Documentation

Data Model

`User-

```
{
  "_id": "jhadsgjsjdba",
  "firstName": "Harsh",
  "lastName": "Singh",
  "email": "harsh@gmail.com",
  "password": "sjdlkjsfd@",

"userType": "Admin",

"isVerified":false
}
```

Admin-

```
{
  "_id": "jhadsgjsjdba",
  "userId": "sdfsdf",
  "address": {
    "buildingNo": "55",
    "street": "kali gali",
    "landMark": "near tripathi bhawan",
    "city": "mirzapur",
    "state": "uttar pradesh",
    "pinCode": "831313"
  },
  "contactNo": "2525252525"
}
```

Comparison-

```
{
  "_id": "hjasghdakjshdkj",
  "userId": "hgfhzgvcx65c4xc", // _id of users collection
  "vehicleIds": [ "id1", "id2"  ], //  _id s of selected vehicles for comparison

"comparisonMatrixName": "Name1",

"createdDate": "date in java util form"
}
```

Customer-

```
{
  "_id": "jhadsgjsjdba",
```

```
"userId": "sdfsdf",
  "alternateEmail": "rohit2@gmail.com",
  "address": {
    "default": {
      "buildingNo": "55",
      "street": "kali gali",
      "landMark": "near tripathi bhawan",
      "city": "mirzapur",
      "state": "uttar pradesh",
      "pinCode": "831313"
    },
    "anotherAddress": {
      "buildingNo": "55",
      "street": "kali gali",
      "landMark": "near tripathi bhawan",
      "city": "mirzapur",
      "state": "uttar pradesh",
      "pinCode": "831313"
    }
  },
  "contactNo": "7898789878",
  "alternateContactNo": "78987898752",
  "wishlist": ["carid", "carid2"],

"isVerified" : true
}
```


Dealer-


```
{
  "_id": "dskfjkjs52225llk",

"userId": "sddfsdf",
  "organizationName": "Mahadev Motors",
  "description": "We are selling cars from 15 years.We deals in 10 diferent brands of cars.",
  "organizationEmail": "jayaadi@gmail.com",
  "buisnessContactWithStd": "065423536898",
  "pocs": [
    {
      "firstName": "Jay",
      "lastName": "aditya",
      "designation": "Sales Director",
      "email": "jayaadi@gmail.com",
      "alternateEmail": "aadijay@gmail.com",
      "contactNo": "9855585658",
      "alternateContactNo": "9865658959",
      "aadharNo": "52525252578787"
    },
    {

      "firstName": "Jay",
      "lastName": "aditya",
      "designation": "Sales Director",
```

```
      "email": "jayaadi@gmail.com",
      "alternateEmail": "aman1@gmail.com",
      "contactNo": "9898989898",
      "alternateContactNo": "7878787878",
      "aadharNo": "5252525257875"
    }
  ],
  "salesTaxNo": "sj5s5d2fgg5",
  "salesLiscenceNo": "jtind252dfd858",
  "experience": "15 years",
  "HeadOfficeAddress": {
    "buildingNo": "55",
    "street": "kali gali",
    "landMark": "near tripathi bhawan",
    "city": "mirzapur",
    "state": "uttar pradesh",
    "pinCode": "831313",
    "time": {
      "opening": "9:00AM",
      "closing": "5:00PM"
    }
  },
  "gstIN": "suthid2525di25",
  "salesPanNo": "BLIED52565",
  "location": {
    "latitude": "23.82128298959131",
    "longtitude": "79.43285514996526"
  },
  "sellingBrands": ["mahindra", "bmw", "audi"],
  "pictureUrls": ["link1", "link2"],
  "branches": [
    {
      "buildingNo": "58",
      "street": "durga gali",
      "landMark": "near railway station",
      "city": "luckhnow",
      "state": "uttar pradesh",

      "pinCode": "831313"
    },
    {
      "buildingNo": "51",
      "street": "hanuman gali",
      "landMark": "near chandni chowk",
      "city": "Delhi",
      "state": "Delhi",

      "pinCode": "831313"
    }
  ],
  "socialMedia": {
    "facebook": "",
    "twitter": "",
    "linkedin": ""
  },
  "typeOfCompany": "individual",
  "emiFacility": true,
  "awardsAndRecognition": [
    "Dealer of the Year 2020",
    "Most Selling for Tata 2020"
  ]
}
```


Feedback-


```
{
  "_id": "fksjhjhd",
  "vehicleId": "jyaguyagds4s5d6s45fs",  // _id of vehicles collection
  "userId": "748568erdsgf",  // _id of users collection
  "review": "This is a nice car",
  "rating": 4
}
```


Vehicle-

```
{
  "_id": "aksdsdad45da5a1sd2",
  "dealerId": "4sad46d64zc",
  "tankCapacity": "10L",
  "mediaInterface": "buttons",
  "theftAlarm": true,
  "pictureUrls": ["link1", "link2"],
  "tripMeter": "analog",
  "airBagCount": 5,
  "description": "This is a car",
  "price": 2700000.00,
  "color": "red",



  "vin": "12234",
  "brand": "Honda",
  "model": "C100",
  "year": 2010,
  "vehicleType": "SUV",
  "fuelType": ["diesel", "petrol"],
  "unitsFuelConsumption": "15km/l",
  "transmissionGearType": "auto",
  "steeringWheelType": "power",
  "vehicleSpeed": "100km/h",
  "horn": "air horn",
  "powerTrainTorque": "100Nm",
  "accelaration": "10m/s2",
  "nightMode": true,
  "isABS": true,
  "wheelRadius": "50cm",
  "wheelSpeed": "10km/h",
  "light": {
    "fog": true,
    "hazard": true,
    "parking": true,
    "dynamicHighBeam": true,
    "automaticHeadlights": true
  },
  "ignitionTime": "5s",
  "odometer": true,
  "washerFluid": true,
  "malfunctionIndicator": true,
  "batteryLevel": 40,
  "airConditioning": true,
  "languageConfiguration": ["English", "Hindi"],
  "mirror": "automatic",
  "childSafetyLock": true,
  "topSpeedLimit": "100km/h",
  "climate": {
    "climateControl": true,
    "sunroof": true,
    "defrost": true
  },
  "chime": true
}
```


sponsored_vehicles-


```
{
  "_id": "aksdsdad45da5a1sd2",

"vid": "6108b42865d0e776590cb015"

}
```


VehicleAccessory -


```
{
  "_id": "ak45sdsdad45da5a1sd2",

"name": "Car Duster",

"description": "Special wax-treated cotton strands.; Does not scratch the car's surface.; Lasts for years; Dirtier it gets the better it works",

"price": "699.00",

"pictureUrls": ["link1","link2"],
```

"dealerId": "12dsfsd3434",

"weight": "100g",

"brush-strands": "cotton",

}

Note: In VehicleAccessory, all attributes excluding "weight" and "brush-strands" are common for all the accessories. Additional attributes for an accessory can be added as shown in the example.

Naming Conventions for the project

Spring Boot Project

Follow the points given below in your project

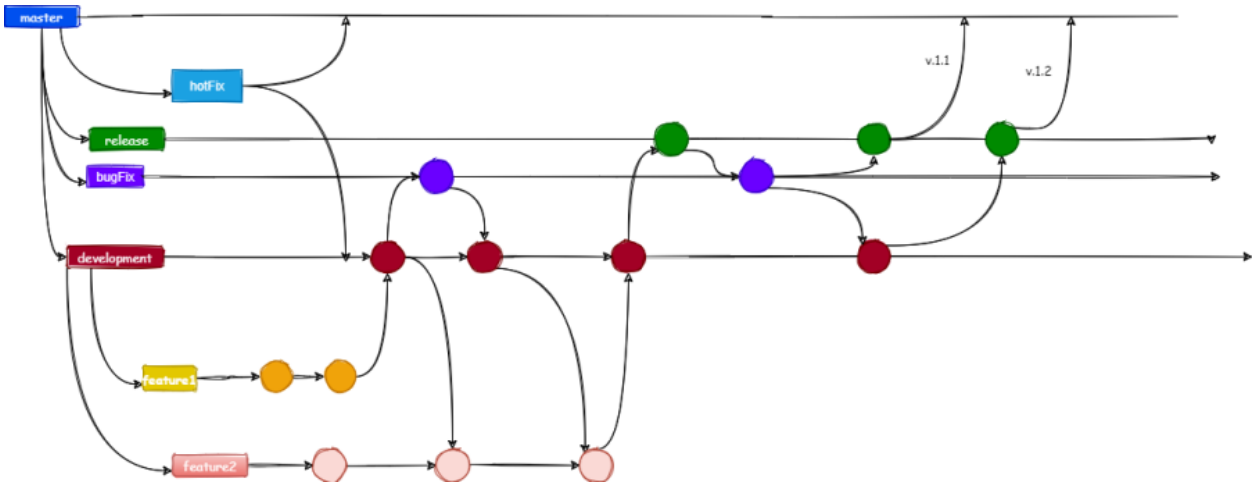
- A package should be named in lowercase characters. There should be only one English word after each dot.
    - com.sapient.asde.batch5.<artifact-id>
    - com.sapient.asde.batch5.<artifact-id>.entity.<EntityClassName>
    - com.sapient.asde.batch5.<artifact-id>.service.<ServiceClassName>
    - com.sapient.asde.batch5.<artifact-id>.controller.<ControllerClassName>
    - com.sapient.asde.batch5.<artifact-id>.repository.<RepositoryInterfaceName>
- All functions of Service should throw ServiceException.(ServiceException should be in service package)
- The name of classes should be in PascalCase.
- The function and variables should have camelCase names.
- The versions of project to be merged with development will end with -SNAPSHOT but the release versions will end with -RELEASE
- Class Names must be singular
- Table names/Collection names must be plural.
- application.yml should be used instead of application.properties

Reactjs Project

Follow the points given below in your Reactjs project

- **Extensions**: Use **.**js extension for React components.
- **Filename:** Use PascalCase for filenames. E.gReservationCard.js.
- **Reference Naming:** Use PascalCase for React components and camelCase for their instances.
- **Component Naming:** Use the filename as the component name. For example, ReservationCard.js should have a reference name of ReservationCard.
- **Folder Naming:** Use PascalCase for folders with the same name as components and keep component's css and test along with the component in the same folder.Example:
- **Function Naming:** Function names should be in camelCase and meaningful.

Branching Strategy (Product Phase)



Branch Description :-

- **master branch** - This is the final product branch.
- **hotfix branch** - If there is a bug in the master branch which needs to be fixed aggressively, then the Hotfix branch can be used for a quick fix in the master and merged back to the master and development branch.
- **release branch** - Development branch to be merged every week to release branch and CI/CD will take place weekly with the help of docker config and Jenkins.
- **bugfix branch** - If there is any bug in the release then we will pull the code to the Bugfix branch to work on the bug and after fixing it changes will be merged to release as well as development. Bug in the development will also be fixed here and after fixing it, It will be merged back to development. Branches will be deleted after merging.
- **development branch**- From this branch, every feature branch will be created, and commits , pushes will happen daily. from this release will happen weekly .
- **feature branch** - In this branch every individual will work on a feature and when completed they will pull , push and merge to development branch.
- feature, hotfix, and bugfix branches to be deleted after a successful merge.
- Delete Branches locally to ensure clean git working.

Note: Subject to change depending on requirements!

Frontend build tools

On the basis of Frontend being developed in **React Js**
Available tools

- Package Manager - npm

npm
npm is a package manager for the JavaScript programming language. It consists of a command line client, also called npm, and an online database of public and paid-for private packages, called the npm registry. When used as a dependency manager for a local project, npm can install, in one command, all the dependencies of a project through the package.json file. npm also provides version-bumping tools for developers to tag their packages with a particular version. npm also provides the package-lock.json file which has the entry of the exact version used by the project after evaluating semantic versioning in package.json.

Frontend Testing Tools

About Frontend Testing

- Front End Testing is a testing technique in which Graphical User Interface (GUI), functionality and usability of web applications or a software are tested.
- The goal of Front end testing is testing overall functionalities to ensure the presentation layer of web applications or a software is defect free with successive updates.

Why To Do Frontend Testing

- Through front end tests, development teams can pinpoint problems on the client-side to ensure they do not jeopardize critical workflows in the application.
- To verify the behavior of web applications on different operating systems, browsers, and devices. It helps validate the functioning and responsiveness of the application over different system architectures.
- To make sure a website or application renders the same across different devices and browser engines.

- To reduce the application load time, ensure the application's content is displayed correctly, and give the interface a unified look across different devices and browsers.
- To Ensure Seamless Integration of Third-Party Services for better user experience.

Frontend Testing Tools Considered

The following performance testing tools have been considered for this project:
1. Jest
Jest is a JavaScript testing framework maintained by Facebook, Inc. Jest was created with a focus on simplicity and support for large web applications. It works with projects using Babel, TypeScript, Node.js, React, Angular, Vue.js and Svelte. Jest does not require a lot of configuration for first time users of a testing framework.
2. Mocha
Mocha is a feature-rich JavaScript test framework running on Node.js featuring browser support, asynchronous testing, test coverage reports, and use of any assertion library. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.
3. Jasmine
Jasmine is a behavior-driven development framework for testing JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM. And it has a clean, obvious syntax so that can easily write tests. It also has the support for asynchronous testing.

Frontend Testing Tools Comparison

Metric	Jest	Mocha	Jasmine
Latest Release	v27.0.6 (28/06/2021)	v9.0.2 (03/07/2021)	v3.7.1 (19/03/2021)
Open Source	Yes	Yes	Yes
Inbuilt Assertions	Yes	No	Yes
Inbuilt Mocks	Yes	No	Yes
Inbuilt Spies	Yes	No	Yes
Inbuilt Stubs	Yes	No	Yes
Inbuilt Code Coverage	Yes	No	Yes
Community Support	Better	Poor	Good
Simplicity	Low	High	Low
Parallel Test Execution	Yes	No	No
Snapshot Testing	Yes	No	No

Finalized Frontend Testing Tool: Jest

The following reasons make Jest a suitable frontend testing tool for the project:

- More features compared to other frontend testing tools.
- Better community support and documentation.
- Jest along with Enzyme is a great fit for testing React Applications.

Frontend Development Tools

Frontend development is a part of web development that focuses on what users see at their end. It refers to converting the data coming from backend to graphical interface with the help of HTML, CSS and Javascript.

TOOLS AVAILABLE

- HTML
- CSS
- JAVASCRIPT
- BOOTSTRAP
- SASS

FRAMEWORKS AND LIBRARIES

ANGULAR JS

AngularJS is an open-source JavaScript framework used to build a dynamic web application Developed in 2009, it is now maintained by Google. It is based on HTML and JavaScript and mostly used for building a Single Page Application.It can be included on an HTML page with a <script> tag. It extends HTML by adding built-in attributes with the directive and binds data to HTML with Expressions.

PROS OF ANGULAR JS

- AngularJS provides an efficient testable framework.
- AngularJS has vast libraries.
- AngularJS also has a spectacular UI design.
- AngularJS also provides an automatic Data Synchronization system between the components and model view.
- AngularJS also helps in data binding.
- Simple routing is possible using AngularJS.
- AngularJS allows the users to easily create the Customized Document Object Model.

CONS OF ANGULAR JS

- Angular JS has performance issues with DOM
- Angular JS offers limited routing.
- Angular JS is not fast with pages embedding interactive components.
- Angular JS has a very complex third party integration.
- Learning Angular JS is time consuming.

REACT JS

React JS is a JavaScript library rather than a simple framework. React JS was developed and maintained by *Facebook* and a community of individual developers and corporates. It is mainly used to create Interactive User Interfaces with an absolute focus on rendering performance. React JS is more reliant on 'V' in Model View Controller (MVC) architecture

PROS OF REACT JS

- React JS provides its users with faster updates.
- It is relatively very easy to import components in React JS
- We can reuse the code with the help of React JS.
- React JS provides a smooth, JavaScript debugging system.
- React JS also provides smooth interface designs and easy learning APIs.
- React JS has an architecture completely based on components.

CONS OF REACT JS

- React JS is just a library, not a framework.
- Some configurations will be required if we integrate React JS into a Model View Controller (MVC) framework.
- Learning React JS is easy to learn as it is just a library and not a framework.

REACT JS	ANGULAR JS	JAVA SERVER PAGES(JSP)
• It is a just a Library	• It is a full fledged Framework	• It is a technology based on servlet container
• It uses JSX which combines logic and markup in a single file(making components easier to read).	• The rendered HTML and Javascript maintains a physical separation	• JSP is Java inside HTML
• App size is smaller than angular so it is a bit fast	• App size is greater than react so longer load times and hence poor performance as compared to react.	• It is slow as need to be compiled into servlet before running on a server.
• Learning curve is smaller but you only get a View so you have to configure with other 3rd party libraries.	• Learning Curve is higher, but once you understand it, you have an entire MVC framework	• Easy to learn as it is just an HTML embedded inside Java
• It has a virtual DOM which render updates faster than regular DOM	• It has just the regular DOM.	• It needs a Tomcat server to run
• It uses Javascript	• It uses Typescript	• It used Java and HTML

• Testing - Jest and Enzyme	• Testing - Jasmine and Mocha	• JSP gets compiled into servlets, then can be tested using Junit or Mockito

Finalized Frontend Development Tool: ReactJS

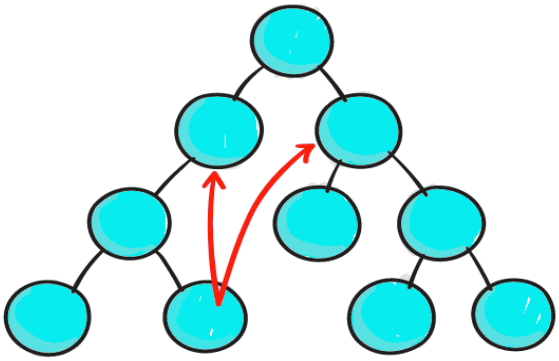
AngularJS is a fully-featured MVC framework with a vibrant and large supporting community. Whereas using React JS doesn't require much code writing and perform more. Moreover, React JS is better than AngularJS when it comes to performance. JSP is not good for making large scale projects as firstly it is very slow(first it needs to be compiled into a servlet then only it can run on a server) and secondly it has only limited features . Considering all the facts and their practical applications required for our use case, **ReactJS** is the perfect choice at this point of time.

Frontend Framework

REDUX

Why ?

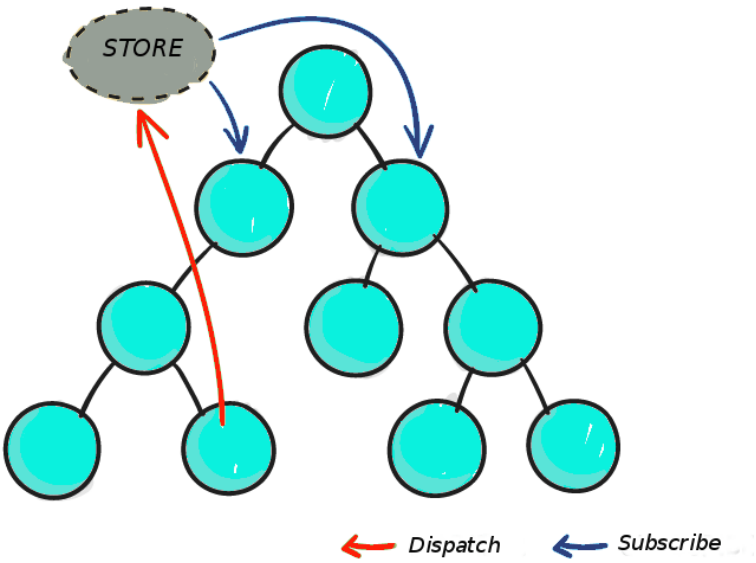
Let's consider why you might need a state management tool. Most libraries like React, Angular, etc. are built with a way for components to internally manage their state without any need for an external library or tool. It does well for applications with few components but as the application grows bigger, managing states shared across components becomes a chore. In an app where data is shared among components, it might be confusing to actually know where a state should live. Ideally, the data in a component should live in just one component. So sharing data among sibling components becomes difficult. For instance, in React, to share data among siblings, a state has to live in the parent component. A method for updating this state is provided by this parent component and passed as props to these sibling components.



Now imagine what happens when a state has to be shared between components that are far apart in the component tree. The state has to be passed from one component to another until it gets to where it is needed. Basically, the state will have to be lifted up to the nearest parent component and to the next until it gets to an ancestor that is common to both components that need the state and then it is passed down. This makes the state difficult to maintain and less predictable. It also means passing data to components that do not need such data. It's clear that state management gets messy as the app gets complex. This is why you need a state management tool that makes it easier to maintain these states.

What is Redux?

Simply put, Redux is a state management tool. While it's mostly used with React, it can be used with any other JavaScript framework or library. It is lightweight at 2KB (including dependencies), so you don't have to worry about it making your application's asset size bigger.



With Redux, the state of your application is kept in a store and each component can access any state that it needs from this store.

How Redux works

The way Redux works is simple. There is a central store that holds the entire state of the application. Each component can access the stored state without having to send down props from one component to another. There are three building parts: **actions**, **store** and **reducers**.

Actions in Redux

Simply put, **actions** are events. They are the only way you can send data from your application to your Redux store. Actions are sent using store.dispatch() method. Actions are plain JavaScript objects and they must have a type property to indicate the type of action to be carried out. They must also have a payload that contains the information that should be worked on by the action. Actions are created via an action creator.
Example:

```
const action-creator = (data1,data2) =>
{
return {
type: "action-type",
payload: {
data1: "foo",
data2: "bar"
}
}
}
```

Reducers in Redux

Reducers are pure functions that take the current state of an application, perform an action and returns a new state. These states are stored as objects and they specify how the state of an application changes in response to an action sent to the store.

```
const reducer-function = (state = initialState, action) =>
{
switch (action.type) {
case "action-type":
return //to do
default: return state } };
```

Store in Redux

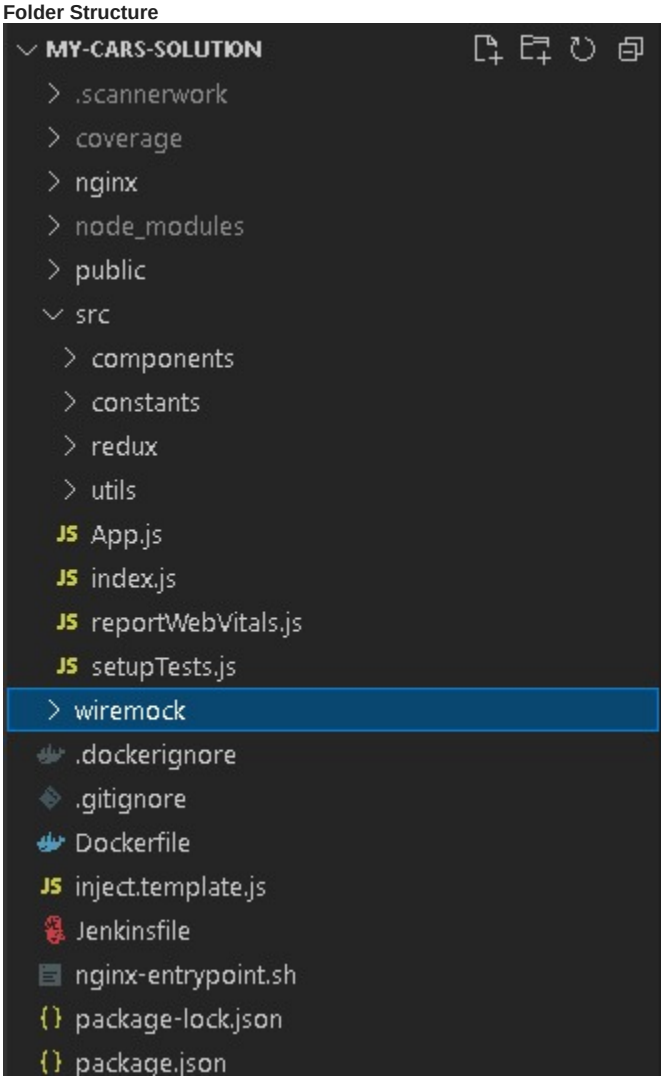
The store holds the application state. There is only one store in any Redux application. You can access the state stored, update the state, and register or unregister listeners via helper methods. Let's create a store for our login app: const store = createStore(rootReducer); Actions performed on the state always returns a new state. Thus, the state is very easy and predictable. **When** We have discussed major features of Redux and why Redux is beneficial to your app. While Redux has its benefits, it that does not mean you should go about adding Redux in all of your apps. Your application might still work well without Redux if any of these scenarios are true for you:

- If your app is going to consist of mostly simple actions such as UI changes, these don't really have to be a part of the Redux store and can be handled at the component level.
- You don't need to manage server side events (SSE) or websockets.
- You fetch data from a single data source per view

Guidelines for implementing Redux

- Single source of truth. The state of your whole application is stored in an object tree within a single store. ...
- State is read-only. The only way to change the state is to emit an action, an object describing what happened. ...
- Changes are made with pure functions.

Folder Structure



Using the framework

Steps to setup the react app

1. Clone the repository.
2. Ensure their is no package-lock.json .
3. Then run "npm install".
4. Project is ready.
5. Different Scripts are available to run, build and test the code.

Backend Build Tools

We have to choose between a **Maven** and a **Gradle** Project.

- Gradle and Maven are both tools to automate the build of your application. We tell it what dependencies we need for our project, such as Spring and it then bundles everything together (builds it).

Both Gradle and Maven provide convention over configuration. However, Maven provides a very rigid model that makes customization tedious and sometimes impossible.Both Gradle and Maven employ some form of parallel project building and parallel dependency resolution. The biggest differences are Gradle's mechanisms for work avoidance and incrementality

- Spring Boot looks at our application, makes checks, assumptions and provides standard configurations to make our app run. For instance, it has a server-functionalities embedded, so you don't have to worry about it. Gradle then takes all of that and builds a java application out of it.

Basis	Gradle	Maven
Based on	Gradle is based on developing domain-specific language projects.	Maven is based on developing pure Java language-based software.
Configuration	It uses a Groovy-based Domain-specific language for creating project structure.	It uses Extensible Markup Language(XML) for creating project structure.
Focuses on	Developing applications by adding new features to them.	Developing applications in a given time limit.
Performance	It performs better than maven as it optimized for tracking only current running task.	It does not create local temporary files during software creation hence uses more time.
Java Compilation	It avoids compilation.	It is necessary to compile.
Customization	This tool is highly customizable as it supports a variety of IDE's.	This tool serves a limited amount of developers and is not that customizable.

We have decided to go with Maven considering all above points which is better suited for our backend Java Springboot needs.

Backend Unit Testing Tools

What is Unit testing?

Unit testing is a software testing technique in which individual components/parts of the software is tested, i.e., a group of computer programs, usage procedure, etc. Unit testing of an object is done during the development of an application or project. The aim of unit testing is to isolate a segment of code (unit) and verifies its correctness. A unit is referred to as an individual function or procedure (program). The developers usually perform it during testing.

Why To Do Backend Unit Testing

- Inappropriate unit testing leads to high cost defect fixing during System Testing, Integration Testing after application is built.
- Unit tests help to fix bugs early in the development cycle and save costs.
- It helps the developers to understand the testing code base and enables them to make changes quickly.
- Due to the modular nature of the unit testing, we can test parts of the project without waiting for others to be completed.

Backend Unit Testing Tools Considered

The following performance testing tools have been considered for this project:
1. JUnit
JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit. JUnit is linked as a JAR at compile-time.
2. TestNG
TestNG is a testing framework for the Java programming language inspired by JUnit and NUnit. The design goal of TestNG is to cover a wider range of test categories: unit, functional, end-to-end, integration, etc., with more powerful and easy-to-use functionalities.
3. Spock
Spock is a testing and specification framework for Java and Groovy applications. What makes it stand out from the crowd is its beautiful and highly expressive specification language. Spock is compatible with most IDEs, build tools, and continuous integration servers. Spock is inspired from a lot of frameworks including JUnit, jMock, RSpec, Groovy, Scala.

Backend Unit Testing Tools Comparison

Metric	JUnit	TestNG	Spock
Latest Release	v5.7.2 (15/05/2021)	v7.4.0 (27/02/2021)	v2.0 (17/05/2021)
Language	Java	Java	Java
Open Source	Yes	Yes	Yes
Community Support	Better	Good	Poor
Client Side Testing	Yes	Yes	Yes
Server Side Testing	Yes	Yes	Yes
Fixtures*	Yes	Yes	Yes
Group Fixtures*	Yes	Yes	Yes
Generators*	Yes (JUnit - QuickCheck)	No	No
Mocks*	Using 3rd party libraries	Using 3rd party libraries	Inbuilt
Test Suite	Yes	Yes	Yes
Ignore Test	Yes	Yes	Yes
Exception Test	Yes	Yes	Yes
Timeout Test	Yes	Yes	Yes
Parameterized Test	Yes	Yes	Yes
Dependency Test	No	Yes	No
Group Test	No	Yes	No
Inbuilt Parallel Execution	No	Yes	No
Exporting Reports	Using Plugins	Inbuilt	Using Plugins

Finalized Backend Unit Testing Tool: JUnit

The following reasons make JUnit a suitable backend unit testing tool for the project:

- Features JUnit is an open source framework which is used for writing & running tests.
- Provides Annotation to identify the test methods.
- Provides Assertions for testing expected results.
- Provides Test runners for running tests.

What is Mocking?

Mocking is a process of developing the objects that act as the **mock** or **clone** of the real objects. In other words, mocking is a testing technique where mock objects are used instead of real objects for testing purposes. Mock objects provide a specific (dummy) output for a particular (dummy) input passed to it.

Need for mocking

Before using the Mocking technique, we should know the reasons for using mocking, which are as follows:

- If we want to test a component that depends on the other component, but it is under development. It generally uses when working in a team and parts are divided between several team-mates. In this case, mocking plays an essential role in the testing of that component. Without mocking, we need to wait for the completion of the required elements for testing.

- If the real components perform slow operations while dealing with database connections or another complex read/ write operation. Sometimes the database queries can take 10, 20, or more seconds to execute. In such cases, we require mock objects to perform testing, and it can be done via mocking.
- If there is an infrastructure concern that makes the testing impossible. It is very similar to the first case. For example, when we create a connection to the database, some issues related to configurations occur. It requires mocking for creating mock components to provide unit testing.

What is Mockito?

- Mockito is a Java-based mocking framework used for unit testing of Java application. Mockito plays a crucial role in developing testable applications.It internally uses the Java Reflection API to generate mock objects for a specific interface. Mock objects are referred to as the dummy or proxy objects used for actual implementations.
- The main purpose of using the Mockito framework is to simplify the development of a test by mocking external dependencies and use them in the test code. As a result, it provides a simpler test code that is easier to read, understand, and modify. We can also use Mockito with other testing frameworks like **JUnit** and **TestNG**.

To write unit tests, add the following dependencies to POM:
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
Sample test:

```
@SpringBootTest
utoConfigureMockMvc
public class AdvancedSearchVehicleAttributeControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @MockBean
    @Qualifier("pjmb1529_advancedsearchvehicleattributeserviceimpl")
    private AdvancedSearchVehicleAttributeSerice service;

    @Test
    void shouldReturnOk_whenAttributesAreRequested() throws Exception {
        JsonObject jb = new JsonObject("{\"attribute\": \"brand\"}");
        Mockito.when(service.getAdvancedSearchAttributes()).thenReturn(Arrays.asList(jb));

        String url = "/api/vehicles/search-attributes";

        mockMvc.perform(get(url)).andExpect(status().isOk());
    }
}
```

API Documentation

Available tools:

- SpringFox Swagger
- Spring REST docs

SpringFox Swagger

Swagger UI allows anyone to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from the OpenAPI (formerly known as Swagger) Specification.

Spring REST docs

It aims to produce accurate and readable documentation for our RESTful services. Spring REST Docs uses Asciidoctor by default. Asciidoctor processes plain text and produces HTML. Spring REST Docs uses snippets produced by tests written with Spring MVC's test framework. This test-driven approach helps to guarantee the accuracy of our service's documentation. If a snippet is incorrect, the test that produces it fails.

Discussion

Criteria	Swagger	REST Docs	Comments
Use	Visual and interactive API documentation	Textual API Documentation	Rest Docs don't provide interactive tools to modify and try out requests
Approach	detection from main files	Test Driven Documentation	Documentation is written in the test code so 'REST docs' does not overload main code with lots of annotations and descriptions
Ease	lots of annotations in the main code	It also requires more work	

There are pros and cons for both the tools. They can be used together as well for added functionality. But we don't need both for this project.
So we choose Swagger. We can also use it for the simple testing of an API on the running application besides the documentation.

Setting Up Swagger

Adding dependency

We will use the Springfox implementation of the Swagger specification. We are going to use **version 2.7.0** for this project. To add it to our Maven project, we need the following dependencies in the *pom.xml* file:

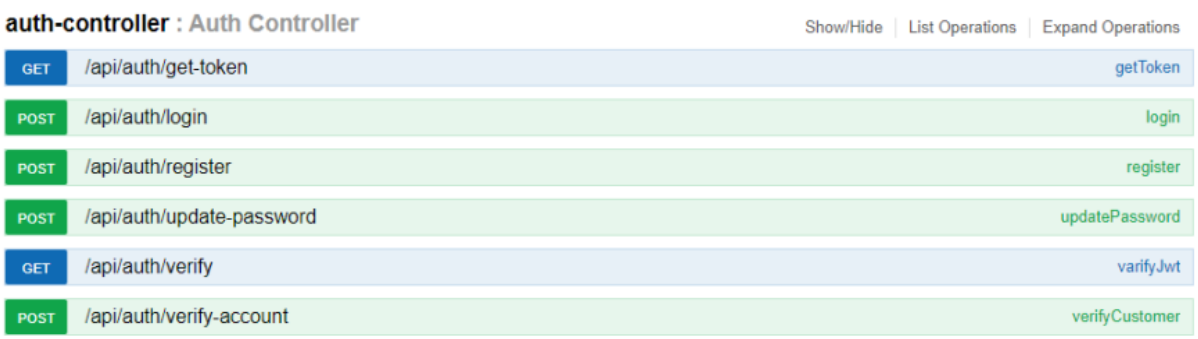
```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.7.0</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.7.0</version>
</dependency>
```

Java Configuration

Add the annotation @EnableSwagger2 (from springfox.documentation.swagger2.annotations.EnableSwagger2) to the main class of Application.
optional - add the following Bean

```
@Configuration
@EnableSwagger2
public class SwaggerConfiguration {
   @Bean
   public Docket docket() {
      return new Docket(DocumentationType.SWAGGER_2).select().apis(RequestHandlerSelectors.basePackage("com.sapient"))
            .build();
   }
}
```

Auth Service



Customer Service

customer-controller : Customer Controller — Show/Hide | List Operations | Expand Operations

GET	/api/customers	getCustomerByUserId
PUT	/api/customers	updateCustomerById
POST	/api/customers/add-customer/	addCustomer
POST	/api/customers/change-password	changePassword
POST	/api/customers/reset-password	resetPassword
GET	/api/customers/reset-password-link	sendResetPasswordLink

favorite-vehicle-controller : Favorite Vehicle Controller — Show/Hide | List Operations | Expand Operations

GET	/api/customers/favorites	getCustomerWishlist

update-wish-list-controller : Update Wish List Controller — Show/Hide | List Operations | Expand Operations

PUT	/api/customers/favorites/{vehicleId}	updateCustomerWishList

user-controller : User Controller — Show/Hide | List Operations | Expand Operations

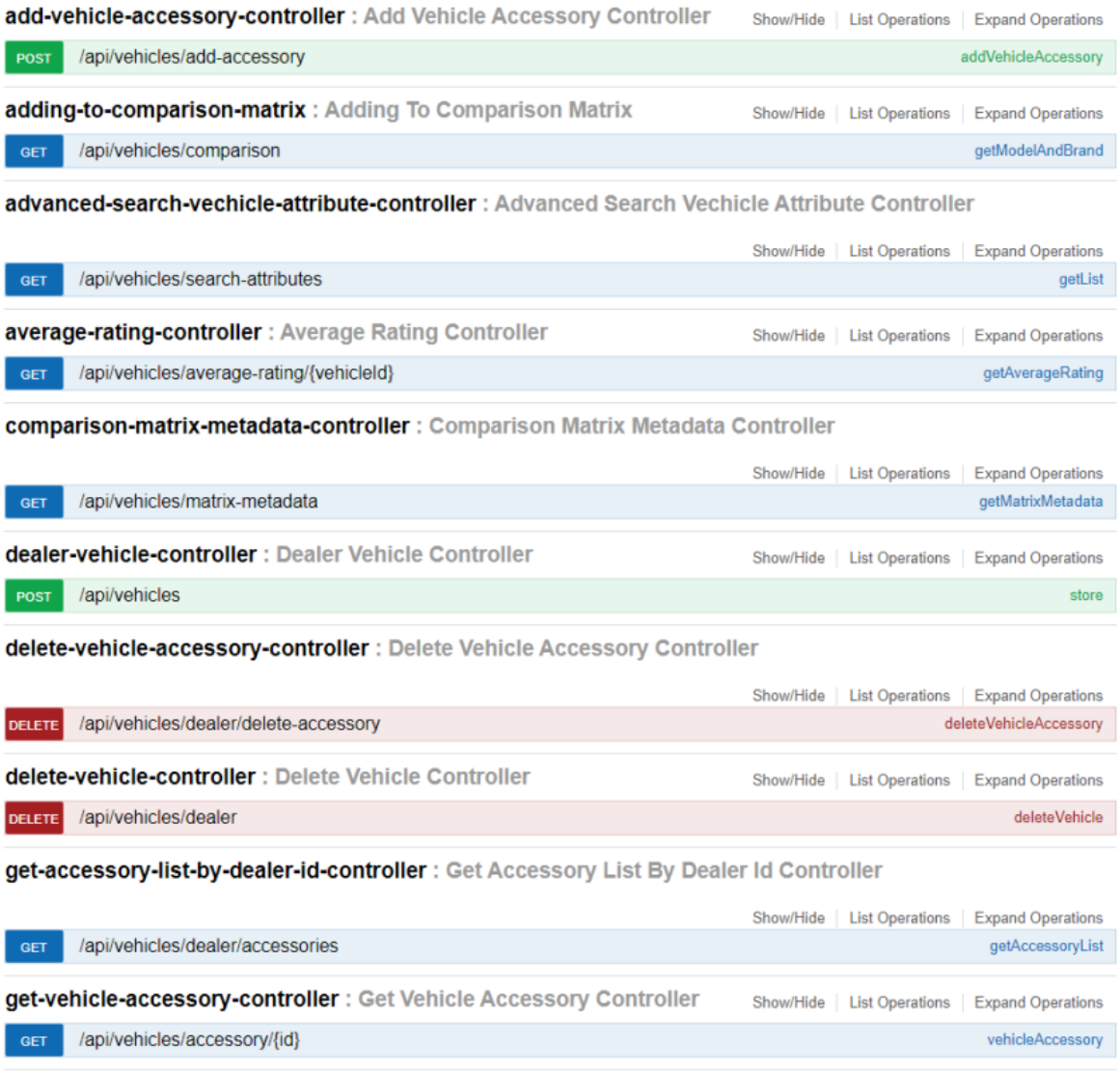
GET	/api/users	nameOfUser

Email Service

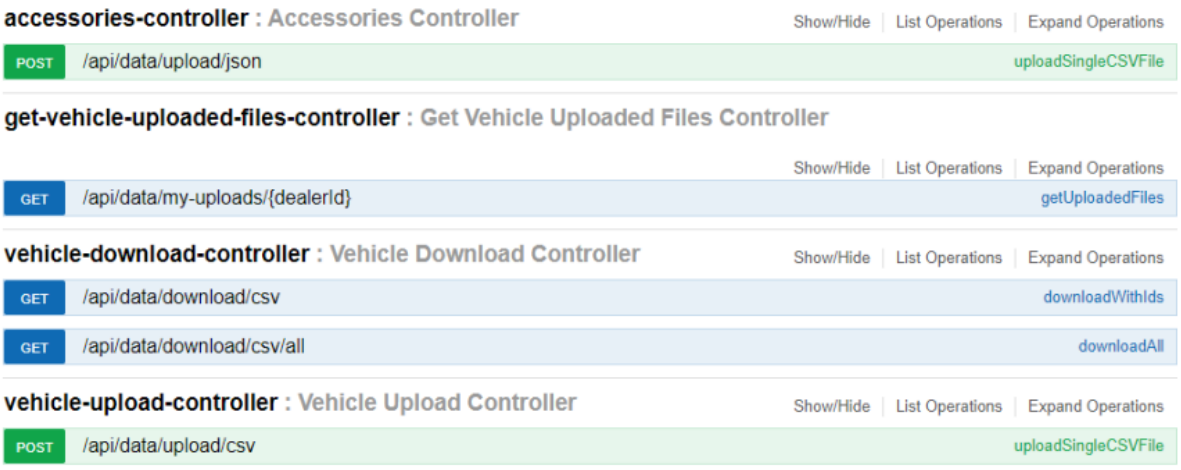
simple-mail-controller : Simple Mail Controller — Show/Hide | List Operations | Expand Operations

POST	/api/mail	sendMail

Vehicle Service

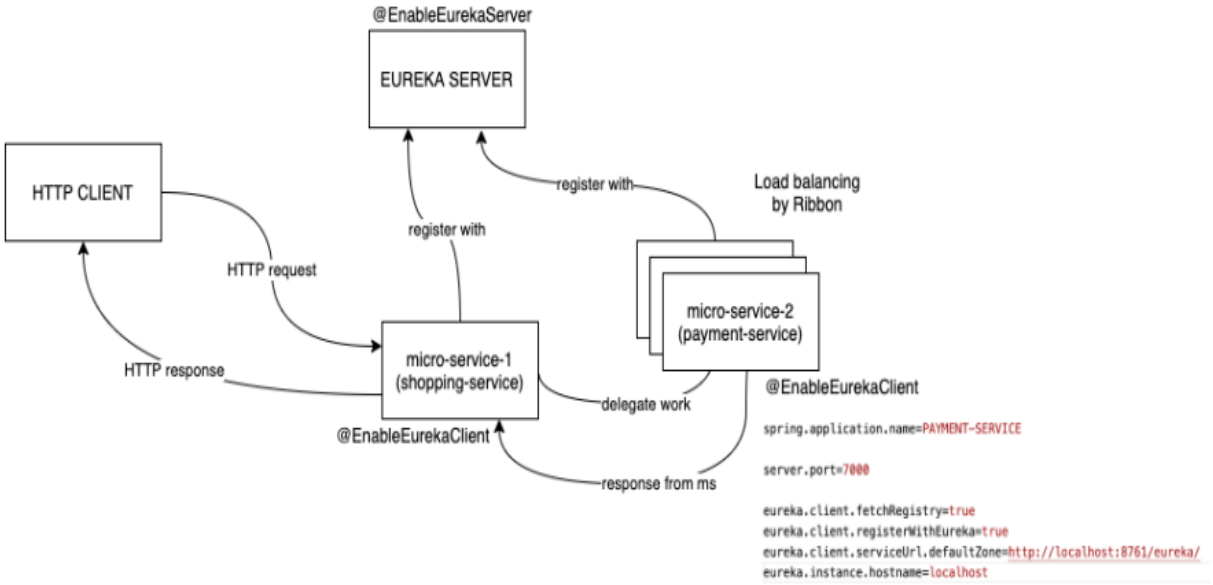


Vehicle Data Service



Eureka

An application that holds information about all client-service applications. Every micro service
Will register itself with the eureka server and the eureka server knows all the micro services running on specific ip-address and port number. Eureka server is also known as discovery server.



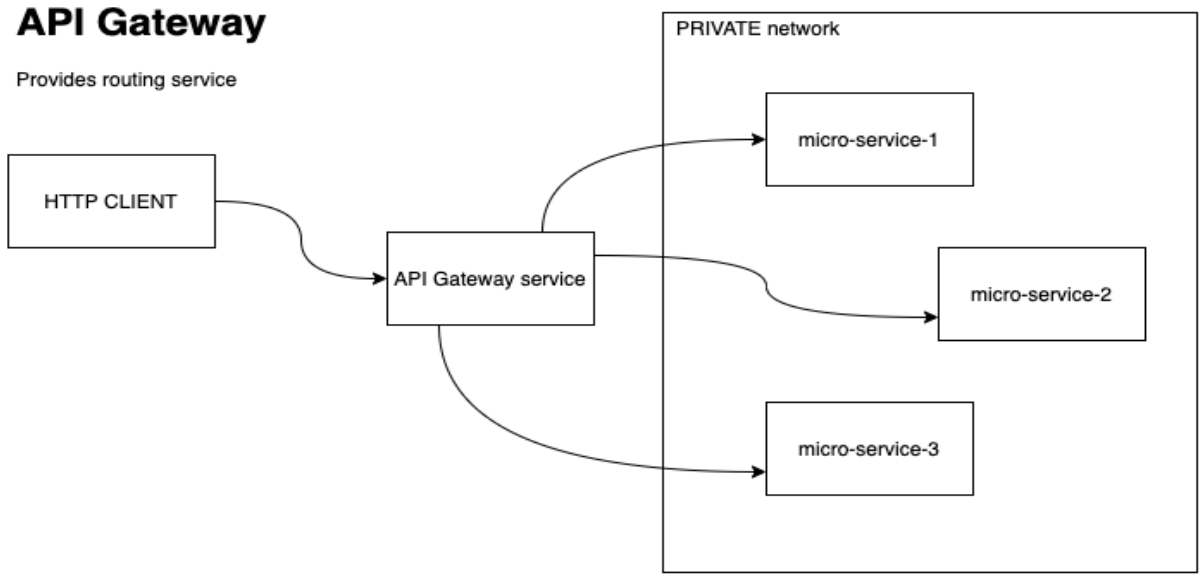
Java Configuration
Add the annotation @EnableEurekaServer to the main class of Application.

```java
@SpringBootApplication
@EnableEurekaServer
public class EurekaServiceDemoApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(EurekaServiceDemoApplication.class, args);
    }

}
```

Adding dependency
To add it to our Maven project, we need the following dependencies in the *pom.xml* file:

```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

Gateway
This project provides a library for building an API Gateway on top of Spring WebFlux. Spring Cloud Gateway aims to provide a simple, yet effective way to route to APIs and provide cross cutting concerns to them such as: security, monitoring/metrics, and resiliency.



Java Configuration
Add the annotation @EnableEurekaClient to the main class of Application.

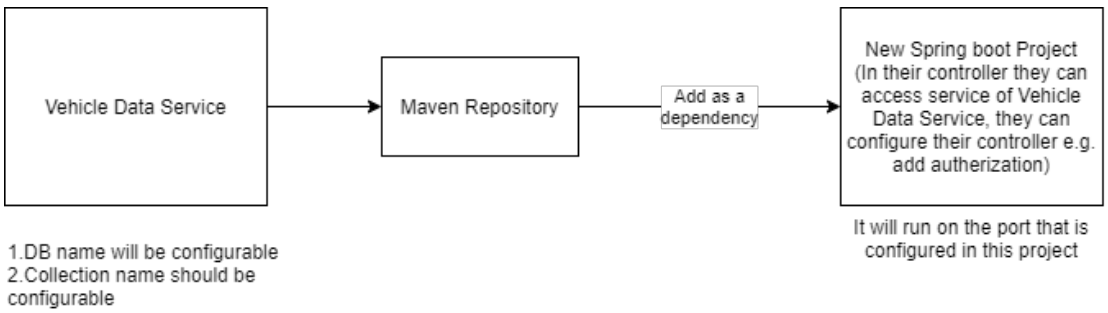
```
@SpringBootApplication
@EnableEurekaClient
public class GatewayServiceApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(GatewayServiceApplication.class, args);
    }

}
```

Adding dependency
To add it to our Maven project, we need the following dependencies in the *pom.xml* file:

```
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
    </dependency>
```

Import/Export by Vehicle Data Service

Architecture

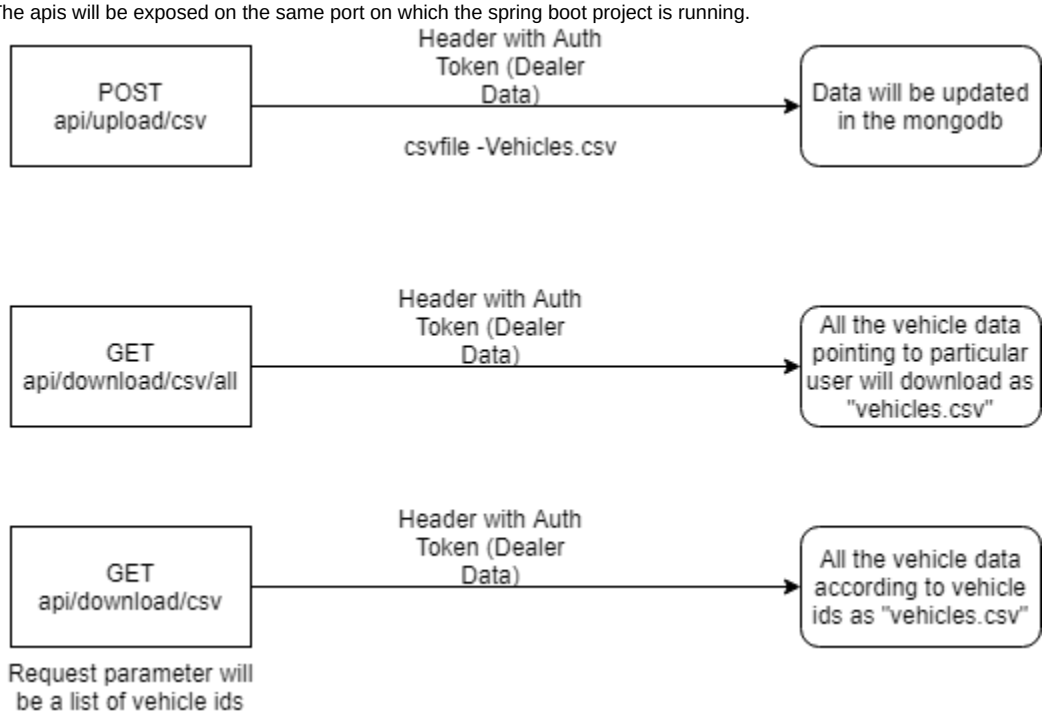


1.DB name will be configurable
2.Collection name should be configurable

We will make a spring boot project that can be added as a dependency in any other spring boot project. Which will expose 3 REST endpoints-

1. Import from csv
2. Export all data to csv for a particular dealer
3. Export data to csv for list of vehicle ids

The apis will be exposed on the same port on which the spring boot project is running.



DB name and collection name can be configured through application.properties.
Add the dependency in pom.xml and add @ComponentScan(basePackages = { "com.sapient.asde.batch5.vehicledataservice.controller" }) in the Application file.

Vehicle Data

_id	aksdsdad45da5a1sd2
dealerId	4sad46d64zc
tankCapacity	10L
mediaInterface	buttons
theftAlarm	true
tripMeter	analog
airBagCount	5
description	This is a car
price	2700000
color	red
vin	12234
brand	Honda
model	C100
year	2010
vehicleType	SUV
unitsFuelConsumption	15km/l
transmissionGearType	auto
steeringWheelType	power
vehicleSpeed	100km/h
horn	air horn
powerTrainTorque	100Nm
accelaration	10m/s2
nightMode	true
isABS	true
wheelRadius	50cm
wheelSpeed	10km/h
ignitionTime	5s
odometer	true
washerFluid	true
malfunctionIndicator	true
batteryLevel	40
airConditioning	true
languageConfiguration	English, Hindi
mirror	automatic
childSafetyLock	true
topSpeedLimit	100km/h
chime	true

pictureUrls[0]

link1

pictureUrls[1]

link2

fuelType[0]

diesel

fuelType[1]

petrol

light

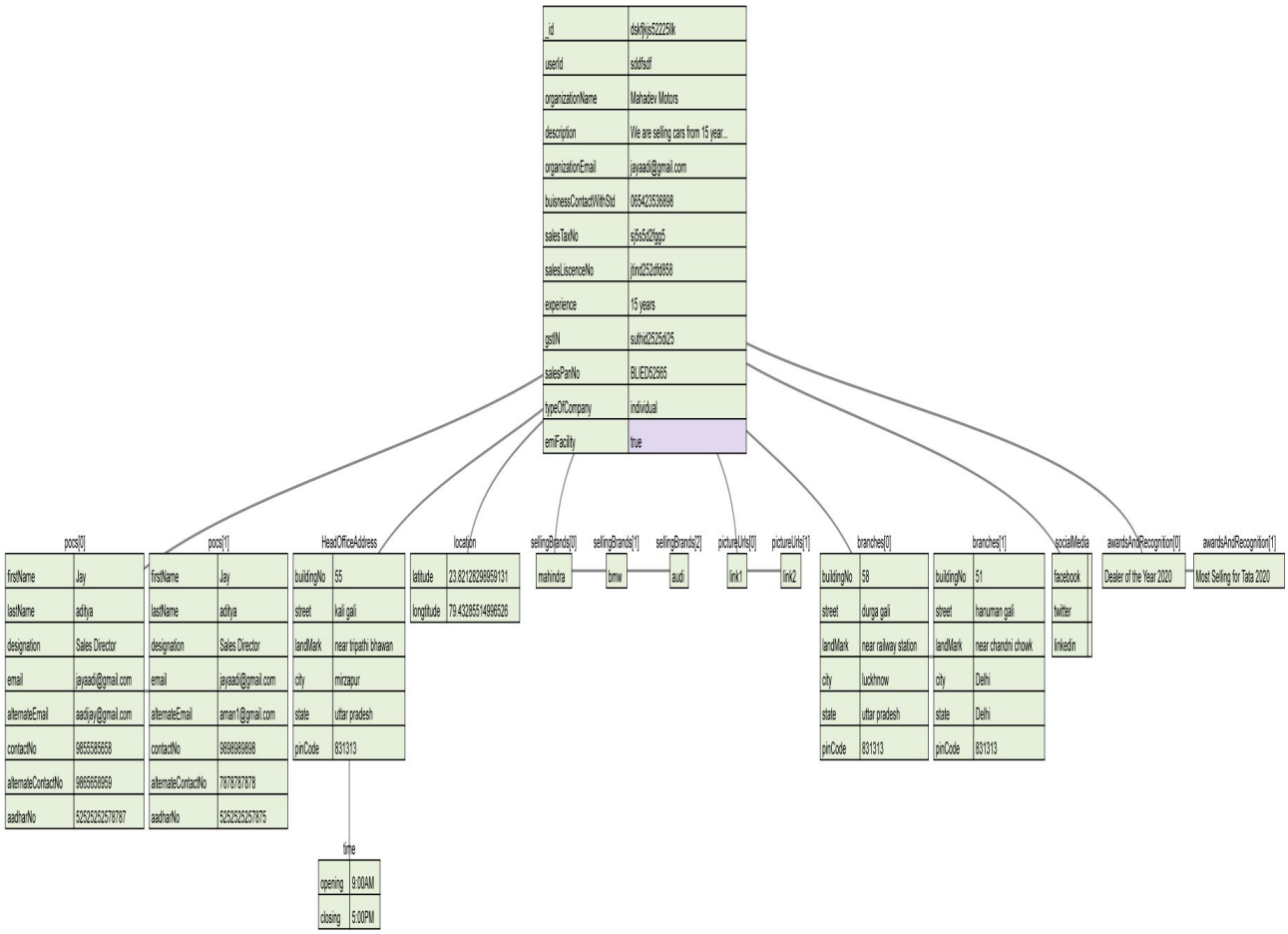
fog	true
hazard	true
parking	true
dynamicHighBeam	true

climate

climateControl	true
sunroof	true
defrost	true

automaticHeadlights	true

Dealer Data:



Green - String
Violet - Boolean
Light Green - Numberh2.Vehicle Data in JSON Format


```
{
   "_id": {
      "$oid": "610d13ba774d3042d588dca1"
   },
   "dealerId": "abc",
   "tankCapacity": "28 L",
   "mediaInterface": "button",
   "theftAlarm": false,
   "pictureUrls":

[
      "https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/2019-honda-civic-sedan-1558453497.jpg?crop=1xw:0.9997727789138833xh;
center,top&resize=480:*",
   ],
   "tripMeter": "display",
   "airBagCount": 5,
   "description": "This is a car",
```

```
  "price": 492036.27,
  "color": "blue",
  "vin": "610b67b4ea45d242e603a1bb",
  "brand": "Audi",
  "model": "r100",
  "year": 2000,
  "vehicleType": "prius",
  "fuelType":

[

  "diesel",
     "petrol"
  ],
  "unitsFuelConsumption": "27 km/l",
  "transmissionGearType": "manual",
  "steeringWheelType": "power",
  "vehicleSpeed": "220 km/h",
  "horn": "air horn",
  "powerTrainTorque": "100Nm",
  "accelaration": "10m/s2",
  "nightMode": false,
  "isABS": false,
  "wheelRadius": "50cm",
  "wheelSpeed": "10km/h",
  "light": {
     "fog": false,
     "hazard": true,
     "parking": true,
     "dynamicHighBeam": false,
     "automaticHeadlights": true
  },
  "ignitionTime": "5s",
  "odometer": false,
  "washerFluid": true,
  "malfunctionIndicator": true,
  "batteryLevel": 48,
  "airConditioning": false,
  "languageConfiguration": "English, Hindi",
  "mirror": "auto",
  "childSafetyLock": false,
  "topSpeedLimit": "160 km/h",
  "climate": {
     "climateControl": false,
     "sunroof": true,
     "defrost": false
  },
  "chime": true
}
```


Dealer Data in JSON Format -


```
{
  "_id": "dskfjkjs52225llk",
  "userId": "sddfsdf",
  "organizationName": "Mahadev Motors",
  "description": "We are selling cars from 15 years.We deals in 10 diferent brands of cars.",
  "organizationEmail": "jayaadi@gmail.com",
  "buisnessContactWithStd": "065423536898",
  "pocs":

[     {
        "firstName": "Jay",
        "lastName": "aditya",
        "designation": "Sales Director",
        "email": "jayaadi@gmail.com",
        "alternateEmail": "aadijay@gmail.com",
        "contactNo": "9855585658",
        "alternateContactNo": "9865658959",
        "aadharNo": "52525252578787"
     },
     {
        "firstName": "Jay",
        "lastName": "aditya",
        "designation": "Sales Director",
        "email": "jayaadi@gmail.com",
        "alternateEmail": "aman1@gmail.com",
        "contactNo": "9898989898",
        "alternateContactNo": "7878787878",
        "aadharNo": "5252525257875"
     }
```

```
    ],
    "salesTaxNo": "sj5s5d2fgg5",
    "salesLiscenceNo": "jtind252dfd858",
    "experience": "15 years",
    "HeadOfficeAddress": {
        "buildingNo": "55",
        "street": "kali gali",
        "landMark": "near tripathi bhawan",
        "city": "mirzapur",
        "state": "uttar pradesh",
        "pinCode": "831313",
        "time": {
            "opening": "9:00AM",
            "closing": "5:00PM"
        }
    },
    "gstIN": "suthid2525di25",
    "salesPanNo": "BLIED52565",
    "location": {
        "latitude": "23.82128298959131",
        "longtitude": "79.43285514996526"
    },
    "sellingBrands":

[
        "mahindra",
        "bmw",
        "audi"
    ],
    "pictureUrls":

[
        "link1",
        "link2"
    ],
    "branches":

[

        {
            "buildingNo": "58",
            "street": "durga gali",
            "landMark": "near railway station",
            "city": "luckhnow",
            "state": "uttar pradesh",
            "pinCode": "831313"
        },
        {
            "buildingNo": "51",
            "street": "hanuman gali",
            "landMark": "near chandni chowk",
            "city": "Delhi",
            "state": "Delhi",
            "pinCode": "831313"
        }
    ],
    "socialMedia": {
        "facebook": "",
        "twitter": "",
        "linkedin": ""
    },
    "typeOfCompany": "individual",
    "emiFacility": true,
    "awardsAndRecognition":

    [

        "Dealer of the Year 2020",
        "Most Selling for Tata 2020"
    ]
}
```


vehicle data service

-It will be an import/export csv files of vehicle details.
-Upload

- Userid(Dealers's) should be in header as part of autherized token
- We find Dealerid using Userid.
- CSV document should have all the details except Dealerid.
- We are going to convert the CSV documents into list of vehicle objects.
- If the vehicleid is already present, it will update the data. If not, it will store the data.
- All this data should be in a predefined attributes.
- For arrays, the format will be "[abc,...]".
- For objects, the format will be "abc,...".
  -Download
- Userid(Dealers's) should be in header as part of autherized token
- We find Dealerid using Userid.
- CSV document should have all the details except Dealerid.
- We are going to convert the objects into csv file
- For arrays, the format will be "[abc,...]".
- For objects, the format will be "abc,...".

CSV Example

In download id will be there, in upload if id is there it is considered as updating the data, if the id is not present in the db, it will create a new data. To maintain redundant data, we consider "vin" as unique .

Header-

Id,tankCapacity,mediaInterface,theftAlarm,pictureUrls,tripMeter,airBagCount,description,price,color,vin,brand,model,year,vehicleType,fuelType, unitsFuelConsumption,transmissionGearType,steeringWheelType,vehicleSpeed,horn,powerTrainTorque,accelaration,nightMode,isABS, wheelRadius,wheelSpeed,light,ignitionTime,odometer,washerFluid,malfunctionIndicator,batteryLevel,airConditioning,languageConfiguration, mirror,childSafetyLock,topSpeedLimit,climate,chime

Body-

abcd,31L,digital,true,"[ss,dd]",30Km,2,geej,234.3,blue,6ghhj,audi,c100,2010,macro,"[diesel,petrol]",30km/L,auto,power,90km/hr,air,30N,90m/s, true,true,30cm,30m,"true,true,true,true,true",2s,true,true,true,80,true,"[english,hindi]",auto,true,50Km/hr,"true,true,true",true

Performance Testing Tools

About Performance Testing

- Performance testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload.
- The goals of performance testing include evaluating application output, processing speed, data transfer velocity, network bandwidth usage, maximum concurrent users, memory utilization, workload efficiency, and command response times.

Why To Do Performance Testing

- To determine whether the application satisfies performance requirements.
- To locate computing bottlenecks within an application.
- To measure stability under peak traffic events.

Acceptable Performance Criteria For The Project

1. Processing = 10000 orders / second.
2. Page view times <= 3s.
3. Response Time < 50ms / request.
4. Throughput = 10000 requests / second.
5. Availability = 99.999%.
6. Concurrent users = 1000.

Performance Testing Tools Considered

The following performance testing tools have been considered for this project:
1. Apache JMeter
The Apache JMeter application is open source software, a 100% pure Java application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions. Apache JMeter may be used to test performance both on static and dynamic resources, Web dynamic applications.
2. MicroFocus LoadRunner
LoadRunner is a software testing tool from Micro Focus. It is used to test applications, measuring system behaviour and performance under load.

LoadRunner can simulate thousands of users concurrently using application software, recording and later analyzing the performance of key components of the application.

3. Locust

Locust is an easy to use, scriptable and scalable performance testing tool. Developers can define the behaviour of the users in regular Python code, instead of using a clunky UI or domain specific language. This makes Locust infinitely expandable and very developer friendly.

4. Gatling

Gatling is an open-source load- and performance-testing framework based on Scala, Akka and Netty. Gatling is designed for continuous load testing and integrates seamlessly with development pipeline. It also offers a remarkable user interface and visualizations for performance analysis.

Performance Testing Tools Comparison

Metric	JMeter	Locust	LoadRunner	Gatling
Latest Release	v5.4.1 (22/01/2021)	v1.6.0 (26/06/2021)	v2021 R1 (29/04/2021)	v3.6.1 (06/07/2021)
Operating System	Windows, Linux, Mac	Windows, Linux, Mac	Windows, Mac	Windows, Linux, Mac
Open Source	Yes	Yes	No	Yes
Community Support	Good	Poor	Average	Average
Inbuilt Protocol Support	Best	Poor	Good	Good
GUI	Less Efficient GUI	No GUI (Code Based, Python)	Efficient GUI	Efficient GUI
Configuration	Easy	Difficult	Moderate	Moderate
Scripting Requirement	Low	High	Moderate	Moderate
Result Visualization	Good	Average	Better	Good
Ramp-up Flexibility	Yes	No	Yes	Yes
Analysis	Easy	Easy	Comparatively Difficult	Easy
CPU Utilization	Moderate	High	Very High	Moderate
Memory Utilization	High	Very Low	Very High	Moderate
Network Utilization	Moderate	NA	NA	Comparatively High
Disk Usage	Moderate	NA	NA	Low

Finalized Performance Testing Tool : Jmeter

The following reasons make Jmeter a suitable performance testing tool for the project:

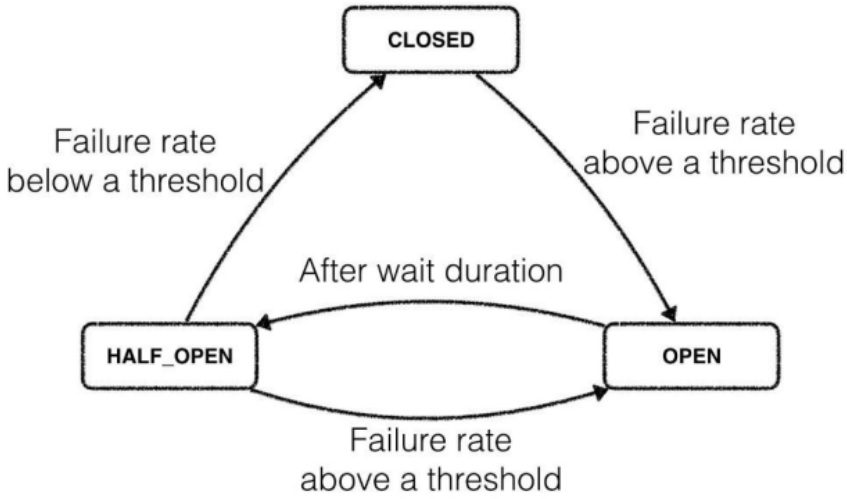
- Lesser CPU & disk utilization.
- Comparatively faster than JMeter for same number of requests.
- Inbuilt support for several protocols.
- Multiple integration facilities with Continuous Integration tools.
- Better visualization features with real-time monitoring.

Circuit Breaker – Resilience4j:

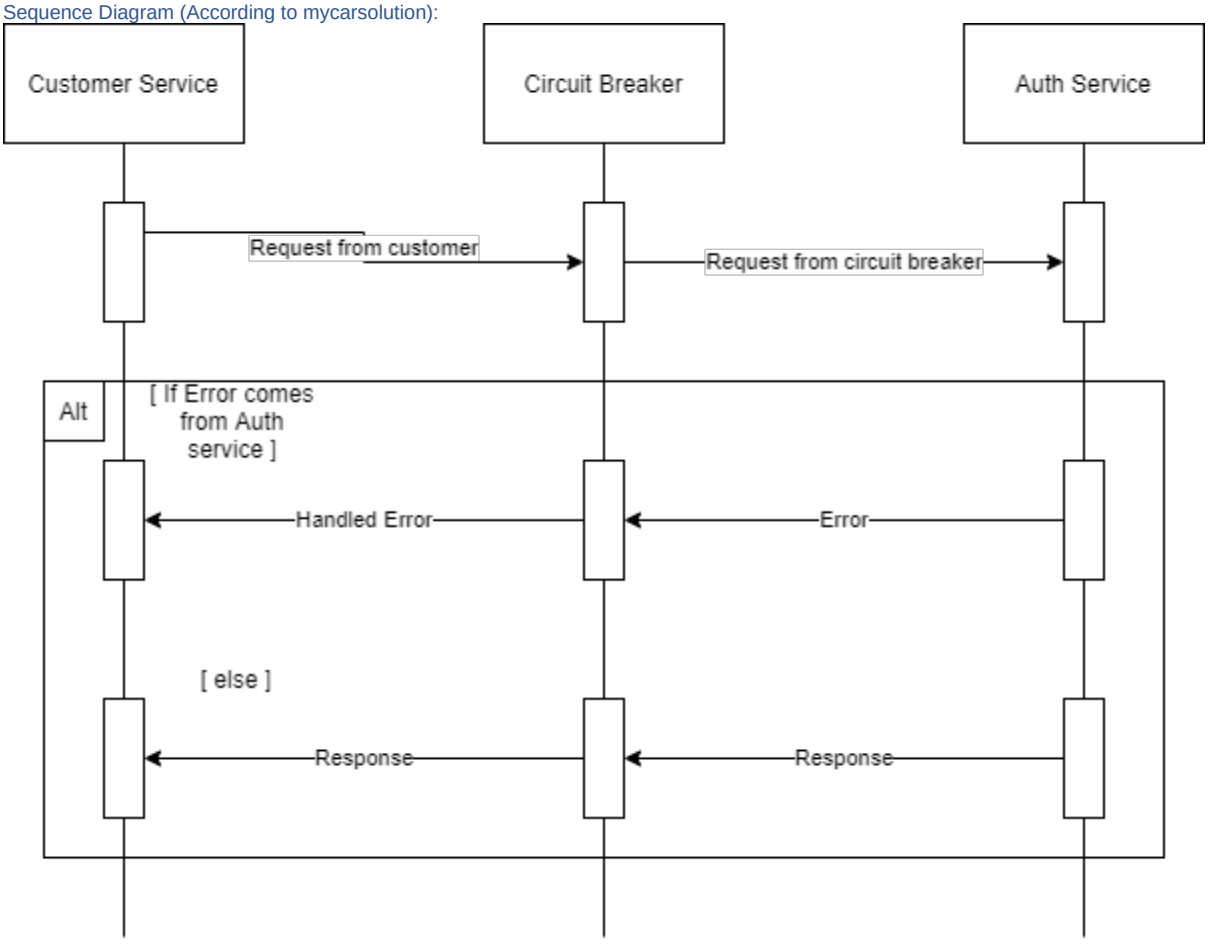
Introduction:

Spring cloud has different types of circuit breakers. In this document , we are going to see about resilence4j. Circuit breakers are like the name suggest if one of your microservice is depending on other microservices , if dependent services are down the circuit breaker will handle the error and make your service more robust.

Architecture:



Sequence Diagram (According to mycarsolution):



Procedure:
1.Adding Dependency:

```
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>
<version>1.0.2.RELEASE</version>
</dependency>
```

2.Wherever your service is dependent on other service like calling another service , you can implement circuit breaker:
Here in this below code the template with url of service B is called by other service A, if any error comes from service B , that is handled by the circuit breaker or the response is returned.

```
Main{
@Autowired
private CircuitBreakerFactory circuitBreakerFactory;

void function(){
CircuitBreaker circuitBreaker = circuitBreakerFactory.create("instance name");
String response = circuitBreaker.run(() -> template.getForObject(url, String.class),
(Throwable e) -> { log.trace("Error from request: {}", e.getMessage()); return "Error: " + e.getMessage
();
});
}
}
```

3.Configuring application.yml:

```
resilience4j.circuitbreaker:
instances: claimscircuitbreaker:
registerHealthIndicator: true slidingWindowSize: 10 permittedNumberOfCallsInHalfOpenState: 3 slidingWindowType: TIME_BASED
minimumNumberOfCalls: 20 waitDurationInOpenState: 50s failureRateThreshold: 50 eventConsumerBufferSize: 10
```

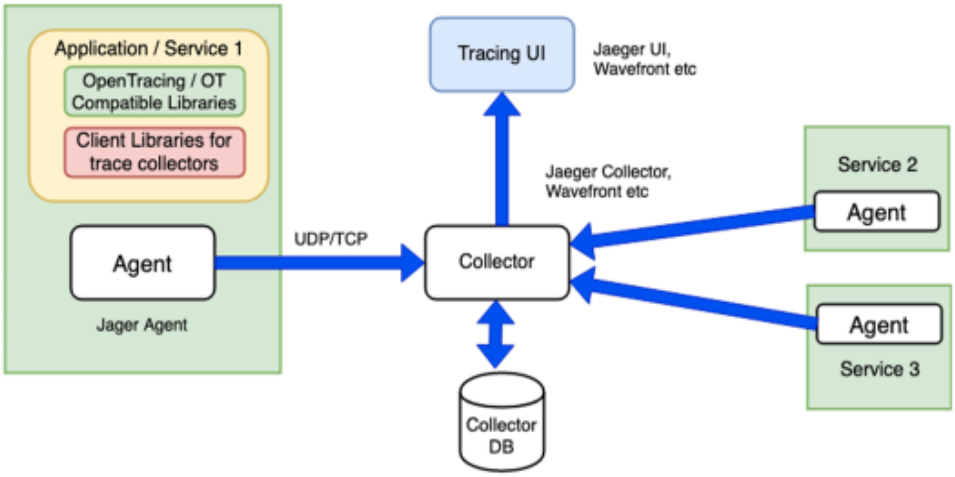
In this above configuration , we are overriding the default limits that are set by spring boot.
The threshold , waiting time for response are defined.

Distributed Tracing Using Jaeger

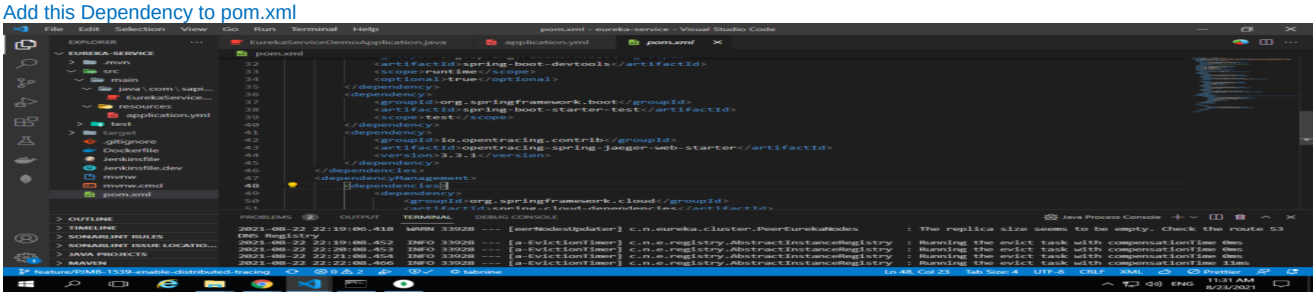
Frameworks used:

- **Opentracing**- API provides a standard, vendor-neutral framework for instrumentation.
- **Jaeger-** Used for monitoring and troubleshooting microservices-based

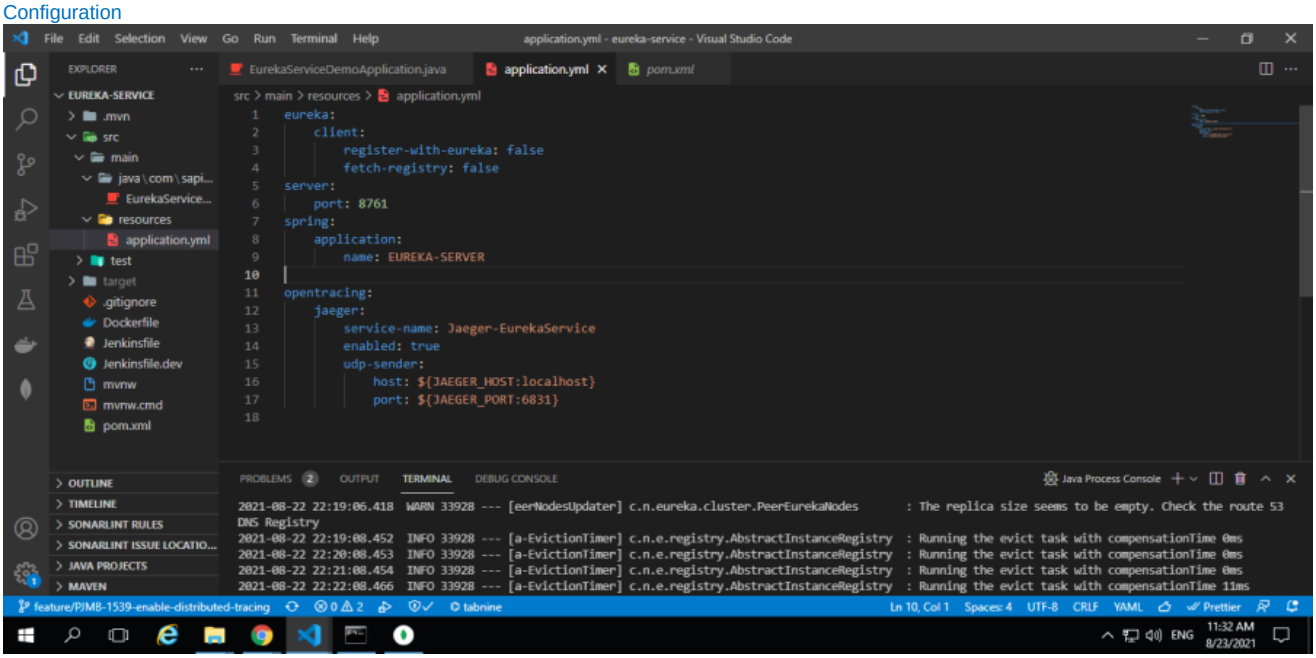
distributed systems.



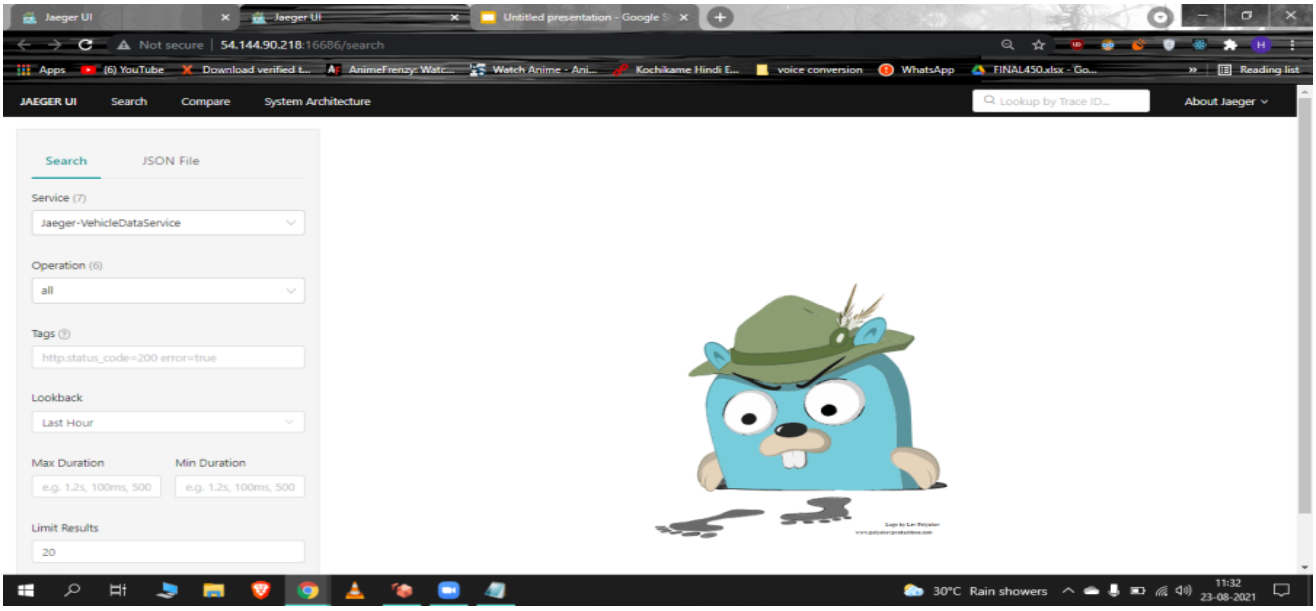
Add this Dependency to pom.xml



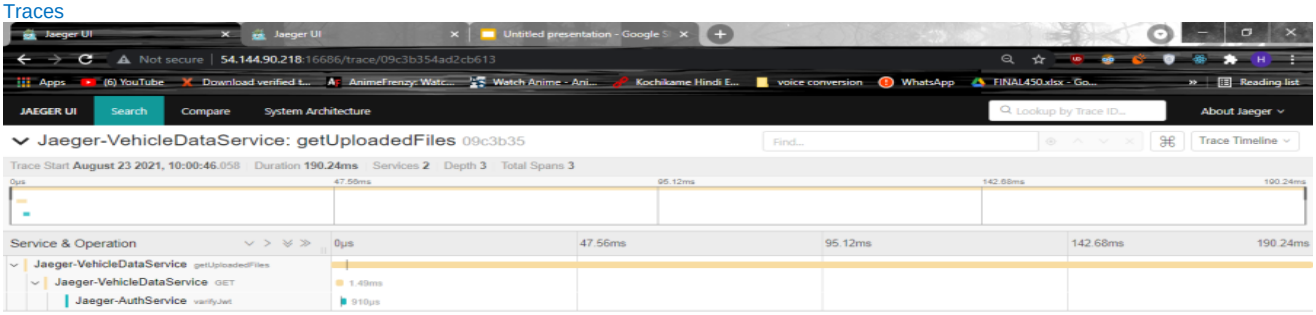
Configuration



Jaeger UI



Traces



Grafana & Prometheus

- Data visualization via charts, graphs, and maps, helps us in understanding the complex form of the data in a simple and better manner.
- Grafana and Prometheus, both help us in tackling issues related to complex data in a simplified manner.
- Grafana is an open-source visualization software, which helps the users to understand the complex data with the help of data metrics.
- It equips the users with features like alert systems, sharing of the dashboard.
- It also helps users in studying time-series analytics.
- Prometheus is an open-source event monitoring and alerting tool.
- It stores the majority of the data locally after scrapping metrics, and after that, to generate alerts, it runs rules over data.
- It came into existence because of the need for monitoring multiple microservices running within the systems.It was initially built at SoundCloud.

Prometheus Config:

Maven dependency:-

Maven Dependency:

<dependency>
<groupId>io.micrometer</groupId>
<artifactId>micrometer-registry-prometheus</artifactId>
<version>1.7.3</version>
</dependency>

Spring application properties:

management:
metrics:
tags:
application: <service-name>
endpoint:
health:
show-details: always
endpoints:
web:
exposure:
include: '*'
base-path: /api/<service-endpoint>/metrics

prometheus.yml

1. my global config
   global:
   scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
   evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
2. scrape_timeout is set to the global default (10s).
3. Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
   rule_files:
4. - "first_rules.yml"
5. - "second_rules.yml"
6. A scrape configuration containing exactly one endpoint to scrape:
7. Here it's Prometheus itself.
   scrape_configs:
8. The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.

- job_name: 'prometheus'

1. metrics_path defaults to '/metrics'
2. scheme defaults to 'http'.
   static_configs:

- targets: ['127.0.0.1:9090']
- job_name: 'gateway-actuator'
  metrics_path: '/actuator/prometheus'
  scrape_interval: 5s
  static_configs:
- targets: ['api.mycarsolutions.net']
  scheme: https
- job_name: 'customer-actuator'
  metrics_path: '/api/customers/metrics/prometheus'
  scrape_interval: 5s
  static_configs:
- targets: ['api.mycarsolutions.net']
  scheme: https
- job_name: 'auth-actuator'
  metrics_path: '/api/auth/metrics/prometheus'
  scrape_interval: 5s
  static_configs:
- targets: ['api.mycarsolutions.net']
  scheme: https
- job_name: 'mail-actuator'
  metrics_path: '/api/mail/metrics/prometheus'
  scrape_interval: 5s
  static_configs:
- targets: ['api.mycarsolutions.net']
  scheme: https
- job_name: 'vehicle-actuator'
  metrics_path: '/api/vehicles/metrics/prometheus'
  scrape_interval: 5s
  static_configs:

- targets: ['api.mycarsolutions.net']
  scheme: https

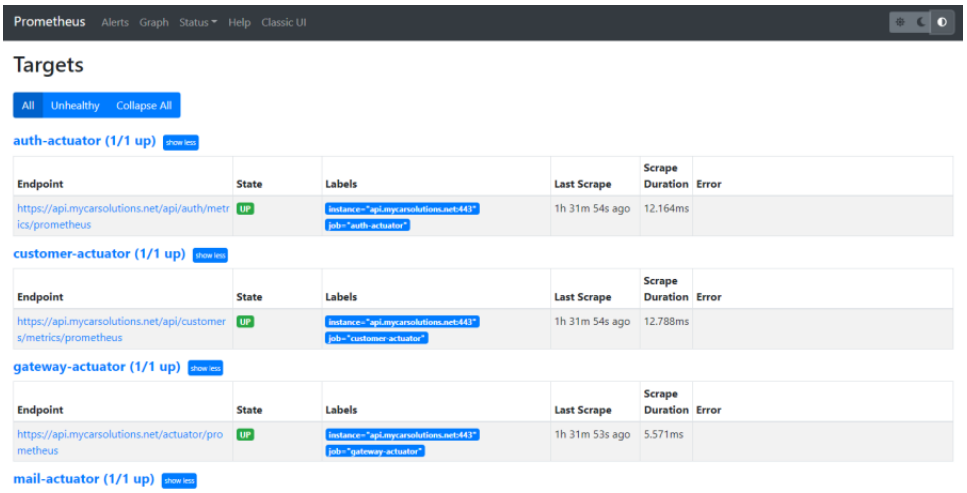
- job_name: 'vehicle-data-actuator'
  metrics_path: '/api/data/metrics/prometheus'
  scrape_interval: 5s
  static_configs:
    - Using Redis

      targets: ['api.mycarsolutions.net']
      scheme: https

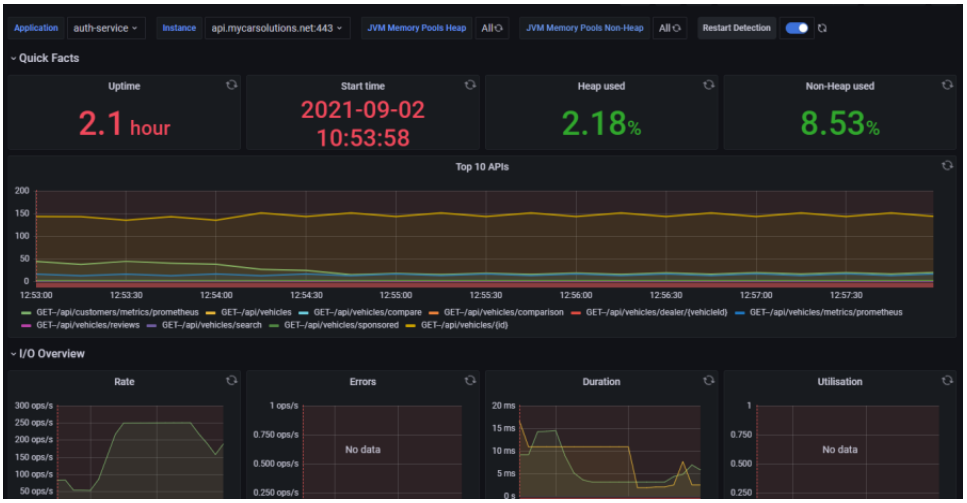
Docker Commands:

sudo docker run -d --name=grafana -p 3000:3000 grafana/grafana
sudo docker run -d --name=prometheus -p 9090:9090 -v //home/ubuntu/grafana/prometheus.yml:/etc/prometheus/prometheus.yml prom
/prometheus

Prometheus Dashboard:

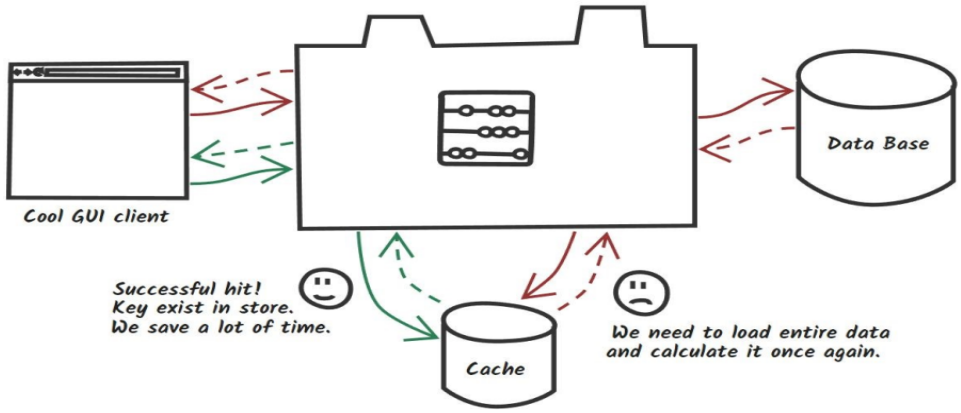


Grafana Dashboard:



Caching in spring boot(Using Redis)

How it works



Why Caching?
Caching is used to enhance the performance of a system. It is a temporary memory that lies between the application and the persistent database. Cache memory stores recently used data items in order to reduce the number of database hits as much as possible.

Few of the features with caching implemented are

- Search Vehicle By text
- Get Comparison
- Get Vehicle Data By Id
- Search Accessories By text

Application.yml :

```yaml
spring:
    application:
        name: vehicle-service

    data:
        mongodb:
            username: admin
            password: Welcome#123
            database: ${DB_NAME:mycarsolution}
            host: ${DB_URL:54.234.219.42}
            port: ${DB_PORT:27017}
    redis:
        port: ${REDIS_PORT:6379}
        host: ${REDIS_HOST:localhost}
        jedis:
            pool:
                maxActive: -1 # no limit to the number of active connections
                maxWait: 30000 # time limit to get a connection - only applies i

eureka:
    client:
        fetchRegistry: true
        registerWithEureka: true
        serviceUrl:
            defaultZone: http://${EUREKA_URL:localhost}:${EUREKA_PORT:8761}/eurel
    instance:
        hostname: ${EUREKA_URL:localhost}
        prefer-ip-address: true
```

Maven dependency:

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-cache</artifactId>
    <version>2.4.3</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-redis</artifactId>
    <version>2.4.3</version>
</dependency>
```

Amazon SES (Simple Email Service)

- Amazon Simple Email Service (SES) is a cost-effective, flexible, and scalable email service that enables developers to send mail from within any application.
- Amazon SES offers several methods of sending email, including the Amazon SES console, the
- Simple Mail Transfer Protocol (SMTP) interface, and the Amazon SES API. We can access the API using the AWS Command Line Interface (AWS CLI), or by using an **AWS Software Development Kit (SDK).**

Dependencies that we used

```xml
<dependency>
    <groupId>io.awspring.cloud</groupId>
    <artifactId>spring-cloud-aws-dependencies</artifactId>
    <version>2.3.1</version>
    <type>pom</type>
</dependency>
<dependency>
    <groupId>io.awspring.cloud</groupId>
    <artifactId>spring-cloud-starter-aws-ses</artifactId>
    <version>2.3.1</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
</dependency>
```

Configuring the Mail Sender Beans

Spring Cloud AWS provides **SimpleEmailServiceMailSender** which is an implementation of the **MailSender** interface from Spring's mail abstraction. **SimpleEmailServiceMailSender** sends emails with Amazon SES using the AWS SDK for Java. It can be used to send simple email messages in plain text without any attachments. A configuration with the necessary elements will look like this:

```java
@Configuration
public class AwsConfig {
    @Bean
    public AmazonSimpleEmailService amazonSimpleEmailService() {

        AWSCredentialsProvider awsCreds = new ClasspathPropertiesFileCredentialsProvider();

        return AmazonSimpleEmailServiceClientBuilder.standard().withCredentials(awsCreds).withRegion(Regions.US_EAST_1)
                .build();

    }

    @Bean
    public JavaMailSender javaMailSender(AmazonSimpleEmailService amazonSimpleEmailService) {
        return new SimpleEmailServiceJavaMailSender(amazonSimpleEmailService);
    }
}
```

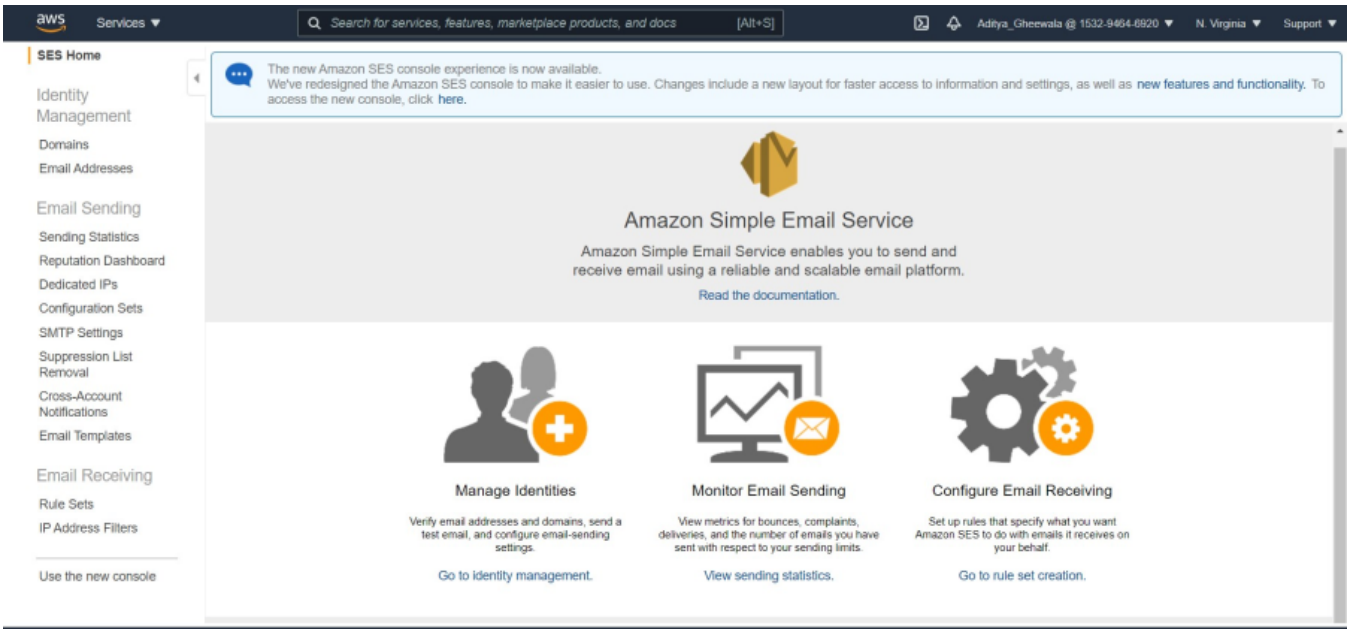
The Service Implementation

```java
@Override
public void sendMailMessage(Mail mail) throws ServiceException {
    log.info("SimpleEmailServiceImpl.sendMailMessage(), mail is {}", mail);
    try {
        MimeMessage mimeMessage = javaMailSender.createMimeMessage();
        MimeMessageHelper messageHelper = new MimeMessageHelper(mimeMessage);
        if (mail == null || mail.getTo() == null || mail.getSubject() == null || mail.getMessage() == null) {
            throw new ServiceException();
        }
        messageHelper.setFrom(sendEmailUsing);
        messageHelper.setTo(mail.getTo().toArray(new String[mail.getTo().size()]));
        messageHelper.setSubject(mail.getSubject());
        messageHelper.setText(mail.getMessage());
        javaMailSender.send(mimeMessage);
    } catch (Exception e) {
        throw new ServiceException(e);
    }

}
```

- We can also use **SendEmailRequest** or **SimpleMailMessage instead of MimeMessage**
- Mail parameter is the entity we've created

```java
@Data
public class Mail {
    private List<String> to = new ArrayList<>();
    private String subject;
    private String message;
}
```

SES Home



Production Access	Sandbox
Mail type	Transactional
Website URL	http://www.mycarsolutions.net
Use case description	The SES will be used to send mails tp verify users and reset passwords when requested.
Additional contact addresses	aditya.gheewala@publicissapient.com
Preferred contact language	English

Forums
Contact Us
Request Increased Sending Limits

Your Amazon SES Sending Limits

Below are the latest statistics and metrics related to your Amazon SES Usage.

Sending Quota: send 200 emails per 24 hour period
Quota Used: 0% as of 2021-08-18 09:03 UTC+5:30
Max Send Rate: 1 email/second
Last updated: 2021-08-18 09:03 UTC+5:30
Learn more about your sending limits.

Your Amazon SES Metrics

The following charts show the number of emails delivered, as well as the rejection, bounce and complaint rates for your account. These charts display data for the past two weeks, aggregated daily.

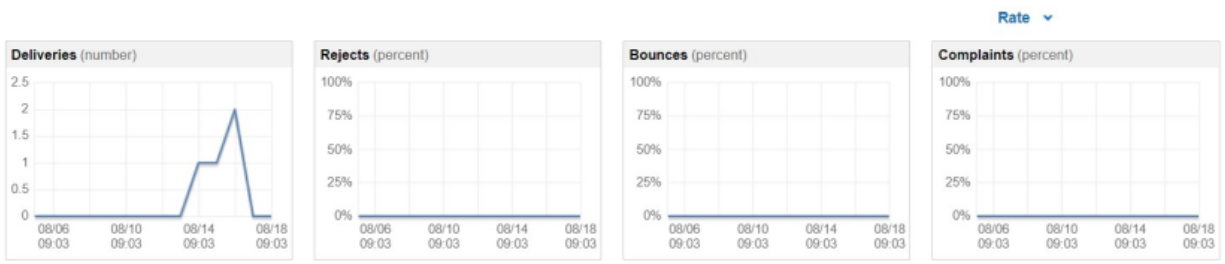
You can change the Rejects, Bounces, and Complaints charts to display a number instead of a percentage. You can also click any chart to view it in a larger window, and to change the aggregation settings.

Sending Statistics

Your Amazon SES Metrics

The following charts show the number of emails delivered, as well as the rejection, bounce and complaint rates for your account. These charts display data for the past two weeks, aggregated daily.

You can change the Rejects, Bounces, and Complaints charts to display a number instead of a percentage. You can also click any chart to view it in a larger window, and to change the aggregation settings.



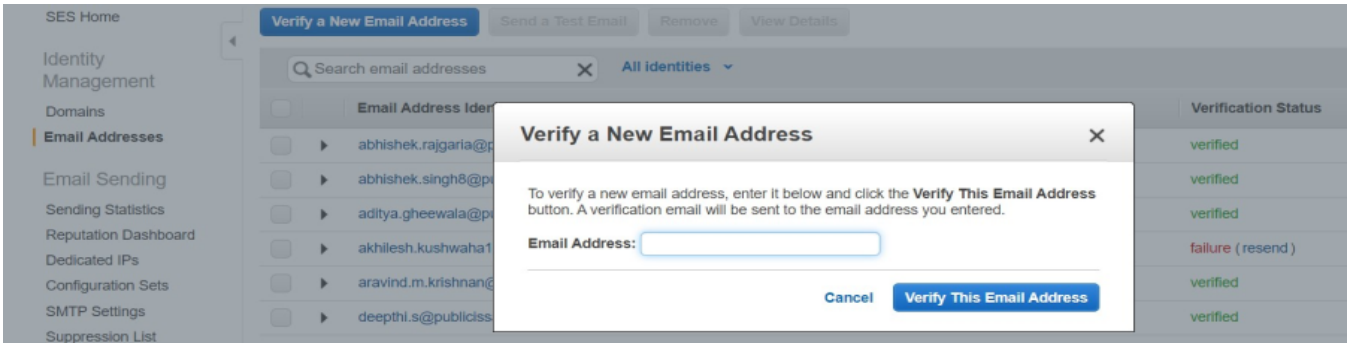
To view the overall bounce and complaint rates for your account, as well as other metrics that may impact your ability to send email using Amazon SES, choose **Reputation Dashboard** in the navigation column to the left.

Amazon says

To help prevent fraud and abuse, and to help protect your reputation as a sender, we apply certain restrictions to new Amazon SES accounts.
We place all new accounts in the Amazon SES ***sandbox***. While your account is in the sandbox, you can use all of the features of Amazon SES. However, when your account is in the sandbox, we apply the following restrictions to your account:

- You **can only send mail to verified email addresses and domains**, or to the Amazon SES mailbox simulator.
- You **can only send mail from verified email addresses and domains**.
- You **can send a maximum of 200 messages per 24-hour period**.
- You **can send a maximum of 1 message per second**.

Verifying Emails



References

- Sending Emails with Amazon SES and Spring Cloud AWS
- Understanding and getting your AWS credentials
- Moving out of the Amazon SES sandbox

Automating AWS Fargate using Terraform

Introduction

Terraform is HashiCorp's infrastructure as code tool. It lets you define resources and infrastructure in human-readable, declarative configuration files, and manages your infrastructure's lifecycle. Using Terraform has several advantages over manually managing your infrastructure:

- Terraform can manage infrastructure on multiple cloud platforms.
- The human-readable configuration language helps you write infrastructure code quickly.
- Terraform's state allows you to track resource changes throughout your deployments.
- You can commit your configurations to version control to safely collaborate on infrastructure.

Installing Terraform

To install Terraform, find the appropriate package for your system and download it as a zip archive.

After downloading Terraform, unzip the package. Terraform runs as a single binary named `terraform`. Any other files in the package can be safely removed and Terraform will still function.

Finally, make sure that the `terraform` binary is available on your `PATH`. This process will differ depending on your operating system.

Getting started with Terraform

- Make a *config.tf* file specifying your provider.

```
provider "aws" {
   region = "your-aws-region"
   access_key = "your-aws-console-access-key"
   secret_key = "your-aws-console-secret-key"
}
```

- Run command *terraform init* for terraform to initialize a working directory containing Terraform configuration files.
- Run command *terraform plan* to evaluate a Terraform configuration to determine the desired state of all the resources it declares, then compares that desired state to the real infrastructure objects being managed with the current working directory and workspace.
- Run command *terraform apply to* execute the actions proposed in a Terraform plan.
- You can use command *terraform destroy* to terminate resources managed by your Terraform project.

Automation of ECS Fargate

- First create *config.tf* file to specify provider and the credentials to log in to AWS console.
- Create *data.tf* file for specifying VPC, Security Groups and ECS task execution role. The file will look something like :-

```
data "aws_subnet_ids" "this" {
  vpc_id = "my-vpc-id"
}

data "aws_iam_role" "ecs_task_execution_role" {
  name = "ecsTaskExecutionRole"
}

data "aws_security_groups" "this" {
  filter {
    name   = "group-name"
    values = ["my-security-group"]
  }
  filter {
    name   = "vpc-id"
    values = ["my-vpc-id"]
  }
}
```

- Create *alb.tf* to configure your load balancer specifications. Our alb.tf file looks like:-

```
resource "aws_alb" "frontend" {
  name            = "tf-frontend-lb"
  subnets         = data.aws_subnet_ids.this.ids
  security_groups = data.aws_security_groups.this.ids
}



resource "aws_alb_target_group" "tg" {
  name        = "tf-frontend-tg"
  port        = 3000
  protocol    = "HTTP"
  vpc_id      = var.vpc_id
  target_type = "ip"

  health_check {
    healthy_threshold   = "2"
    interval            = "30"
    protocol            = "HTTP"
    matcher             = "200"
    timeout             = "10"
    path                = "/"
    unhealthy_threshold = "3"
  }
}

# Redirect all traffic from the ALB to the target group
resource "aws_alb_listener" "front_end" {
  load_balancer_arn = aws_alb.frontend.id
  port              = 3000
  protocol          = "HTTP"
  default_action {
    target_group_arn = aws_alb_target_group.tg.id
    type             = "forward"
  }
}
```

- Create *main.tf* to specify your cluster, services and task related information. An example follows:-

```
locals {
  application_name = "tf-fargate-mycarsolution"
  launch_type      = "FARGATE"
}
```

```
resource "aws_ecs_cluster" "this" {
  name = local.application_name
  setting {
    name  = "containerInsights"
    value = "enabled"
  }
}

resource "aws_ecs_service" "this" {
  name        = "terraform-frontend-service"
  cluster     = aws_ecs_cluster.this.arn
  launch_type = local.launch_type

  deployment_maximum_percent         = 200
  deployment_minimum_healthy_percent = 0
  desired_count                      = 1
  task_definition                    = "${aws_ecs_task_definition.this.family}:${aws_ecs_task_definition.this.revision}"

  network_configuration {
    assign_public_ip = true
    subnets          = data.aws_subnet_ids.this.ids
    security_groups  = data.aws_security_groups.this.ids
  }

  load_balancer {
    target_group_arn = aws_alb_target_group.tg.id
    container_name   = "tf-frontend-container"
    container_port   = 3000
  }
}
```

- Create *main-tasks.tf* file to specify your images to be containerized in the tasks. Example:-

```
resource "aws_ecs_task_definition" "this" {
    family = local.application_name
    requires_compatibilities = [local.launch_type]
    execution_role_arn       = "${data.aws_iam_role.ecs_task_execution_role.arn}"
    network_mode = "awsvpc"
    cpu       = "4096"
    memory    = "8192"
    container_definitions = jsonencode([
    {
      name      = "tf-frontend-container"
      image     = "my-image"
      essential = true
      portMappings = [
        {
          containerPort = 3000
          hostPort      = 3000
        }
      ]
      "environment": [
            {"name": "REACT_APP_SERVICES_HOST", "value": "abcd"},
            {"name": "REACT_APP_SERVICES_PORT", "value": "3000"}
      ],
      "logConfiguration": {
        "logDriver": "awslogs",
        "options": {
            "awslogs-group": "/ecs/tf-frontend",
            "awslogs-region": "us-east-1",
            "awslogs-stream-prefix": "ecs"
        }
      }
    }

])

}
```

Deployment Servers

Given that we are using React for frontend, Java Spring Boot for backend and AWS as our deployment server. Based on these, we will be having certain deployment tools available for us.

Frontend Deployment Servers :

1. nginx
2. Node.js
3. Apache HTTP Server

Frontend Deployment Servers	Open Source	Response Time	Handling Request	Concurrency	Configuration
nginx	Yes	Faster (Even with big load)	Event-driven mechanism (multiple request within one thread)	Better performance	Not additionally configurable
Node.js	Yes	Slow with big load	A single-thread and an event loop	Slow performance	Configurable
Apache HTTP Server	Yes	Slower than nginx	Process-driven mechanism (new thread for each request)	Not that good as nginx	Configurable with .htaccess file

It is always suggested to have a reverse proxy whenever possible.

Nginx was created to overcome the drawbacks of httpd (Apache HTTP Server).

Nginx can also be used as a load balancer, HTTP cache, mail proxy, or reverse proxy in a server.

It is best suited for our need since it is faster in response time even with large requests.

Hence we decide to **use nginx as frontend server** for our application.

Backend Deployment Servers:

Jetty

Apache Tomcat

Backend Deployment Servers	Open Source	Production Tested	Preferences (perception)	On Spring Boot
Jetty	Yes	Yes	Performance	Have to configure
Apache Tomcat	Yes	Yes	Specification	By default

Jetty have be there before Apache Tomcat but Apache Tomcat has found wider use . Developer perceive that Jetty is performance intensive but it depends on application needs and design. For our project we are using Spring Boot , by default it provides an integrated Apache Tomcat server. Following Convention over configuration, we prefer to use spring boot **embedded Apache Tomcat server**.

Conclusion:

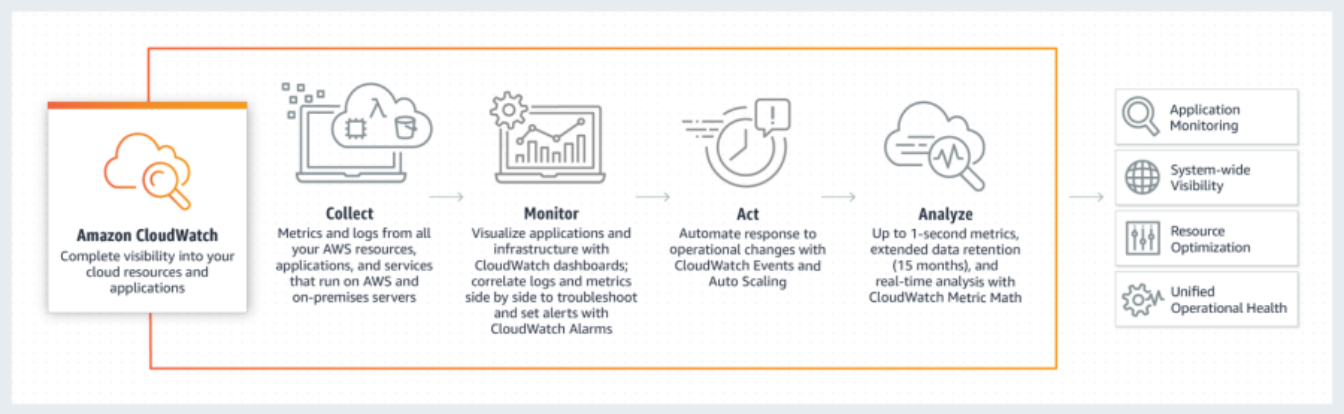
- For frontend we decided to use **nginx** as server.
- For backend we decided to use embedded **Apache Tomcat** Server (which comes along spring boot).

Amazon CloudWatch

Amazon CloudWatch is a monitoring and observability service built for DevOps engineers, developers, site reliability engineers (SREs), and IT managers.
CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, providing you with a unified view of AWS resources, applications, and services that run on AWS and on-premises servers. You can use CloudWatch to detect anomalous behavior in your environments, set alarms, visualize logs and metrics side by side, take automated actions, troubleshoot issues, and discover insights to keep your applications running smoothly.

How it works

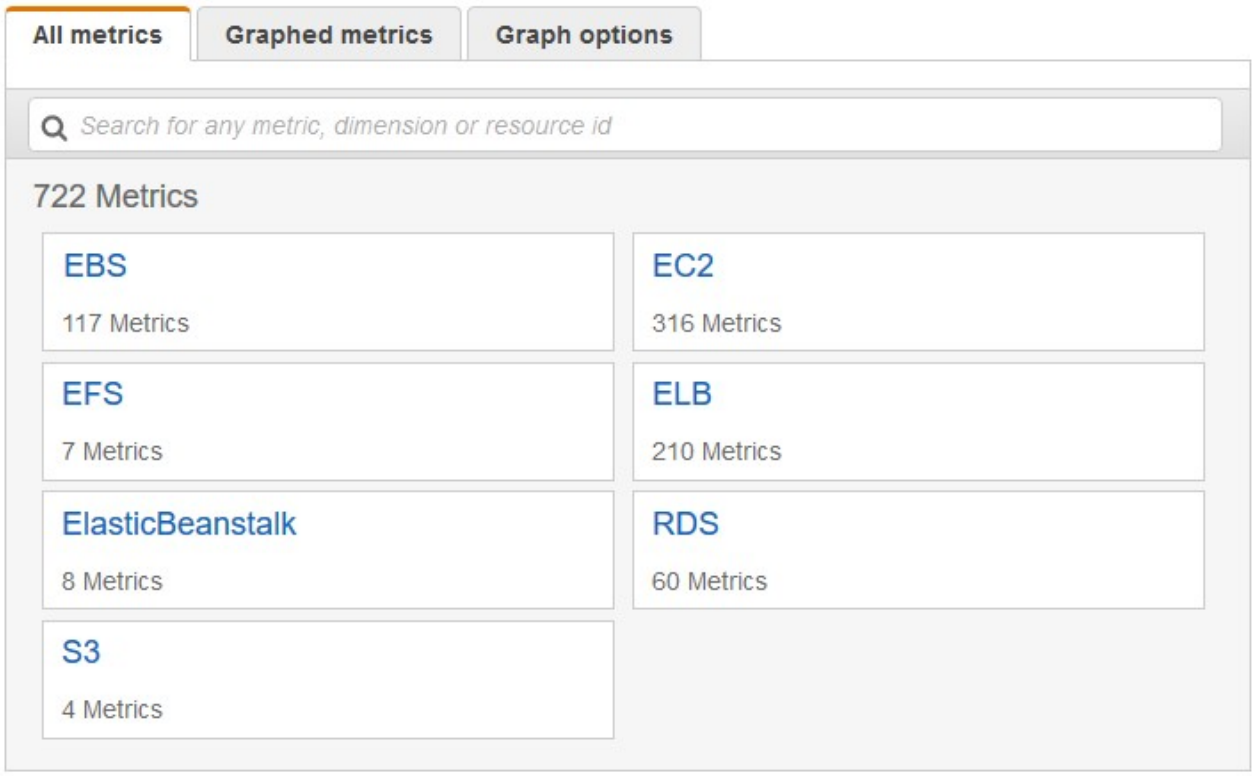
CloudWatch collects monitoring and operational data in the form of logs, metrics, and events, and visualizes it using automated dashboards so you can get a unified view of your AWS resources, applications, and services that run in AWS and on-premises.

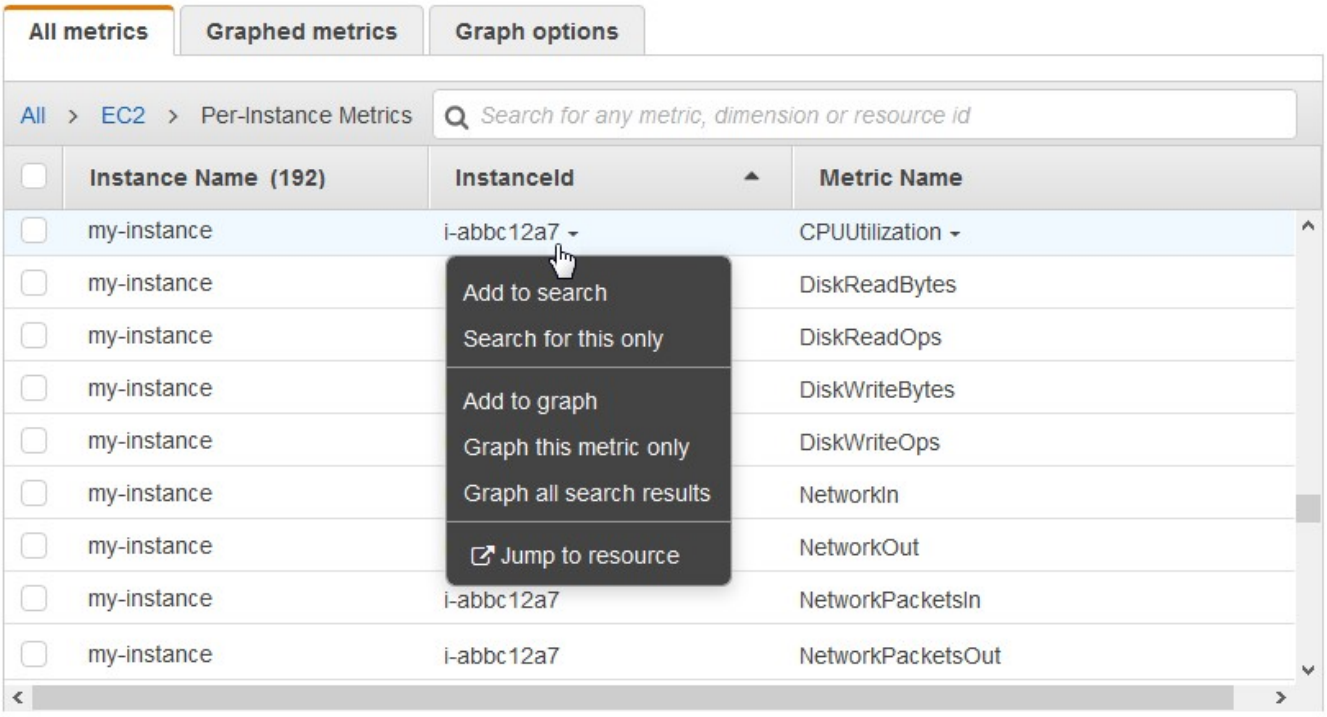


Viewing Metrics

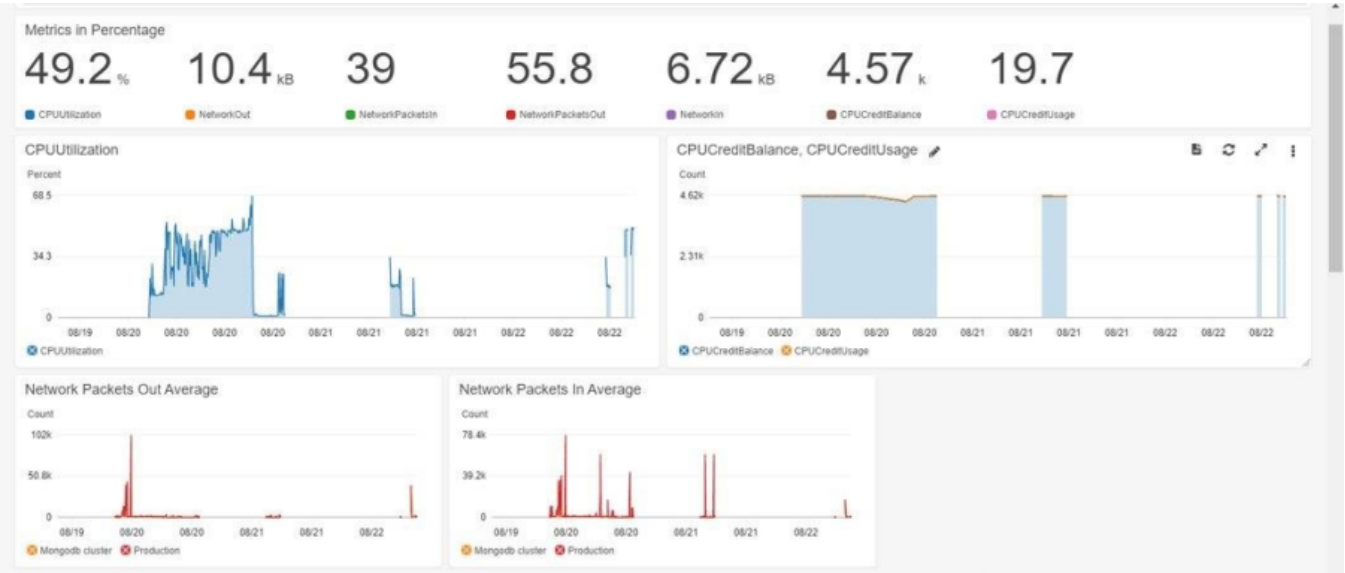
Metrics are grouped first by namespace, and then by the various dimension combinations within each namespace. For example, you can view all EC2 metrics, EC2 metrics grouped by instance, or EC2 metrics grouped by Auto Scaling group.

Only the AWS services that you're using send metrics to Amazon CloudWatch.





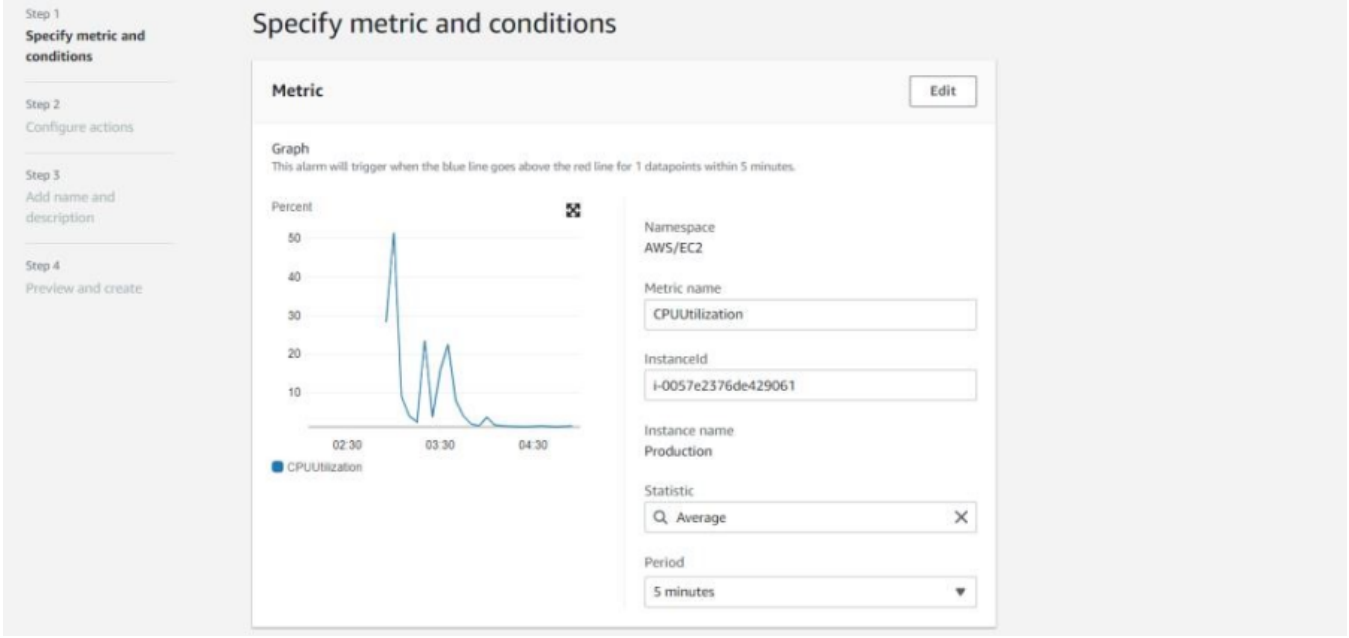
Example Dashboard

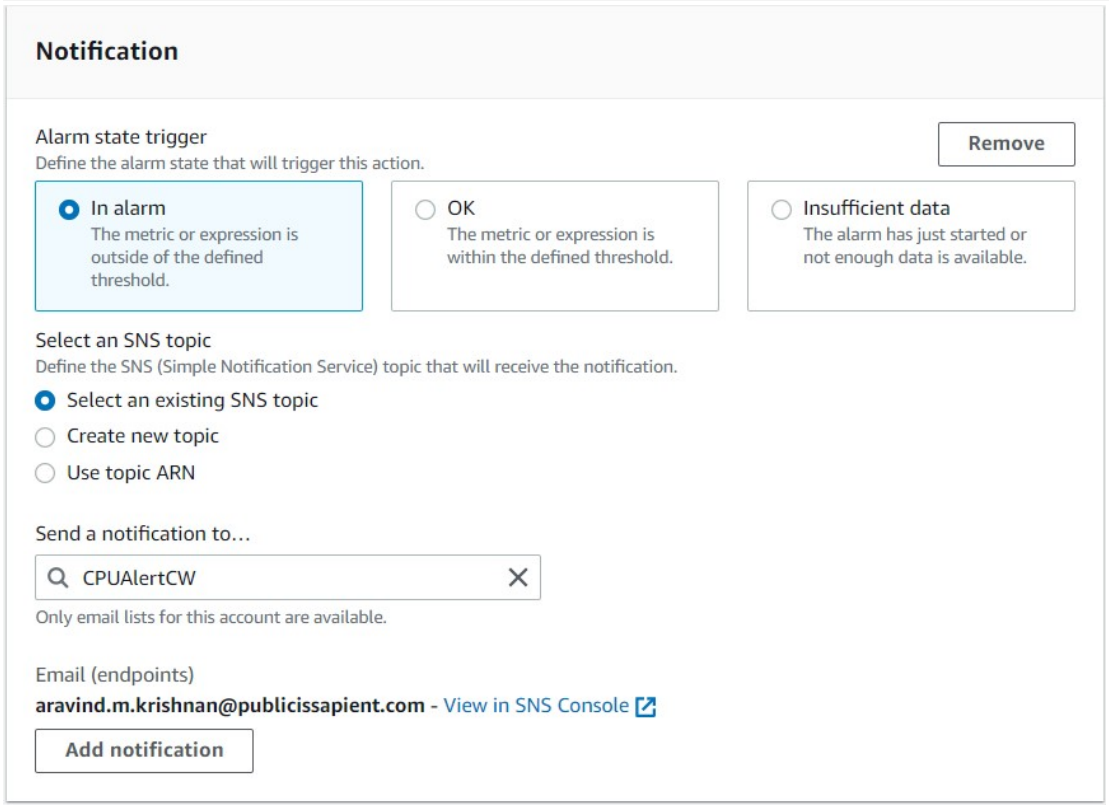


Alarms

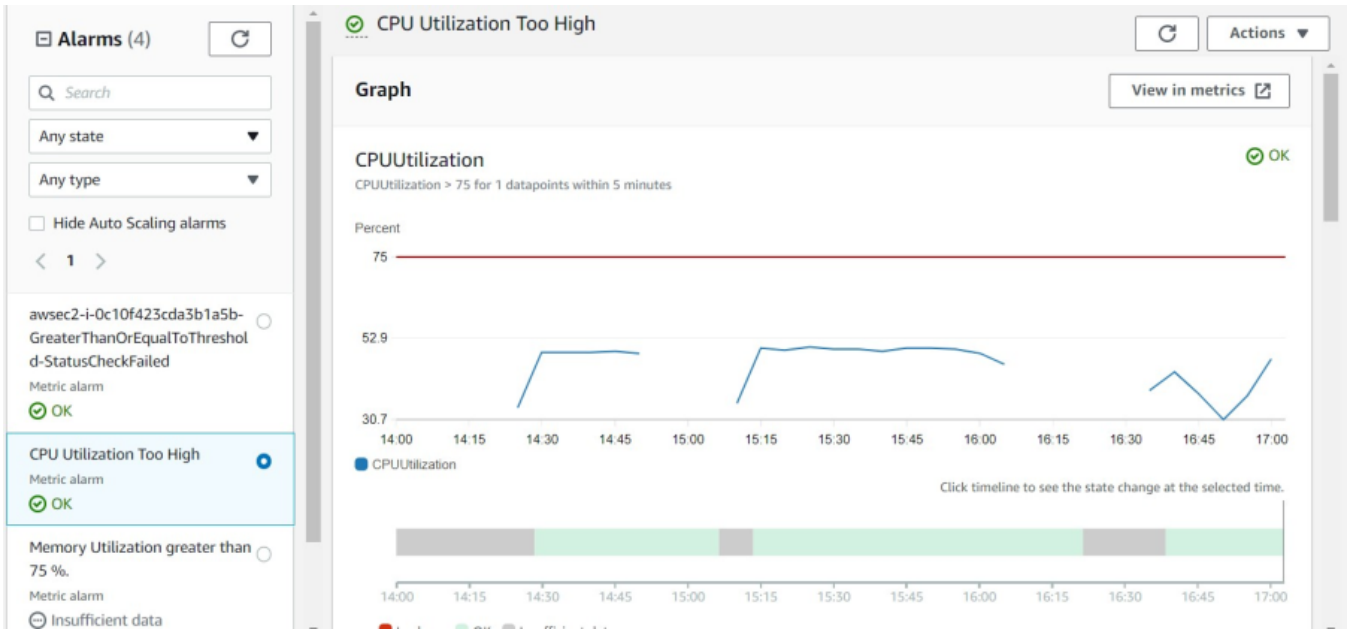
A *metric alarm* watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. The action can be sending a

notification to an Amazon SNS topic, performing an Amazon EC2 action or an Auto Scaling action, or creating an OpsItem or incident in Systems Manager.





Alarms



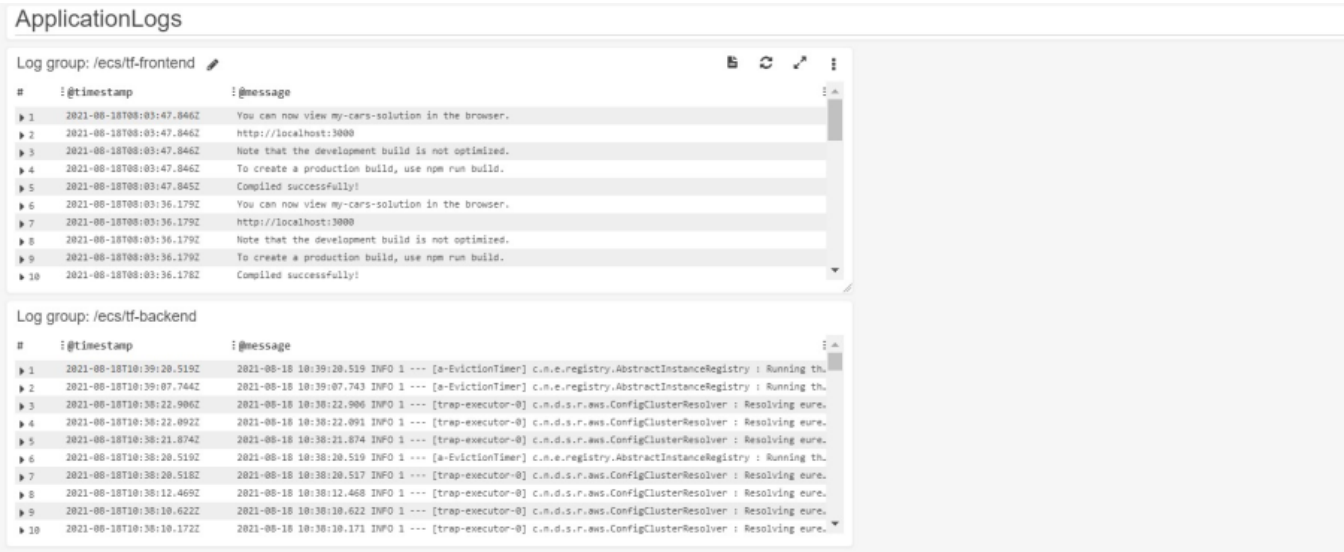
Logging

You can configure the containers in your tasks to send log information to CloudWatch Logs. If you are using the Fargate launch type for your tasks, this

allows you to view the logs from your containers. Before your containers can send logs to CloudWatch, you must specify the awslogs log driver for containers in your task definition.

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "wordpress",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "awslogs-wordpress",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
```

Example Log Files



Building and Pushing docker images to ECR:

1. A new docker image is generated at each Jenkins build for the particular micro-service or frontend.
2. In ECR we have to create the repository through the AWS console, where the docker images are being stored.
3. The generated docker images are pushed to ECR (Elastic Container Registry) and each new image is assigned two tags (one, associated with Jenkins build number eg. "4.0.12" and other "latest").
4. The "latest" tag is used to identify the latest image being pushed to the ECR.

```
def build =docker.build(<ECR-repository url>:${<Version-num>.0.${BUILD_NUMBER}",
".")

withDockerRegistry(credentialsId: 'ecr:us-east-1:<credential>', toolName:
DOCKER',url:<ECR-repository url>){
docker.image(<ECR-repository url>:${<Version-num>.0.${BUILD_NUMBER}").push()
docker.image(<ECR-repository
url>:${<Version-num>.0.${BUILD_NUMBER}").push('latest')
}
```

Deploying ECR images on EC2

1. Elastic Compute Cloud (EC2) is a virtual server in AWS used for running applications.
2. We have selected Ubuntu 20.04 LTS as the AMI and the instance type is t3.2xlarge for running various micro-services, frontend and tools.
3. By default an unique IP address is assigned whenever the instance is started, to fix the IP address we have assigned an Elastic IP address 54 .144.90.218 which remains fixed in all conditions.
4. To containrise the images of micro-services stored in ECR, we have created a docker-compose.yml file where we specify the image, volume, environment variable if any, also dependencies on other services etc.

```
aws ecr get-login-password | sudo docker login --username AWS --password-stdin
<ECR-repository url>
```

5. A shell script is also written which will pull newer/latest images from ECR into the EC2 instance.
6. After pulling the images, docker-compose.yml file is executed using the comman.

```
sudo docker pull<ECR-repository image url>
docker-compose up -d
```

7. A security group is defined which exposes the required ports to public

Port range	Protocol	Source
80	TCP	0.0.0.0/0
80	TCP	::/0
8400	TCP	0.0.0.0/0
8400	TCP	::/0
8761	TCP	0.0.0.0/0
8761	TCP	::/0
22	TCP	0.0.0.0/0
16686	TCP	0.0.0.0/0
16686	TCP	::/0
5601	TCP	0.0.0.0/0

8. A total of 14 images are being maintained in the containerised form in the particular EC2 instance (Production server).

Deploying ECR images on ECS Fargate

1. Elastic Container Service (ECS) is the Container Orchestration service provided by AWS which allows us to run several containers without having the need to manage servers or clusters. With Fargate, we no longer need to configure instance type scale, it will be managed by Fargate.
2. Structure of ECS:
    a. Container Definition - here we define container image, size etc.

    b. Task Definition - It is the blueprint of the application and logical grouping of containers.
    c. Service - It allows us to maintain desired instances of tasks.
    d. Cluster - logical grouping of services.
3. We had define two task definition:
    a. micro-service-TD : here we define the containers for all micro services and additional tools like redis, kafka etc.
    b. Frontend -TD : here we define the container definition of front end only.
4. In the container definition we have specified the image url from the ECR. and to load image into ECS we require to provide an IAM role ecsTaskExecutionRole.
5. Similarly two services are being defined one for backend and frontend and 3-3 instances of them are being instantiated.
6. While defining the service we need to define the security group which would be mostly the same as we defined in the EC2 instance. Also an Application Load Balancer is defined for scalability and efficient management of micro services.
7. The above forms an ECS cluster where the application is running.