

Introduction

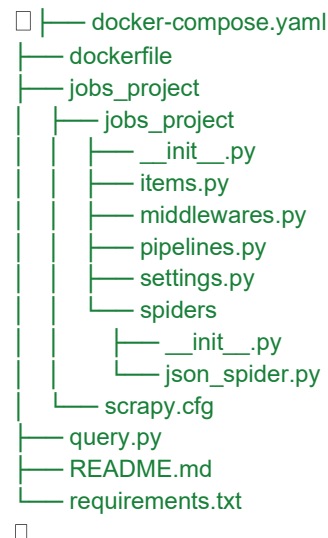
The objective of this project is to develop a scraping pipeline using Python and [Scrapy](#), which extracts data from a json file, stores it in PostgreSQL and (optional) MongoDB databases, and (optional) uses Redis for caching. The entire application should be containerized using Docker and orchestrated with Docker Compose.

Note: PostgreSQL is required for this project. MongoDB and Redis are optional (bonus points).

Note: Please include a 2 min video to demo your work

Part 1 - Setting Up Environment

Have Docker and Docker Compose installed on your system. Create a project folder with the following structure:



- **Compose File:** Create a `docker-compose.yaml` file to define services: Scrapy, PostgreSQL, MongoDB, Redis.
 - **Scrapy Service:** Either use a Python / Conda image or mount an Ubuntu / Debian image.
 - **PostgreSQL Service:** Use `PostgreSQL` image, set necessary environment variables.
 - **MongoDB Service (Optional):** Use `MongoDB` image.
 - **Redis Service (Optional):** Use `redis` or `redis-stack-server` image.
- **Dockerfile:** Create a `dockerfile` for the scrapy service.
- **Requirements:** include a `requirements.txt` file in your project that include Scrapy, psycopg2 (or an equivalent PostgreSQL adapter), PyMongo (optional), and any other libraries you use in your project. Specify the versions of these packages.

- **Documentation:** Document the setup process, how to run the project, and any configurations needed. Provide a [README.md](#) file explaining:
 - Setup instructions.
 - A brief description of the pipeline process.
- **Submission:**
 - The application should work as intended.
 - Clean, well-organized, and documented code.
 - Proper handling of common issues in database operations.
 - Submit the entire project folder as a compressed file (ZIP or TAR) or provide a link to a Git repository. Only share the scripts, do not share large files or any data.

Part 2 - Raw Data

The raw JSON files can be found at [s01.json](#) and [s02.json](#). Within these files, the most important information we are interested in is nested under the `jobs` key. Iterate over each entry in the `jobs` array and use the item pipeline you will work on to store them in the database.

Part 3 - Scrapy Pipeline

- **Initialize Scrapy Project:** Initialize a Scrapy project, name it `jobs_project`.
- **Create a Spider:** Write a spider in `json_spider.py` to scrape data from this html template. Use `json` format to retrieve the items.

```
import json
import scrapy
```

```
class Jobpider(scrapy.Spider):
    name = 'job_spider'
    custom_settings = {
        'ITEM_PIPELINES': {
            'local_spider.pipelines.yourPipeline': 300,
        },
    }

    def __init__(self, **kwargs):
        # your code here
        pass

    def start_requests(self):
        # your code here
        # make sure you can send a request locally at the file
        # if you can't get this to work, do not waste too much time here
        # instead load the json file inside parse_page
        yield scrapy.Request(
            url='file:///home/PATH_TO_JSON',
```

```

        callback=self.parse_page,
    )

def parse_page(self, response):
    # your code here
    # load json files using response.text
    # loop over data
    # return items
    pass

```

□

- **Define Items:** In `items.py`, define the structure of the items you will scrape.
 - Make sure that date fields are in the appropriate date format.
- **Pipelines:** Modify `pipelines.py` and within `json_spider.py` activate `ITEM_PIPELINES` component(s) to handle:
 - **Database-Raw:** Configure the pipeline to store all scraped data into `PostgreSQL`, store the items in `raw_table`.
 - **MongoDB (Optional Bonus):** Store items in `raw_collection`.
 - **Cache (Optional Bonus):** Use Redis to cache items that have already been scraped to avoid duplicate processing.

Part 4 - Database Query

- Include a `query.py` script with Database class(es) designed to establish connections to both `PostgreSQL` and `MongoDB` (optional) servers. The goal of this script is to retrieve all processed data and organize it into a `CSV` file format.