

Documentation for 50.043 Project

Group 7:

Kwa Li Ying (1003833)

Heng Jing Han (1003590)

Koh Xian Ming (1003448)

Lim Yang Zhi (1003458)

Tiang Pei Yuan (1003323)

Github link:

<https://github.com/liying-kwa/SUTD-50.043-Database-and-Big-Data-Systems-Project>

Project page:

<https://istd50043.github.io/project>

Table content

[Prerequisite](#)

[Steps to create the system](#)

[Automation Scripts](#)

[Analytics task](#)

[Web Application](#)

[Extra features and efforts](#)

Prerequisite

Ensure you have your aws account access key and secret key (Not awseducate please)

<https://docs.aws.amazon.com/powershell/latest/userguide/pstools-appendix-sign-up.html>

Use Singapore region (ap-southeast-1)

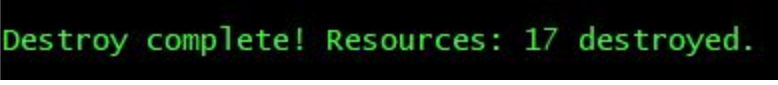
Ensure that you don't have kp.pem or kp_main.pem name as key-pair in aws

Ensure that you have at most 3 existing vpc, as the code will create 2 vpc

Steps to create the system

Step 1	Description: Ensure the prerequisites above are met
Step 2	Description: Create a new ubuntu ec2 instance in ap-southeast-1, ssh into it
Step 3	Description: Download the starter script and run it. It will do all necessary installations and downloads. You should see the folder "SUTD-50.043-Database-and-Big-Data-Systems-Project" and this output All installations and downloads done!
	Command: wget -c https://jinghanbucket1997.s3-ap-southeast-1.amazonaws.com/script.sh -O script.sh bash ./script.sh
Step 4	Description: Change directory into the folder
	Command: cd SUTD-50.043-Database-and-Big-Data-Systems-Project
Step 5	Description: Create the main structure (Databases and Production system without the analytics portion) When prompted, input in your access key and secret key
	Command: bash ./create_main.sh
Step 6	Description: Let the code run finish and you should see a similar screen as shown.

	<pre> Apply complete! Resources: 17 added, 0 changed, 0 destroyed. Outputs: MongoDBLogs_public_dns = ec2-13-212-123-51.ap-southeast-1.compute.amazonaws.com MongoDBMetadata_public_dns = ec2-13-212-5-238.ap-southeast-1.compute.amazonaws.com WebApp_elastic_ip = ec2-3-1-115-28.ap-southeast-1.compute.amazonaws.com mysql_public_dns = ec2-54-151-164-74.ap-southeast-1.compute.amazonaws.com vpc_id = vpc-03c7b3cc79eb8ba60 </pre> <p>Link to the Web App is the WebApp_elastic_ip as shown. (It takes about 10mins to set up.)</p>
Step 7	<p>Description:</p> <p>Create the analytics structure and declare the number of data nodes you want. It's roughly 15mins for 3 data nodes.</p> <p>The system already have your access key and secret key, there is no need to key it in again</p> <p>Note: This step takes quite a long time and there may be a prompt sometimes, it's uncontrollable. If there are any errors seen, just run step 11 and step 8 again.</p> <p>Command:</p> <pre>bash ./create_analytics.sh</pre>
Step 8	<p>Description:</p> <p>Do data ingestion. You should see the output below when it's done.</p> <pre>Data Ingestion done</pre> <p>Command:</p> <pre>bash ./data_ingestion.sh</pre>
Step 9	<p>Description:</p> <p>To run analytics task and see the output. (Take note you should do step 9 before step 10)</p> <p>Anytime you want to see the output again, re-run step 9 followed by step 10</p> <p>Command:</p> <pre>bash ./analytics_task.sh</pre>
Step 10	<p>Description:</p> <p>To destroy the analytics structure, run the destroy code.</p> <p>You may re-create it by running the creation code (Step 7) again after destroying if you want a different number of data nodes</p> <p>Command:</p> <pre>bash ./destroy_analytics.sh</pre>
Step 11	<p>Description:</p> <p>After destroying the analytics structure, if you want to tear down the whole system.</p> <p>(Note: Please destroy the analytics structure first before destroying the main structure)</p>

	
	Command: bash ./destroy_main.sh
Step 12	Description: Check your AWS console, at this point you should only have your client node and no other EC2 instance. You are done!

Automation Scripts

<https://www.terraform.io/>

We use terraform, an open-source infrastructure as code software tool, to automate the creation of the aws resources. To introduce isolation into the system, we separate the automation code for the production system and analytics system.

It will create the networks first, such as VPC, subnets as well as security groups and routing access. The secret key is stored in the directory and its corresponding public key is uploaded to aws through terraform. (kp_main.pem for production system and kp.pem for analytics system) Finally, it will create the different EC2 instances and run the individual bash scripts for each of them.

EC2 instances: Web App, mysql database, mongodb for metadata, mongodb for user logs, name node and data nodes.

We have a variable script for storing common variables such as EC2 instance ami and type. This enables any user to easily change the variable from one single file instead of having to change them in every instance one by one. (Filename is variables.tf)

We also have an output script which will determine what are the outputs to be displayed. The user is interested in some information after the creation of the whole system. Perhaps the output that users are most interested in is the link for the web app which is shown as the elastic IP. (Filename is output.tf)

When grading is done, you will want to destroy all the aws resources to save cost. We wrote automation scripts for destroying the resources for your convenience. If not, you may need to terminate instances one by one in the AWS console, which is tedious.

Below is a list of problems faced and the solution we developed

	Problem	Solution
1	Database creation is slow and we have no idea when it's done. Query from a database that is still creating in progress will result in an error.	Use google firebase to store variables indicating when the database is fully created. (By busy waiting) When it is created, the database endpoints can be extracted from firebase.
2	With the creation of many aws resources, the terraform code eventually looks very huge with few thousands of code. This makes code development hard and finding sections tedious.	We study terraform project structure and use it to our advantage by separating the code into different files. (Terraform command will read all terraform type files in the current directory!)

For security reasons, the access key and secret key are only stored in the directory and nowhere else.

Analytics task

Process

First, we use terraform to spin up $(n+1)$ number of EC2 instances (where n is user input) for 1 name node and n data nodes. A VPC and subnet is created and these instances will be allocated to the same security group with this VPC and subnet. The network for this security group is configured such that the inbound rules allow for all ports from all IP addresses to be accessed within the same security group (172.2.0.0/24) and only the SSH port (i.e. 22) is exposed to the public. This is to allow for the client machine to SSH into the name node and data nodes when setting up the analytics system and to allow the name node and data nodes to communicate with each other for HDFS and Spark, because these processes require communication through different ports.

Before the libraries are set up, we made some additional configurations. The private IP addresses of all nodes are written into the `/etc/hosts` file of all nodes so that the nodes will be able to address each other by their given hostname (e.g. `com.analytics.datanode1`) instead of their IP address. A new user called `hadoop` is created on all nodes for security purposes, and every installation, configuration and startup will be executed via this `hadoop` user. The `hadoop`

user of the name node is configured to be able to ssh into the hadoop user of all the data nodes without using a password. Java compiler and runtime is installed on all nodes.

Next, Hadoop is installed and HDFS is set up. The Name Node downloads the hadoop library using wget and edits the hadoop configuration files. It is then zipped and distributed to the Data Nodes via SCP. In all nodes, the library is moved to the appropriate location and the directory to store HDFS files is created. The Name Node is formatted and the HDFS cluster is started up using start-dfs.sh and start-yarn.sh.

Following this, the Name Node downloads the Spark library and configures the variables required for Spark to interact with Java, Python, and Hadoop. The configured folder is then zipped and distributed to the data nodes via SCP. After distribution, we unzip the files, then run start-all.sh to spin up the spark cluster.

For the ingestion, we first install Sqoop on the name node. Then, the environment is configured and drivers are installed. After which, we run a Python script to communicate with a Firebase database to check if the MySQL and MongoDB databases are up. If the databases are not up yet, the python script busy waits for the databases to be up. When the databases are up we then proceed with the ingestion of data from mySQL into /user/hadoop/KRTable (using Sqoop) and MongoDB into /user/hadoop/kindle_metadata (using pyspark). The script also stores their endpoints in a local text file.

Lastly, we run the Python scripts to calculate Pearson correlation and TF-IDF. We utilise pyspark to read data from our HDFS and do calculations. Pearson correlation coefficient is calculated between the prices and the average length of the reviews. For TF-IDF, for brevity, we limit the vocabulary size to the top 20 most common words.

```
Final Table
+-----+
| pearson_coefficient|
+-----+
|0.025058674923425922|
+-----+
```

```
id tfidf
+-----+
+-----+
1 | {'the': 0.7125142491256569, 'and': 0.4143946221242928, 'to': 0.25136109817667285, 'a': 0.2482229328244232, 'i': 0.5911805132020355,
  | 'this': 0.3134078786259924, '': 2.0136253335849914, 'in': 1.1841624074296702, 'it': 0.5880089552508699, 'was': 1.448228131663754, 'for'
  | : 0.7121458226048759, 'but': 0.892073026240443}|
2 | {'the': 0.4275085494753941, 'and': 0.6215919331864392, 'to': 0.5027221963533457, 'a': 0.7446687984732696, 'i': 0.5911805132020355,
  | 'of': 0.8481466264175123, 'this': 0.3134078786259924, '': 3.020438000377487, 'it': 0.5880089552508699, 'was': 1.448228131663754, 'that':
  | 0.7540632886447778, 'for': 0.7121458226048759}|
3 | {'the': 0.14250284982513137, 'i': 0.29559025660101773, 'of': 0.42407331320875613, 'this': 0.3134078786259924, '': 1.006812666792495
  | 7}
4 | {'the': 0.14250284982513137, 'of': 0.42407331320875613, 'this': 0.3134078786259924, '': 1.0068126667924957, 'you': 2.23972638888315
  | 5}
5 | {'the': 0.14250284982513137, 'a': 0.7446687984732696, 'of': 0.42407331320875613, 'is': 0.5499162154997934, 'it': 1.1760179105017399
  | }
6 | {'the': 0.14250284982513137, 'and': 0.2071973110621464, 'i': 0.29559025660101773, 'this': 0.3134078786259924, 'in': 0.5920812037148
  | 351, 'it': 1.1760179105017399, 'was': 0.724114065831877}|
7 | {'the': 0.5700113993005255, 'and': 0.2071973110621464, 'to': 0.7540832945300185, 'a': 0.2482229328244232, 'i': 0.29559025660101773,
  | 'of': 0.42407331320875613, 'this': 0.6268157572519848, 'is': 0.5499162154997934, 'in': 0.5920812037148351, 'but': 0.892073026240443}|
8 | {'a': 0.2482229328244232, 'of': 0.42407331320875613, 'is': 0.5499162154997934, '': 1.0068126667924957, 'that': 0.7540632886447778}
  |
9 | {'the': 0.5700113993005255, 'to': 0.7540832945300185, 'a': 0.4964458656488464, 'this': 0.6268157572519848, 'is': 1.6497486464993802
  | , 'in': 1.1841624074296702, 'that': 0.7540632886447778, 'but': 0.892073026240443}|
10 | {'and': 0.2071973110621464, 'a': 0.2482229328244232, 'i': 0.5911805132020355, 'this': 0.3134078786259924, 'is': 0.5499162154997934,
    | 'in': 0.5920812037148351, 'it': 0.5880089552508699, 'book': 0.7520319052445371}|
+-----+
+-----+
only showing top 10 rows
```

Areas to Improve

The execution duration could be reduced if some processes were run in parallel, e.g. the wget commands for downloading the hadoop and spark libraries.

A lot of automation and bash scripts can be removed if we use a prepared image with Hadoop and Spark installed.

Web Application

The Web Application is developed in accordance with the project requirements.

Adding reviews

To add a review to the book, users have to be logged in. Users have to navigate to the book they want to leave a review on, scroll to the bottom of the web page to add a new review.

Leave a comment

☆☆☆☆☆ (3.72/5, 5 votes) You rated this: 5 star

Title:

Good book!

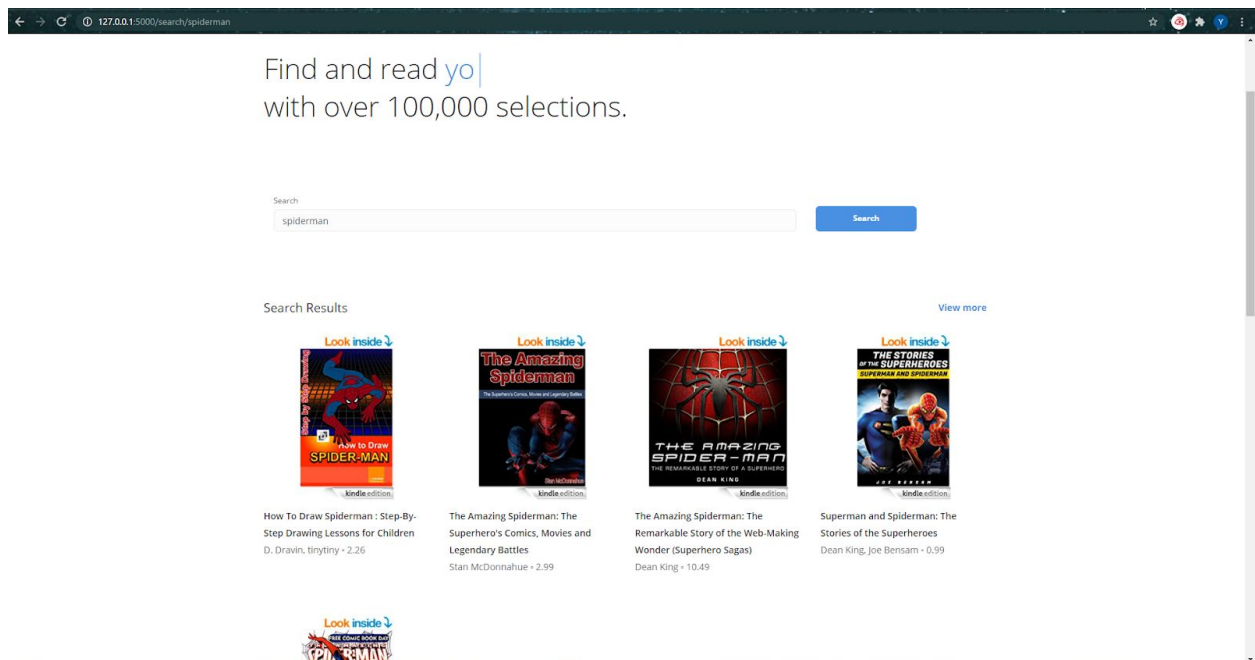
Comment:

This book is one of the best book I've ever read

Submit Comment

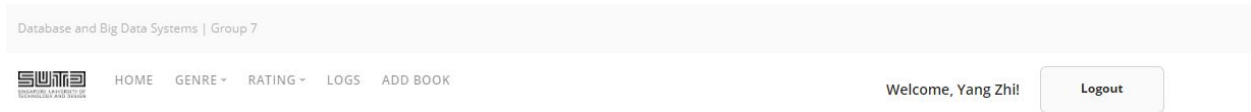
Search function

The web application contains a search function for users to search for existing books by author and title. Users can also search for books by their ASIN.

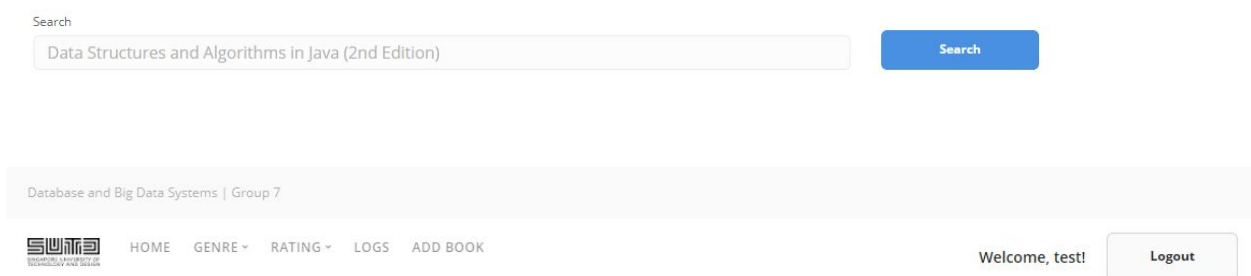


Login and web logging

Interface for users to login and logout. Logs for user activity are stored in mongodb. Users without an account can sign up for one. Users would be able to see their own web logs.



Find and read
with over 100,000 selections.



Logs


User	Datetime	Action	Response	Content
test@test.com	06/12/2020, 10:13:49	remove_book	401	B00E5GSS5E
test@test.com	06/12/2020, 10:13:26	view	200	Big Data and Hadoop
test@test.com	06/12/2020, 10:13:17	view	404	A1003448

Adding a new book

Admin would be able to add a new book to the database

(Use the following credentials to log in as admin, email: admin@admin.com, password: admin)

Database and Big Data Systems | Group 7

 HOME GENRE ▾ RATING ▾ LOGS ADD BOOK

Welcome, xm! Logout

Add new book


ASIN	Title	Author	Price	Genre
<input type="text" value="A0262033844"/>	<input type="text" value="Introduction to Algorithms"/>	<input type="text" value="Thomas H. Cormen"/>	<input type="text" value="85.15"/>	<input type="text" value="academic"/>





Description

The latest edition of the essential text and professional reference, with substantial new material on such topics as vEB trees, multithreaded algorithms, dynamic programming, and edge-based flow. Some books on algorithms are rigorous but incomplete; others cover masses of material but lack rigor. Introduction to Algorithms uniquely combines rigor and comprehensiveness. The book covers a broad range of algorithms in depth, yet makes their design and analysis accessible to all levels of readers. Each chapter is relatively self-contained and can be used as a unit of study. The algorithms are described in English and in a pseudocode

Book Cover Image URL

Add Book

 Get Started help@sutd.edu.sg

[Privacy Policy](#) [Legal](#)

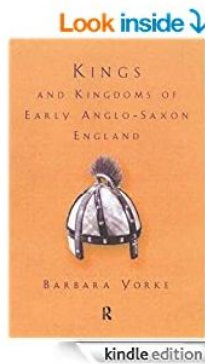
Extra features and efforts

We did web scraping to get the missing and incomplete metadata data.

Web scraping code can be found in github.

https://github.com/liying-kwa/SUTD-50.043-Database-and-Big-Data-Systems-Project/blob/iac2/web_scrape_metadata.py

Suggested books from related genres



Kings and Kingdoms of Early Anglo-Saxon England

~~\$549.99~~ \$10.22

For the first time we have in one volume histories of the six major kingdoms, with discussions of the sources and of all the major problems of reconstructing dynasties and of dating assembled together in one place.--History

^ Category

^ Books from the same genre

[The Holy Bible \(Illustrated, King James Version \[KJV\]\)](#)

[George Washington, Volumes I-II, Complete](#)

[Save Me the Waltz: A Novel](#)

[I Promise I Won't Kill You: A Hitchhiking Adventure](#)


[Revelations of Divine Love](#)

^ Shipping Info

Sleek and intuitive UI

Much effort and considerations have been put in place when designing the website to make it user friendly

Database and Big Data Systems | Group 7

HOMEGENRE *RATING *LOGSADD BOOK

Welcome, xm!Logout

Find and read **thriller** with over 100,000 selections.


Search

Data Structures and Algorithms in Java (2nd Edition)

Search


Featured Books

Look inside ↴



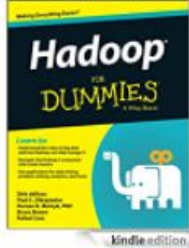
Big Data and Hadoop
WAGmob • 18.35

Look inside ↴



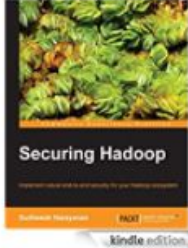
Microsoft SQL Server 2012 with Hadoop
Debarchan Sarkar •

Look inside ↴



Hadoop For Dummies
Dirk DeRoos • 0.99

Look inside ↴



Securing Hadoop
Sudheesh Narayanan • 12.64

Popular Books

