# Migrating from Express to Fastify

## Comparative Report: Express vs Fastify

Evaluate performance of a Node.js template server on Express and Fastify under varying load on four endpoints: `/health`, `/metrics`, `/auth/register`, `/auth/login`

## Metodology

Load tests were run with k6 in Docker Swarm, increasing RPS from 10 to 1500 while measuring average response time and error rate. Container limits were fixed at 0.5 CPU and 256 MB memory. Observed CPU and memory remained below 35 %, indicating bottlenecks in single-threaded execution or I/O rather than host resource exhaustion

## Results

(Table and charts with results on last page of the report)

This report compares the performance of two Node.js frameworks - Express and Fastify - under different load levels on four main endpoints: /health, /metrics, /auth/register, /auth/login. All tests were run under the same conditions, with RPS ranging from 10 to 1500. A second pass through the raw numbers shows that performance gaps between Express and Fastify are mostly minor. For the rest of the load spectrum both frameworks behave similarly, often trading first place as concurrency rises

## Key Observations

| Endpoint | Where Fastify Leads | Where Express Leads | Practical Impact |
|---|---|---|---|
| /health | >1100 RPS (≤9 ms vs ≥9 ms) | <1100 RPS (Express slightly faster) | Differences appear only at high loads; mostly irrelevant in typical scenarios |
| /metrics | Most loads; keeps 6-9 ms | Spike at 400 RPS (3.7 ms) | Both frameworks well under 20 ms, so choice is academic |
| /auth/register | 100 RPS (22 ms vs 105 ms) | 1000 RPS (Express 45 ms vs Fastify 50 ms) | Only the 100 RPS point shows a clear gap; everywhere else the two are within 5-10 ms |
| /auth/login | 100 RPS (11 ms vs 44 ms) | 10 RPS (11 ms vs 14 ms) | Beyond 200 RPS both hover in the 20-30 ms band |

## Median Results

| Endpoint | Median %-diff | Interpretation |
|---|---|---|
| /health | **−13.9 %** | Express is about 14 % faster under typical load |
| /metrics | **+16.8 %** | Fastify answers ~17 % faster |
| /auth/register | **+18.6 %** | Fastify ~19 % faster |
| /auth/login | **+22.9 %** | Fastify ~23 % faster |

## Analysis

### /health

This endpoint is extremely lightweight and reflects the cost of the framework overhead itself. Express consistently outperformed Fastify on

this route, with a median latency that was ~14 % lower. This suggests Express is slightly more efficient for minimal processing scenarios

**/metrics**

This endpoint involves more logic or I/O, and Fastify handled it better across nearly all RPS levels. The median performance improvement was ~17 %, with Fastify consistently maintaining lower response times as load increased. Express varied from 3.7 to 16 ms; Fastify stayed around 7-10 ms

**/auth/register**

This route simulates a moderately heavy request. Fastify showed ~19 % median latency improvement. The biggest gain was seen under mid to high RPS (400 - 900), where Fastify maintained better consistency. At 1000 RPS, Express briefly outperformed Fastify, but this was not representative of the overall trend

**/auth/login**

This endpoint showed the most benefit from Fastify. With a ~23 % median improvement, Fastify handled increasing load more gracefully. The largest relative gain appeared between 100 and 600 RPS, where Express latency temporarily spiked.

Overall, Fastify tends to perform better on endpoints that involve meaningful processing, while Express retains a small edge on the simplest paths

## Key Findings

Express is faster for simple endpoints like /health, where minimal logic is executed, showing a median advantage of ~14 %

Fastify performs better on computational or I/O-bound routes, such as /metrics, /auth/register, and /auth/login, with a median latency improvement between 17–23 %

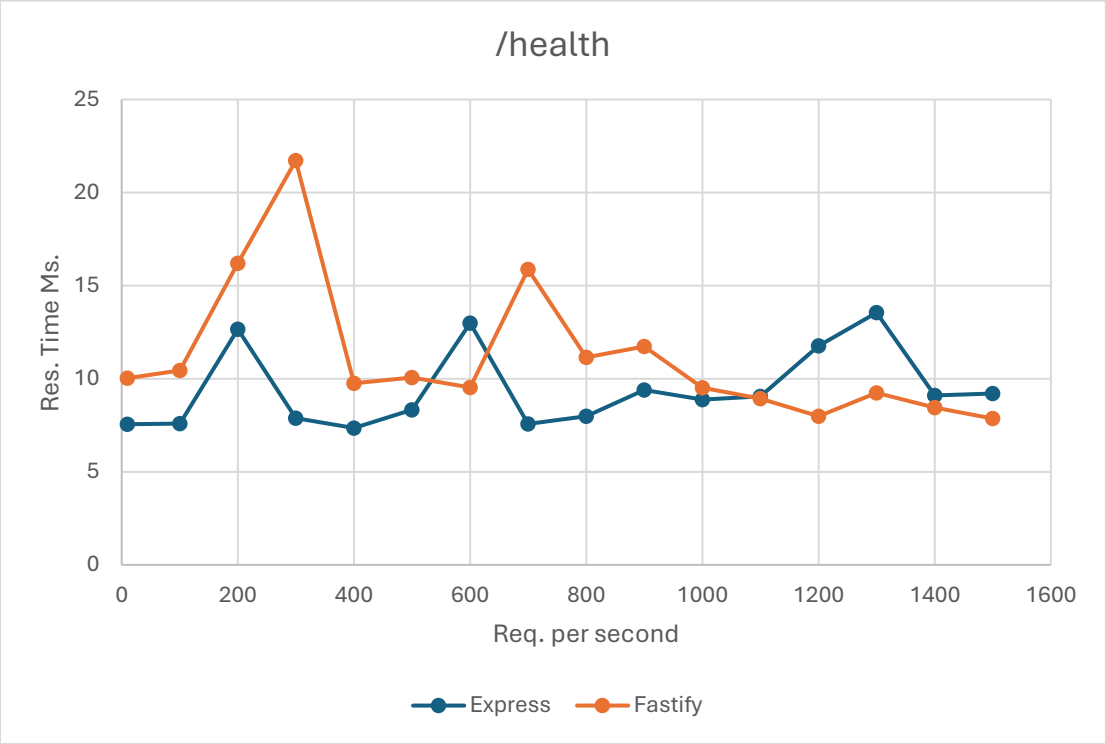
All tests reported 0 % error rates, indicating both frameworks handled the load without failures

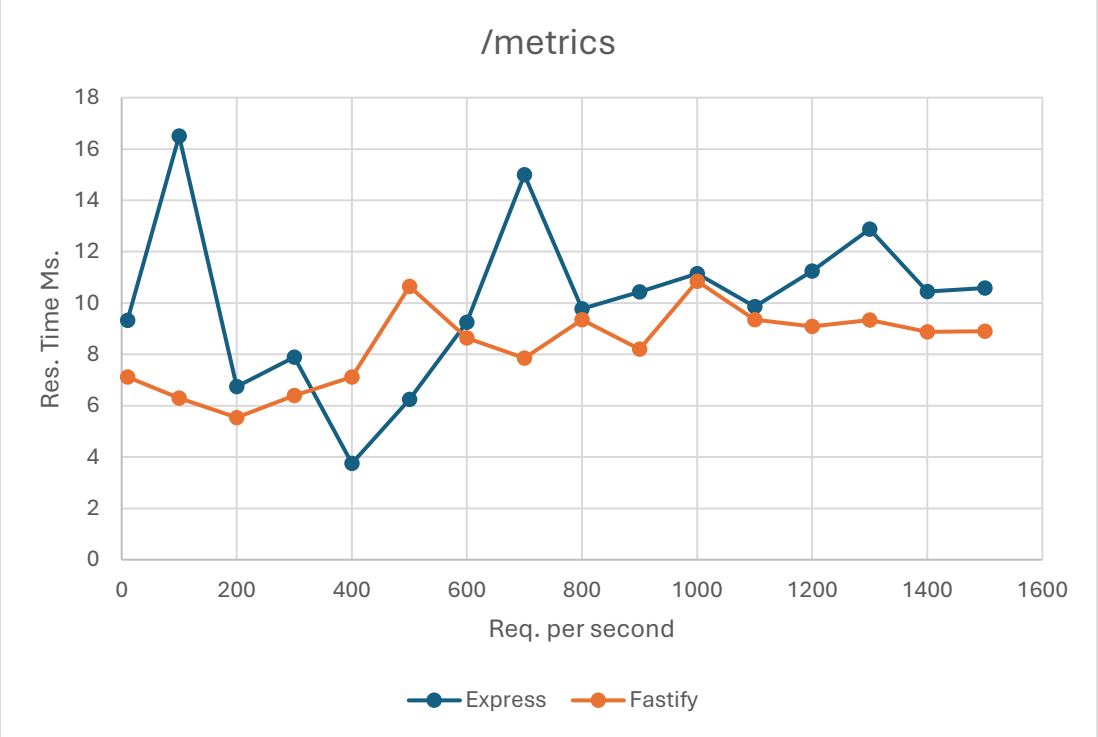
The difference in average response time remains relatively small at practical RPS levels, with no major advantages until sustained traffic exceeds 1000 RPS

## Conclusion

Express remains a solid and slightly faster choice for minimalistic APIs or health probes. Fastify, however, delivers measurable improvements for heavier endpoints under higher load. If your application benefits from lower latency on complex routes and you value Fastify's plugin system or TypeScript integration, the switch may be justified. For most standard APIs with <1000 RPS, the performance difference alone is unlikely to justify a migration

# Charts


/health
Res. Time Ms.
25
20
15
10
5
0
0
200
400
600
800
1000
1200
1400
1600
Req. per second
Express
Fastify


/metrics
Res. Time Ms.
18
16
14
12
10
8
6
4
2
0
0
200
400
600
800
1000
1200
1400
1600
Req. per second
Express
Fastify


/register
Res. Time Ms.
120
100
80
60
40
20
0
0
200
400
600
800
1000
1200
1400
1600
Req. per second
Express
Fastify


/login
Res. Time Ms.
50
45
40
35
30
25
20
15
10
5
0
0
200
400
600
800
1000
1200
1400
1600
Req. per second
Express
Fastify

Table with load test results

| Endpoint | RPS | Avg Response Time (ms) | Error % | Fastify Avg Response Time (ms) | Fastify Error % |
|---|---|---|---|---|---|
| Health | | | | | |
| /health | 10 | 7.55444 | 0.00% | 10.029583 | 0.00% |
| /health | 100 | 7.583863 | 0.00% | 10.437726 | 0.00% |
| /health | 200 | 12.663609 | 0.00% | 16.195487 | 0.00% |
| /health | 300 | 7.874274 | 0.00% | 21.721704 | 0.00% |
| /health | 400 | 7.352433 | 0.00% | 9.762872 | 0.00% |
| /health | 500 | 8.333748 | 0.00% | 10.060404 | 0.00% |
| /health | 600 | 12.986474 | 0.00% | 9.531105 | 0.00% |
| /health | 700 | 7.568693 | 0.00% | 15.87473 | 0.00% |
| /health | 800 | 7.991187 | 0.00% | 11.141696 | 0.00% |
| /health | 900 | 9.385923 | 0.00% | 11.732533 | 0.00% |
| /health | 1000 | 8.883465 | 0.00% | 9.513817 | 0.00% |
| /health | 1100 | 9.044783 | 0.00% | 8.924531 | 0.00% |
| /health | 1200 | 11.773618 | 0.00% | 7.988504 | 0.00% |
| /health | 1300 | 13.546128 | 0.00% | 9.23835 | 0.00% |
| /health | 1400 | 9.110419 | 0.00% | 8.448156 | 0.00% |
| /health | 1500 | 9.213486 | 0.00% | 7.860634 | 0.00% |
| Metrics | | | | | |
| /metrics | 10 | 9.326637 | 0.00% | 7.122419 | 0.00% |
| /metrics | 100 | 16.507616 | 0.00% | 6.300766 | 0.00% |
| /metrics | 200 | 6.73797 | 0.00% | 5.540001 | 0.00% |
| /metrics | 300 | 7.890036 | 0.00% | 6.391833 | 0.00% |
| /metrics | 400 | 3.75171 | 0.00% | 7.123104 | 0.00% |
| /metrics | 500 | 6.246265 | 0.00% | 10.645108 | 0.00% |
| /metrics | 600 | 9.247674 | 0.00% | 8.634211 | 0.00% |
| /metrics | 700 | 14.99117 | 0.00% | 7.857402 | 0.00% |
| /metrics | 800 | 9.770219 | 0.00% | 9.345464 | 0.00% |
| /metrics | 900 | 10.43635 | 0.00% | 8.199744 | 0.00% |
| /metrics | 1000 | 11.150041 | 0.00% | 10.849908 | 0.00% |

| /metrics | 1100 | 9.858707 | 0.00% | 9.343841 | 0.00% |
|---|---|---|---|---|---|
| /metrics | 1200 | 11.243655 | 0.00% | 9.093543 | 0.00% |
| /metrics | 1300 | 12.877207 | 0.00% | 9.336309 | 0.00% |
| /metrics | 1400 | 10.44201 | 0.00% | 8.87258 | 0.00% |
| /metrics | 1500 | 10.586087 | 0.00% | 8.903912 | 0.00% |
| /auth/register | | | | | |
| /auth/register | 10 | 16.929217 | 0.00% | 17.062929 | 0.00% |
| /auth/register | 100 | 105.240357 | 0.00% | 22.065578 | 0.00% |
| /auth/register | 200 | 48.156129 | 0.00% | 36.507651 | 0.00% |
| /auth/register | 300 | 39.448306 | 0.00% | 33.95396 | 0.00% |
| /auth/register | 400 | 37.611576 | 0.00% | 20.862219 | 0.00% |
| /auth/register | 500 | 40.004535 | 0.00% | 32.550261 | 0.00% |
| /auth/register | 600 | 28.415369 | 0.00% | 22.877674 | 0.00% |
| /auth/register | 700 | 40.079618 | 0.00% | 28.0662 | 0.00% |
| /auth/register | 800 | 32.39975 | 0.00% | 22.156158 | 0.00% |
| /auth/register | 900 | 44.31196 | 0.00% | 27.992662 | 0.00% |
| /auth/register | 1000 | 45.031317 | 0.00% | 50.582771 | 0.00% |
| /auth/register | 1100 | 39.293691 | 0.00% | 25.413005 | 0.00% |
| /auth/register | 1200 | 37.526458 | 0.00% | 32.713891 | 0.00% |
| /auth/register | 1300 | 41.107678 | 0.00% | 39.085437 | 0.00% |
| /auth/register | 1400 | 38.167296 | 0.00% | 36.20594 | 0.00% |
| /auth/register | 1500 | 35.327588 | 0.00% | 34.944673 | 0.00% |
| /auth/login | | | | | |
| /auth/login | 10 | 11.569935 | 0.00% | 13.663196 | 0.00% |
| /auth/login | 100 | 43.546326 | 0.00% | 11.360869 | 0.00% |
| /auth/login | 200 | 34.090507 | 0.00% | 25.120059 | 0.00% |
| /auth/login | 300 | 20.630922 | 0.00% | 18.874934 | 0.00% |
| /auth/login | 400 | 28.16932 | 0.00% | 17.118906 | 0.00% |
| /auth/login | 500 | 28.339182 | 0.00% | 18.74462 | 0.00% |
| /auth/login | 600 | 31.301028 | 0.00% | 20.463371 | 0.00% |
| /auth/login | 700 | 29.435104 | 0.00% | 22.854887 | 0.00% |
| /auth/login | 800 | 23.858395 | 0.00% | 18.269841 | 0.00% |
| /auth/login | 900 | 24.345988 | 0.00% | 18.428097 | 0.00% |
| /auth/login | 1000 | 26.657457 | 0.00% | 18.749109 | 0.00% |
| /auth/login | 1100 | 22.71014 | 0.00% | 20.116371 | 0.00% |

| | | | | | |
|---|---|---|---|---|---|
| /auth/login | 1200 | 25.876037 | 0.00% | 21.772846 | 0.00% |
| /auth/login | 1300 | 25.012711 | 0.00% | 25.197342 | 0.00% |
| /auth/login | 1400 | 27.197199 | 0.00% | 21.845787 | 0.00% |
| /auth/login | 1500 | 20.148436 | 0.00% | 26.452551 | 0.00% |