Performance test report - May 9, 2024 (#1)

Open in Postman

Postman collection: express-nodejs-dynamodb-api

Report exported on: May 9, 2024, 20:00:39 (GMT-3)

Test setup

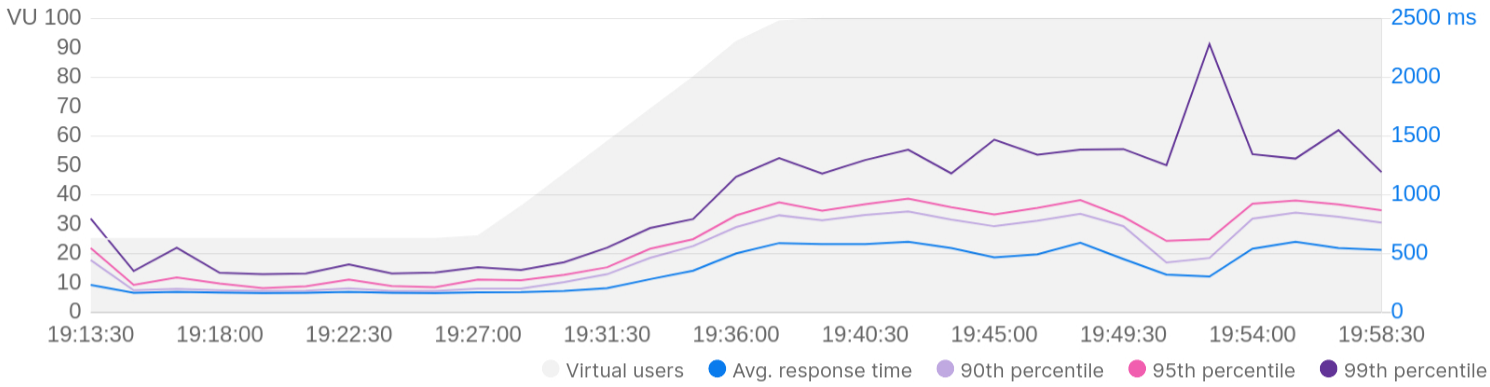
Virtual users	Start time	Load profile
100 VU	May 9, 19:14:47 (GMT-3)	Ramp up (10 minutes)
Duration	**End time**	**Environment**
45 minutes	May 9, 19:59:54 (GMT-3)	express-nodejs-dynamodb-api

1. Summary

Total requests sent	Throughput	Average response time	Error rate
184,226	68.06 requests/second	419 ms	0.10 %

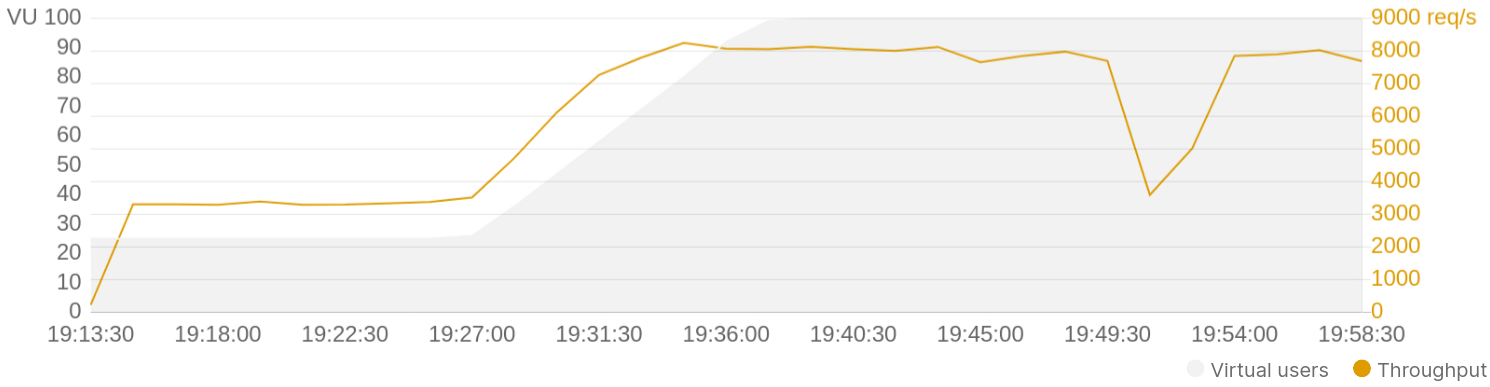
1.1 Response time

Response time trends during the test duration.



1.2 Throughput

Rate of requests sent per second during the test duration.



1.3 Requests with slowest response times

Top 5 slowest requests based on their average response times.

Request	Resp. time (Avg ms)	90th (ms)	95th (ms)	99th (ms)	Min (ms)	Max (ms)
GET getCashInById {{host}}/api/v1/cashIn/id/:id	425	746	855	1,249	143	19,889
POST createCashIn {{host}}/api/v1/cashIn/	413	721	808	1,065	145	19,975

1.4 Requests with most errors

Top 5 requests with the most errors, along with the most frequently occurring errors for each request.

Request	Total error count	Error 1	Error 2	Other errors
GET getCashInById {{host}}/api/v1/cashIn/id/:id	157	ECONNRESET (155)	-	0
POST createCashIn {{host}}/api/v1/cashIn/	28	ECONNRESET (26)	-	0

2. Metrics for each request

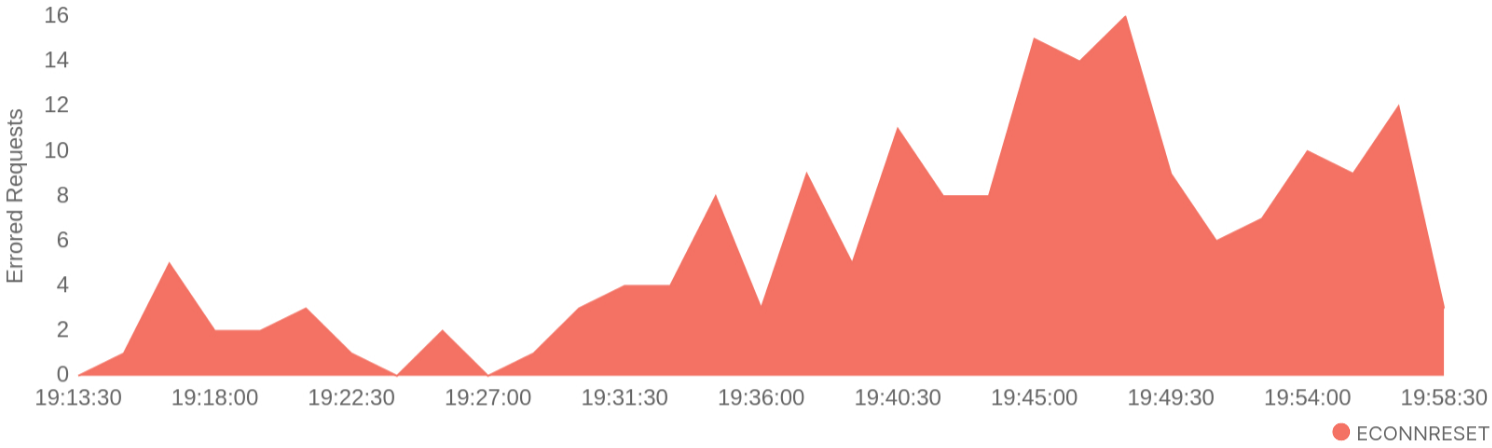
The requests are shown in the order they were sent by virtual users.

Request	Total requests	Requests/s	Min (ms)	Avg (ms)	90th (ms)	Max (ms)	Error %
GET getCashInById {{host}}/api/v1/cashIn/id/:id	92,122	34.03	143	425	746	19,889	0.17
POST createCashIn {{host}}/api/v1/cashIn/	92,104	34.03	145	413	721	19,975	0.03

3. Errors

3.1 Error distribution over time

Top 5 error classes observed during the test duration.



3.2 Error distribution for requests

Errored requests grouped by error class, along with the error count for each class.

Error class	Total counts
ECONNRESET	181
GET getCashInById	155
POST createCashIn	26

Testing API performance on Postman

Postman enables you to simulate user traffic and observe how your API behaves under load. It also helps you identify any issues or bottlenecks that affect performance.

Learn more about testing API performance.