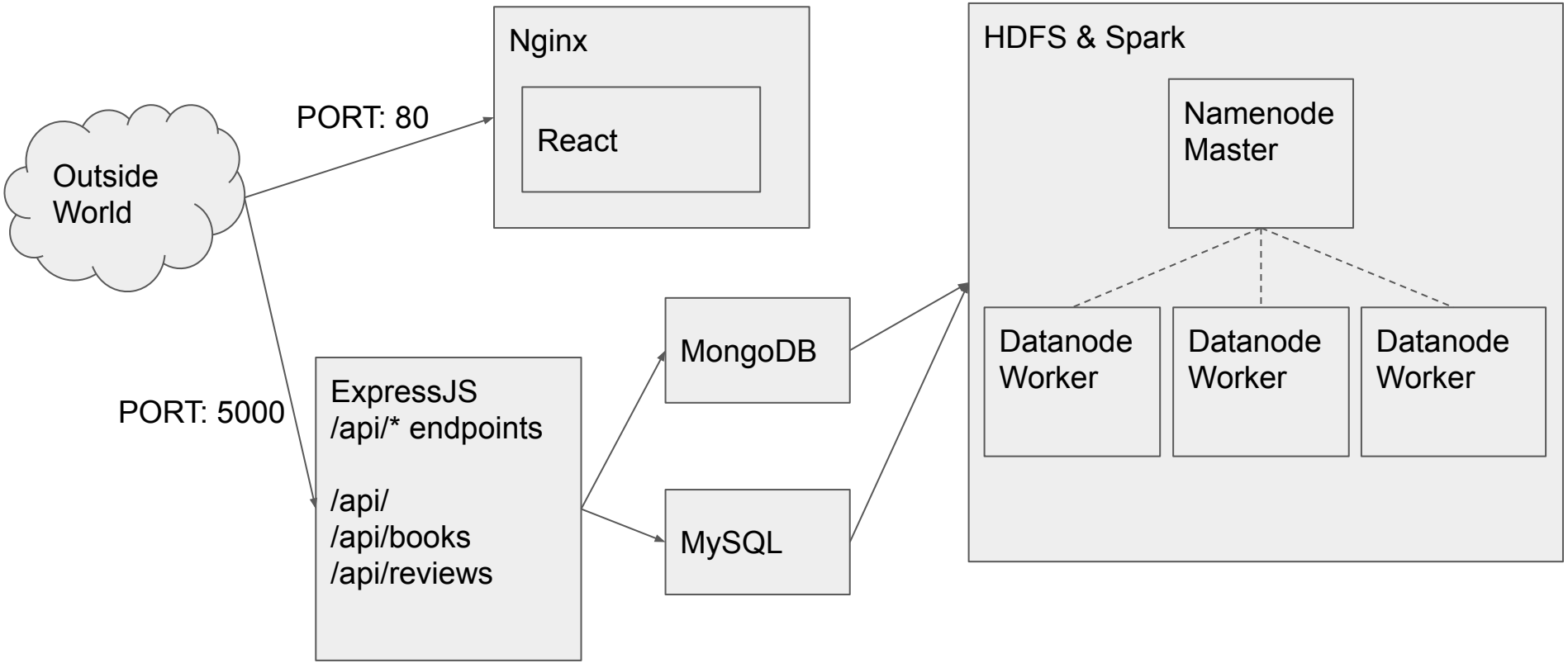
Betterread

System Overview



Website Overview

Home

BetterRead

Add Book

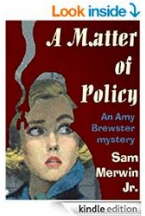
Search by title or author

Refine your search APPLY

Genre

- Art
- Biography
- Business
- Chick Lit
- Children's
- Christian
- Classics
- Comics
- Contemporary
- Cookbooks
- Crime
- Ebooks
- Fantasy
- Fiction
- Gay and Lesbian
- Graphic Novels
- Historical Fiction
- History
- Horror
- Humor and Comedy
- Manga
- Memoir
- Music
- Mystery
- Nonfiction

Showing 1 - 10 of 115 results found



A Matter of Policy [An Amy Brewster Mystery]

by Sam Merwin Jr.

overall ★★★★★

Reviews 8

Categories

Books | Literature & Fiction | Mystery, Thriller & Suspense | Thrillers & Suspense | Suspense | Kindle Store | Kindle eBooks

Saboteur (Star Wars: Darth Maul #1)

by James Luceno

overall ★★★★★

Reviews 5

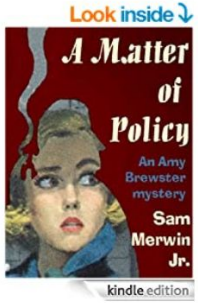
Categories

Books | Comics & Graphic Novels | Science Fiction & Fantasy | Science Fiction | Teen & Young Adult

Review

BetterRead

Add Book



A Matter of Policy [An Amy Brewster Mystery] Title placeholder

by Sam Merwin Jr. Author placeholder

Ratings ★★★★★

Reviews 8

Categories

Books | Literature & Fiction | Mystery, Thriller & Suspense | Thrillers & Suspense | Suspense | Kindle Store | Kindle eBooks

ADD REVIEW

Showing 8 of 8 results found

A

Avidreader

Summary

"Nice vintage story"

rated

Review

★★★★★

"I enjoy vintage books and movies so I enjoyed reading this book. The plot was unusual. Don't think killing someone in self-defense but leaving the scene and the body without notifying the police or hitting someone in the jaw to knock them out would wash today.Still it was a good read for me."

dot

Also bought

Buy after viewing

Add Review

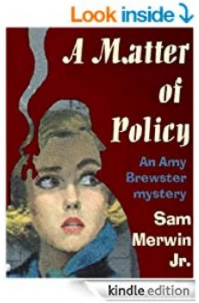
BetterRead

Add Book

What do you think about the book

A Matter of Policy [An Amy Brewster Mystery]

by Sam Merwin Jr.



AA

Anonymous

GENERATE

Ratings

Summary

Review

ADD REIVEW

Add Book


BetterRead
Add Book
Add a New Book
Author
Title
Description
ADD

Endpoints Overview

Features

1. Add new book
2. Add new review
3. Get reviews by asin
4. Search book from author and title
5. Filter books by genre and ratings
6. Random name generator for reviewer

Databases

Data Schema

1. Books Metadata (MongoDB)

- `id`: ObjectId, primary key
- `asin`: String
- `price`: Double
- `imUrl`: String
- `related`: Object
  - `also_bought`: String[]
  - `buy_afterviewing`: String[]
- `categories`: String[]
- `title`: String
- `author`: String

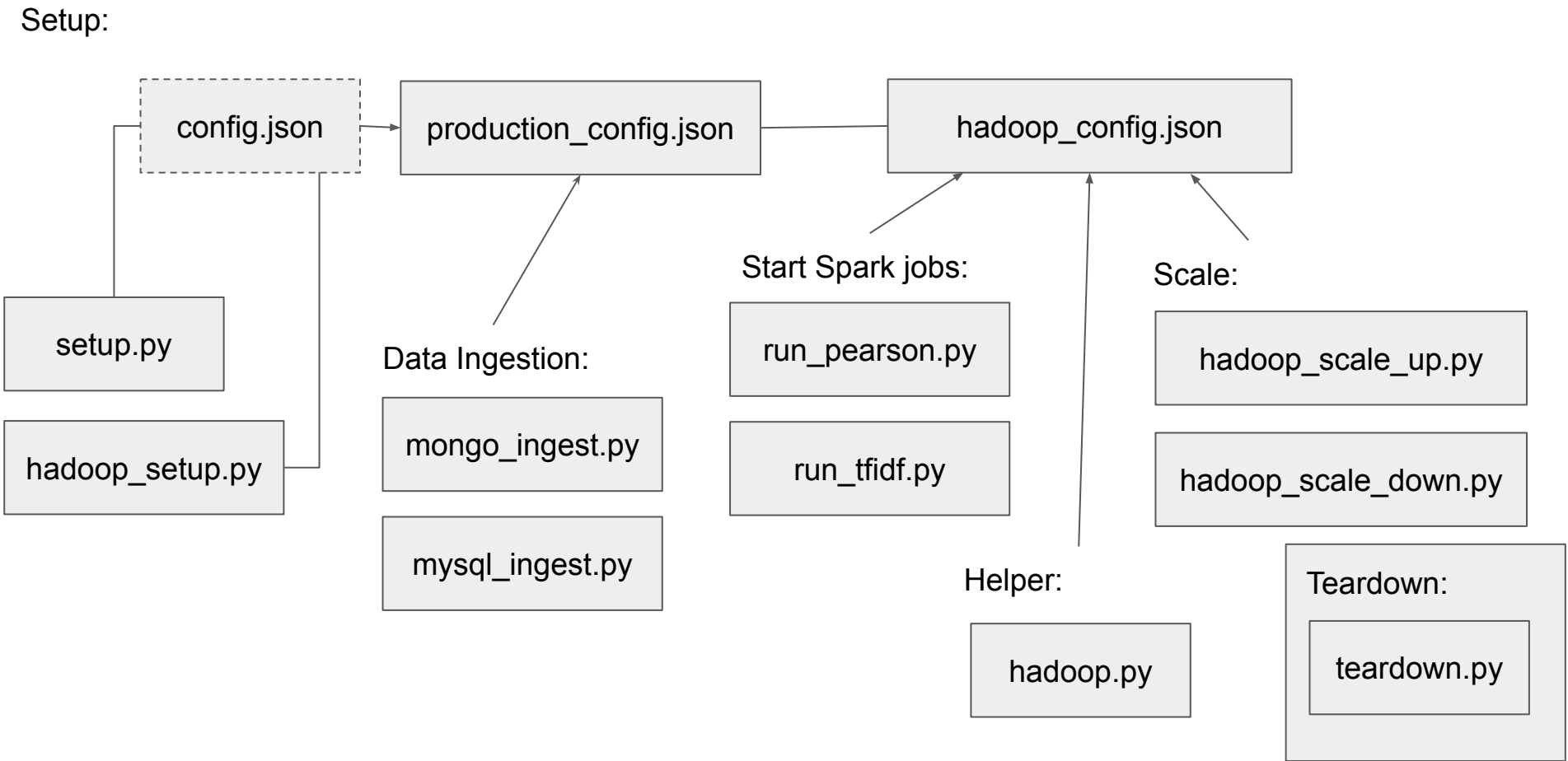
2. Logs (MongoDB)

- `timeStamp`: String
- `reqType`: String
- `resCode`: Number
- `url`: String

3. Review Data (MySQL)

- `id` INT(11) NOT NULL AUTO_INCREMENT
- `asin` VARCHAR(255) NOT NULL,
- `helpful` VARCHAR(255) NOT NULL,
- `overall` INT(11) NOT NULL,
- `reviewText` TEXT NOT NULL,
- `reviewTime` VARCHAR(255) NOT NULL,
- `reviewerID` VARCHAR(255) NOT NULL,
- `reviewerName` VARCHAR(255) NOT NULL,
- `summary` VARCHAR(255) NOT NULL,
- `unixReviewTime` INT(11) NOT NULL,PRIMARY KEY (`id`));

Automation


Setup:
config.json
production_config.json
hadoop_config.json
setup.py
hadoop_setup.py
Data Ingestion:
mongo_ingest.py
mysql_ingest.py
Start Spark jobs:
run_pearson.py
run_tfidf.py
Scale:
hadoop_scale_up.py
hadoop_scale_down.py
Helper:
hadoop.py
Teardown:
teardown.py

Spark jobs

Tfidf

1. Import MYSQL table containing the reviewText into hadoop
2. tokenize the data into a list of words.
3. calculate the term frequency(tf) values for each word storing them as vectors
4. calculate their inverse document frequency(idf), treating each review as 1 document
5. get the actual word from their index in the vectors
6. return it as a csv file.

Pearson Correlation

1. Get the reviewText from MySQL database
2. compute the average review length for each book
3. get the price of book from MongoDB books metadata
4. combine both the values into one dataframe
5. execute the calculation for Pearson Correlation according to this formula:

$$r = \frac{n\sum xy - \sum x \sum y}{\sqrt{(n\sum x^2 - (\sum x)^2)(n\sum y^2 - (\sum y)^2)}}$$

Improvements made

1. Title, author crawler (from Goodreads)
2. Reduced initial query time from ~45 seconds to < 3 seconds, by grouping queries & creating index
3. Extra feature: Random name generator

Future works

1. Scale Spark workers together when hadoop is scaled
2. Upload results of Spark jobs to cloud for access
3. Better way of organizing data for easier access
   (e.g. collections of books)
4. Prefetch queries
5. Better scripts organization