High Availability Review Checklist

Check only what you can prove (implemented, tested, and observable), not what "should" be there.

01 Redundancy and topology

- [ ] No critical component is a single point of failure (compute, network, data, DNS, secrets).
- [ ] Replicas spread across distinct availability zones; the system tolerates losing one zone without breaching the SLO.
- [ ] Load balancing removes unhealthy instances automatically and never routes to a target still recovering.
- [ ] The critical path does not depend on a shared resource with no replica (a single database, queue, or cache).

02 Health checks and failover

- [ ] Liveness and readiness are distinct (restarting ≠ being ready to receive traffic).
- [ ] Failover was tested with a real outage, not just configured; detection + switchover time is measured.
- [ ] Flapping is prevented (hysteresis on thresholds; a partial recovery does not reopen all traffic at once).
- [ ] A partially degraded health signal does not drag the whole process to DOWN.

03 Capacity and load control

- [ ] There is headroom for the worst case: losing a zone during a traffic peak.
- [ ] Pools, threads, connections, and queues have explicit, sized limits.
- [ ] Autoscaling has upper and lower bounds; it cannot amplify an outage.
- [ ] The system was tested near its load limit, not only under normal conditions.

04 Dependency resilience

- [ ] Every remote call has a per-attempt timeout and the operation a propagated end-to-end deadline.
- [ ] Retries use backoff with jitter and a bounded budget (the worst-case total number of attempts is known).
- [ ] Circuit breakers watch the right signal (failure rate and slow-call rate) with a minimum sample.
- [ ] Resource isolation exists (bulkhead/dedicated pool) so a slow dependency does not starve the rest.
- [ ] There is a defined functional degradation for when a dependency does not respond.

05 SLOs and error budget

- [ ] There are SLOs per critical operation, not a single global number.
- [ ] The error budget is defined and its consumption is monitored.
- [ ] Alerts are based on symptoms and SLOs, not on resource metrics with no impact (CPU alone, for example).
- [ ] There is a policy for what gets frozen or prioritized when the budget is exhausted.

06 Observability

- [ ] RED metrics per service (rate, errors, duration) and USE per resource (utilization, saturation, errors).
- [ ] Distributed traces with context propagation; a request can be followed across services.
- [ ] Structured logs correlatable by trace identifier.
- [ ] Dashboards answer on-call questions (what is failing, where, since when), not just show graphs.
- [ ] Telemetry does not disappear during the outage (sufficient retention and sampling).

07 Incident response

- [ ] Runbooks are executable and current, not a dead wiki.
- [ ] There is an on-call rotation and a clear escalation path.
- [ ] Mitigations are reversible (feature flags, rollback, rate limit, traffic shedding).
- [ ] Every incident closes with a blameless postmortem whose corrective actions are actually carried out.

08 Data, deployment, and continuous validation

- [ ] Backups are actually restored periodically; RPO and RTO are defined and measured.
- [ ] Critical writes are idempotent or deduplicated (a lost response does not duplicate the effect).
- [ ] Deployments are gradual (canary or blue-green), with a health gate and one-step rollback.
- [ ] Configuration changes are treated with the same care as a code deployment.
- [ ] Recovery drills are run (game days / DR drills) and alerts are verified to actually fire.