Origin shielding и устойчивость — чек-лист

До запуска стриминга или live-события: классифицируйте и размерьте origin, поставьте перед ним один shield, задайте TTL так, чтобы shield держал кэш, и спроектируйте failover, чтобы один сервер не был единой точкой отказа. Инженерное руководство — дефолты CDN сверяйте вживую, они меняются.

1 · КЛАССИФИЦИРУЙТЕ И РАЗМЕРЬТЕ ORIGIN

- ☐ **Поймите, какой у вас origin.** Статический (object storage, готовые сегменты) боится байт/egress. Just-in-time packaging (MediaPackage и аналоги) собирает сегмент на лету и боится запросов — каждый промах это задача CPU.
- ☐ **Размеряйте JIT-origin по числу запросов, а не только по egress.** План «только egress» не видит всплеска packaging-CPU на старте live. Тип origin: _____ бюджет пиковых req/s: _____
- ☐ **Посчитайте fan-in.** Регионы × CDN, что промахнутся по одному новому сегменту. Столько дублирующих запросов к origin на сегмент вы получите без shield.

2 · ВКЛЮЧИТЕ И РАЗМЕСТИТЕ SHIELD (одна точка консолидации)

- ☐ **Включите shielding** (CloudFront Origin Shield / Cloudflare Tiered Cache / Akamai Tiered Distribution / Fastly Shielding). Все промахи к origin идут через одно место, которое тянет origin один раз на объект.
- ☐ **Ставьте shield ближе к ORIGIN, а не к зрителям.** Выберите регион/POP с наименьшей задержкой до origin; для облачных origin за anycast задайте cloud region hint, чтобы выбрался верный tier.
- ☐ **В multi-CDN сделайте один shielded CDN общей «парадной дверью»**, чтобы промахи всех CDN шли через него (и общий ключ кэша). Размещение shield: _____

3 · ЗАДАЙТЕ TTL, ЧТОБЫ SHIELD ДЕРЖАЛ КЭШ

- ☐ **Длинный TTL на неизменяемых сегментах, короткий на live-манифесте.** Shield консолидирует только то, что ему разрешено хранить; слишком короткий TTL сегмента зря промахивается в origin.
- ☐ **Изменения — через новый путь; purge — для удаления.** Новый путь чисто заполняет shield; surrogate keys / cache tags очищают весь тайтл одним вызовом.
- ☐ **Смотрите на byte-based origin offload, а не только на request hit ratio** — попадания в shield после промаха edge выглядят как промахи, хотя origin их не видел.

4 · FAILOVER И МУЛЬТИРЕГИОННЫЙ ORIGIN

- ☐ **Shield перед одним origin кэширует аварию.** Добавьте origin group: primary + secondary, failover по выбранным кодам (напр. 502 / 503 / 504), для GET / HEAD / OPTIONS.
- ☐ **Ужмите тайминг failover для live.** Веб-дефолт может дать ~30 с чёрного экрана; сократите таймауты origin и число попыток, чтобы переключение заняло секунды.
- ☐ **Держите secondary в отдельном домене отказа.** Минимум другой AZ, для важных событий другой регион; active-active за балансировщиком убирает разрыв. Коды failover: ___ таймаут: ___

ОДНА ПРОВЕРКА ПЕРЕД LIVE-СОБЫТИЕМ

CDN широк на edge и узок у origin: трафик расходится к зрителям и сходится в один сервер. Пример: live для 100 000 зрителей на 4-секундных сегментах, доставка через 8 региональных кэшей в 3 CDN. Новый сегмент выходит каждые 4 секунды, а HLS-манифест заставляет каждый плеер перезапрашивать его, поэтому все регионы промахиваются по новому сегменту разом — thundering herd. Без shield это ~8 регионов × 3 CDN = 24 дублирующих запроса к origin на сегмент, и для just-in-time origin каждый — задача packaging — ~6/сек, умноженные на все рендишны ladder. Поставьте один origin shield — он схлопнет все 24 в один запрос на объект: origin видит ~1 обращение на сегмент вместо 24, сокращение ~24× без потери ценности, ведь все эти запросы возвращают одинаковые байты. Затем убедитесь, что есть второй origin за ужатой origin group, чтобы мёртвый origin был failover, а не мёртвой трансляцией. Числа иллюстративны; дефолты и имена CDN сверяйте вживую, они меняются.