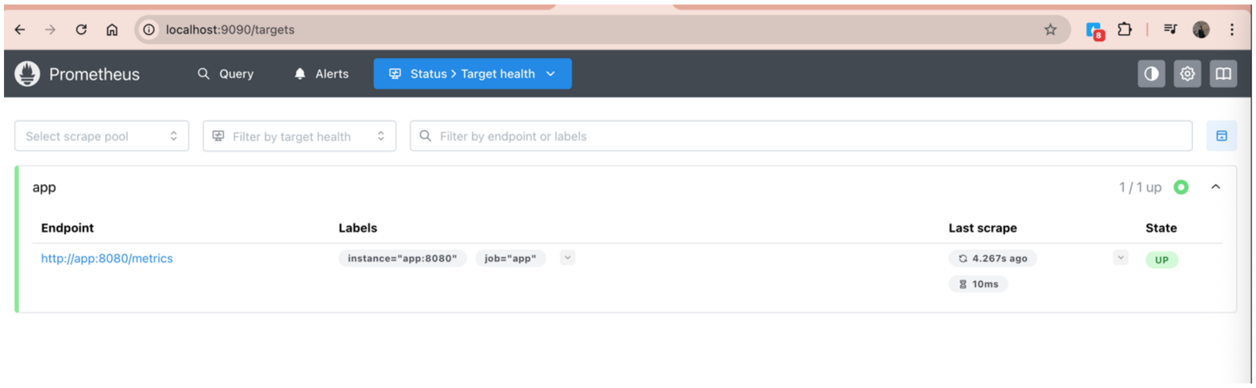
Figure 1 – Prometheus Target Health

Prometheus successfully scrapes the application /metrics endpoint and the target is in **UP** state.
This confirms that the metrics pipeline is working correctly and the application is ready for monitoring.



app 1 / 1 up

Endpoint	Labels	Last scrape	State
http://app:8080/metrics	instance="app:8080" job="app"	4.267s ago 10ms	UP

Figure 2 – User Request Rate (req/s)

This metric represents the number of user requests handled by the application per second. It reflects the **traffic load** on the service and is used to understand how busy the application is over time.
The observed increase and fluctuations indicate varying user demand generated during the test

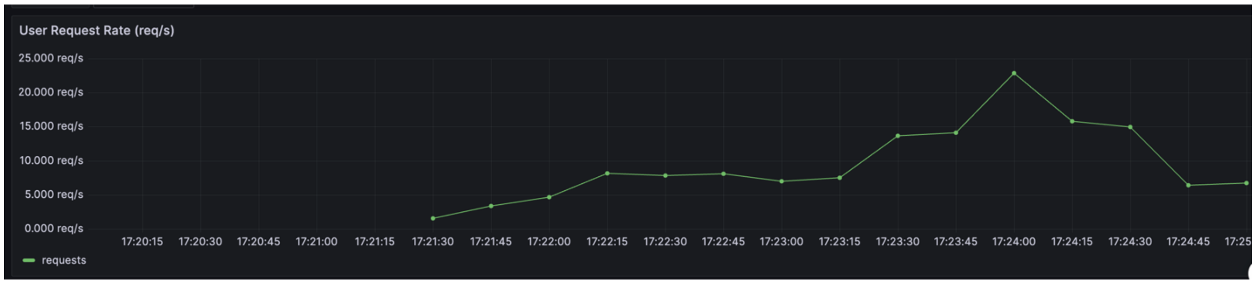


Figure 3 – User 5xx Error Rate (%)

This metric shows the percentage of user requests that failed due to **server-side errors (HTTP 5xx)**.
It indicates the **reliability of the application**, as 5xx errors represent failures caused by the service rather than user input.
The increase in error rate during the test reflects intentionally generated server errors to validate error monitoring.

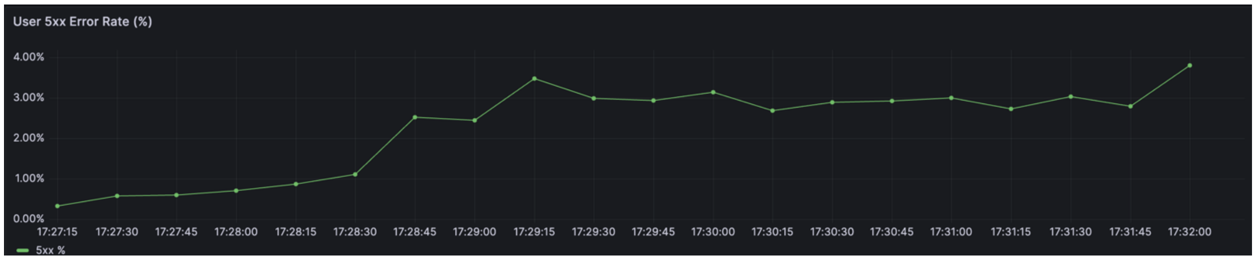


Figure 4 – p95 Latency (User Traffic)

This metric represents the response time within which **95% of user requests are completed**.
It reflects the **user experience under load**, highlighting performance degradation that may not be visible in average latency.
The stable p95 values indicate consistent request handling during the test.

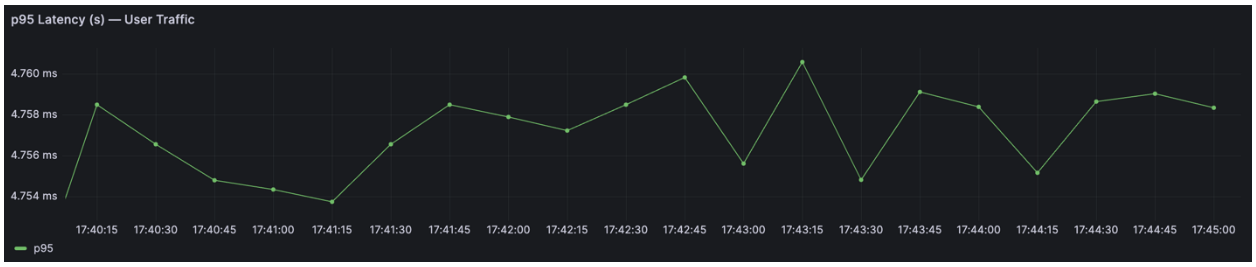
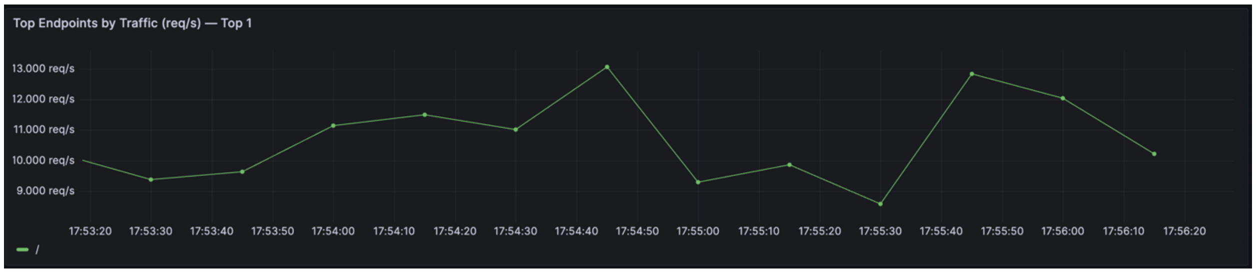


Figure 5 – Top Endpoint by Traffic (req/s)

This metric identifies the **single endpoint receiving the highest request rate** at any point in time.
It helps pinpoint the **hot path** of the application where most traffic is concentrated.

During testing, traffic was intentionally focused on different endpoints (/ and /items).
As a result, each endpoint appeared as the top endpoint at different times, demonstrating how this metric dynamically highlights the most heavily used part of the application.



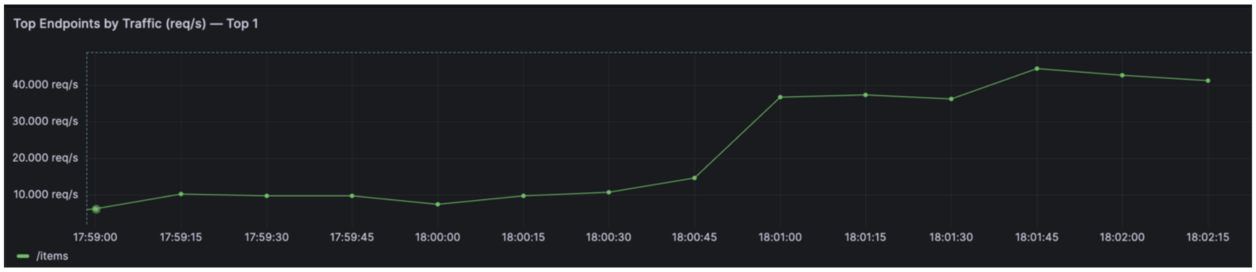
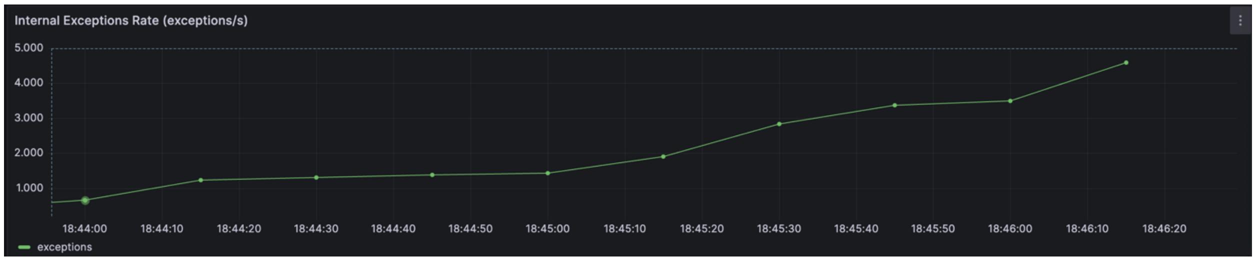
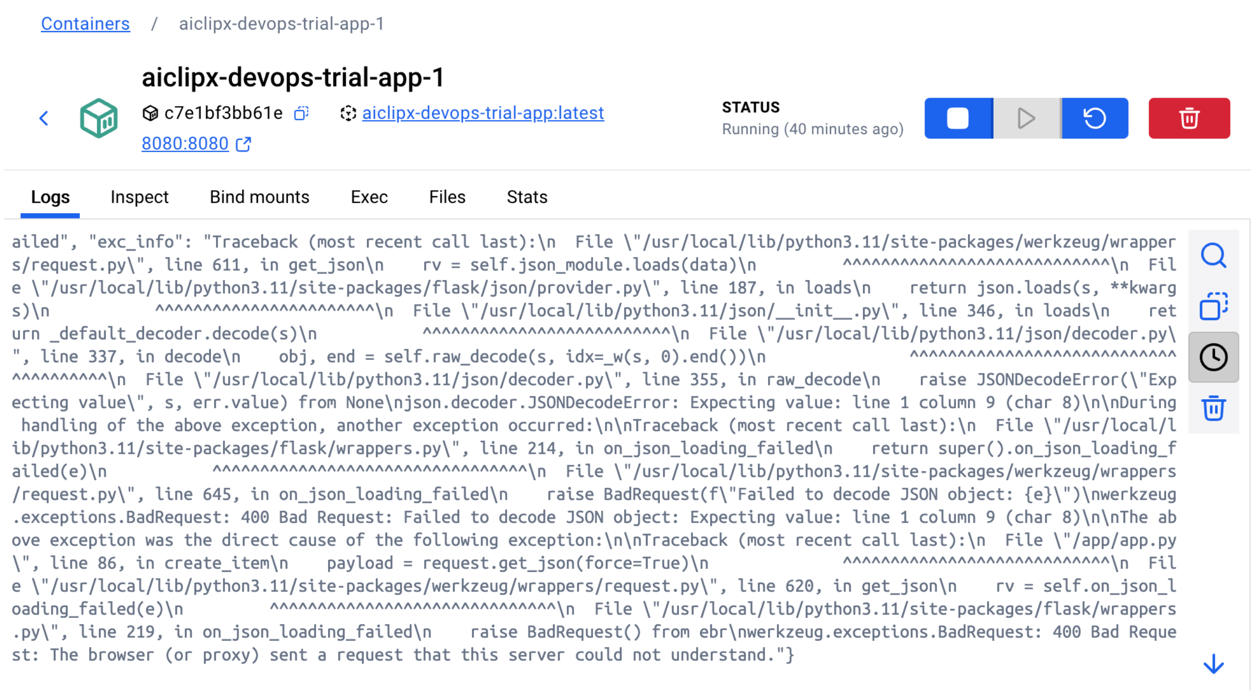


Figure 6 – Internal Exceptions Observability (Metrics & Logs)

This figure demonstrates that internal application errors are **captured in logs and reflected in metrics**.
The exception stack trace in the application logs corresponds to the increase in the **Internal Exceptions Rate** on the Grafana dashboard, confirming correct error observability and monitoring integration.





aiclipx-devops-trial-app-1

c7e1bf3bb61e aiclipx-devops-trial-app:latest

8080:8080

STATUS

Running (40 minutes ago)

```
ailed", "exc_info": "Traceback (most recent call last):\n  File \"/usr/local/lib/python3.11/site-packages/werkzeug/wrapper
s/request.py\", line 611, in get_json\n    rv = self.json_module.loads(data)\n         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n  Fil
e \"/usr/local/lib/python3.11/site-packages/flask/json/provider.py\", line 187, in loads\n    return json.loads(s, **kwarg
s)\n          ^^^^^^^^^^^^^^^^^^^^^^^\n  File \"/usr/local/lib/python3.11/json/__init__.py\", line 346, in loads\n    ret
urn _default_decoder.decode(s)\n          ^^^^^^^^^^^^^^^^^^^^^^^^^^\n  File \"/usr/local/lib/python3.11/json/decoder.py\
", line 337, in decode\n    obj, end = self.raw_decode(s, idx=_w(s, 0).end())\n              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^\n  File \"/usr/local/lib/python3.11/json/decoder.py\", line 355, in raw_decode\n    raise JSONDecodeError(\"Exp
ecting value\", s, err.value) from None\njson.decoder.JSONDecodeError: Expecting value: line 1 column 9 (char 8)\n\nDuring
 handling of the above exception, another exception occurred:\n\nTraceback (most recent call last):\n  File \"/usr/local/l
ib/python3.11/site-packages/flask/wrappers.py\", line 214, in on_json_loading_failed\n    return super().on_json_loading_f
ailed(e)\n          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n  File \"/usr/local/lib/python3.11/site-packages/werkzeug/wrappers
/request.py\", line 645, in on_json_loading_failed\n    raise BadRequest(f\"Failed to decode JSON object: {e}\")\nwerkzeug
.exceptions.BadRequest: 400 Bad Request: Failed to decode JSON object: Expecting value: line 1 column 9 (char 8)\n\nThe ab
ove exception was the direct cause of the following exception:\n\nTraceback (most recent call last):\n  File \"/app/app.py
\", line 86, in create_item\n    payload = request.get_json(force=True)\n              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n  Fil
e \"/usr/local/lib/python3.11/site-packages/werkzeug/wrappers/request.py\", line 620, in get_json\n    rv = self.on_json_l
oading_failed(e)\n         ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^\n  File \"/usr/local/lib/python3.11/site-packages/flask/wrappers
.py\", line 219, in on_json_loading_failed\n    raise BadRequest() from ebr\nwerkzeug.exceptions.BadRequest: 400 Bad Reque
st: The browser (or proxy) sent a request that this server could not understand."}
```