# Część 3: Infrastruktura — Metricbeat (hosty, kontenery, reverse proxy, DB)

Scenariusz prowadzącego (20–30 min) — stack: .NET Core + Angular + Node.js

## 0) Cel i efekty

• Zrozumieć, jak w Kibanie korzystać z widoków Infrastructure/Metrics do oceny zdrowia hostów, kontenerów, reverse proxy oraz baz danych.

• Umieć szybko zidentyfikować wąskie gardła zasobowe (CPU, RAM, dyski, sieć) oraz problemy na poziomie usług (IIS/Nginx/Traefik, MSSQL/Postgres/Mongo/Redis).

• Nauczyć się wykonywać „drill-down”: Infra → Logs/Discover oraz Infra → APM i Uptime, aby potwierdzić hipotezy.

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

• Metricbeat: włączone moduły `system`, `windows`/`linux`, `docker`/`kubernetes`, reverse proxy (`iis`, `nginx`, `traefik`), bazy (`mssql`, `postgresql`, `mongodb`, `redis`).

• Uzgodnione etykiety: `labels.environment`, `labels.team` na hostach/kontenerach; spójne `service.name`/`service.environment` z APM.

• Data Views: `metrics-\*` (Metricbeat), `logs-\*` (logi), `traces-\*` (APM), `synthetics-\*` (Uptime) — sprawdź dostęp.

• Saved Objects: import NDJSON (Starter Dashboard + panele; Saved Queries z poprzednich części).

• Test: wygeneruj niewielkie obciążenie i 1–2 restarty kontenera, aby mieć materiał do pokazu.

## 2) Start sesji – ustawienia w Kibanie

• Zakres czasu: Last 24 hours (lub 7 days dla trendu).

• Filtry globalne: `labels.environment : "prod"` (lub `staging`), opcjonalnie `labels.team`.

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

## 3) Widok Infrastructure/Metrics — orientacja w UI

• Inventory: kafelki hostów/kontenerów z kolorowaniem wg CPU/Memory/Load — szybka mapa hot-spotów.

• Metrics Explorer: szybkie wykresy dla dowolnych metryk (agregacje/filtry) z możliwością zapisania paneli do dashboardu.

• Hosts: szczegół karty hosta (CPU, pamięć, dyski, sieć, procesy).

• Kubernetes/Containers: pody, nody, restarty, limity i throttling CPU/memory (jeśli masz kube-state-metrics/metricsety).

• Integracje usług: IIS, Nginx, Traefik — żądania/saturacja/błędy; bazy: MSSQL, Postgres, Mongo, Redis — metryki wydajności.

• Przejścia: z hosta/poda klikniesz do „View logs” (Discover), „APM traces” (jeśli powiązane) oraz do Uptime (monitor zasobu).

## 4) Scenariusz pokazu — krok po kroku

• Krok A — Inventory: sortuj/koloruj po CPU, potem Memory. Zidentyfikuj 1 host i 1 kontener z najwyższym zużyciem.

• Krok B — Host detail: otwórz kartę hosta → sprawdź CPU (total.pct), Memory (actual.used.pct), Disk IO (i/o time), Network (errors/drops).

• Krok C — Containers/K8s: pokaż kontenery/pody z restartami; przejdź do szczegółów (limity, throttling, restarts).

• Krok D — Reverse proxy: otwórz panel dla IIS/Nginx/Traefik i pokaż requests/min, 5xx/min, ewentualnie latency/active connections.

• Krok E — Bazy danych: MSSQL/PostgreSQL/MongoDB/Redis — wskaż kluczowe symptomy (waits/locks, evictions, replication lag).

• Krok F — Drill-down do APM: jeśli 5xx lub wysokie latency na proxy, otwórz APM usługę backendową i porównaj p95/p99 i error rate.

• Krok G — Logs/Discover: z hosta lub usługi kliknij „View logs”, użyj zapisanych KQL (np. „DB – MSSQL (24h)”, „NGINX – Metryki (24h)”).

• Krok H — Uptime: sprawdź, czy monitor zasobu (host/API) wskazywał niedostępność w tym samym oknie czasu.

## 5) Kluczowe wskaźniki i progi referencyjne

• Host: CPU `system.cpu.total.pct` (⚠ >85%), Memory `system.memory.actual.used.pct` (⚠ >90%), Disk `system.filesystem.used.pct` (⚠ >85%), I/O wait, Network errors/drops.

• K8s/Docker: `kubernetes.container.restart\_count` (⚠ >0 w 24h), throttling CPU, `container.cpu.usage` i `container.memory.usage`.

• IIS/Nginx/Traefik: 5xx/min (nagły wzrost), requests/min (nagły spadek), active connections/queue length.

• MSSQL: log flush waits, Page life expectancy (spadki), blocking sessions; Postgres: locks/errors; Mongo: replication lag; Redis: `evictions` i latency.

## 6) Saved Queries — KQL dla Infra

• Hosty — CPU > 85% (24h)

event.dataset : "system.cpu" and system.cpu.total.pct >= 0.85 and @timestamp >= now-24h

• Hosty — Pamięć > 90% (24h)

event.dataset : "system.memory" and system.memory.actual.used.pct >= 0.90 and @timestamp >= now-24h

• K8s — Restarty kontenerów (24h)

event.dataset : "kubernetes.\*" and kubernetes.container.restart\_count > 0 and @timestamp >= now-24h

• Docker — Restarty (24h)

event.dataset : "docker.\*" and container.status : "restarted" and @timestamp >= now-24h

• IIS — metryki/5xx (24h)

event.dataset : "iis.\*" and @timestamp >= now-24h

• NGINX — metryki (24h)

event.dataset : "nginx.\*" and @timestamp >= now-24h

• Traefik — HTTP 5xx (24h)

event.dataset : "traefik" and http.response.status\_code >= 500 and @timestamp >= now-24h

• DB — MSSQL (24h)

event.dataset : "mssql.\*" and @timestamp >= now-24h

• DB — PostgreSQL (24h)

event.dataset : "postgresql.\*" and @timestamp >= now-24h

• DB — MongoDB (24h)

event.dataset : "mongodb.\*" and @timestamp >= now-24h

• DB — Redis (24h)

event.dataset : "redis.\*" and @timestamp >= now-24h

## 7) Alerty — zestaw startowy

• Host: CPU > 90% przez 10 min; Pamięć > 90% przez 10 min; Dysk wolny < 15%.

• Kontenery: restarty > 3 w 30 min; throttling CPU > 10% przez 10 min.

• Reverse proxy: 5xx/min powyżej progu bazowego przez 5 min; wysoki backlog/kolejka.

• Bazy: Redis evictions > 0 przez 5 min; MSSQL log flush waits skok > zdefiniowany baseline; Postgres locks rosnące w czasie.

• Korelacyjne: jeśli Infra alert aktywny i jednocześnie APM error rate ↑ — eskaluj do P1.

## 8) Lab — ćwiczenia praktyczne

• Ćw. 1: Znajdź host z najwyższym `system.cpu.total.pct` i sprawdź procesy obciążające CPU.

• Ćw. 2: W K8s/Docker wyszukaj pody/kontenery z restartami; sprawdź logi i przyczynę.

• Ćw. 3: Na panelu Nginx/IIS znajdź skok 5xx i skoreluj z APM (p95/error rate) w tym samym oknie czasu.

• Ćw. 4: Dla Redis wykryj `evictions` i sprawdź zużycie pamięci oraz politykę; dla MSSQL sprawdź waits/flush.

• Ćw. 5: Utwórz alert CPU>90%/10 min i zasymuluj obciążenie; zweryfikuj, że alert zadziałał.

## 9) Pułapki i dobre praktyki

• Kolorowanie Inventory zależy od wybranej metryki — zawsze powiedz, po czym kolorujesz (CPU/Memory/Load).

• Pamiętaj o różnicach w `system.memory.actual.used.pct` vs `system.memory.used.pct` (cache/buffers).

• W K8s sprawdzaj throttling CPU i limity — częsta przyczyna „niewyjaśnionej” latencji.

• Reverse proxy: 5xx mogą pochodzić z upstreamu — zawsze koreluj z APM usługi backend.

• Bazy: metryki są symptomem — do przyczyny potrzebne są zarówno metryki, jak i logi oraz plany zapytań.

## 10) Mini-ściąga dla prowadzącego

• Inventory → wybierz hot-spot host/kontener.

• Host/K8s detail → CPU/Memory/Restarts/Throttling.

• Reverse proxy → 5xx/min i requests/min.

• DB → symptomy (waits/locks/evictions/lag).

• Drill-down: Logs (Discover) i APM; potwierdź hipotezę; w razie potrzeby Uptime.