

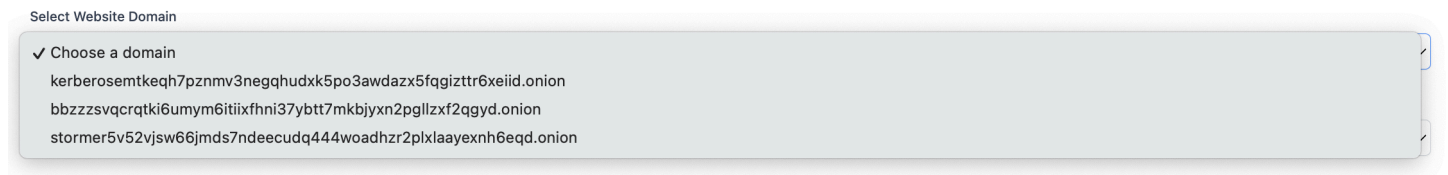
Antarmuka ini digunakan untuk melakukan deploy spider dalam sebuah aplikasi scraping. Berikut adalah penjelasan untuk setiap elemen pada halaman ini:



Select Website Domain

Dropdown ini digunakan untuk memilih domain website yang akan menjadi target scraping. Pilih salah satu domain yang telah terdaftar sebelumnya.

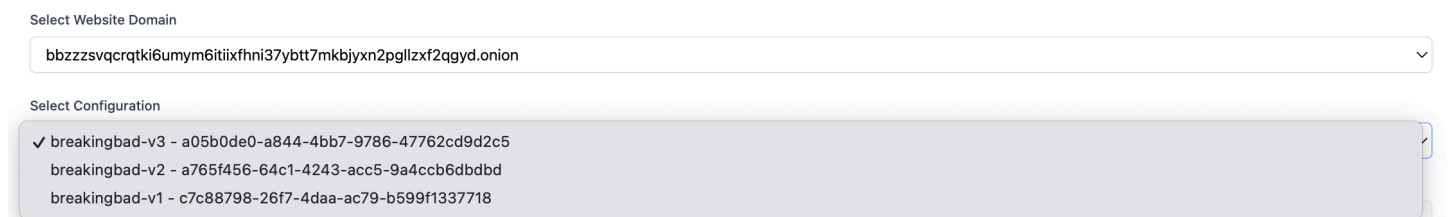
Ilustrasi



Select Configuration

Dropdown ini digunakan untuk memilih konfigurasi yang akan digunakan untuk spider. Konfigurasi mencakup pengaturan seperti XPath, header, cookie, dan pengaturan lainnya.

Ilustrasi

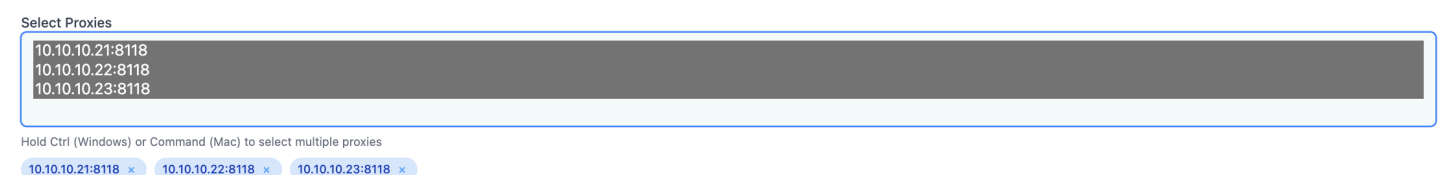


Select Proxies

Bagian ini digunakan untuk menentukan proxy yang akan digunakan selama proses scraping. Proxy membantu menyembunyikan identitas IP asli dan mencegah pemblokiran.

Proxies Available

Ilustrasi





Penting: Proxy ini mencakup **1 Browser TOR** dan **1 Reverse Tor Privoxy** , karena Spider tidak mendukung SOCKS5 secara langsung dan membutuhkan reverse proxy untuk digunakan.

- Jika tidak ada proxy yang tersedia, akan muncul pesan:

*"No proxies available at the moment. Please check the **proxy status page** for available proxy information."*

Proxies No Available

Ilustrasi

Select Proxies

Please select an item in the list.

No proxies available at the moment.
Please check the [proxy status page](#) for available proxy information.

Add Rotating Proxies



Penting: Jika **Rotating Tor Proxy** tidak tersedia/kosong kunjung tab **Proxies** untuk menambahkan **Rotating Tor Proxy** .

Ilustrasi

Advanced Proxy Management

Add New Proxy

Enter proxy address : 127.0.0.1:8118

HTTP +

Current Proxies

Test All Proxies

● Used ● Online ● Offline ● Checking ● Unchecked

● 10.10.10.21:8118 (http)	ⓘ	⌵
● 10.10.10.22:8118 (http)	ⓘ	⌵
● 10.10.10.23:8118 (http)	ⓘ	⌵

Cookie JSON (Optional)

Di sini, pengguna dapat menambahkan cookie dalam format JSON untuk dikirim bersama permintaan HTTP.

Contoh format JSON:

json

```
{
  "cookie_1": "value_1",
  "cookie_2": "value_2",
  "cookie_3": "value_3",
  "cookie_4": "value_4"
}
```

Output Destination

Dropdown ini digunakan untuk memilih tujuan penyimpanan data hasil scraping. Beberapa opsi umum mungkin termasuk:

Local Storage Server

Ilustrasi

Choose an output destination

☒ Local

☐ Kafka

Note: Jika memilih Output Destination `local` ini akan menyimpan result didalam `server` .

Kafka Brokers

Kafka Brokers Available

Ilustrasi

Output Destination

Kafka

Kafka Configuration

Kafka Brokers

3 broker(s) selected

Search brokers...

Deselect All

☒ bintaro-cluster | 10.10.10.7:9092

☒ bintaro-cluster | 10.10.10.8:9093

☒ bintaro-cluster | 10.10.10.9:9094

Kafka Topics

example-topics

Tips: Jika memilih Output Destination **Kafka** ini akan menyimpan result didalam **Kafka Brokers** yang sudah dikonfigurasi.

Kafka Brokers No Available

Ilustrasi

Kafka Configuration

Kafka Brokers

Select Kafka Brokers

Search brokers...

Select All

Kafka Topics

Penting: Jika **Kafka Brokers & Kafka Topics** tidak tersedia/kosong kunjung tab **Kafka Brokers** untuk menambahkan **Kafka Brokers** .

Add Kafka Brokers

Ilustrasi

Kafka Brokers

Search brokers...

Add New Broker

BROKER ID	BROKER NAME	HOST	PORT	STATUS	ACTIONS
66eb8529-5b55-4dd5-a3fb-d9bad5b100d7	bintaro-cluster	10.10.10.7	9092	TODO: IMPLEMENT	<a>Edit <a>Remove
a50eb311-d5c2-4175-9060-aec8cd86b681	bintaro-cluster	10.10.10.8	9093	TODO: IMPLEMENT	<a>Edit <a>Remove
d8dc755a-4364-452b-9f08-129b34a80f90	bintaro-cluster	10.10.10.9	9094	TODO: IMPLEMENT	<a>Edit <a>Remove

Note: Gambar dibawah ini untuk menambahkan **Kafka Brokers** .

Ilustrasi

Kafka Broker Configuration

Broker Group

cluster-example

Kafka group name for the broker (e.g., bintaro-cluster).

Broker Addresses

10.10.10.7:9092, 10.10.10.8:9093, 10.10.10.9:9094

Comma-separated list of host:port pairs for Kafka brokers.

Save Kafka Broker Configuration

Deployment Time

Pengguna dapat menentukan waktu deploy spider

Run Now

Spider langsung dijalankan setelah deploy.

Ilustrasi

Deployment Time

Choose when to deploy the spider



Run Now



Schedule Run

Tips: Untuk melihat contoh `Deployment Spider Time Now`, dapat mengunjungi [Time Now Spider](#).

Schedule Run

Spider dijadwalkan untuk dijalankan pada waktu tertentu.

Ilustrasi

Deployment Time

Choose when to deploy the spider

☐ Run Now ☒ Schedule Run

Schedule Type

Choose between simple or advanced scheduling

Minute
Every minute

Hour
Every hour

Day of Month
Every day

Month
Every month

Day of Week
Every day

Generated Cron Expression

Cron: * * * * *

This is the cron expression generated from your selections above.

Next Run Time

12/19/2024, 9:48:36 PM

This is when the spider will next run based on the current schedule.

Tips: Untuk melihat contoh `Deployment Spider Scheduled` , dapat mengunjungi [Scheduled Spider](#).

Additional Custom Settings

Bagian ini memungkinkan pengguna untuk menambahkan pengaturan tambahan dalam format JSON. Pengaturan ini dapat mencakup:

DOWNLOAD_DELAY: Waktu jeda antar permintaan (dalam detik). CONCURRENT_REQUESTS: Jumlah permintaan yang dijalankan secara bersamaan. Contoh format JSON:

```
json
{
  "DOWNLOAD_DELAY": 1,
  "CONCURRENT_REQUESTS": 16,
  "CONCURRENT_ITEMS": 100,
  "CONCURRENT_REQUESTS_PER_DOMAIN": 8
}
```

Tips: Jika ingin melihat `additional custom settings` lebih lanjut kunjungi [Scrapy Settings](#).

Monitoring And Management Spider General Engine



User Interface Spider

Bagian ini menunjukkan antarmuka pengguna (User Interface) untuk memantau dan mengelola spider yang sedang berjalan atau terjadwal.

The screenshot shows the 'Spiders' section of the Scrapy Dashboard. On the left is a sidebar with navigation links: Overview, Home, Spiders (selected), Config, Deploy, Proxies, and Kafka broker. The main content area is titled 'Spiders' and includes a 'Refresh' button. Below the title is a 'Domain List' table with two entries. Each entry shows a spider name, a status icon, and a 'View Details' button. The first spider, 'bbzzsvqcrqtki6umym6ititxfhnl37ybt7mkbjyn2pgllxf2qgyd.onion', has a status of 'Last crawled: 7 hours 0 minutes 22 seconds ago' and shows 0 Active, 0 Pending, 1 Finished, and 0 Scheduled spiders. The second spider, 'stormer5v52vjsw66jmds7ndeecudq444woadhxr2plxaayexnh6eqd.onion', has a status of 'Last crawled: Never been run before :)' and shows 0 Active, 0 Pending, 0 Finished, and 0 Scheduled spiders. There is also an 'Add Domain' button in the top right corner of the table area.

Domain List
<div><div></div><div>bbzzsvqcrqtki6umym6ititxfhnl37ybt7mkbjyn2pgllxf2qgyd.onion</div><div>View Details</div></div> <div>Last crawled: 7 hours 0 minutes 22 seconds ago 0 Active Spiders 0 Pending Spiders 1 Finished Spiders 0 Scheduled Spiders</div>
<div><div></div><div>stormer5v52vjsw66jmds7ndeecudq444woadhxr2plxaayexnh6eqd.onion</div><div>View Details</div></div> <div>Last crawled: Never been run before :) 0 Active Spiders 0 Pending Spiders 0 Finished Spiders 0 Scheduled Spiders</div>

Tampilan antarmuka utama yang mencakup daftar spider, statusnya, dan opsi pengelolaan seperti menjalankan, menghentikan, atau menjadwalkan spider.

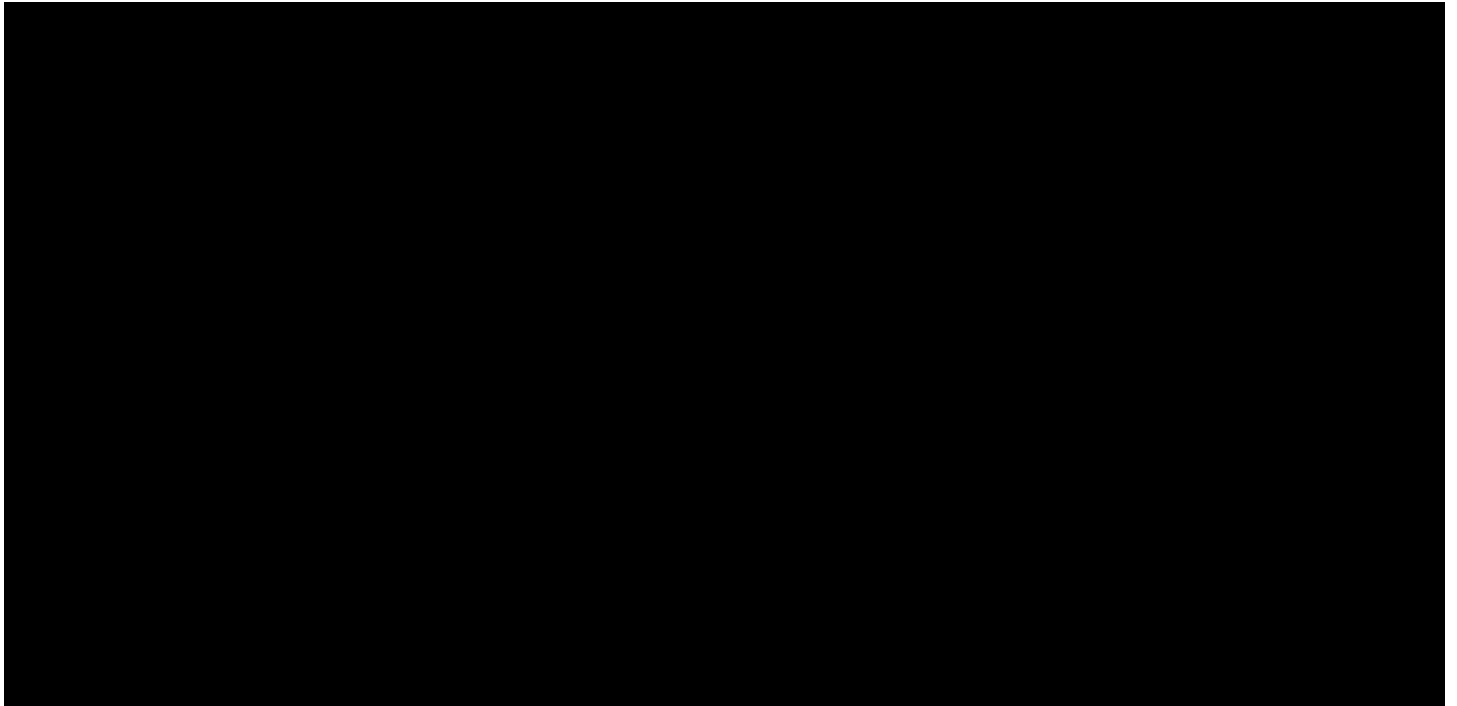
Running Now Spider Details



Bagian ini menampilkan detail spider yang sedang berjalan secara real-time. Video yang disematkan memberikan panduan visual untuk memahami informasi berikut:

- Status spider yang sedang berjalan.
- Statistik seperti jumlah halaman yang sudah di-scrape, error yang ditemukan, dan waktu eksekusi.
- Fitur penghentian atau penyesuaian langsung pada spider yang aktif.

Scheduled Spider Details



Bagian ini menampilkan detail spider yang telah dijadwalkan untuk berjalan pada waktu tertentu. Video yang disematkan memberikan panduan visual untuk memahami fitur berikut:

- Daftar spider yang dijadwalkan beserta waktu eksekusinya.
- Kemampuan untuk mengedit jadwal, menunda, atau menghapus spider dari daftar jadwal.
- Informasi status seperti "pending" atau "queued."