

Eureka Server (service-registry):

Instances currently registered with Eureka

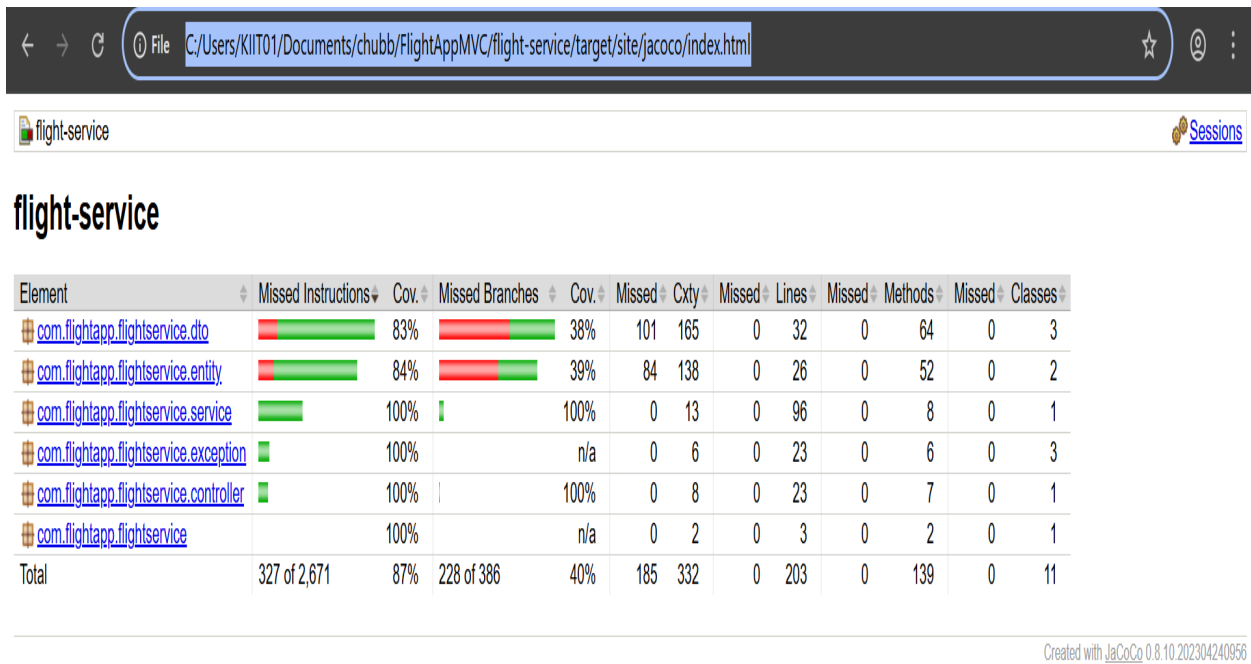
Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - KIIT01.mshome.net:api-gateway:8081
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - KIIT01.mshome.net:booking-service:8082
FLIGHT-SERVICE	n/a (1)	(1)	UP (1) - KIIT01.mshome.net:flight-service:8083

General Info

Name	Value
total-avail-memory	90mb
num-of-cpus	8
current-memory-usage	53mb (58%)
server-uptime	01:07
registered-replicas	http://localhost:8761/eureka/
unavailable-replicas	http://localhost:8761/eureka/,

JACOCO-REPORTS:

Flight-service(87%)



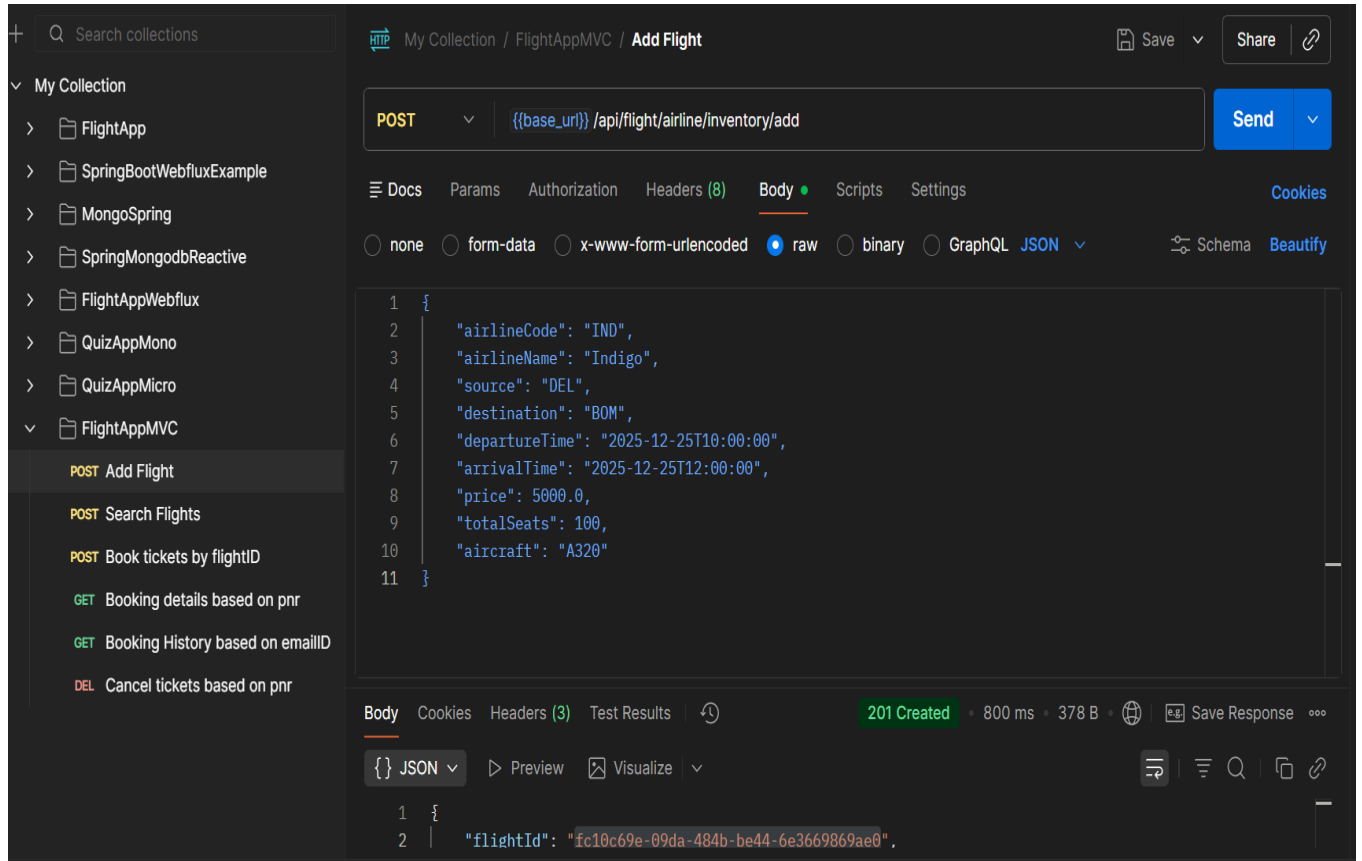
Booking-service:

booking-service

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.flightapp.bookingservice.exception	<div><div></div></div>	31%	<div><div></div></div>	0%	41	51	7	26	17	27	0	5
com.flightapp.bookingservice.dto	<div><div></div></div>	80%	<div><div></div></div>	41%	106	165	0	32	11	64	0	3
com.flightapp.bookingservice.entity	<div><div></div></div>	82%	<div><div></div></div>	41%	74	120	1	23	4	46	0	2
com.flightapp.bookingservice.feign	<div><div></div></div>	78%	<div><div></div></div>	40%	48	69	0	13	7	26	0	1
com.flightapp.bookingservice.messaging	<div><div></div></div>	80%	<div><div></div></div>	43%	35	64	2	27	2	28	0	3
com.flightapp.bookingservice.service	<div><div></div></div>	94%	<div><div></div></div>	78%	4	21	4	108	1	14	0	2
com.flightapp.bookingservice.controller	<div><div></div></div>	88%	<div><div></div></div>	100%	0	8	3	25	0	6	0	1
com.flightapp.bookingservice.config	<div><div></div></div>	100%		n/a	0	7	0	7	0	7	0	2
com.flightapp.bookingservice		100%		n/a	0	2	0	3	0	2	0	1
Total	838 of 3,796	77%	348 of 574	39%	308	507	17	264	42	220	0	20

Postman Screenshots:

1. Adding Inventory



2. Searching Flights:

The screenshot displays the Postman application interface. On the left sidebar, the 'My Collection' is expanded, showing a list of API collections. The 'FlightAppMVC' collection is selected, and the 'POST Search Flights' request is highlighted. The main panel shows the details of this request. The URL is set to `{{base_url}} /api/flight/search`. The request body is a JSON object with the following fields: `"source": "DEL", "destination": "BOM", "departureDate": "2025-12-25", "journeyType": "ONE_WAY"`. The response status is `200 OK` with a response time of 95 ms and a size of 375 B. The response body is a JSON object with the following fields: `"flightId": "fc10c69e-09da-484b-be44-6e3669869ae0", "airlineCode": "IND", "airlineName": "Indigo", "source": "DEL", "destination": "BOM", "departureTime": "2025-12-25T10:00:00", "arrivalTime": "2025-12-25T12:00:00", "price": 5000.0, "availableSeats": 100`.

SONAL KUMARI_3114's Workspace New Import

My Collection / FlightAppMVC / Search Flights

POST `{{base_url}} /api/flight/search` Send

Docs Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON Schema

```
1 {
2   "source": "DEL",
3   "destination": "BOM",
4   "departureDate": "2025-12-25",
5   "journeyType": "ONE_WAY"
6 }
```

Body Cookies Headers (3) Test Results 200 OK 95 ms 375 B Save Response

{ } JSON Preview Visualize

```
2 {
3   "flightId": "fc10c69e-09da-484b-be44-6e3669869ae0",
4   "airlineCode": "IND",
5   "airlineName": "Indigo",
6   "source": "DEL",
7   "destination": "BOM",
8   "departureTime": "2025-12-25T10:00:00",
9   "arrivalTime": "2025-12-25T12:00:00",
10  "price": 5000.0,
11  "availableSeats": 100,
```

3. Book tickets by providing flight ID:

The screenshot shows the Postman interface with a collection named "My Collection" containing a folder "FlightAppMVC". Inside this folder, the endpoint "POST Book tickets by flightID" is selected. The request is a POST to the URL `{{base_url}}/api/booking/flight/fc10c69e-09da-484b-be44-6e3669869ae0`. The request body is raw JSON, containing the following data:

```
1 {
2   "userEmail": "john@test.com",
3   "userName": "John Miller",
4   "numberOfSeats": 2,
5   "passengers": [
6     {
7       "name": "John",
8       "gender": "Male",
9       "age": 30
10    },
11    {
12      "name": "Jane",
13      "gender": "Female",
14      "age": 28
15    }
16  ]
17 }
```

The response status is "201 Created" with a time of 29.09 s and a size of 478 B. The response body is JSON, containing the following data:

```
1 {
2   "bookingId": "c40e0d50-646c-493a-b979-ec6e6f148f68",
3   "pnr": "PNR17646171725096a748e",
4   "flightId": "fc10c69e-09da-484b-be44-6e3669869ae0",
5   "userEmail": "john@test.com",
6   "userName": "John Miller",
7   "numberOfSeats": 2,
8   "selectedSeats": [
9     "1A",
10    "1B"
11  ],
12   "mealPreference": "VEG",
13   "totalPrice": 10000.0,
14   "bookingStatus": "CONFIRMED",
15   "journeyDate": "2025-12-25",
16   "cancelDate": "2025-12-25"
17 }
```

The screenshot shows the Postman interface with the same collection and endpoint as the previous screenshot. The response status is "201 Created" with a time of 29.09 s and a size of 478 B. The response body is JSON, containing the following data:

```
1 {
2   "bookingId": "c40e0d50-646c-493a-b979-ec6e6f148f68",
3   "pnr": "PNR17646171725096a748e",
4   "flightId": "fc10c69e-09da-484b-be44-6e3669869ae0",
5   "userEmail": "john@test.com",
6   "userName": "John Miller",
7   "numberOfSeats": 2,
8   "selectedSeats": [
9     "1A",
10    "1B"
11  ],
12   "mealPreference": "VEG",
13   "totalPrice": 10000.0,
14   "bookingStatus": "CONFIRMED",
15   "journeyDate": "2025-12-25",
16   "cancelDate": "2025-12-25"
17 }
```

4. GET Booking details based on PNR:

The screenshot displays a REST client interface with a sidebar on the left listing various API endpoints under the 'My Collection' folder. The selected endpoint is 'GET Booking details based on pnr' under the 'FlightAppMVC' folder. The main panel shows the request details: a GET method with the URL `{{base_url}}/api/booking/ticket/PNR17646171725096a748e`. The response is a 200 OK status, received in 5.61 seconds with a body size of 473 B. The response body is displayed in JSON format, showing the following details:

```
1 {
2   "bookingId": "c40e0d50-646c-493a-b979-ec6e6f148f68",
3   "pnr": "PNR17646171725096a748e",
4   "flightId": "fc10c69e-09da-484b-be44-6e3669869ae0",
5   "userEmail": "john@test.com",
6   "userName": "John Miller",
7   "numberOfSeats": 2,
8   "selectedSeats": [
9     "1A",
10    "1B"
11  ],
12   "mealPreference": "VEG",
13   "totalPrice": 10000.0,
14   "bookingStatus": "CONFIRMED",
15   "journeyDate": "2025-12-25",
16   "ticketId": "17646171725096a748e"
17 }
```

5. Booking history based on emailID

The screenshot shows the Postman interface with a GET request to `{{base_url}}/api/booking/history/john@test.com`. The response is a 200 OK status with a 90 ms response time and 1.56 KB of data. The response body is a JSON array containing one booking record.

```
1 [
2   {
3     "bookingId": "8bef14e7-e723-4b84-86dd-754357fb7813",
4     "pnr": "PNR176461212744013dcac",
5     "flightId": "fc10c69e-09da-484b-be44-6e3669869ae0",
6     "userEmail": "john@test.com",
7     "userName": "John Miller",
8     "numberOfSeats": 2,
9     "selectedSeats": [
10      "1A",
11      "1B"
12    ],
13     "mealPreference": "VEG",
14     "totalPrice": 10000.0,
15     "bookingStatus": "CONFIRMED",
16     "journeyDate": "2025-12-25",
17     "createdAt": 1764612127464
18   },
19 ]
```

6. Cancel ticket by giving pnr:

The screenshot shows a REST client interface with a sidebar on the left containing a collection of API endpoints. The selected endpoint is `DEL Cancel tickets based on pnr`. The main panel displays a DELETE request to `{{base_url}}/api/booking/cancel/PNR17646171725096a748e`. Below the request bar, tabs for `Docs`, `Params`, `Authorization`, `Headers (6)`, `Body`, `Scripts`, and `Settings` are visible. The `Query Params` section is empty. The response section shows a `200 OK` status with a response time of `4.52 s` and a size of `473 B`. The response body is displayed in JSON format:

```
8  "selectedSeats": [  
9    "1A",  
10   "1B"  
11  ],  
12  "mealPreference": "VEG",  
13  "totalPrice": 18888.0,  
14  "bookingStatus": "CANCELLED",  
15  "journeyDate": "2025-12-25",  
16  "createdAt": 1764617172509  
17  }
```

JMETER TESTING:

20 requests:

The screenshot displays the Apache JMeter interface with the 'View Results Tree' window open. The left sidebar shows a test plan structure with three thread groups: 'Thread Group (20 requests)', 'Thread Group (50 requests)', and 'Thread Group (100 requests)'. The 'View Results Tree' window is titled 'View Results Tree' and contains a search bar, a 'Log/Display Only' section with checkboxes for 'Errors' and 'Successes', and a 'Write results to file / Read from file' section with a 'Filename' field and a 'Browse...' button. The main area shows a list of 20 test results, all of which are 'Search ticket based on pnr' and have a green checkmark icon, indicating success. The results are displayed in a table with columns for 'Text' and 'Sampler result'. The 'Text' column contains the text 'Search ticket based on pnr' and the 'Sampler result' column contains the text 'Raw' and 'Parsed'. The 'Scroll automatically?' checkbox is checked.

Test Plan

- Thread Group (20 requests)
 - Search ticket based on pnr
 - Summary Report
 - View Results Tree
- Thread Group (50 requests)
 - Get booking history by emailID
 - View Results Tree
 - Summary Report
- Thread Group (100 requests)
 - Search flights
 - HTTP Header Manager
 - Summary Report
 - View Results Tree

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes Configure

Search: ☐ Case sensitive ☐ Regular exp. Search Reset

Text

Sampler result

- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr
- Search ticket based on pnr

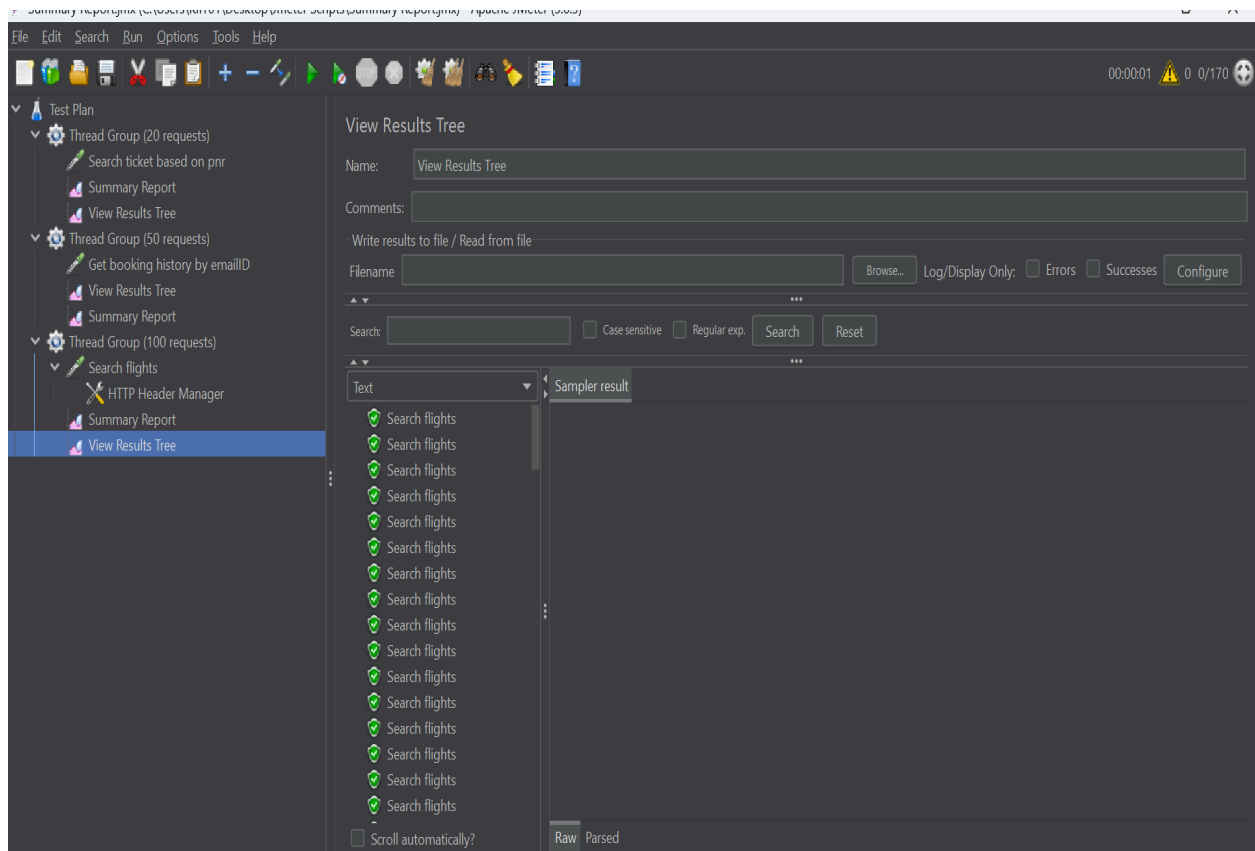
☒ Scroll automatically?

Raw Parsed

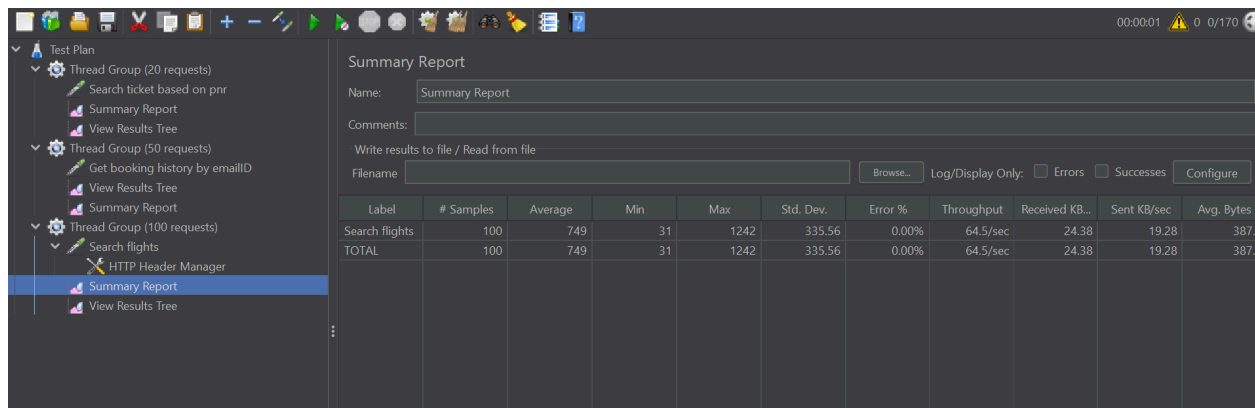
50 requests:

[illegible]

100 requests:



The screenshot shows the JMeter 'View Results Tree' window. The left sidebar displays a test plan with three thread groups: 'Thread Group (20 requests)', 'Thread Group (50 requests)', and 'Thread Group (100 requests)'. The 'Thread Group (100 requests)' is expanded, showing a 'Search flights' sampler with 100 samples, all marked with green checkmarks. The main panel shows the details for a selected 'Search flights' sampler, including fields for Name, Comments, and Filename, and a search bar. The 'Raw' tab is selected, displaying a list of 100 'Search flights' entries.



The screenshot shows the JMeter 'Summary Report' window. The left sidebar is the same as the previous screenshot. The main panel displays a table with performance metrics for the 'Search flights' sampler. The table has columns for Label, # Samples, Average, Min, Max, Std. Dev., Error %, Throughput, Received KB..., Sent KB/sec, and Avg. Bytes. The data shows 100 samples with an average of 749, a minimum of 31, and a maximum of 1242. The error rate is 0.00% and the throughput is 64.5/sec.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB...	Sent KB/sec	Avg. Bytes
Search flights	100	749	31	1242	335.56	0.00%	64.5/sec	24.38	19.28	387.
TOTAL	100	749	31	1242	335.56	0.00%	64.5/sec	24.38	19.28	387.