CapregSoft

Backend & AI Engineering

The Backend Scaling Checklist

The 12 checks we run before a system hits 10k req/sec.

01 **Load test before you ship**

Prove your target throughput (req/sec) under realistic traffic on real infra don't discover the ceiling in production.

02 **Find the actual bottleneck**

Profile CPU, I/O, locking, and N+1 queries with data. Most 'scaling' problems are one hot path, not the whole system.

03 **Add caching at the right layer**

Redis for hot reads, HTTP caching at the edge, and query result caching. Cache invalidation strategy decided up front.

04 **Index and tune the database**

Indexes on every query path, connection pooling sized correctly, and slow-query logging on in production.

05 **Make services stateless**

Move session/state out of the app process (Redis/DB) so you can scale horizontally instead of vertically.

06 **Set timeouts and retries everywhere**

Every external call has a timeout, bounded retries with backoff, and a circuit breaker. No unbounded waits.

The Backend Scaling Checklist

Part 2 — checks 7 to 12.

07 **Add backpressure to queues**

Producers slow down before queues overflow. Dead-letter handling for poison messages. Kafka/NATS sized for peak.

08 **Autoscale on the right signal**

Scale on the metric that actually correlates with load (queue depth, p95 latency) — not just CPU.

09 **Instrument with metrics + tracing**

p50/p95/p99 latency, error rates, and distributed traces. You can't fix what you can't see.

10 **Harden security (OWASP Top 10)**

Input validation, secrets in a vault (never in code), least-privilege DB users, and dependency scanning.

11 **Plan zero-downtime deploys**

Rolling/blue-green deploys, backward-compatible migrations, and health checks that gate traffic.

12 **Ship tests + docs with the code**

80%+ coverage on critical paths and a runbook, so the next engineer (or you at 3am) can change it safely.

Stack buckling under load? Let's fix it.

Book a 30-minute technical discovery call at capregsoft.com/contact