Origin Shielding & Resilience Checklist

Before a streaming launch or live event: classify and size the origin, put one shield in front of it, set TTLs so the shield holds, and design failover so one server is never the single point of failure. Engineering guidance — confirm CDN defaults live, they change.

1 · CLASSIFY & SIZE THE ORIGIN

- ☐ **Know which origin you have.** Static (object storage, pre-packaged segments) fears bytes/egress. Just-in-time packaging (MediaPackage and peers) builds each segment on demand and fears requests — each miss is a CPU job.
- ☐ **Size a JIT origin for request rate, not just egress.** An egress-only capacity plan misses the packaging-CPU spike a live start creates. Origin type: _____ peak req/s budget: _____
- ☐ **Map the fan-in.** Count regions × CDNs that can miss the same new segment. That is how many duplicate origin requests you get per segment without a shield.

2 · ENABLE & PLACE THE SHIELD (one consolidation point)

- ☐ **Turn on shielding** (CloudFront Origin Shield / Cloudflare Tiered Cache / Akamai Tiered Distribution / Fastly Shielding). All origin-bound misses route through one place that fetches the origin once per object.
- ☐ **Place the shield nearest the ORIGIN, not the viewers.** Pick the region/POP with lowest latency to your origin; for cloud origins behind anycast, set a cloud region hint so the right tier is chosen.
- ☐ **In multi-CDN, make one shielded CDN the shared front door** so every CDN's misses funnel through it (and share a common cache key). Shield location: _____

3 · SET TTL SO THE SHIELD HOLDS

- ☐ **Long TTL on immutable segments, short TTL on the live manifest.** The shield can only consolidate what it is allowed to keep; a too-short segment TTL re-misses to origin needlessly.
- ☐ **Version by new path for changes; purge for removal.** A new path fills the shield cleanly; surrogate keys / cache tags purge a whole title in one call.
- ☐ **Watch byte-based origin offload, not just request hit ratio** — shield hits after an edge miss can look like misses while the origin never saw them.

4 · FAILOVER & MULTI-REGION ORIGIN

- ☐ **A shield in front of one origin caches the outage.** Add an origin group: primary + secondary, failover on chosen status codes (e.g. 502 / 503 / 504), for GET / HEAD / OPTIONS.
- ☐ **Tighten failover timing for live.** The web-tuned default can mean ~30 s of black screen; shorten origin timeouts and connection attempts so a switch takes seconds.
- ☐ **Put the secondary in a separate failure domain.** Different AZ minimum, different region for high-stakes events; active-active behind a load balancer removes the gap. Failover codes: ___ timeout: ___

THE ONE SANITY CHECK BEFORE A LIVE EVENT

A CDN is wide at the edge and narrow at the origin: traffic fans out to viewers and fans in to one server. Worked: a live event to 100,000 viewers on 4-second segments, delivered through 8 regional caches across 3 CDNs. Each new segment is released every 4 seconds, and the HLS manifest makes every player re-fetch to find it, so all the regions cold-miss the brand-new segment together — the thundering herd. Without a shield, that is about 8 regions × 3 CDNs = 24 duplicate origin requests per segment, and for a just-in-time origin each is a packaging job — ~6/second, multiplied across every rendition in the ladder. Put one origin shield in front and it consolidates all 24 into one request per object: the origin sees ~1 fetch per segment instead of 24, a ~24× cut for zero lost value because every one of those requests returns identical bytes. Then make sure a second origin exists behind a tightened origin group, so a dead origin is a failover, not a dead broadcast. Figures illustrative; confirm CDN defaults and names live, they change.