**Tech Stack:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Framework/Tools** |  | **Description** | **References** |
| **Crawler/Scraper** | Scrapy Framework | Scrapy is an open-source framework for crawling websites and extracting structured data from them. | <https://scrapy.org/> |
| **Headless Browser** | Aquarium container – (Multiple Splash instances with load balancer) | Aquarium helps us to have multiple Splash (headless browser) instances with load balancing(HAProxy) to increase the rendering speed. | <http://splash.readthedocs.io/en/stable/index.html>  <https://github.com/TeamHG-Memex/aquarium> |
| **Codebase** | GitHub | We use github and manage deployments from trunk. Jenkins builds artifacts from code base and stores them in artifactory. Service is deployed by pulling artifacts from artifactory | <https://github.com/> |
| **DB** | MongoDB | Using mongodb to store the scraped product information like product name , price and available SKUs etc. | <https://www.mongodb.com/> |
| **Build & Release** | Jenkins | Build done on Jenkins server.  Same artifacts deploy through environments as promoted. | <https://jenkins.io/> |
| **Dependency Management** | Pygradle | PyGradle leverages [Gradle](https://gradle.org/) to empower Python's existing ecosystem to solve problems like dependency management (**But need to setup artifactory with needed python packages with enhanced ivy metadata**) | <https://github.com/linkedin/pygradle> |
| **Artifactory** | JFrog Artifactory | Artifactory Server | <https://www.jfrog.com/> |
| **Deploy** | Scrapyd (or) ScrapingCloud | Now we are using ‘Scrapyd’ to deploy Scrapy service and for basic job monitoring if required we can leverage the cloud solution for scheduling and monitoring purpose. | [https://scrapyd.readthedocs.io/en/latest/index.html#](https://scrapyd.readthedocs.io/en/latest/index.html)  <https://scrapinghub.com/scrapy-cloud> |

**Scrapy High-level Architecture:**

****

**Brief introduction of the approach:**

Creating a Scrapy project requires one to implement a spider to define which webpages should be crawled and what data should be extracted, as explained above. Spiders are the integral part of scrapy.

Spider can crawl for a brand or can crawl for division or/and category for a brand. The high-level context is explained in Fig 1.

We have created dedicated Spiders for different tasks.

1. Crawling Spider
2. Scraping Spider

**Crawling Spider** intended to crawl and fetch the product URLs based on the brand and division combination (these URLs will be saved to disk/db)

**Scarping Spider** will consider the results of Crawling Spider and start scraping actual product data like product id, name, price and available SKU list.

Any failures due to network / rendering issues will be captured in a separate failed URL list and will be retried once the Scraping Spider is finished it’s job.

And Scrapy comes with variety of settings like logging, performance, export feeds etc. as shown below



**Splash**

Splash is a javascript rendering service. It’s a lightweight web browser with an HTTP API. Some of Splash features:

* process multiple webpages in parallel
* get HTML results and/or take screenshots;
* turn OFF images or use Adblock Plus rules to make rendering faster
* execute custom JavaScript in page context
* write Lua browsing scripts
* get detailed rendering info in HAR format.

**Aquarium:**

Aquarium helps us to have multiple Splash instances with load balancing(HAProxy) to increase the rendering speed by handling multiple requests parallelly and we can run it as a Docker container.

**Scrapyd**

Scrapyd is an application for deploying and running Scrapy spiders. It enables you to deploy (upload) your projects and control their spiders using a JSON API.

It can manage multiple projects and each project can have multiple versions uploaded, but only the latest one will be used for launching new spiders.

It is an application (typically run as a daemon) that listens to requests for spiders to run and spawns a process for each one

Scrapyd also runs multiple processes in parallel, allocating them in a fixed number of slots given by the [max\_proc](https://scrapyd.readthedocs.io/en/latest/config.html#max-proc) and [max\_proc\_per\_cpu](https://scrapyd.readthedocs.io/en/latest/config.html#max-proc-per-cpu) options, starting as many processes as possible to handle the load.

Scrapyd comes with a minimal web interface (for monitoring running processes and accessing logs) which can be accessed at <http://localhost:6800/>

**Scrapy-cloud:**

‘Scrapy-cloud’ is the cloud based solution for deploying, scheduling and managing the spiders we have written and is a paid service.