Cluster and Cloud Computing

Sarathi T S	Harish Kannan	Ajith SP	Eby Thomas	Dash Park
1410995	1534410	1539182	1536916	1171850

May 22, 2024

Assignment 2 Report

Group 72

GitHub and Video Link

- GitHub: `https://github.com/Harish-2502/CCC-Team-72`
- Video link: `https://youtu.be/o9gjHp0zj1E`

Team Member Emails

- sthirumalais@student.unimelb.edu.au
- harishk@student.unimelb.edu.au
- aselvarajpra@student.unimelb.edu.au
- thomaset@student.unimelb.edu.au
- dash.park@student.unimelb.edu.au

Master of Data Science
University of Melbourne

Contents

1 Introduction **3**

2 Architecture **3**

3 System Design **3**

3.1 ABS Region Population Data (SUDO) 3

3.2 Rental Affordability Index (SUDO) 4

3.3 Melbourne Building Permits (DataVic) 5

3.4 Melbourne Liveability Indicators (DataVic) 6

3.5 Use of the Melbourne Research Cloud, Kubernetes, Fission, and ElasticSearch 7

3.6 ElasticSearch Index Design 8

3.7 Kibana Index View 13

3.8 Jenkins Working and GitHub Integration 15

4 System Functionality **20**

4.1 Public Keys 20

4.2 Fission Scripts 21

4.3 Specs 24

4.4 Fission Function - Implementation 25

4.5 Fission Function - Validation and Testing 27

4.6 .gitignore 28

5 Error Handling **28**

5.1 Single Point of Failure 29

5.2 GitHub and Jenkins 30

5.3 Scalability 30

6 Data Analysis **31**

6.1 Scenario 1: Estimated Regional Population & Building Permits 31

6.2 Scenario 2: Liveability & Rental Affordability Index 38

6.3 Scenario 3: Rental Affordability Index & Population 41

6.4 Scenario 4: Analysis of Liveability Percentage & Population in Melbourne 43

7 GUI Implementation **44**

1 Introduction

As part of the University of Melbourne's Master of Data Science course, the subject Cluster and Cloud Computing offers students the ability to learn and explore the working systems of cloud computing. As one of its assignments, students are asked to create a simplified version of an end-to-end cloud computing system for data analysis. The following report contains information regarding the system and the analysis made by students in group 72.

2 Architecture

The cloud architecture was created using a combination of Kubernetes, Fission, and ElasticSearch, with resources provided by the Melbourne Research Cloud (MRC).

As a rough overview, the roles of each component are as follows. The MRC provided the underlying cloud infrastructure for all groups to use. This allowed the groups to deploy Kubernetes clusters, Fission functions, and ElasticSearch instances. Kubernetes was used to enable the deployment of both Fission functions and ElasticSearch indexes. Fission functions allowed the project to have an easy method of streaming data, both via an API and older JSON files from different sources, into the ElasticSearch database. Finally, ElasticSearch was used as our NoSQL type database, allowing the storage of both structured and unstructured data in a format that was ideal for data analysis.

Furthermore, to assist with the code's version control among all group members, GitHub was used. GitHub allowed the group members to not only collaborate on code in real-time and manage code version control among members but also to integrate with other services such as Jenkins for the project. Additionally, to assist with error handling, Fission functions for unit tests were created to handle the Fission functions.

3 System Design

Firstly, the datasets chosen should be shown so that the system design can be discussed in detail. In total, four datasets were chosen. Two datasets were chosen from the Spatial Urban Data Observatory (SUDO), and two datasets were chosen from DataVic.

3.1 ABS Region Population Data (SUDO)

This dataset contains estimates of the resident population and estimates of the components of population change as of 30 June for the years 2001-2021. The data is aggregated to the 2021 Australian Statistical Geography Standard (ASGS) Local Government Areas (LGA).

This data is sourced from the Australian Bureau of Statistics (Catalogue Number: 3218.0).

For more information, please visit the Regional Population Methodology.

While a multitude of columns were provided by the dataset, only the following was used:

- **erp_YYYY**

These columns held the information regarding the final population for greater Melbourne per year, where YYYY is the year between 2001 to 2021, inclusive of both ends.

The trend for the population is as follows:

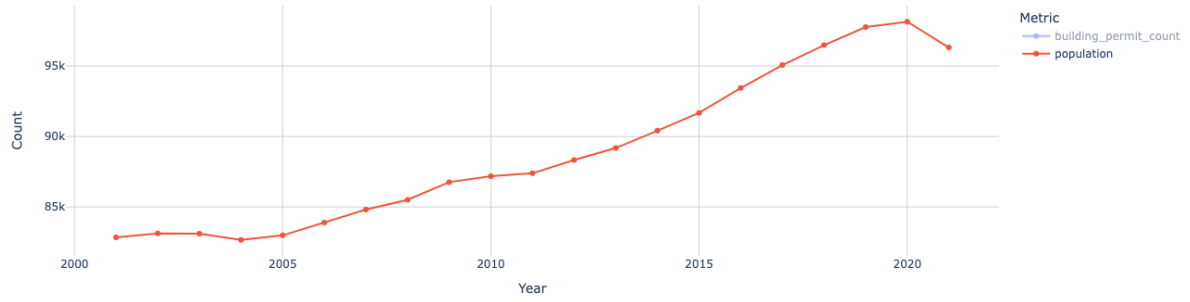


Figure 1: Building Permits and Population Over the Years

3.2 Rental Affordability Index (SUDO)

This dataset presents the Rental Affordability Index (RAI) for all dwellings. The data uses a single median income value for all of Australia (enabling comparisons across regions) and spans the quarters Q1 2011 to Q2 2021. The RAI covers all states with available data; the Northern Territory does not form part of this dataset.

It is generally accepted that if housing costs exceed 30% of a low-income household's gross income, the household is experiencing housing stress (30/40 rule). That is, housing is unaffordable and housing costs consume a disproportionately high amount of household income. The RAI uses the 30 percent of income rule. Rental affordability is calculated using the following equation, where 'qualifying income' refers to the household income required to pay rent where rent is equal to 30% of income:

$$\text{RAI} = \left(\frac{\text{Median Income}}{\text{Qualifying Income}} \right) \times 100$$

In the RAI, households who are paying 30% of income on rent have a score of 100, indicating that these households are at the critical threshold for housing stress. A score of 100 or less indicates that households would pay more than 30% of income to access a rental dwelling, meaning they are at risk of experiencing housing stress.

For more information on the Rental Affordability Index, please refer to SGS Economics and Planning.

For the sake of the analysis, we have combined all the quarterly information per year into RAI for that year by taking the average of four quarters. While this removes more detailed information about this dataset, for this dataset to be compared against the other three datasets, it needed to be simplified.

The trend for this dataset appears as follows:

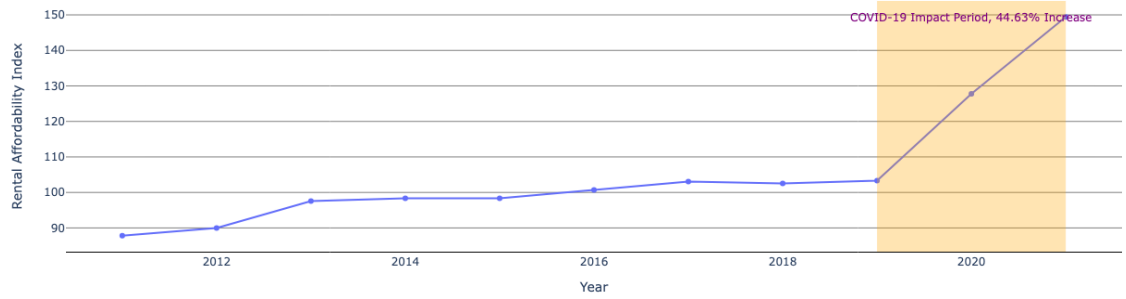


Figure 2: Rental Affordability Index Over the Years

3.3 Melbourne Building Permits (DataVic)

This public register of building permits lists all building permits for building works issued by Melbourne Certification Group, in and outside of the City of Melbourne, and also any building works issued by Private Building Surveyors within the City of Melbourne.

Similar to the other datasets, some cleaning was necessary for the dataset to be used as part of the analysis. The following cleanup was conducted:

- Any rows where the issue date was later than either the commencement or completion date were removed, as a permit is required to be issued before any work is conducted.
- Any building permits where the issued date was in the future were removed.

Following the cleanup, the dataset is finalized as follows:

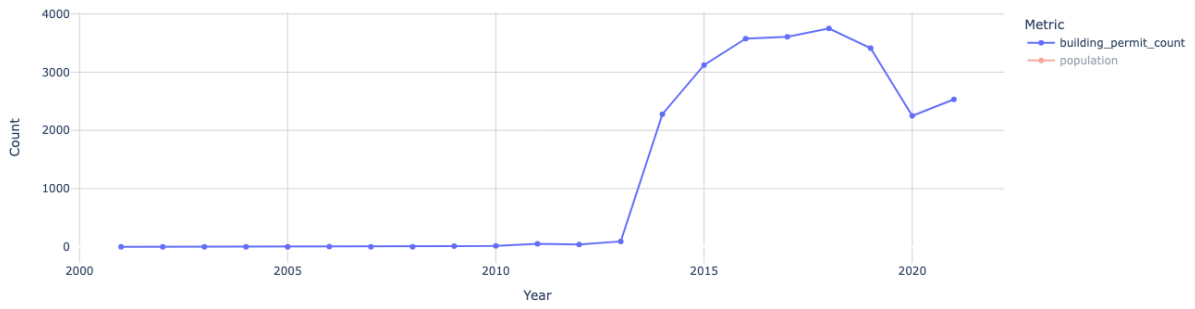


Figure 3: Building Permits and Population Over the Years

3.4 Melbourne Liveability Indicators (DataVic)

The following paragraph is a summary of the dataset provided by DataVic.

Indicators help us to assess our municipality's performance, measure progress, and compare with other cities. The dataset provided contains information for both liveability indicators and social indicators between the years 2006 and 2019. However, the dataset has gaps in the data, and not all of the provided data is needed. Thus, the following cleaning was done to the dataset:

- Only liveability scores were kept - social indicators were not used for this analysis, as that was not within the scope of the analysis conducted with this dataset.
- Outliers were removed. For the next step in cleaning the dataset for usage, some large numbers needed to be removed. It was observed that the liveability scoring calculated for the GDP was much too large, and the values for numerator and denominator columns were swapped around. Thus, these rows were removed from the dataset.
- Some data which were split over the years (both quarterly and financial years) were combined into a single year. This was possible as the dataset offered separate columns for numerator and denominator for the percentage calculation. By adding all the numerators and denominators respectively for each year, a new percentage was able to be calculated for the year.

Following this cleanup, the dataset over the years is as follows:

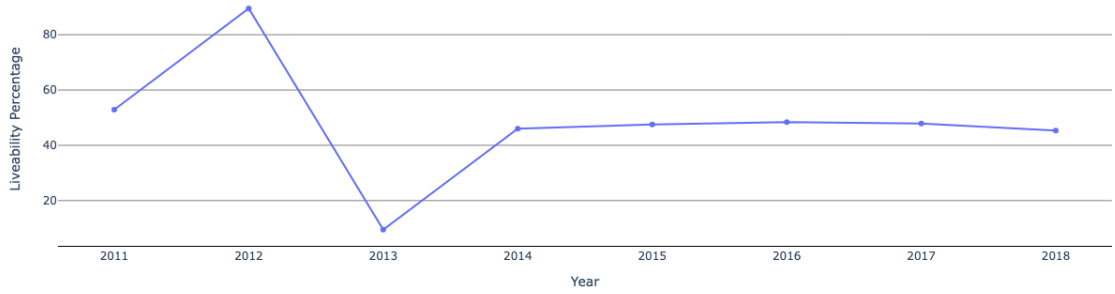


Figure 4: Liveability Score over the Years

3.5 Use of the Melbourne Research Cloud, Kubernetes, Fission, and ElasticSearch

As mentioned previously, a combination of Kubernetes, Fission, and ElasticSearch was used as part of the cloud computing solution for the data analytics project. The Melbourne Research Cloud (MRC) was utilized as the cloud computing solution. As a cloud computing service, it brings all the usual benefits. MRC provides already existing powerful computing resources for any authorized individual to run Kubernetes clusters, serverless functions with Fission, and data indexing with ElasticSearch. Additionally, with the use of public keys, a collaborative environment can easily be created and managed. As a group of five members, ease of accessibility by all group members is a consideration that must be accounted for. While MRC is also scalable and cost-effective, as students are not required to buy and manage their own hardware, for the scope of this project, 11 virtual CPUs and 700GB of storage were given to every group. Thus, these benefits of a usual cloud provider are not applicable.

However, MRC holds its own drawbacks alongside its advantages. First comes complexity - someone with little to no knowledge of the cloud infrastructure would have trouble setting up the appropriate environment without proper help from someone with experience. With the resources provided to each group being limited, it may also cause additional areas of concern if a group were to use most of the resources given, as this will slow down the entire pipeline. Finally, it comes with the dependencies of using a cloud service. While the benefits of using a service provided by a third party are worth considering, any downtime or complications raised on the cloud service's end will impact the project. In this particular project, an issue was raised from the MRC, where problems with the hypervisors caused unresolvable issues from the students' end.

Similar benefits and issues arise with Kubernetes. While it offers scalability, high availability, and good resource management, it comes at the cost of the complexity of setting up and managing the service. Further, security starts becoming an issue if the setup for Kubernetes is not managed properly.

For Fission, the biggest benefit is the event-driven architecture and the serverless model. Event-driven architecture allows ease of use with big data, as any updates to the dataset can be easily set up to be

retrieved via an API, or a cron job can be used to schedule more manual checks with equal ease. The serverless aspect of Fission allows the students to not worry about the inner workings of the cloud infrastructure and reduces operational overhead. Fission functions also scale automatically, adjusting based on the incoming data for processing. It works with Kubernetes to handle the scaling of the pods.

Finally, we have ElasticSearch. Due to its capabilities to handle real-time big data, using indexes to search real-time data, and a flexible query language system, it is an appropriate choice for the project's database. Additionally, its included ecosystem, such as Kibana, allows for easy visualization and logging for easier management of the database. However, ElasticSearch has its own disadvantages. This becomes apparent for big data which are only large in volume but nothing else. Due to the NoSQL type structuring, dealing with multiple tables that are related to each other becomes much more difficult, as the structuring needs to be modified to accommodate any typical 'joins' that one would use for a relational database such as PostgreSQL.

3.6 ElasticSearch Index Design

Schema and Index Creation

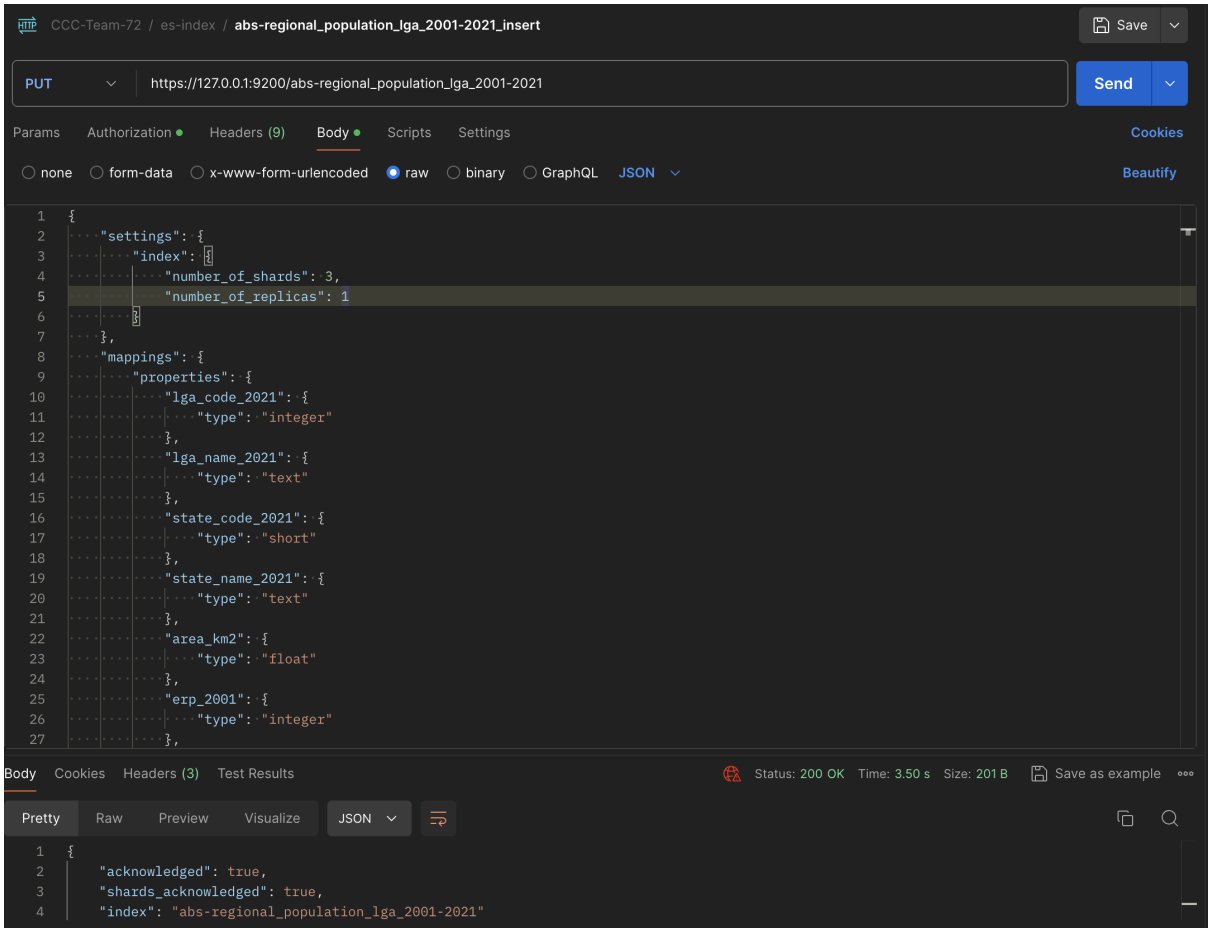


Figure 5: Create Index abs-regional_population_lga_2001-2021 in Elasticsearch

We used Postman for creating and deleting ElasticSearch indices mainly because it supports various HTTP methods (GET, POST, PUT, DELETE, PATCH, etc.) with a lot more user-friendly features. It also streamlines the process of creating, testing, and documenting APIs, enhancing and ensuring productivity and the reliability and quality of APIs.

The `PUT` request is used in HTTP to update or replace a resource at a specified URL. In our case, we used it to create an index in Elasticsearch. The URL in this case is:

`https://127.0.0.1:9200/abs-regional_population_lga_2001-2021`

Where `https://127.0.0.1:9200` specifies that Elasticsearch is running on the local system on port 9200, and `abs-regional_population_lga_2001-2021` is the name of the index we want to create. Basic Authorization is also added with the username and password in order to connect to Elasticsearch.

Then, we use the JSON body to create the index in Elasticsearch with specific settings and mappings.

In the settings part, we set `number_of_shards` as 3, which specifies that the index will be divided into three primary shards. Shards are considered the basic units of storage in Elasticsearch and allow the index to be distributed across multiple nodes for scalability and performance. `number_of_replicas` is set to 1, meaning each primary shard will have one replica, providing redundancy and high availability. If a node holding a primary shard fails, the replica shard can be promoted to primary.

In the mappings section, we define the schema of the documents that will be stored in the index. It specifies the suitable and appropriate data types for each field in the documents.

Index Deletion

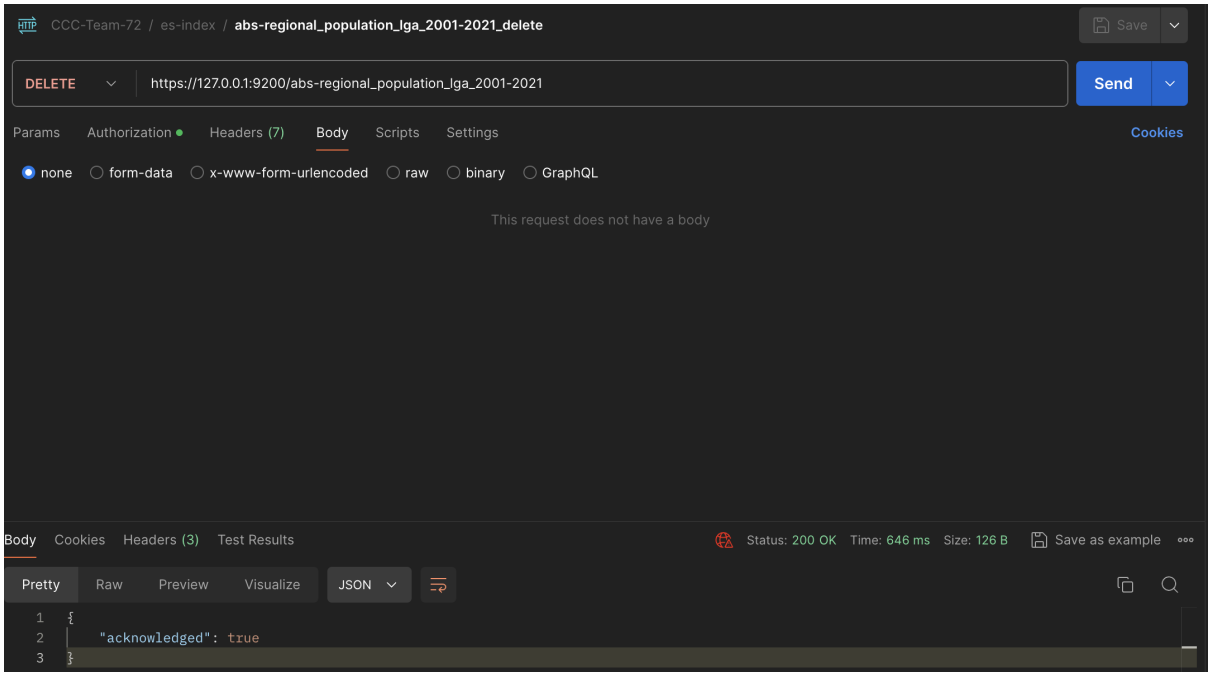


Figure 6: Delete Index abs-regional_population_lga_2001-2021 from Elasticsearch

The DELETE request is used in HTTP to delete or remove a resource at a specified URL. In our case, we used it to delete an index in Elasticsearch. The URL in this case is:

`https://127.0.0.1:9200/abs-regional_population_lga_2001-2021` , where

`https://127.0.0.1:9200` specifies that Elasticsearch is running locally on port 9200, and

`abs-regional_population_lga_2001-2021` is the name of the index we want to delete. Basic authorization is added with the username and password to connect to Elasticsearch. Unlike the PUT request above, it doesn't require a JSON body.

The process for creating and deleting an index is repeated while working on the rest of the indices (`building-permits`, `rental_affordability`, `liveability-index`) similar to

`abs-regional_population_lga_2001-2021`.

The mappings for `abs-regional_population_lga_2001-2021` are as follows:

```
"properties": {
    "lga_code_2021": { "type": "integer" },
    "lga_name_2021": { "type": "text" },
    "state_code_2021": { "type": "short" },
    "state_name_2021": { "type": "text" },
```

```
    "area_km2": { "type": "float" },
    "erp_2001": { "type": "integer" },
    "erp_2002": { "type": "integer" },
    "erp_2003": { "type": "integer" },
    "erp_2004": { "type": "integer" },
    "erp_2005": { "type": "integer" },
    "erp_2006": { "type": "integer" },
    "erp_2007": { "type": "integer" },
    "erp_2008": { "type": "integer" },
    "erp_2009": { "type": "integer" },
    "erp_2010": { "type": "integer" },
    "erp_2011": { "type": "integer" },
    "erp_2012": { "type": "integer" },
    "erp_2013": { "type": "integer" },
    "erp_2014": { "type": "integer" },
    "erp_2015": { "type": "integer" },
    "erp_2016": { "type": "integer" },
    "erp_2017": { "type": "integer" },
    "erp_2018": { "type": "integer" },
    "erp_2019": { "type": "integer" },
    "erp_2020": { "type": "integer" },
    "erp_2021": { "type": "integer" }
}
```

The settings for `building-permits` include an additional field:

```
"max_result_window": 500000
```

This sets the maximum number of search results that can be paginated through using the `from/size` parameters to 500000. By default, this value is set to 10,000, meaning we can only retrieve 10,000 documents.

The mappings for `building-permits` are as follows:

```
"properties": {
    "council_ref": { "type": "keyword" },
    "permit_number": { "type": "text" },
    "issue_date": { "type": "date" },
    "address": { "type": "text" },
    "estimated_cost_of_works": { "type": "integer" },
    "rbs_number": { "type": "text" },
    "commence_by_date": { "type": "date" },
```

```
    "completed_by_date": { "type": "date" },
    "permit_certificate_type": { "type": "text" }
}
```

The mappings for `rental_affordability` are as follows:

```
"properties": {
    "geography_name": { "type": "integer" },
    "rai_cityadjusted_total_2011_q2": { "type": "float" },
    "rai_cityadjusted_total_2011_q3": { "type": "float" },
    "rai_cityadjusted_total_2020_q1": { "type": "float" },
    "rai_cityadjusted_total_2011_q1": { "type": "float" },
    "rai_cityadjusted_total_2020_q3": { "type": "float" },
    "rai_cityadjusted_total_2019_q4": { "type": "float" },
    "rai_cityadjusted_total_2020_q2": { "type": "float" },
    "rai_cityadjusted_total_2011_q4": { "type": "float" },
    "rai_cityadjusted_total_2020_q4": { "type": "float" },
    "city": { "type": "text" },
    "unique_id": { "type": "integer" },
    "rai_cityadjusted_total_2017_q4": { "type": "float" },
    "rai_cityadjusted_total_2017_q3": { "type": "float" },
    "rai_cityadjusted_total_2017_q2": { "type": "float" },
    "rai_cityadjusted_total_2018_q1": { "type": "float" },
    "rai_cityadjusted_total_2018_q2": { "type": "float" },
    "rai_cityadjusted_total_2021_q2": { "type": "float" },
    "rai_cityadjusted_total_2013_q1": { "type": "float" },
    "rai_cityadjusted_total_2021_q1": { "type": "float" },
    "rai_cityadjusted_total_2013_q3": { "type": "float" },
    "rai_cityadjusted_total_2013_q2": { "type": "float" },
    "rai_cityadjusted_total_2013_q4": { "type": "float" },
    "rai_cityadjusted_total_2017_q1": { "type": "float" },
    "rai_cityadjusted_total_2018_q4": { "type": "float" },
    "rai_cityadjusted_total_2018_q3": { "type": "float" },
    "rai_cityadjusted_total_2019_q3": { "type": "float" },
    "rai_cityadjusted_total_2019_q2": { "type": "float" },
    "rai_cityadjusted_total_2019_q1": { "type": "float" },
    "rai_cityadjusted_total_2016_q3": { "type": "float" },
    "rai_cityadjusted_total_2016_q4": { "type": "float" },
    "rai_cityadjusted_total_2016_q1": { "type": "float" },
    "rai_cityadjusted_total_2016_q2": { "type": "float" },
    "rai_cityadjusted_total_2012_q2": { "type": "float" },
    "rai_cityadjusted_total_2015_q2": { "type": "float" },
```

```
    "rai_cityadjusted_total_2012_q1": { "type": "float" },
    "rai_cityadjusted_total_2015_q3": { "type": "float" },
    "rai_cityadjusted_total_2012_q4": { "type": "float" },
    "rai_cityadjusted_total_2014_q3": { "type": "float" },
    "rai_cityadjusted_total_2012_q3": { "type": "float" },
    "rai_cityadjusted_total_2014_q4": { "type": "float" },
    "rai_cityadjusted_total_2015_q1": { "type": "float" },
    "rai_cityadjusted_total_2014_q1": { "type": "float" },
    "rai_cityadjusted_total_2014_q2": { "type": "float" },
    "rai_cityadjusted_total_2015_q4": { "type": "float" }
}
```

The mappings for `liveability-index` are as follows:

```
"properties": {
    "type": { "type": "text" },
    "topic": { "type": "text" },
    "id": { "type": "text" },
    "indicator": { "type": "text" },
    "period": { "type": "text" },
    "numerator": { "type": "text" },
    "denominator": { "type": "text" },
    "value": { "type": "float" },
    "value_type": { "type": "text" },
    "sources": { "type": "text" }
}
```

3.7 Kibana Index View

Kibana provides a comprehensive, powerful, and flexible platform for data visualization and analysis, making it an essential tool when working with Elasticsearch due to its ability to provide an intuitive interface for querying Elasticsearch.

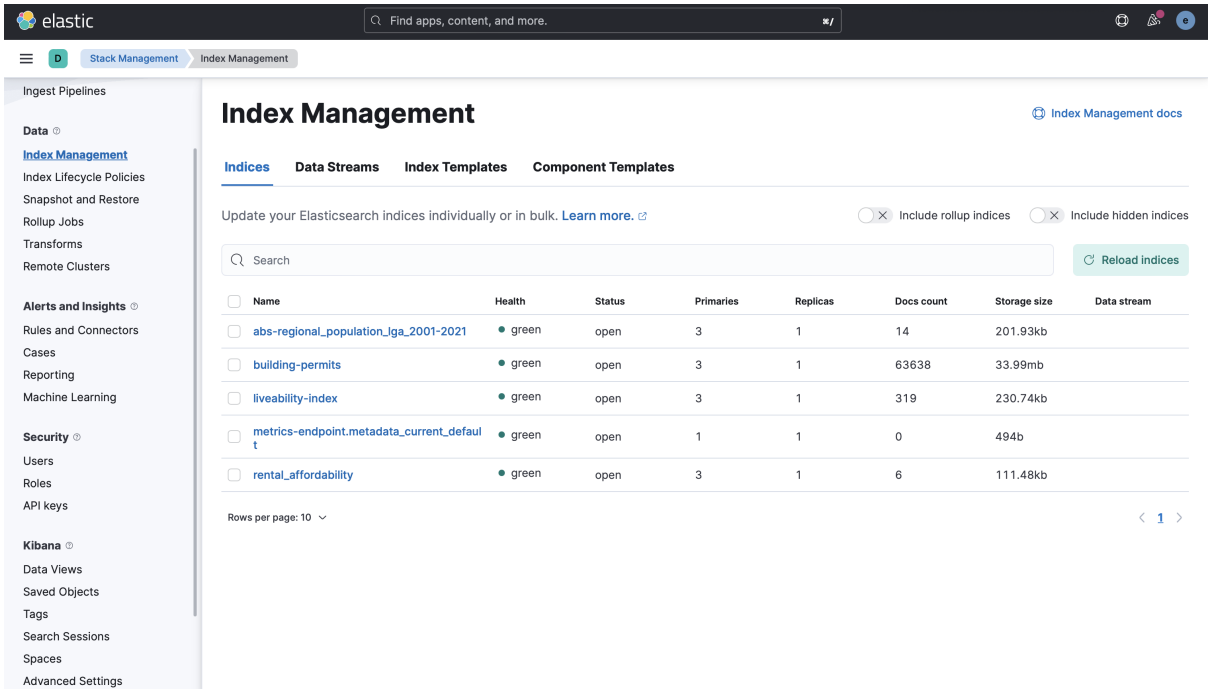


Figure 7: Index Management in Kibana

The Index Management option under the Stack Management section allows us to efficiently track all the indices available, displaying the number of primary shards and replica count for each primary shard used to create each index. It also provides the total number of documents present in an index and its storage size.

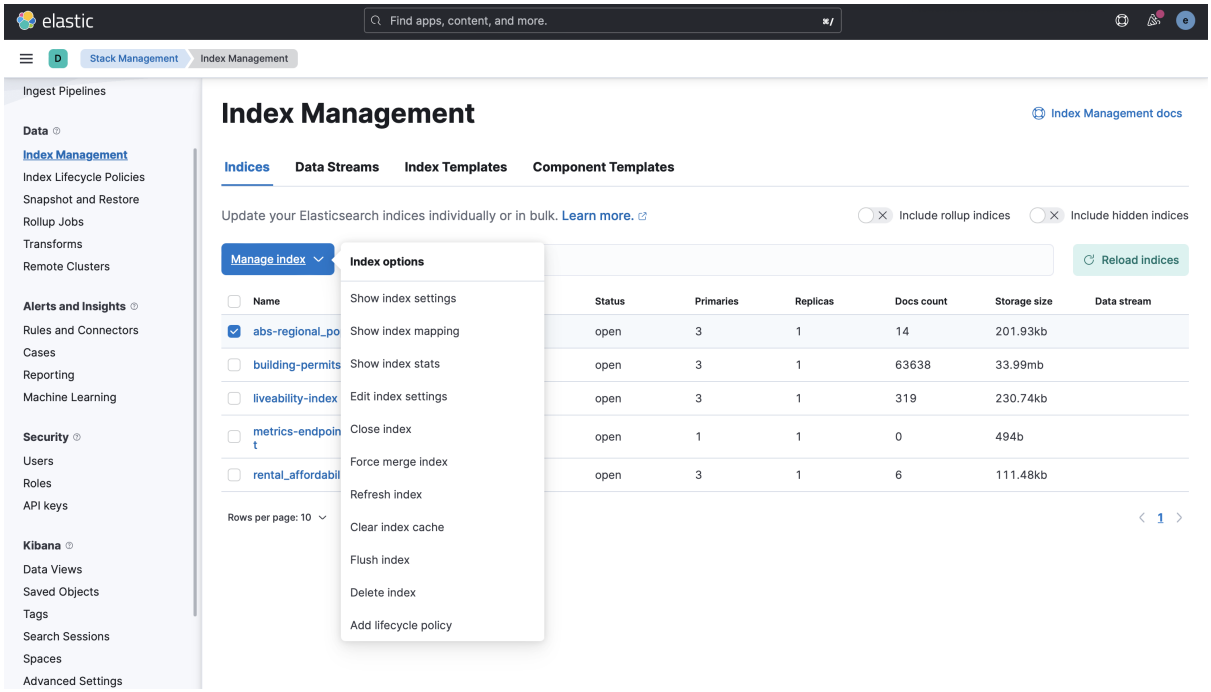


Figure 8: Index operations in Index Management

It is possible to select and perform operations on an individual index, such as deleting, refreshing the index, and more.

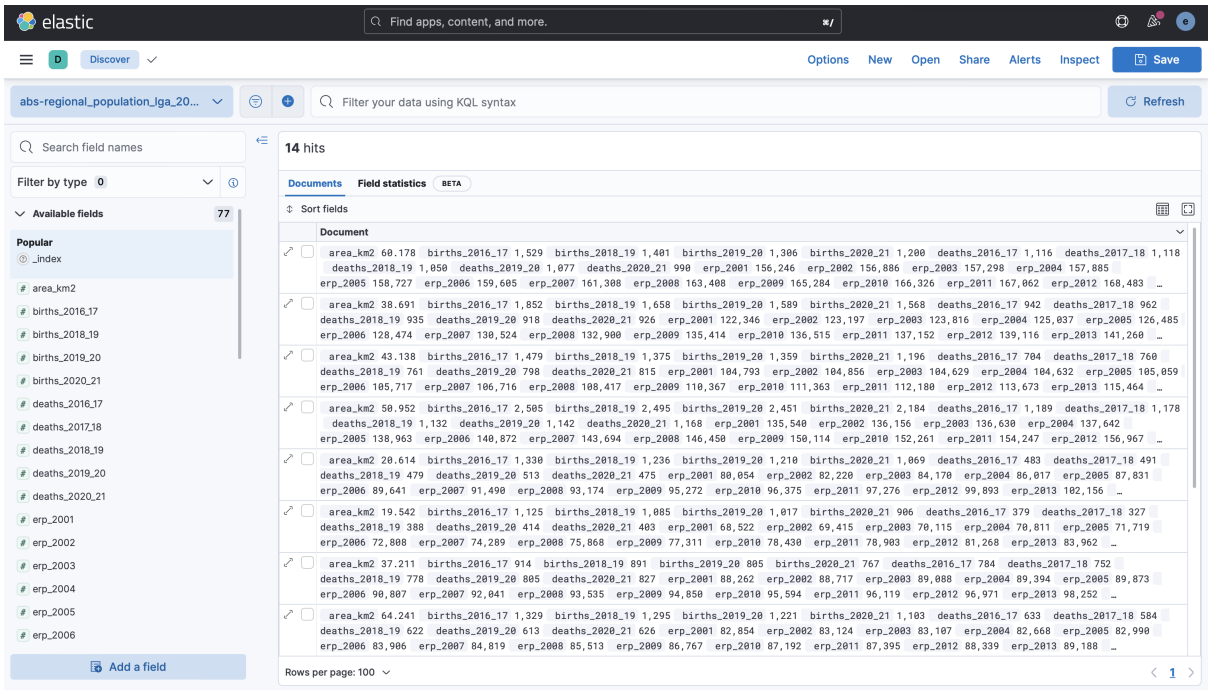


Figure 9: Data view for abs-regional_population_lga_2001-2021 index in Discover

We should then create a data view for each index to view the documents in each index and perform basic filtering operations. Each data view can be accessed from the Discover section.

3.8 Jenkins Working and GitHub Integration

Jenkins is a powerful and flexible open-source automation server widely used for continuous integration (CI) and continuous delivery (CD) of software projects. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and delivery. It is considered to be a crucial part of the agile cycle during software development.

Jenkins Installation and Working

The commands used for installing and running Jenkins locally are listed below:

```
Install the latest LTS version: brew install jenkins-lts
Start the Jenkins service: brew services start jenkins-lts
Restart the Jenkins service: brew services restart jenkins-lts
Update the Jenkins version: brew upgrade jenkins-lts
```

After starting the Jenkins service, browse to `http://localhost:8080` and follow the instructions to complete the installation.

Ngrok Installation and Working

Ngrok allows us to create secure tunnels to our localhost, making our local web server accessible over the internet. This is particularly useful during testing and development purposes.

The Ngrok agent has to be authenticated using our personal authtoken.

```
ngrok http http://localhost:8080
```

Ngrok starts a tunnel that forwards requests from a public URL to our local server running at `http://localhost:8080` (where Jenkins is running).

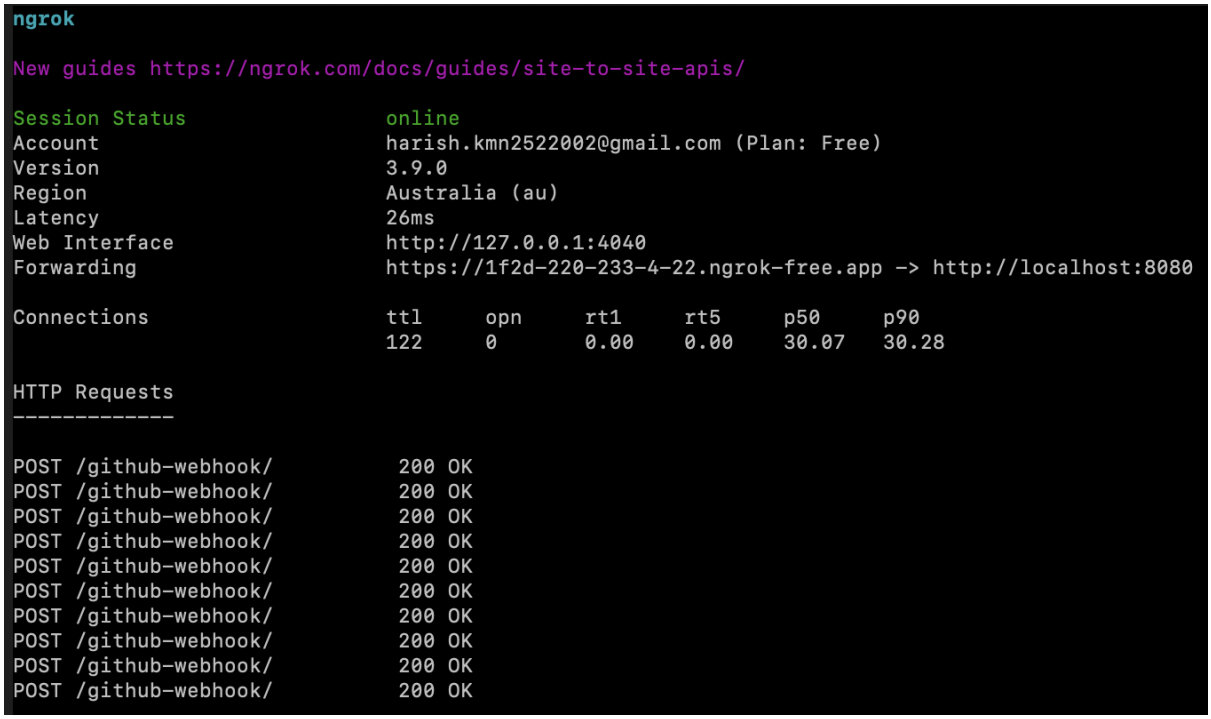


Figure 10: ngrok secure tunnel running for port 8080

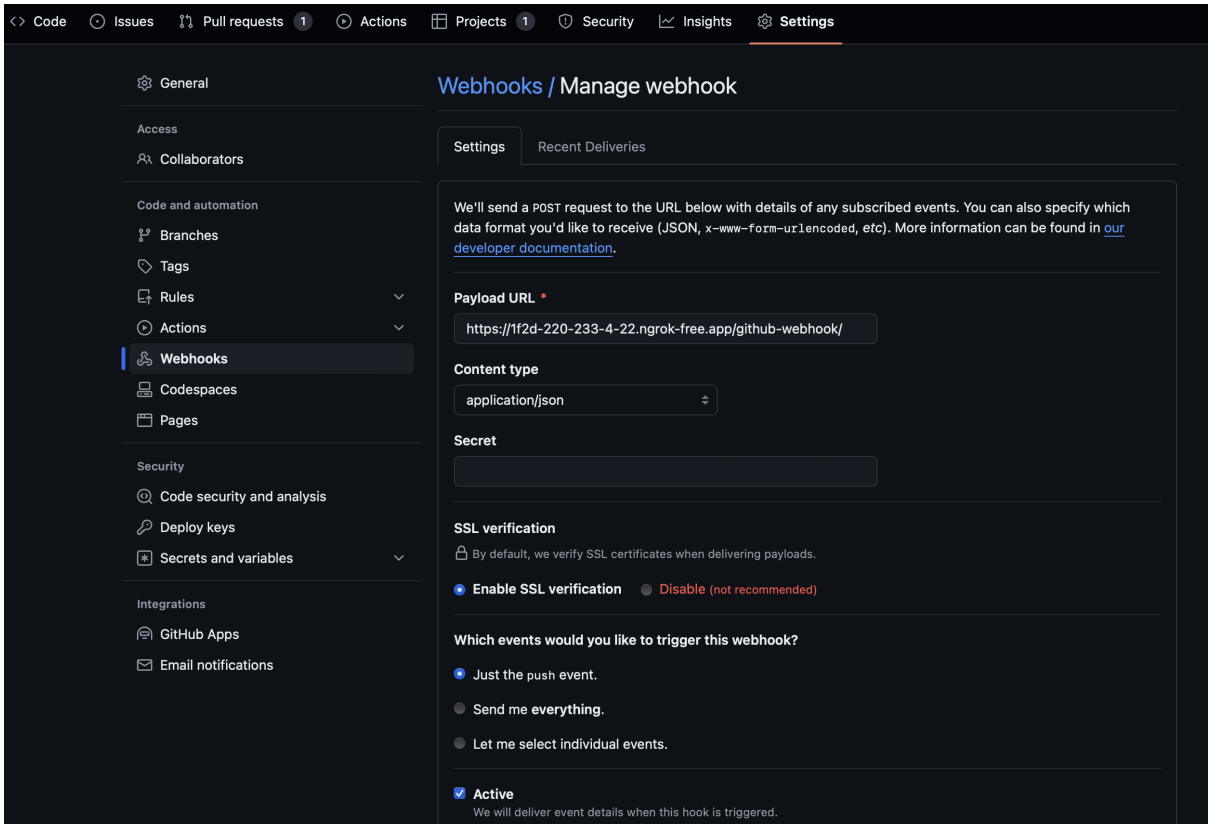


Figure 11: setting up GitHub webhook to connect to Jenkins project

A new freestyle Jenkins project was created with the same name as our GitHub repository. Under the configure option, it was linked to our GitHub project; the same was done for the source code management with the GitHub credentials as it is a private repository. Then the build trigger was enabled for the master branch. The GitHub hook trigger for GITScm polling option was enabled under the build triggers section, which functions as follows: when Jenkins receives a GitHub push hook, the GitHub Plugin checks to see whether the hook came from a GitHub repository that matches the Git repository defined in the SCM/Git section of this job. If they match and this option is enabled, the GitHub Plugin triggers a one-time polling on GITScm. When GITScm polls GitHub, it finds that there is a change and initiates a build.

The following commands are then executed in order to run a Docker container that runs the unit test cases for the Python scripts created and publishes the coverage report with the help of the `coverage` and `pytest-cov` modules:

```
echo $WORKSPACE
export PATH="/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin"
/usr/local/bin/docker build -t my-image ./test/
/usr/local/bin/docker run -v $WORKSPACE:/workspace --rm my-image
```

Next, we installed the Cobertura module in Jenkins to generate a coverage report dashboard with the

`cobertura.xml` that was added to the Jenkins workspace as a result of the execution of the Docker container. This setup enables hardgating the build with a minimum of 90% coverage required, else resulting in failure of the build. Also, the execution of the build triggers an automated email to each of the team members irrespective of the success or failure of the corresponding build, with the build log and coverage output as an HTML file, due to enabling the E-mail notification option. All these options are enabled under the post-build action section.

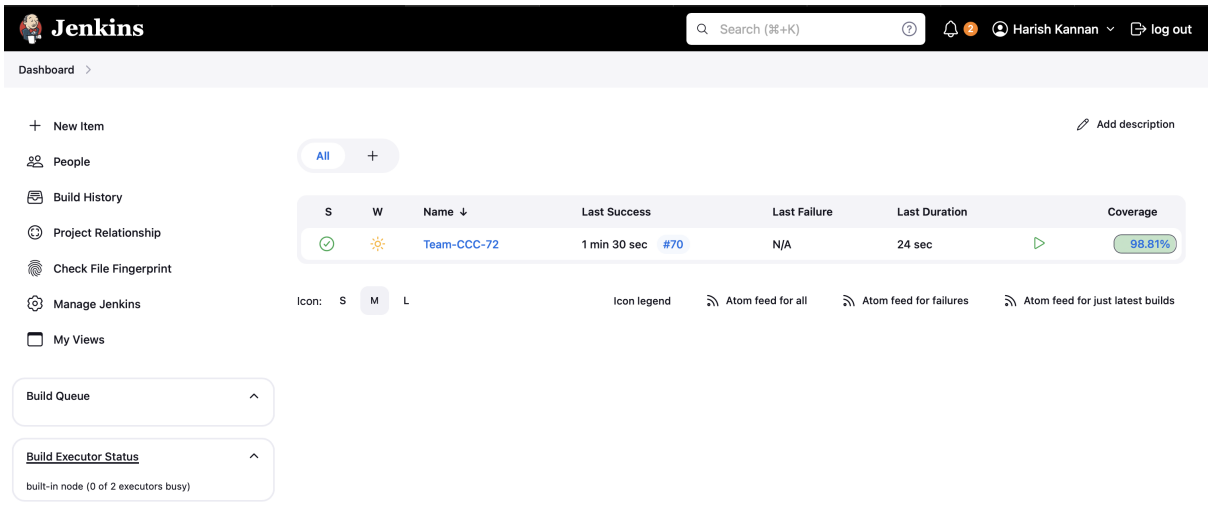


Figure 12: Jenkins project

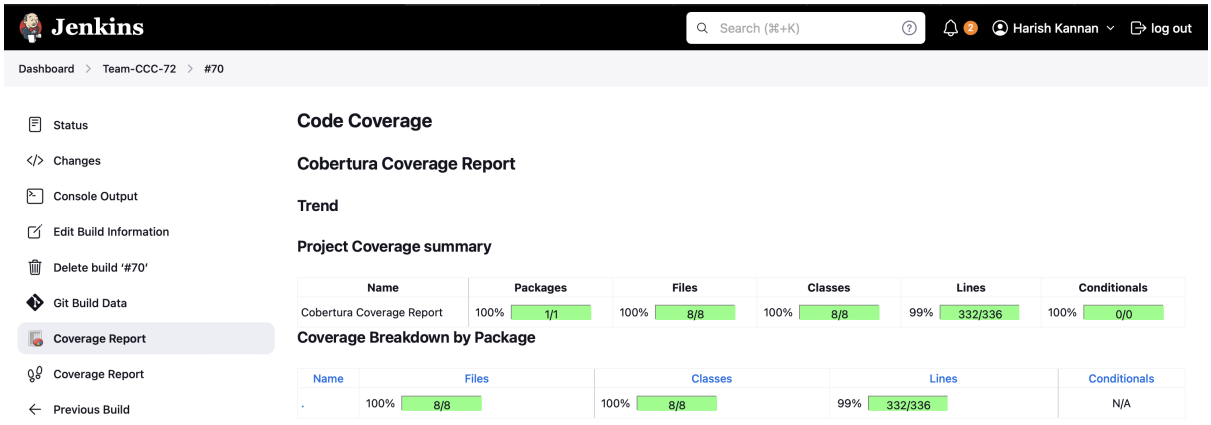


Figure 13: Code Coverage Report

Significance of Unit Tests

The unit test cases are written to capture the functionality of each function in the original Python script. These cases ensure that the application works as intended by verifying that each function behaves correctly according to its specifications. Unit testing is mainly used in every software development pipeline to improve code quality and detect bugs early.

In our case, we have written four unit test scripts for each of the four Fission scripts. As mentioned above, we are using a Dockerfile to create a Docker image which copies all the original Python scripts as well as the unit test scripts into its working directory, runs Python version 3.9, and installs the required packages to run the unit tests and generate a code coverage report.

For example, if the data that we export and use changes, the unit test cases might not be able to capture the change, eventually causing the Jenkins pipeline to fail. If someone makes a change that alters the Python scripts in the master branch, it might again cause the pipeline to fail. In this case, we would revert the changes made to the master branch. Additionally, if someone adds more functionality to the Python script, it might not be captured by the unit test cases, which will cause the code coverage rate to drop and can once again cause the pipeline to fail if the code coverage is less than 90%.

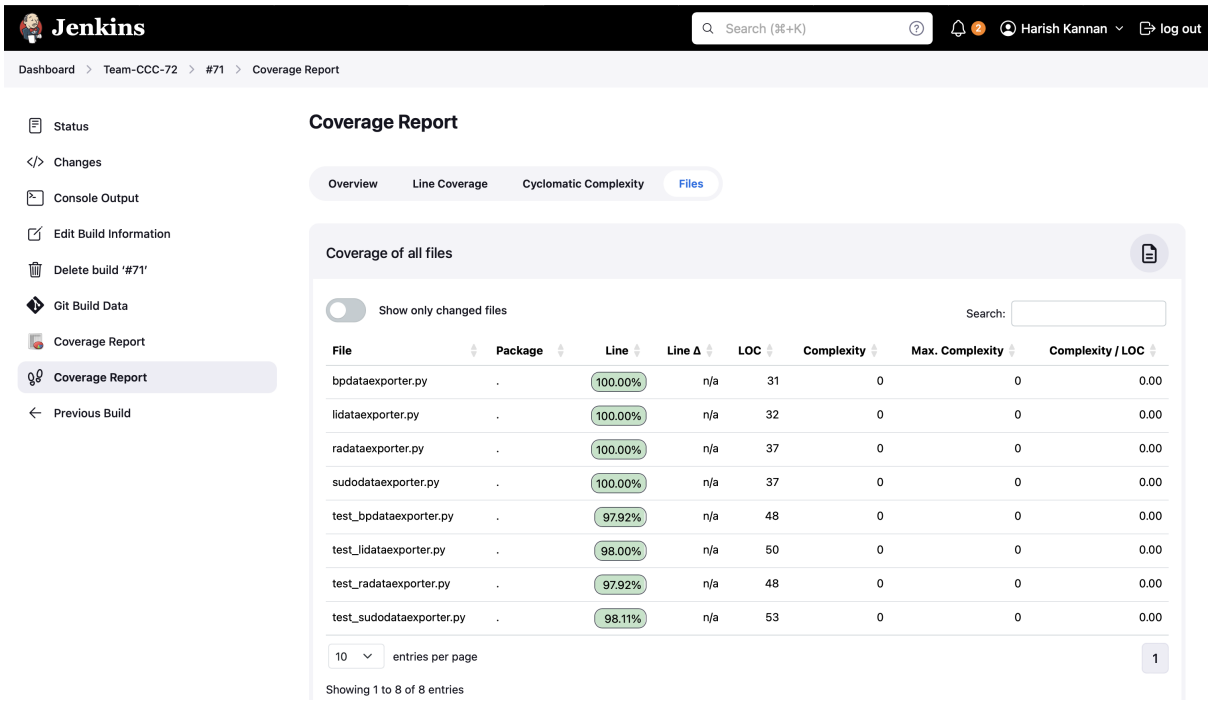


Figure 14: Detailed Coverage Report

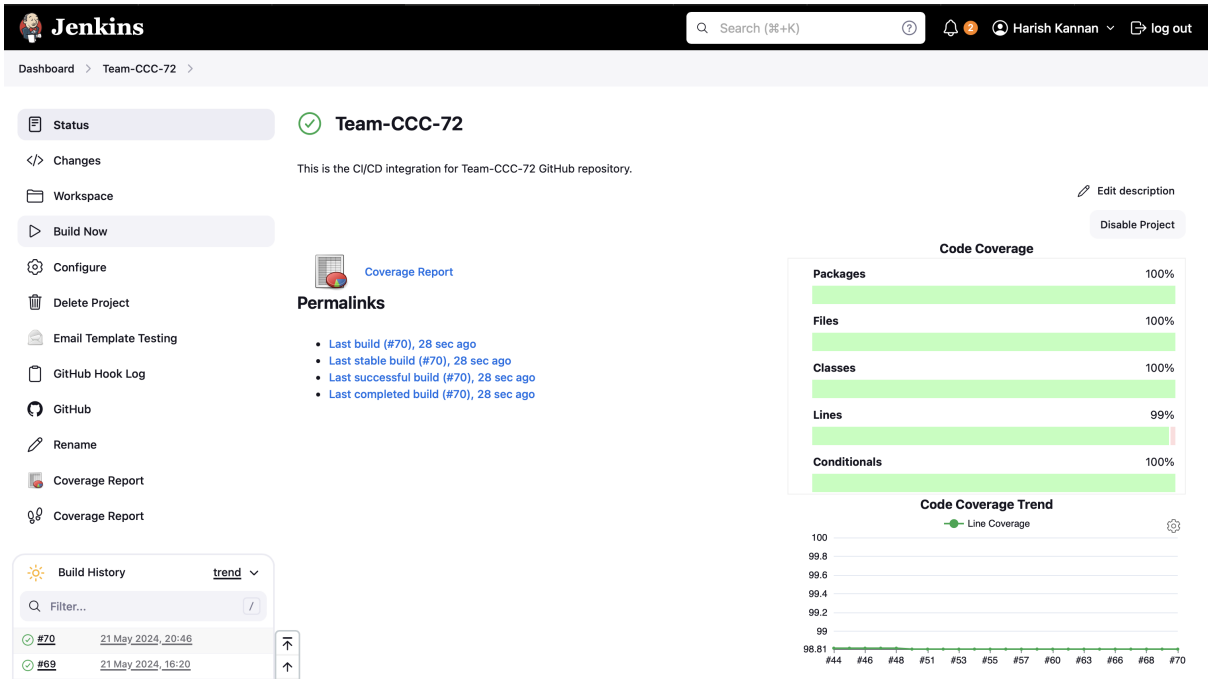


Figure 15: Jenkins project overview

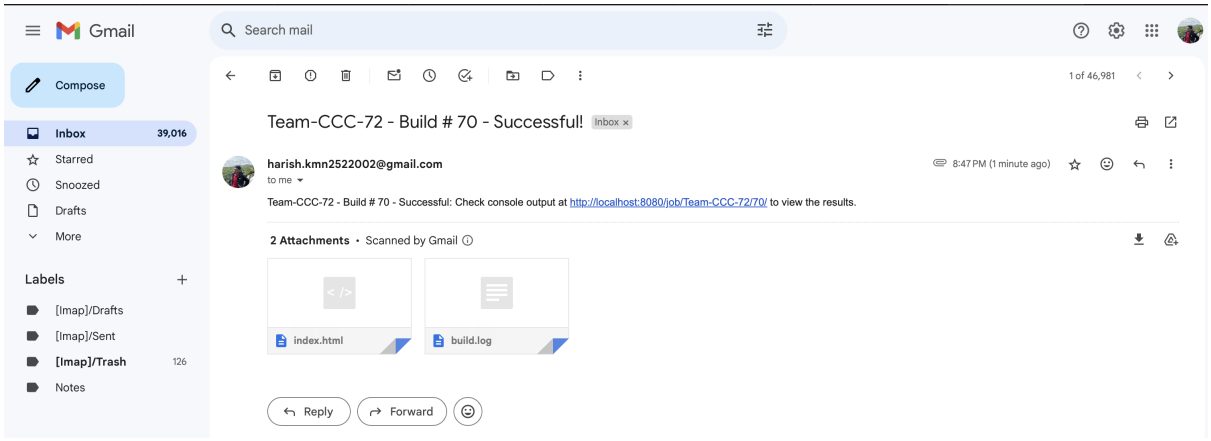


Figure 16: Email alert

4 System Functionality

4.1 Public Keys

To ensure all team members, except the creator of the Kubernetes cluster and the bastion node, could efficiently connect to the bastion node and utilize the functionalities of Elasticsearch, Kibana, and Fission pods for assignment 2, the creator added all the members' public keys to the `authorized_keys` file of the bastion node.

4.2 Fission Scripts

These Python scripts are the core of the backend solution that we implemented as a team. The hierarchy of the backend directory is as follows:

```
backend
- fission
    - specs
- functions
- sudodataexporter
    __init__.py
    requirements.txt
    build.sh
    sudodataexporter.py
- sudodataexporter.zip
- bpdataexporter
- bpdataexporter.zip
- radataexporter
- radataexporter.zip
- lidataexporter
- lidataexporter.zip
```

The `bpdataexporter`, `radataexporter`, `lidataexporter` directories also contain similar structured files like `sudodataexporter` as mentioned in the hierarchy structure above.

Prerequisites

- It is mandatory to make sure the UniMelb Student VPN (if off-campus) connection is enabled.
- Obtain the OpenStack RC file and API password, and source them in the current shell.
- Install OpenStack clients.
  Note: Please ensure the following OpenStack clients are installed:
  - `python-cinderclient`
  - `python-keystoneclient`
  - `python-magnumclient`
  - `python-neutronclient`
  - `python-novaclient`
  - `python-octaviaclient`
- Install necessary components (enabling command line functionalities):

1. JQ
2. Kubectl
3. Helm
4. Fission

- A Kubernetes cluster created on NeCTAR.

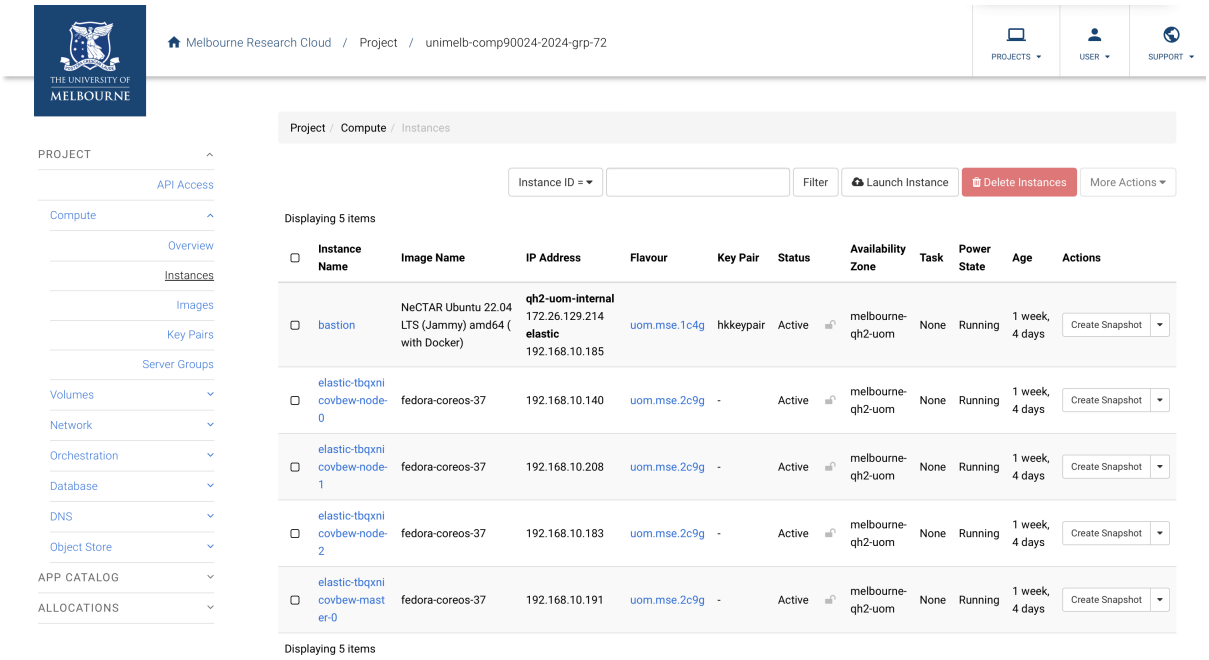


Figure 17: Instances running in Team 72's MRC project

Then ensuring SSH tunneling is enabled by logging into the bastion server (or jump server) which is responsible for a reliable and secure connection from the local system to the services running in the Kubernetes cluster.

Checking all the Elasticsearch, Kibana, and Fission pods are up and healthy using the following commands: Checking if the indexes for storing the respective data are created with their unique schemas.

For Elasticsearch and Kibana

```
kubectl get pods -n elastic
```

```
NAME                                READY   STATUS    RESTARTS   AGE
elasticsearch-master-0              1/1     Running   0          11d
elasticsearch-master-1              1/1     Running   0          11d
kibana-kibana-749869bc5c-g6vbk      1/1     Running   0          11d
```

The above command lists all the pods under the namespace `elastic` and their current status.

For Fission

```
kubectl get pods -n fission
```

```
NAME                              READY   STATUS    RESTARTS   AGE
buildermgr-b9d6b56f9-djgm7        1/1     Running   0          11d
executor-8b5c8489c-qcmf9          1/1     Running   0          11d
kubewatcher-5dbb4c466-hdfcw       1/1     Running   0          11d
mqtrigger-keda-7c8b8f5ddc-vpl5z   1/1     Running   0          11d
router-747967986f-nfqvp           1/1     Running   0          11d
storagesvc-5d98cd6f54-7d8sp       1/1     Running   0          11d
timer-66c797b5c6-c78sh            1/1     Running   0          11d
webhook-7bb6d4875b-5v7j7          1/1     Running   0          11d
```

The above command lists all the pods under the namespace `fission` and their current status.

Then port forward the Fission, Elasticsearch, and Kibana services running as pods in the Kubernetes cluster to the localhost using the commands listed below.

For Fission

```
kubectl port-forward service/router -n fission 9090:80
```

The functionality of Fission (accessing the Fission routes) from the localhost via port 9090 can be ensured with this.

For Elasticsearch

```
kubectl port-forward service/elasticsearch-master -n
elastic 9200:9200
```

The read and write access to the Elasticsearch database from the localhost via port 9200 can be ensured with this.

For Kibana

```
kubectl port-forward service/kibana-kibana -n elastic 5601:5601
```

The read and write access to Kibana (GUI for Elasticsearch) from the localhost via port 5601 can be ensured with this.

4.3 Specs

The `fission` directory has two subdirectories, namely `specs` and `functions`. The `specs` (default name) directory was created in order to hold our specifications. It was done with the following command:

```
(
  cd fission
  fission specs init
)
```

From now on, our actions will add YAML files under the `specs` directory, and the YAML files will then be applied to the cluster with the `kubectl apply` command.

We started by creating the specs for the Python environment (since our scripts are entirely based on Python):

```
(
  cd fission
  fission env create --name python --builder fission/
  python-builder-3.9 --image fission/python-env-3.9
)
```

Passing Parameters to Functions with ConfigMaps Parameters (such as usernames and passwords) can be passed to functions through the environment rather than being hard-coded in the source code (which has to be avoided, especially for sensitive information). A common way to share parameters in Kubernetes is through ConfigMaps, which are key-value pairs that can be accessed by all pods within a namespace. Fission functions can read ConfigMaps as files, so we can create a ConfigMap with the Elasticsearch username and password and also the GitHub username and the personal access token of the GitHub repository owner. This was done because two of our scripts require the datasets from SUDO, and this was ensured by creating a session to pull the data from our private repository with the GitHub credentials.

Let's start by creating a ConfigMap as `shared-data.yaml` file under the `specs` directory:

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: default
  name: shared-data
data:
  ES_USERNAME: ""
```

```
  ES_PASSWORD: ""
  GITHUB_USERNAME: ""
  GITHUB_TOKEN: ""
```

Since ConfigMaps are not directly managed by Fission, we have to apply them to the cluster with the `kubectl` command:

```
kubectl apply -f ./fission/specs/shared-data.yaml
```

To use these values, we have to make sure our function uses the ConfigMap. In addition, we have to create the function definition so that the ConfigMap is mounted as volume

`/configs/default/shared-data`.

4.4 Fission Function - Implementation

The main purpose of the Fission function is to store the data into Elasticsearch. To be able to pack libraries (such as the Elasticsearch Python client) together with the function source code,

a `requirements.txt` file must be created in the same directory as the function. Then a `build.sh` command must be created to install the libraries, and finally, the function must be packaged in a ZIP file as shown below:

```
(
  cd ./fission/functions/sudodataexporter
  zip -r sudodataexporter.zip .
  mv sudodataexporter.zip ../
)
```

Note: The ZIP file must be relative to the directory the function is in; otherwise, the package build will fail. To avoid this, it is advised to always build the ZIP file from within the directory the function is in and then move it somewhere else to avoid a recursive ZIP file (see the shell commands above).

Creation of a Function with Dependencies This function depends on the Elasticsearch client package to add data to Elasticsearch. The following code is run in order to create a package, a function, and also a route, all named `sudodataexporter`, in order to populate the

`abs_regional_population_lga_2001_2021` dataset obtained from SUDO into the Elasticsearch index. The following code consists of four individual parts, and each of them will be elaborated in detail below:

```
(
  cd fission

  fission package create --sourcearchive
  ./functions/sudodataexporter.zip\
    --spec\
    --env python\
    --name sudodataexporter\
    --buildcmd './build.sh'

  fission fn create --name sudodataexporter\
    --spec\
    --pkg sudodataexporter\
    --env python\
    --entrypoint "sudodataexporter.main"\
    --configmap shared-data

  fission route create --spec --url /sudodataexporter --function
  sudodataexporter --name sudodataexporter --createingress
)
```

1. To move into the fission directory.

2. This command creates a Fission package named `sudodataexporter` using the source code from the path `./functions/sudodataexporter.zip`. The package is intended to run in a Python environment that was created earlier. The build command is specified to run any necessary build steps defined inside the `./build.sh` file. The `--spec` flag ensures that a spec file is generated, allowing for declarative management of the Fission package inside the `specs` directory that was created.

3. This command creates a Fission function named `sudodataexporter` which refers to the

`sudodataexporter` package and runs in a Python environment created earlier. The `spec` argument generates a spec file for declarative management of the function. The `entrypoint` argument executes the `main` function from the `sudodataexporter` module when being invoked. Finally, the `shared-data` ConfigMap is used for accessing secrets and credentials.

4. This command helps to set up a route (`/sudodataexporter`) that directs HTTP requests from the route to the `sudodataexporter` function. The `createingress` argument ensures that the route is accessible externally, exposing the Fission functions to external clients. The `--spec` flag ensures that the route configuration is saved in a spec file in the `specs` folder for easy management and reproducibility.

4.5 Fission Function - Validation and Testing

```
(
  cd fission
  fission spec validate
)
```

The `validate` command is used to check if everything is fine with our specs.

```
fission spec apply --specdir fission/specs --wait
```

The `spec apply` command is used to apply the specs to the cluster provided that no errors were reported from the `validate` command.

```
fission fn log -f --name sudodataexporter
```

```
[2024-05-19 05:42:17,872] INFO in server: specialize called with
filepath = "/userfunc/deployarchive" handler =
"sudodataexporter.main"
2024-05-19 05:42:17,872 - INFO - specialize called with
filepath = "/userfunc/deployarchive" handler =
"sudodataexporter.main"
[2024-05-19 05:42:17,873] DEBUG in server: moduleName =
"sudodataexporter" funcName = "main"
2024-05-19 05:42:17,873 - DEBUG - moduleName =
"sudodataexporter" funcName = "main"
[2024-05-19 05:42:17,873] DEBUG in server: __package__ = "None"
2024-05-19 05:42:17,873 - DEBUG - __package__ = "None"
Success - 14, Failed - 0
Finished
Documents indexed in 0.154311 seconds
```

The log will start streaming the logs of either the Fission function or route named

`sudodataexporter` in real-time. It allows us to monitor the function or route's execution and output continuously, which can be used for debugging, monitoring, and understanding the behavior of the function as it processes requests. With the successful execution, the

`abs_regional_population_lga_2001_2021` data was successfully populated to the respective Elasticsearch index.

```
curl "http://127.0.0.1:9090/sudodataexporter"
```

```
OK
```

The `curl` command sends an HTTP GET request to the `sudodataexporter` endpoint on a server running on the local machine and listening on port 9090. This command can be used to trigger the respective function and view its output directly in the terminal.

The Fission function implementation, validation, and testing steps are carried out in a similar manner for the rest of the modules, namely:

- `bpdataexporter` (for indexing the building permits dataset from DataVic)
- `radataexporter` (for indexing the rental affordability index dataset from SUDO)
- `lidataexporter` (for indexing the Melbourne liveability indicators dataset from DataVic)

4.6 .gitignore

The `.gitignore` file is an essential component of Git and GitHub repositories. The main purpose of the `.gitignore` file is to prevent sensitive information from being tracked, exclude unnecessary files or directories, improve the performance of Git operations, and finally, maintain a clean project.

In our case, these are the contents that were added to the `.gitignore` file and were exempted from being displayed from the GitHub repository:

```
backend/fission/specs
backend/fission/functions/sudodataexporter.zip
backend/fission/functions/bpdataexporter.zip
backend/fission/functions/radataexporter.zip
backend/fission/functions/lidataexporter.zip
```

The reason for including these contents in the `.gitignore` file is to avoid displaying the `ConfigMap` (`shared-data.yaml`) as it contains secrets and credentials and all the YAML files generated as a result of using the `spec` argument while creating packages, functions, and specs for the respective scripts. Also, not displaying the zipped version of each script directory, namely `sudodataexporter.zip`, `bpdataexporter.zip`, `radataexporter.zip`, and `lidataexporter.zip`, since the exact contents inside them are already present in the script directories.

5 Error Handling

The following sections include information regarding the error handling mechanisms implemented in this project.

5.1 Single Point of Failure

For this project's cloud solution, there are a a few areas of concern when it comes to the single point of failure. The following list encapsulates said concerns.

Master Node: The master node in a Kubernetes setup is a single point of failure. If it goes down, the entire cluster's orchestration and management capabilities are lost.

Bastion Node: If the bastion node goes down, access to the Kubernetes cluster is interrupted.

Melbourne Research Cloud (MRC): The cloud provider infrastructure can still be a single point of failure. Issues at the cloud provider level can impact the entire setup.

As previously mentioned, the project has implemented several mechanisms to improve error handling and make the application fault-tolerant. Some of these cases are detailed below:

- We have set the `number_of_replicas` as 2 for the databases to ensure that each primary shard has 2 replica shards. This ensures high availability of the data as even if one or two nodes fail, the data can still be obtained from the remaining shards. Having more replicas can help distribute query loads more effectively, enhancing read performance.

Figure 18: abs-regional_population_lga_2001-2021 index

- For the building permits dataset from DataVic, where the document to be queried from the respective API was limited to 10,000, which was much less than the actual document count. After some research, we were able to get the API to export the entire dataset as a JSON file with the help of the `requests` package. With this approach, and considering that the dataset is updated rarely, we can use the corresponding Fission function or route to clean and populate the new data into the `building-permits` index.
- For the building dataset, we wanted to create an interactive map plot for visualizing the building permits allotted to a specific postcode for any given year and one for all years combined. We cleaned the dataset by filtering it to have data with proper dates, and for implementing a map plot, we required coordinates (latitude, longitude). To tackle this, we used the `pgeocode` package, which takes the postcodes as input and outputs the coordinates for them.
- The bastion node (or jump host) that we connect through SSH tunneling acts as a secure gateway to connect the local system to the Kubernetes cluster.

- In all datasets, before populating the data into each respective index, all NaN values and null values were replaced with `None` to ensure no errors occur during the bulk operation.
- Then, unit test scripts were included for the original Python scripts to improve code quality and ensure that the code coverage is at least above 90%. If the coverage falls below this threshold, the build will fail in the Jenkins pipeline. This process is discussed briefly in the above sections.

5.2 GitHub and Jenkins

The use of GitHub and Jenkins further assists with the error handling between team members. This targets more at human errors which may be caused by any group member.

Version Control: GitHub ensures that the codebase is safely stored and can be restored in case of failure.

CI/CD Pipelines: Jenkins automates the build and deployment process, ensuring that changes are tested and deployed reliably. It also notifies all members through their respective emails.

5.3 Scalability

For the technologies used within the scope of this research project, there are methods for them to dynamically scale according to demand. Here is a quick summary of how each tool handles dynamic scaling:

Kubernetes: Kubernetes has numerous built-in features for automatic scaling. This means it can handle changes in workload efficiently without much manual intervention but will require some setup.

Fission: Fission, which runs on top of Kubernetes, is designed to handle serverless functions and comes with its own auto-scaling features:

- **Idle Management:** If a function isn't being used, Fission scales it down to zero instances. This way, resources aren't wasted on inactive functions.
- **On-Demand Scaling:** When there's a sudden increase in demand for a function, Fission can quickly create more instances to handle the load. This ensures that the system can cope with spikes in traffic without any manual adjustments.

Elasticsearch: Elasticsearch is built to scale dynamically by adding shards and replicas. This allows it to manage large data volumes and heavier query loads effectively via the following methods:

- **Shard Allocation:** Data is split into smaller pieces called shards, which are spread across different nodes. Adding more shards helps distribute the load evenly and improves performance.
- **Replica Management:** Replicas are copies of the data that help with redundancy and fault tolerance. By increasing the number of replicas, Elasticsearch can handle more queries and provide better data reliability.

Melbourne Research Cloud (MRC): Although this project doesn't directly involve this, it's worth mentioning that the Melbourne Research Cloud (MRC) offers scalable resources that can be very useful. Some options to be considered for a more unrestricted research project include:

- **Scalable Virtual Machines:** MRC allows you to scale up or down the number of virtual machines based on your needs. This flexibility ensures that you only use and pay for the resources you need at any given time.
- **Flexible Storage Solutions:** MRC provides storage options that can grow with your project. These storage solutions integrate well with Kubernetes, ensuring that your data storage scales with your compute resources.

6 Data Analysis

6.1 Scenario 1: Estimated Regional Population & Building Permits

Overview

We have developed a robust model to predict future population trends in Melbourne. Additionally, we have analyzed and visualized the correlation between building permits and population growth, as well as identified historical trends in both population and building permits. Also, we examined year-to-year changes in these variables. Thus, we are able to provide valuable insights for urban planners, policymakers, developers, and various dependent sectors.

Data Collection

The data are stored in Elasticsearch indices. We used the Elasticsearch Python client to extract the data from the respective indices:

- Regional Population Dataset: Extracted from the index

`abs_regional_population_lga_2001_2021`.

- Building Permits Dataset: Extracted from the index `building-permits`.

Linear Regression Modelling

We developed three different models to analyze the relationship and predict future population trends.

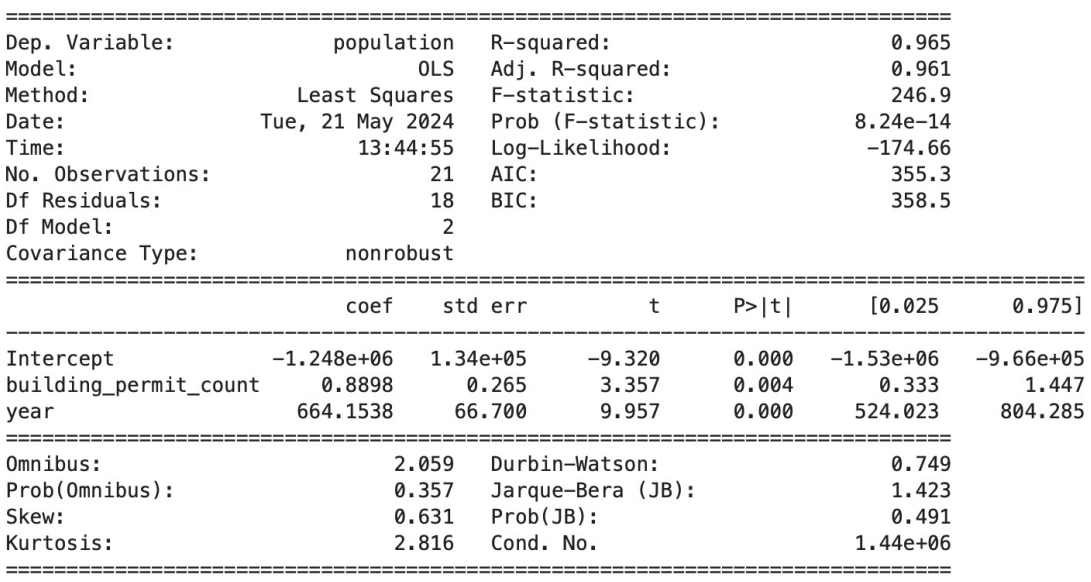
Partial Model (Simple Linear Regression) The first model was a simple linear regression using building permit counts to predict population. This model helps to understand the basic linear relationship between the two variables.

```
==============================================================================
Dep. Variable:             population   R-squared:                       0.771
Model:                            OLS   Adj. R-squared:                  0.759
Method:                 Least Squares   F-statistic:                     64.00
Date:                Tue, 21 May 2024   Prob (F-statistic):           1.68e-07
Time:                        13:44:55   Log-Likelihood:                -194.33
No. Observations:                  21   AIC:                             392.7
Df Residuals:                      19   BIC:                             394.7
Df Model:                           1
Covariance Type:            nonrobust
===============================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
Intercept               8.532e+04    733.055    116.395      0.000    8.38e+04    8.69e+04
building_permit_count     3.0435      0.380      8.000      0.000       2.247       3.840
==============================================================================
Omnibus:                        2.035   Durbin-Watson:                   0.508
Prob(Omnibus):                  0.362   Jarque-Bera (JB):                1.726
Skew:                           0.621   Prob(JB):                        0.422
Kurtosis:                       2.345   Cond. No.                     2.44e+03
==============================================================================
```

Figure 19: OLS Regression Results

The model explains 77.1% of the variance in the population data, indicating a strong relationship. The F-statistic and the corresponding P-value suggest that the model is statistically significant.

Full Model (Multiple Linear Regression) To improve the model's accuracy, we included the year as an additional predictor. This accounts for the time trend in the population data.



Dep. Variable:	population	R-squared:	0.965
Model:	OLS	Adj. R-squared:	0.961
Method:	Least Squares	F-statistic:	246.9
Date:	Tue, 21 May 2024	Prob (F-statistic):	8.24e-14
Time:	13:44:55	Log-Likelihood:	-174.66
No. Observations:	21	AIC:	355.3
Df Residuals:	18	BIC:	358.5
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.248e+06	1.34e+05	-9.320	0.000	-1.53e+06	-9.66e+05
building_permit_count	0.8898	0.265	3.357	0.004	0.333	1.447
year	664.1538	66.700	9.957	0.000	524.023	804.285

Omnibus:	2.059	Durbin-Watson:	0.749
Prob(Omnibus):	0.357	Jarque-Bera (JB):	1.423
Skew:	0.631	Prob(JB):	0.491
Kurtosis:	2.816	Cond. No.	1.44e+06

Figure 20: OLS Regression Results

Including the year as a predictor significantly improved the model, explaining 96.5% of the variance in the population data. The high F-statistic and low P-value indicate that both predictors (building permit count and year) are significant.

Interaction Model The final model included an interaction term between building permit counts and year to capture the interaction effect.

Dep. Variable:	population	R-squared:	0.981
Model:	OLS	Adj. R-squared:	0.978
Method:	Least Squares	F-statistic:	291.3
Date:	Tue, 21 May 2024	Prob (F-statistic):	8.26e-15
Time:	13:44:55	Log-Likelihood:	-168.24
No. Observations:	21	AIC:	344.5
Df Residuals:	17	BIC:	348.7
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-1.066e+06	1.12e+05	-9.493	0.000	-1.3e+06	-8.29e+05
building_permit_count	-397.1321	105.112	-3.778	0.002	-618.900	-175.364
year	573.4903	55.934	10.253	0.000	455.481	691.500
building_permit_count:year	0.1974	0.052	3.787	0.001	0.087	0.307

Omnibus:	1.275	Durbin-Watson:	1.431
Prob(Omnibus):	0.529	Jarque-Bera (JB):	0.251
Skew:	-0.147	Prob(JB):	0.882
Kurtosis:	3.448	Cond. No.	2.47e+09

Figure 21: OLS Regression Results

The interaction model provided the best fit, explaining 98.1% of the variance in the population data.

The interaction term was significant, indicating that the effect of building permits on population changes over time.

Model Evaluation Metrics

Goodness of Fit

- Partial Model (Simple Linear Regression): R-squared = 0.771
- Full Model (Multiple Linear Regression): R-squared = 0.965
- Interaction Model: R-squared = 0.981

AIC Comparison The AIC (Akaike Information Criterion) values were compared to assess model fit. Lower AIC values indicate better model fit. The interaction model had the lowest AIC, indicating the best fit among the three models.

ANOVA Test ANOVA (Analysis of Variance) was used to compare the models. The interaction model showed significant improvement over the full model, confirming the importance of including the interaction term. Therefore, the best model was the interaction model.

Forecasting Future Population

Using the interaction model, we forecasted the population for 2022, 2023, and 2024. The predictions were as follows:

```
future_predictions = interaction_model.predict(predict_df)
```

Predicted Populations:

- 2022: 99,794
- 2023: 100,857
- 2024: 96,387

Analysis

Correlation Analysis We computed the correlation between building permits and population to understand the initial relationship.

```
correlation = analysis_df['building_permit_count'].corr(
analysis_df['population'])
print(f"Correlation between building permit count and
population: {correlation:.2f}")
```

This calculates the Pearson correlation coefficient between `building_permit_count` and

`population` and prints it. The correlation coefficient of 0.88 indicates a strong positive correlation between building permit counts and population. This means that as the number of building permits increases, the population also tends to increase.

Scatter Plot The below scatter plot shows the relationship between building permit counts and population, with each point representing a year from 2001 to 2021. The color gradient indicates the year, with darker colors representing earlier years and lighter colors representing more recent years.

Figure 22: Correlation Analysis: Building Permits vs Population

Trend Analysis This line chart illustrates the trends in building permits and population over the years from 2001 to 2021. The following observations can be made:

- The red line represents the population trend, showing a steady increase over the years.
- The blue line represents the building permit counts, which also show an upward trend, though with more fluctuations compared to the population trend.

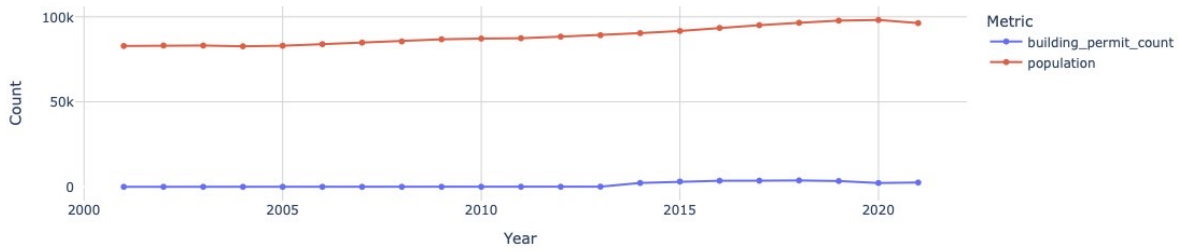


Figure 23: Trend Analysis: Building Permits and Population Over the Years

Year-over-Year Growth Rate The bar chart presents the year-over-year growth rates for building permits and population. Each bar represents the percentage growth for a given year:

- **Permit Growth:** The blue bars show the percentage change in the number of building permits issued each year.
- **Population Growth:** The red bars show the percentage change in the population each year.

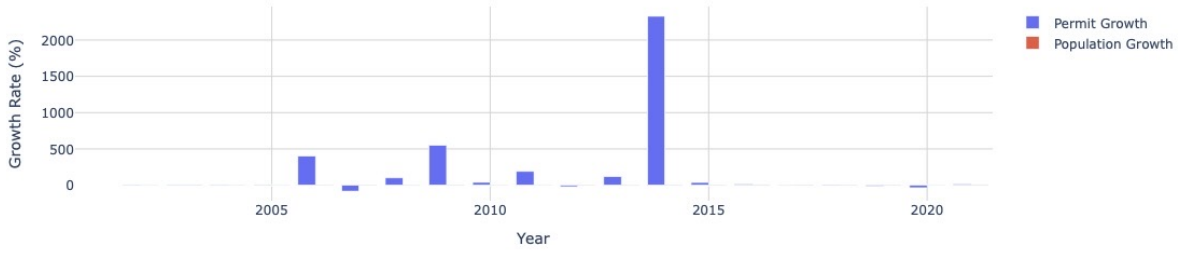


Figure 24: Year-over-Year Growth Rate

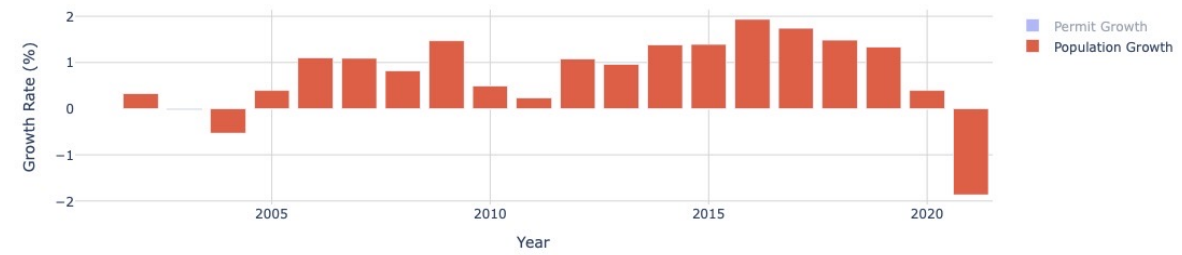


Figure 25: Year-over-Year Growth Rate

The chart reveals significant variability in building permit growth rates, with some years experiencing substantial increases. In contrast, population growth rates are more stable but also show periods of higher growth.

Outlier Detection The box plot identifies outliers in the building permits and population data:

- **Building Permit Count:** The left box plot shows the distribution of building permit counts, with several outliers indicated by individual points.
- **Population:** The right box plot shows the distribution of population data, with fewer outliers compared to building permits.

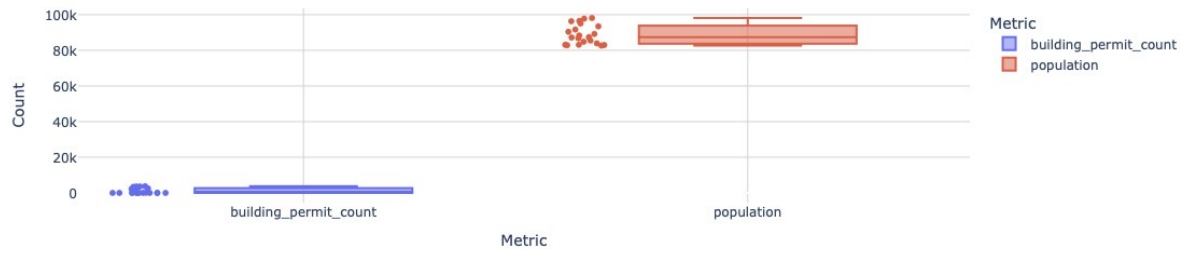


Figure 26: Outlier Detection: Building Permits and Population

This visualization helps to identify years with unusually high or low values, which can be important for understanding anomalies in the data.

Predicted Population vs. Actual Population The predicted population and future population predictions from the interaction model, as well as the actual population, are visualized by an interactive plot to provide better understanding. The chart is divided into three key components:

- **Actual Population:** Represented by the solid blue line with circular markers, this line shows the observed population data from 2001 to 2021.
- **Predicted Population:** Represented by the dashed orange line, this line shows the population predicted by the interaction model based on the building permit counts and year.
- **Future Predictions:** Represented by the green line with triangular markers, this line shows the projected population for the years 2022, 2023, and 2024. The shaded green area indicates the prediction range, showing potential variability in future predictions.

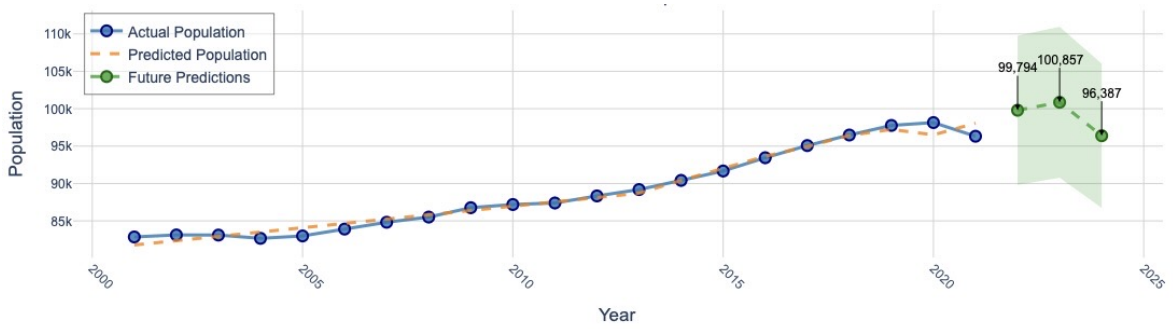


Figure 27: Actual vs Predicted Population Prediction for ’22, ’23, ’24

Results

The visualizations collectively provide a comprehensive view of the relationship between building permits and population in Melbourne. Key takeaways include:

- **Strong Positive Correlation:** There is a significant positive correlation (0.88) between building permits and population, indicating that as building activity increases, the population tends to grow.
- **Predictive Accuracy:** The interaction model, incorporating both building permits and year, provides accurate predictions for future population trends with an R-squared value of 0.981.
- **Future Prediction:** Predicted the population for 2022, 2023, and 2024.
- **Growth Patterns:** Both building permits and population show upward trends over the years, with building permits exhibiting more variability.
- **Outliers and Anomalies:** The box plots identify outliers in the data, highlighting years with unusual building activity or population changes.

These findings underscore the importance of building activity in driving population growth and provide valuable insights for urban planning and policy-making in Melbourne.

6.2 Scenario 2: Liveability & Rental Affordability Index

Overview

With the rental affordability and liveability scores available for data analysis, a natural question that arises is whether Melbourne’s rental affordability index correlates with the liveability of the city. A reasonable assumption is that the broad concept of liveability, which includes aspects of a fulfilling life, such as housing, would correlate with the rental affordability index.

Beyond simple correlation, it was possible to create a logistic model, similar to the one used in the previous scenario, to predict the liveability score for the years 2019 to 2021.

Data Collection

The data is split into two types: streaming and static. The static dataset is simpler with respect to loading and retrieving data, as Fission simply uploads and downloads an existing file rather than using an API to retrieve the dataset. This would be considered the streaming data - which the liveability dataset is. Via an API provided by DataVic, the data flows into Elasticsearch via the API call. Since the dataset is updated yearly, there was no need to schedule any cron jobs to actively consume the data.

Model Analysis

Initially, a full model was fit with rental affordability and year as the predictors and liveability percentage as the response variable, including interaction terms. In the output, 'rai' is the rental affordability index, and year is the year it was recorded.

From the output, we see that the p-value for the interaction term is not significant at the 5% level. Thus, there is not enough evidence to suggest an interaction between the two terms. Then, a model is created to compare the ordering of variables. The first output's formula is `percentage ~ rai + year`, and the second model is `percentage ~ year + rai`.

Output 1:

OLS Regression Results

Dep. Variable:	percentage	R-squared:	0.555
Model:	OLS	Adj. R-squared:	0.220
Method:	Least Squares	F-statistic:	1.660
Date:	Tue, 21 May 2024	Prob (F-statistic):	0.311
Time:	20:39:16	Log-Likelihood:	-32.132
No. Observations:	8	AIC:	72.26
Df Residuals:	4	BIC:	72.58
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-4.098e+04	2.12e+05	-0.194	0.856	-6.28e+05	5.46e+05
rai	170.7480	2079.251	0.082	0.938	-5602.180	5943.676
year	20.7407	105.186	0.197	0.853	-271.301	312.783
rai:year	-0.0886	1.034	-0.086	0.936	-2.959	2.782

Omnibus:	3.335	Durbin-Watson:	3.243
Prob(Omnibus):	0.189	Jarque-Bera (JB):	0.942
Skew:	0.159	Prob(JB):	0.625
Kurtosis:	1.350	Cond. No.	6.13e+09

Figure 28: OLS Regression Results

Output 2:

OLS Regression Results

Dep. Variable:	percentage	R-squared:	0.554
Model:	OLS	Adj. R-squared:	0.375
Method:	Least Squares	F-statistic:	3.101
Date:	Tue, 21 May 2024	Prob (F-statistic):	0.133
Time:	20:39:17	Log-Likelihood:	-32.140
No. Observations:	8	AIC:	70.28
Df Residuals:	5	BIC:	70.52
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.288e+04	1.22e+04	-1.881	0.119	-5.42e+04	8391.750
rai	-7.5100	3.123	-2.405	0.061	-15.537	0.517
year	11.7424	6.174	1.902	0.116	-4.129	27.614

Omnibus:	3.064	Durbin-Watson:	3.226
Prob(Omnibus):	0.216	Jarque-Bera (JB):	0.915
Skew:	0.170	Prob(JB):	0.633
Kurtosis:	1.379	Cond. No.	4.08e+06

Figure 29: OLS Regression Results

From the output, if RAI is in the model, year is not statistically significant and can be removed. However, if year is in the model, RAI should be included, as the statistic is significant at the 7% level. Thus, we compare the AIC of model `percentage ~ year + rai` and `percentage ~ rai`. The results, stored in the frontend notebook for the analysis, show that the AIC for the model `percentage ~ year + rai` was the lowest. Thus, this model is selected for prediction.

For the prediction, the model gave the following output:

- 2019: 45.310173
- 2020: -160.736355
- 2021: -231.603578

Given the coefficients in output 2, we see that both the intercept variable and the RAI's coefficient are negative, explaining the negative results predicted.

Correlation Analysis

The correlation between RAI and liveability is as follows:

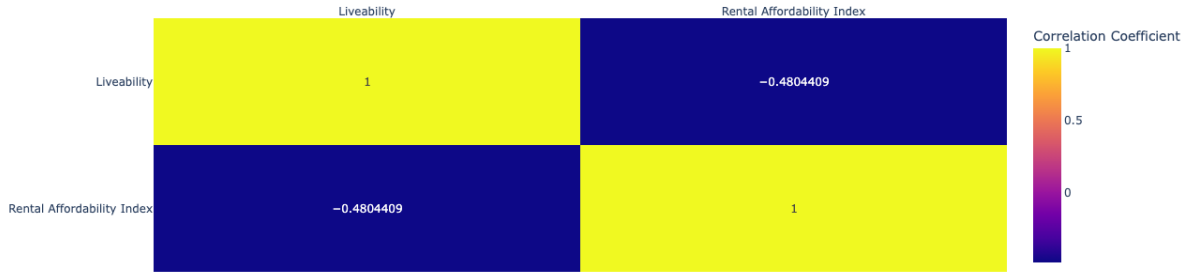


Figure 30: Correlation Heatmap Between RAI and Liveability

From the image, the correlation coefficient for RAI and liveability is -0.4804409. While not statistically significant, this suggests only a small amount of correlation between these two datasets.

Results

From both the heatmap and the modelling, the two datasets do not show enough correlation to draw any reasonable conclusions. This could be due to the oversimplification of the datasets, as more granular data was removed due to the grouping by year. Furthermore, all models used resulted in a large negative coefficient for the logistic regression model. This suggests that the model type used for the analysis of these two datasets may not be optimal, and a better model could be chosen for further analysis.

6.3 Scenario 3: Rental Affordability Index & Population

Overview

This analysis explores the relationship between the population of Melbourne and the rental affordability index over the years 2011 to 2021. Additionally, it examines the impact of the COVID-19 pandemic on the rental affordability index in Melbourne.

Data Collection

Data is retrieved from the `"regional_population_lga_2001-2021"` index using Elasticsearch for the Regional Population Dataset, and the `"rental_affordability"` index from Elasticsearch for the Rental Affordability Index Dataset.

Analysis

Correlation Analysis The correlation between the rental affordability index and population was calculated using the Pearson correlation coefficient, resulting in the following heatmap:

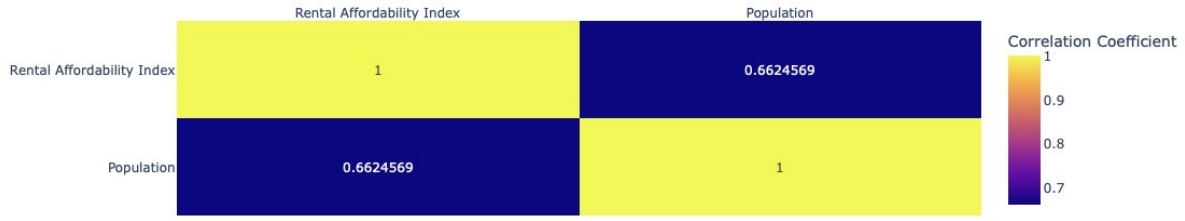


Figure 31: Heatmap of Correlation Between Rental Affordability and Population

The correlation coefficient between the rental affordability index and population is approximately 0.662, indicating a moderate positive correlation.

Trend Analysis We further analyzed the trend of the rental affordability index over the years. This trend was visualized using a line graph, highlighting the impact period of COVID-19 from 2019 to 2021 and annotating the percentage increase in the rental affordability index during this period.

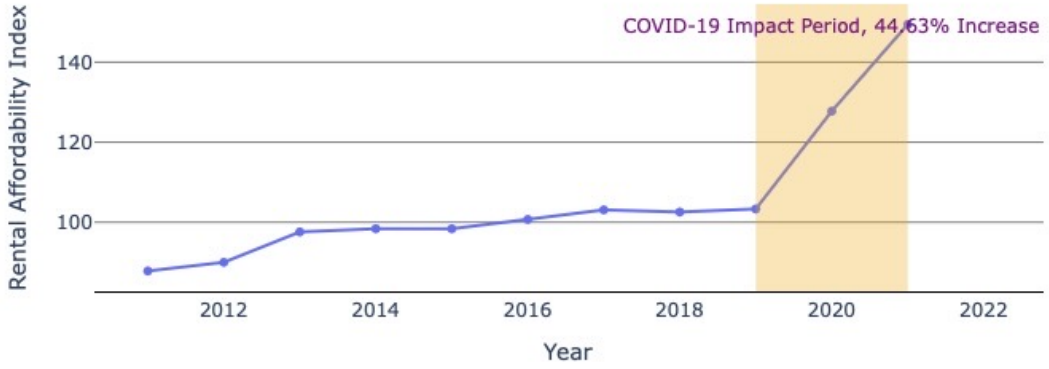


Figure 32: Rental Affordability Index Over the years

The plot shows a significant increase in the rental affordability index during the COVID-19 period (2019-2021). Specifically, there was a 44.63% increase from 2019 to 2021.

Results

Correlation Coefficient The heatmap revealed a moderate positive correlation (0.662) between the rental affordability index and population, indicating that as the population of Melbourne increases, the rental affordability index also tends to increase. This implies a moderately direct relationship between changes in population and changes in rental affordability in Melbourne.

Impact of COVID-19 The rental affordability index increased significantly during the COVID-19 period, with a 44.63% increase from 2019 to 2021. The COVID-19 pandemic had a substantial impact on the rental affordability index, causing a significant increase.

6.4 Scenario 4: Analysis of Liveability Percentage & Population in Melbourne

Overview

This analysis aims to determine whether there is a correlation between the liveability percentage and the population of Melbourne. By examining data from 2011 to 2021, we seek to understand the relationship between these two variables. Utilizing the Regional Population Dataset and the Liveability Index Dataset, we provide insights into how population changes might affect the overall liveability of Melbourne or if it is a valuable factor.

Data Collection

The population data and liveability data respectively were retrieved via the following indexes:

- `abs-regional_population_lga_2001-2021`.
- `liveability-index`.

Analysis

Correlation Calculation A correlation matrix was calculated to determine the relationship between the liveability percentage and population.

```
corr_matrix =
analysis_df[['liveability percentage', 'population']].corr()
```

Heatmap An interactive heatmap was created using Plotly to visualize the correlation between the two variables.

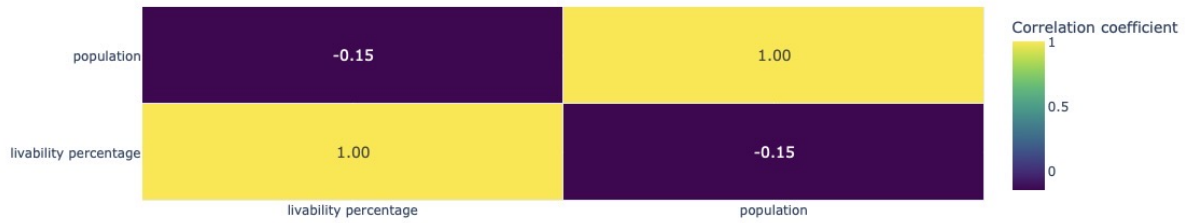


Figure 33: Interactive Correlation heatmap between Livability % and population

Results

There is a negative correlation (-0.15) between the liveability percentage and population in Melbourne over the period from 2011 to 2018. This analysis highlights a weak negative correlation between the liveability percentage and population in Melbourne. The findings suggest that as the population increases, the liveability percentage slightly decreases, but the relationship is not significant.

7 GUI Implementation

For the building permits dataset, after collecting the entire data from the Elasticsearch index, we used the `pgeocode` Python library to create a new pandas dataframe with just the postcodes extracted from the address in the original dataset. The `issue_date` field from the original dataset was also appended to the new dataframe. Then, the `streamlit` package was used to create a simple GUI, the task of which is to obtain the count of postcodes for each year separately and also an "All Years" option, which gets the count for all the years combined.

The GUI was titled "Building Permits Scatter Map Plot by Year" and has the selectbox functionality, which can be used to select a particular year or all years combined, and the GUI compiles and gives the output for each input selected. The output here is an interactive OpenStreetMap by Plotly.

For All Years

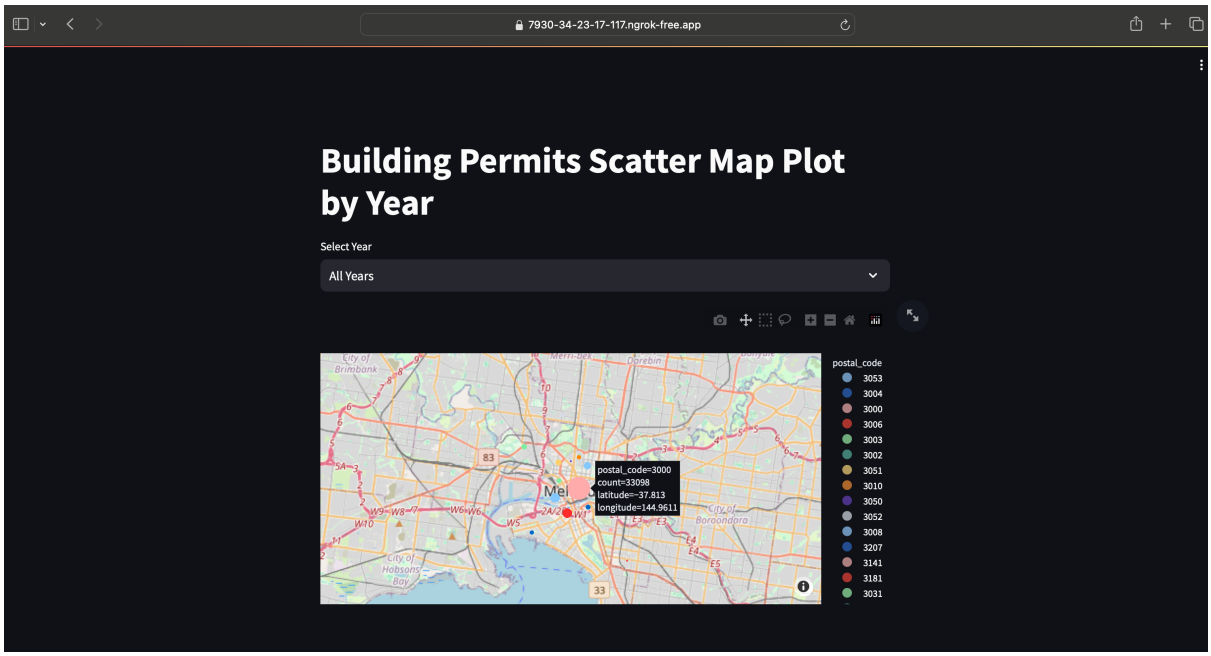


Figure 34: Openstreet map plot for All Years

For 2018 (just a random example)

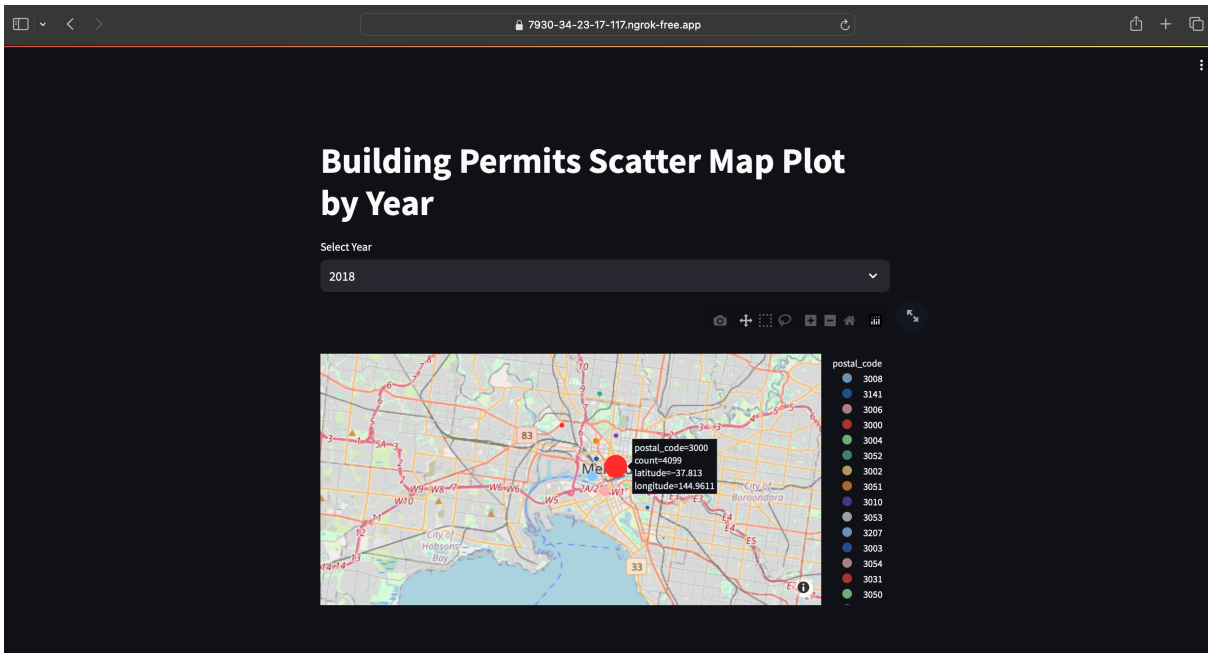


Figure 35: Openstreet map plot for 2018