TEST REPORT · PERFORMANCE AND LOAD

● MEETS SLOS AT EXPECTED PEAK

Acme Ledger API · Load, soak, spike and stress

Environment	Performance env sized as production (2 API pods × 4 vCPU; Postgres 8 vCPU)
Build	4.2.0-rc3
Tool	k6 v0.54 · scripts in `acme/qa-perf` · 2 load generators (eu-central)
Executed	2026-10-21 – 2026-10-22
Traffic model	Mix from 7 days of production logs; 24 endpoints, think time 3–8 s
SLOs	p95 < 500 ms for API calls, p95 < 2 s for PDFs and reports, errors < 0.5%
Prepared by	Tomasz Kordiak (QA lead)

Contents

1. Summary
2. Key numbers
3. Latency as load increases
4. Scenarios
5. Defects
6. Observations
7. Thresholds
8. Ramp steps
9. By endpoint and scenario
10. Errors
11. Resource usage
12. Soak trend
13. Bottlenecks
14. Recommendations
15. Environment and method

Summary

Ready for expected peak, with 1.5× headroom

At the expected peak (300 concurrent users, 236 req/s) all SLOs are met: p95 447 ms, errors 0.04%. The system stays within SLOs up to about **450 users**.

Beyond ~550 users the database connection pool is exhausted and errors climb to 6.8%. The invoice PDF endpoint is the first to degrade: 1.98 s at expected peak against a 2 s SLO.

Recommendation: release 4.2 as planned; fix ACME-525 and ACME-526 before the January billing peak, which marketing expects at roughly 2× today's traffic.

Key numbers

236 req/s
At expected peak

447 ms
p95 at peak

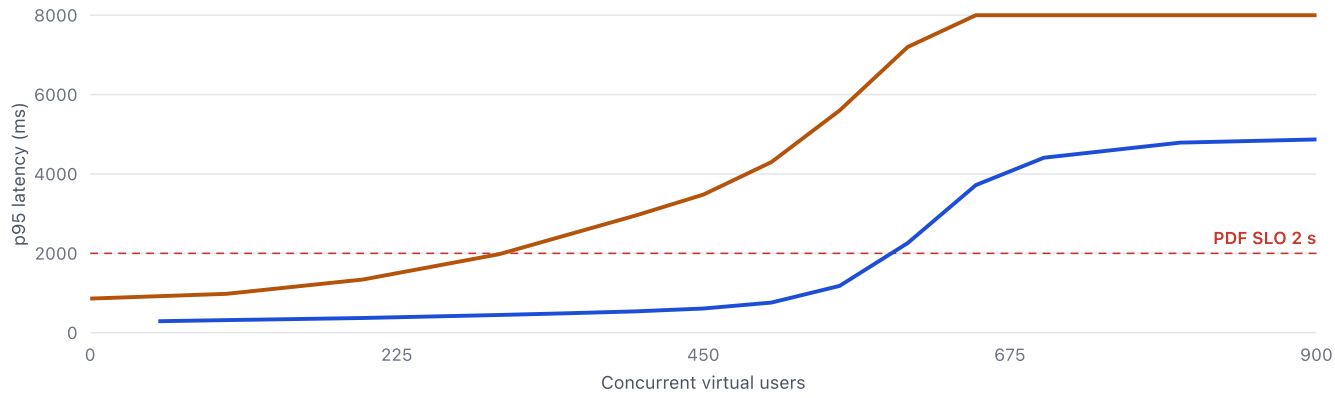
0.04%
Errors at peak

~450
Users within SLO

~550
Breaking point

16/20
Thresholds passed

Latency as load increases



Stress ramp, 0 → 900 users over 25 minutes. PDF requests (6% of traffic) time out at 8 s above 650 users.

Scenarios

SCENARIO	PROFILE	REQ/S	P95 (API)	ERRORS	SLO
Baseline	50 VUs · 10 min	41	290 ms	0.00%	Met
Load (expected peak)	ramp to 300 VUs · 30 min	236	447 ms	0.04%	Met
Soak	200 VUs · 2 h	158	402 ms	0.02%	Met
Spike	50 → 600 VUs in 30 s · 5 min	410	1,840 ms	1.90%	Missed briefly

SCENARIO	PROFILE	REQ/S	P95 (API)	ERRORS	SLO
Stress	ramp to 900 VUs · 25 min	512	4,870 ms	6.80%	Missed (by design)

Stress and spike are meant to go past the limits; they show where and how the system fails.

Defects 6 raised

KEY	SUMMARY	SEVERITY	STATUS	COMPONENT	FOUND BY	OWNER	CREATED
ACME-525	DB connection pool exhausted above ~550 concurrent users (503s) Planned before the January peak.	High	Open	API platform	Stress	Platform	22 Oct
ACME-526	Invoice PDF rendered on every request; p95 1.98 s at peak	Medium	Open	PDF	Load	Backend	22 Oct
ACME-527	Unhandled timeout returns 500 instead of 503 under stress	Medium	Open	API platform	Stress	Platform	22 Oct
ACME-528	Search p95 doubles with tenant size; no index on `invoices.number`	Medium	Open	Search	Load	Backend	22 Oct
ACME-529	Revenue report scans all invoices of the tenant	Low	Open	Reports	Load	Backend	22 Oct
ACME-530	Spike: autoscaling adds pods after 90 s, too late for 30 s bursts	Medium	Open	Infrastructure	Spike	Platform	22 Oct

Observations

ID	AREA	OBSERVATION	IMPACT	SUGGESTED ACTION
`OBS-01`	Memory	Memory grew 16 MB in 2 hours (0.3%/h) and flattened after GC; no leak.	Positive	–
`OBS-02`	Database	DB CPU stays below 50% even under stress; the database is not the bottleneck.	Info	Don't scale the database; fix the pool (ACME-525).
`OBS-03`	Caching	`GET /products` and `/users/me` are good cache candidates (rarely change, 8% of traffic).	Low	Add HTTP caching headers.
`OBS-04`	Rate limiting	Per-client rate limit (100/min) never triggered: load came from many simulated clients.	Info	Test one heavy API client separately if a customer integrates in bulk.
`OBS-05`	Logs	Under stress, request logging produced 3 GB per hour.	Cost	Sample 2xx logs at 10% above 300 rps.
`OBS-06`	Front end	Not in scope: browser rendering and Core Web Vitals.	Coverage	Lighthouse CI on key pages (separate task).

Thresholds 16 of 20 passed

K6 THRESHOLD	SCENARIO	MEASURED	RESULT
`http_req_duration p(95) < 500 ms (API)`	Load	447 ms	Pass
`http_req_duration p(99) < 1000 ms (API)`	Load	812 ms	Pass
`http_req_failed rate < 0.5%`	Load	0.04%	Pass
`checks rate > 99.5%`	Load	99.96%	Pass
`pdf p(95) < 2000 ms`	Load	1,978 ms	Pass
`reports p(95) < 2000 ms`	Load	1,620 ms	Pass
`login p(95) < 500 ms`	Load	210 ms	Pass
`iteration_duration p(95) < 25 s`	Load	19.4 s	Pass

K6 THRESHOLD	SCENARIO	MEASURED	RESULT
`http_req_failed rate < 0.5%`	Soak	0.02%	Pass
`memory growth < 5% per hour`	Soak	0.3%	Pass
`http_req_duration p(95) < 500 ms (API)`	Soak	402 ms	Pass
`db connections < 80% of pool`	Soak	65%	Pass
`http_req_failed rate < 2% during spike`	Spike	1.9%	Pass
`recovery to baseline p95 < 60 s`	Spike	38 s	Pass
`http_req_duration p(95) < 1500 ms during spike`	Spike	1,840 ms	Fail
`no 5xx after spike ends`	Spike	0	Pass
`http_req_failed rate < 1%`	Stress	6.8%	Fail
`http_req_duration p(95) < 2000 ms`	Stress	4,870 ms	Fail
`graceful degradation (429/503, no 500)`	Stress	312 × 500	Fail
`service recovers within 2 min after stress`	Stress	70 s	Pass

Ramp steps

VUS	REQ/S	P50	P95	ERRORS	API CPU	DB CPU	DB CONNS / POD
50	41	118 ms	290 ms	0%	18%	22%	6 / 20
100	82	124 ms	318 ms	0%	27%	24%	9 / 20
150	120	131 ms	344 ms	0%	36%	27%	11 / 20
200	158	141 ms	371 ms	0%	45%	30%	13 / 20
250	197	152 ms	409 ms	0.01%	55%	33%	16 / 20
300	236	164 ms	447 ms	0.04%	63%	35%	18 / 20
350	272	178 ms	490 ms	0.06%	70%	37%	19 / 20
400	306	196 ms	538 ms	0.09%	76%	39%	20 / 20
450	338	221 ms	610 ms	0.18%	81%	41%	20 / 20

VUS	REQ/S	P50	P95	ERRORS	API CPU	DB CPU	DB CONNS / POD
500	366	262 ms	760 ms	0.6%	84%	42%	20 / 20
550	384	340 ms	1,180 ms	1.9%	85%	43%	20 / 20
600	392	520 ms	2,260 ms	3.8%	85%	44%	20 / 20
650	398	760 ms	3,720 ms	5.1%	86%	45%	20 / 20
700	401	905 ms	4,410 ms	5.9%	86%	45%	20 / 20
800	405	1,080 ms	4,790 ms	6.4%	86%	46%	20 / 20
900	408	1,210 ms	4,870 ms	6.8%	87%	46%	20 / 20

Throughput flattens at ~400 req/s once all 20 connections per pod are busy, while DB CPU stays under 50%.

By endpoint and scenario 120 results

ENDPOINT	SHARE	SLO	BASELINE	LOAD	SOAK	SPIKE	STRESS
`GET /invoices`	19%	500 ms	210	336	311	638	2,083
`GET /invoices/{id}`	12%	500 ms	95	143	133	271	884
`POST /invoices`	5%	500 ms	260	468	426	889	2,902
`PATCH /invoices/{id}`	2%	500 ms	230	391	359	743	2,424
`POST /invoices/{id}/send`	2%	500 ms	300	480	444	912	2,976
`GET /invoices/{id}/pdf`	6%	2,000 ms	860	1,978	1,754	3,758	12,264
`GET /customers`	8%	500 ms	140	224	207	426	1,389
`GET /customers/{id}`	5%	500 ms	80	112	106	213	694
`POST /customers`	1%	500 ms	190	304	281	578	1,885
`GET /products`	4%	500 ms	90	117	112	222	725
`GET /payments`	4%	500 ms	150	240	222	456	1,488
`POST /payments`	4%	500 ms	220	396	361	752	2,455

ENDPOINT	SHARE	SLO	BASELINE	LOAD	SOAK	SPIKE	STRESS
`POST /payments/{id}/refund`	1%	500 ms	280	476	437	904	2,951
`GET /credit-notes`	2%	500 ms	140	210	196	399	1,302
`GET /recurring`	2%	500 ms	120	168	158	319	1,042
`GET /reports/revenue`	3%	2,000 ms	640	1,120	1,024	2,128	6,944
`GET /reports/aged-receivables`	1%	2,000 ms	720	1,296	1,181	2,462	8,035
`GET /reports/revenue.csv`	1%	2,000 ms	900	1,710	1,548	3,249	10,602
`GET /dashboard/summary`	6%	500 ms	260	442	406	840	2,740
`GET /users/me`	4%	500 ms	60	78	74	148	484
`POST /oauth/token`	4%	500 ms	140	210	196	399	1,302
`GET /webhooks`	1%	500 ms	90	117	112	222	725
`POST /files`	1%	500 ms	180	288	266	547	1,786
`GET /search`	2%	500 ms	230	483	432	918	2,995

p95 in ms. 120 endpoint × scenario results: green within 90% of the SLO, amber close to it, red over it.

Errors

SCENARIO	500	502/504 (TIMEOUTS)	503 (POOL)	K6 TIMEOUTS	TOTAL
Baseline	–	–	–	–	–
Load (expected peak)	–	61	–	–	61
Soak	–	–	–	228	228
Spike	–	1,104	2,876	–	3,980
Stress	312	4,870	28,140	22,410	55,732

500s under stress come from unhandled timeouts (ACME-527); they should be 503 with Retry-After.

Resource usage

SCENARIO	API CPU (MAX)	API MEMORY	DB CPU (MAX)	DB CONNECTIONS / POD	QUEUE DEPTH
Baseline	18%	410 MB	11%	6 / 20	0
Load (expected peak)	63%	520 MB	35%	18 / 20	0
Soak	46%	505 → 521 MB	30%	13 / 20	0
Spike	85%	560 MB	41%	20 / 20	1,400 peak
Stress	87%	590 MB	46%	20 / 20	9,800 peak

Soak trend

WINDOW	REQ/S	P95	ERRORS	API MEMORY
0:00–0:15	155	398 ms	0.00%	505 MB
0:15–0:30	158	401 ms	0.02%	507 MB
0:30–0:45	158	399 ms	0.01%	509 MB
0:45–1:00	159	404 ms	0.03%	511 MB
1:00–1:15	158	402 ms	0.02%	513 MB
1:15–1:30	157	405 ms	0.02%	516 MB
1:30–1:45	158	403 ms	0.03%	518 MB
1:45–2:00	158	406 ms	0.02%	521 MB

Stable for 2 hours at 200 users: no latency drift, memory flat after garbage collection.

Bottlenecks

High PERF-01 · ACME-525 **Database connection pool exhausted above ~550 users**

Observed	From 550 users, `503` and `timeout acquiring connection`; errors 6.8% at 900 users. DB CPU stays at 46%.
Evidence	Pool metrics: 20/20 connections busy for 11 minutes; API pod logs; Grafana snapshot attached.
Recommendation	Pool 40 per pod (DB allows 200) and PgBouncer; re-run stress. Expected breaking point ~900 users.

Medium PERF-02 · ACME-526 **Invoice PDF is rendered on every request**

Observed	p95 1.98 s at expected peak (SLO 2 s), 4.3 s at 500 users; API CPU 85% when the PDF share rises.
Recommendation	Cache the rendered PDF per invoice version, or render on invoice creation.

Medium PERF-03 · ACME-530 **Autoscaling reacts too slowly to spikes**

Observed	New pods ready after 90 s; a 30-second burst to 600 users pushed p95 to 1.84 s and errors to 1.9%.
Recommendation	Scale on queue depth, pre-scale before known peaks (month-end, campaigns).

Recommendations

CHANGE	DEFECT	EXPECTED GAIN	EFFORT
Raise DB pool to 40 per pod, add PgBouncer	`ACME-525`	Breaking point ~550 → ~900 users	1–2 days
Cache rendered PDFs per invoice version	`ACME-526`	PDF p95 1.98 s → < 300 ms (cached)	2–3 days
Return 503 with Retry-After on timeouts	`ACME-527`	No 500s under overload; clients retry correctly	0.5 day
Index `invoices(tenant_id, number)` for search	`ACME-528`	Search p95 −60% on large tenants	0.5 day
Pre-scale before known peaks; scale on queue depth	`ACME-530`	Spike p95 within 1.5 s	1 day

CHANGE	DEFECT	EXPECTED GAIN	EFFORT
Composite index on `(tenant_id, issued_at)`	`ACME-529`	Report p95 1.6 s → ~0.6 s	0.5 day

Environment and method

Test data	40 tenants, 210,000 invoices, 9,000 customers from the seed script; no production data
Traffic model	Endpoint shares and think times from 7 days of anonymised access logs
Load generators	2 × c6i.2xlarge in eu-central-1, same region as the environment
Monitoring	Grafana (API, DB, pool metrics), k6 output to Prometheus
Not covered	Browser rendering and Core Web Vitals, third-party payment latency (stubbed), multi-region traffic
What you keep	k6 scripts, thresholds and the Grafana dashboard in your repository; a 50-VU load test runs nightly as a regression check

Sample report. Acme Ledger, its data, people and defects are fictional. Your report shows your product, your tickets and your tools.

More sample reports · Book a 15-minute call