# Część 4: Dostępność — Heartbeat/Uptime (monitory, historia awarii)

Scenariusz prowadzącego (15–20 min) — stack: .NET Core + Angular + Node.js

## 0) Cel i efekty

• Zrozumieć, jak w Kibanie korzystać z Uptime/Heartbeat do kontroli dostępności usług i punktów końcowych (HTTP/TCP/ICMP).

• Umieć czytać historię awarii/flappingu, lokalizacje geograficzne i mierzyć dostępność (availability) oraz czas odpowiedzi.

• Skorelować incydenty dostępności z APM (.NET/Node) oraz z metrykami infrastruktury (Metricbeat) i logami (Discover).

## 1) Przygotowanie „dzień wcześniej” (poza sesją)

• Heartbeat: skonfigurowane monitory HTTP/TCP/ICMP do krytycznych endpointów (API .NET/Node, strony Angular, reverse proxy, DB porty — tylko jeśli ma sens). Każdy monitor ma stabilne `monitor.id`, opisową `monitor.name` i `tags` (np. env/team/system).

• Lokalizacje: co najmniej 2 lokalizacje (jeśli masz) — np. regiony chmurowe lub data center; pozwala wykrywać problemy regionalne.

• Dane i widoczność: upewnij się, że dane trafiają do Elasticsearch i widzisz je w Uptime. Jeśli używasz Elastic Synthetics (journeys), sprawdź, że są widoczne w tej samej przestrzeni.

• Alerting: skonfigurowane kanały powiadomień (Slack/Email/Teams/Webhook) i reguły: Monitor status down, TLS certificate expiration, Availability threshold.

• Maintenance windows: zdefiniowane okna utrzymaniowe (jeśli planujesz prace), aby unikać fałszywych alarmów.

• Saved Objects/Queries: zaimportowany NDJSON (dashboard startowy) — będziesz mieć pod ręką gotowe filtry.

## 2) Start sesji – ustawienia w Kibanie

• Zakres czasu: Last 24 hours (na potrzeby pokazu). Dla analizy stabilności rozważ 7 days.

• Filtry: `tags : "prod"` lub `monitor.environment : "prod"` (jeśli używasz), ewentualnie filtr na `monitor.type` (http/tcp/icmp).

• Auto-refresh: 1–5 min na czas pokazu.

## 3) Widok Uptime — orientacja w UI

• Overview: lista monitorów z kolumnami status/availability/avg duration; szybka identyfikacja DOWN/FLAPPING.

• Monitor details: wykres statusu w czasie (historia), czasy odpowiedzi (duration), błędy (error.message), lokalizacje (observer.geo.name).

• TLS/Certyfikaty (dla HTTPS): dni do wygaśnięcia i szczegóły certyfikatu (jeśli monitor http/https).

• Synthetics (opcjonalnie): dla journey’ów przeglądarkowych — waterfall/steps/screenshoty; pozwalają diagnozować z perspektywy użytkownika.

• Nawigacja: linki do APM (jeśli endpoint powiązany z usługą), do Logs/Discover (po host/URL) i do Infrastructure (host/pod).

## 4) Scenariusz pokazu — krok po kroku

• Krok A — Overview: posortuj monitory po statusie; na wierzch trafią DOWN/FLAPPING. Zanotuj, które dotyczą krytycznych API.

• Krok B — Monitor detail: otwórz szczegóły; sprawdź wykres statusu i duration. Odczytaj `error.message` dla najnowszych prób.

• Krok C — Lokalizacje: porównaj status per region (`observer.geo.name`). Jeśli tylko jeden region DOWN, to problem sieciowy/edge.

• Krok D — Historia awarii: zmień zakres na 7 days i oceń powtarzalność. Flapping = częste przejścia UP/DOWN w krótkim czasie.

• Krok E — TLS: jeżeli monitor to HTTPS, otwórz sekcję certyfikatów i sprawdź dni do wygaśnięcia.

• Krok F — Korelacja: kliknij do APM usługi backendowej; sprawdź czy w tych samych godzinach rosną error rate/latency. Następnie sprawdź Infra (CPU/Memory/5xx na proxy).

• Krok G — Alerts: pokaż regułę „Monitor status down” i gdzie sprawdzić historię alertów oraz powiązane Cases.

## 5) Kluczowe wskaźniki i progi (do uzgodnienia w zespole)

• Availability: miesięcznie np. 99.9% (SLO).

• Response time: p95 HTTP duration < 500 ms dla usług API; dla stron publicznych zależnie od SLA.

• Flapping: >3 przejścia UP/DOWN w 15 min — wymagane zwiększenie timeoutu/better retry albo stabilizacja endpointu.

• TLS expiry: ostrzegaj poniżej 30 dni do wygaśnięcia (lub 14 dni dla mniej krytycznych).

## 6) Saved Queries — KQL dla Uptime/Heartbeat

• Monitory DOWN (24h)

monitor.status : "down" and @timestamp >= now-24h

• HTTP monitory DOWN (24h)

monitor.type : "http" and monitor.status : "down" and @timestamp >= now-24h

• Region — EU tylko (24h)

observer.geo.name : "eu" and @timestamp >= now-24h

• Czasy odpowiedzi > 500 ms (24h)

monitor.duration.us >= 500000 and @timestamp >= now-24h

• TLS — wygasa w 14 dni

tls.server.x509.not\_after < now+14d or tls.certificate\_not\_valid\_after < now+14d

• Błędy HTTP 5xx

http.response.status\_code >= 500 and @timestamp >= now-24h

## 7) Alerty — zestaw startowy

• Monitor status: DOWN przez ≥ 2 min (z histerezą na flapping).

• Availability: średnia dostępność < 99.9% w oknie 30 dni (SLO).

• Performance: p95 duration > 1 s przez 5 min (dla krytycznych API).

• TLS: certyfikat wygaśnie w ≤ 14/30 dni.

• Lokalizacja: DOWN w ≥1 regionie przy UP w innych (degradacja regionalna).

## 8) Lab — ćwiczenia praktyczne

• Ćw. 1: Utwórz nowy HTTP monitor do endpointu `/health` usługi .NET i nadaj `tags: ["prod", "api", "team-x"]`.

• Ćw. 2: Zasymuluj krótkie `DOWN` (np. tymczasowa reguła firewall) i sprawdź, jak wygląda historia oraz alert.

• Ćw. 3: Dodaj alert „TLS expiry < 30 dni” dla publicznego hosta i zweryfikuj działanie.

• Ćw. 4: Skoreluj `DOWN` z APM i Metricbeat — czy w tym czasie p95/error rate wzrosły?

• Ćw. 5: Utwórz Case z notatką, właścicielem i linkami do monitoru/alertu.

## 9) Pułapki i dobre praktyki

• Flapping: dostosuj `schedule`, `timeout` i retry; rozważ maintenance window przy planowanych deployach.

• Stabilne identyfikatory: `monitor.id` i `monitor.name` niech będą stałe; zmiany generują chaos w historii.

• Tagowanie: `tags` (env/team/system) ułatwiają filtrowanie i routing alertów.

• Geolokalizacje: problem tylko w jednej lokalizacji często oznacza kłopoty DNS/edge/CDN.

• Zakresy czasu: dla stabilności patrz 7–30 dni; 24h służy szybkiemu przeglądowi.

## 10) Mini-ściąga dla prowadzącego

• Overview → sortuj po statusie → wybierz DOWN/FLAPPING.

• Monitor detail → status w czasie, duration, error.message, lokalizacje.

• TLS → dni do wygaśnięcia (dla HTTPS).

• Korelacja → APM i Infrastructure; w razie potrzeby Logs.

• Alerty → historia i Cases.