# Część 5: Logi i zdarzenia — Discover/Logs (ostatnie 24 h, outliers, winlogi)

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

## 0) Cel i efekty

• Nauczyć zespołowo robić przegląd logów z ostatnich 24 h, wykrywać anomalie/outliers i błyskawicznie znajdować źródła problemów.

• Umieć korzystać z widoków Logs (Stream) i Discover, a także z analizy log rate (outliers/spikes).

• Znać podstawowe kody i wzorce dla winlogów (Windows) oraz szybkie filtry KQL dla Nginx/IIS/aplikacji/DB.

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

• Źródła logów: Filebeat/Winlogbeat/Elastic Agent wysyłają do Elasticsearch; Data Views: `logs-\*` (ogólne), `winlogbeat-\*` lub `windows-\*` (winlogi), ewentualnie dedykowane np. `filebeat-\*`.

• Mapowanie ECS: upewnij się, że kluczowe pola są w ECS (service.name, service.environment, event.dataset, log.level, event.code, event.provider, trace.id).

• Saved Objects: zaimportowany NDJSON (dashboard i zapytania), by mieć pod ręką gotowe filtry.

• Uprawnienia: dostęp do Logs (Stream), Discover, Alerts, Cases.

• Dane testowe: wygeneruj 1–2 kontrolowane błędy (np. Nginx 502, wyjątek w API, Winlog event 1000), by było co analizować.

## 2) Start sesji – ustawienia w Kibanie

• Zakres czasu: Last 24 hours (na początek). Dla trendów — 7 days.

• Filtry globalne: `service.environment : "prod"` (lub `staging`), opcj. `labels.team` albo `service.name`.

• Auto-refresh: 1–5 min na czas pokazu (potem możesz wyłączyć).

## 3) Widok Logs/Discover — orientacja w UI

• Logs → Stream: live’owy strumień logów z filtrowaniem po `log.level`, `service.name`, `event.dataset`, host/kubernetes fields. Szybkie podglądy rekordów.

• Discover: pełnotekstowe zapytania + KQL, tabele kolumn, histogram timeline, zapisywanie widoków i „Saved queries”.

• Highlights i kolumny: dodaj kolumny `service.name`, `event.dataset`, `log.level`, `host.name`, `kubernetes.pod.name`, `trace.id`.

• Linki i korelacje: z logów z polem `trace.id` przejdziesz do APM; z `host.\*` / `kubernetes.\*` do Infrastructure.

• Analyze (Log rate): wykrywanie anomalii w intensywności logów (spikes/dips), change points i kategorie komunikatów.

## 4) Scenariusz pokazu — krok po kroku

• Krok A — Logs → Stream: ustaw filtry `service.environment:"prod"` i `log.level:(error OR fatal)`. Posortuj malejąco po czasie.

• Krok B — Discover: wybierz data view `logs-\*`. Dodaj kolumny: service.name, event.dataset, log.level, host.name, kubernetes.pod.name.

• Krok C — Aplikacyjne błędy: odfiltruj `event.dataset : "apm\*" OR message:"Exception"` i zobacz top źródła/serwisy.

• Krok D — Reverse proxy: `event.dataset : ("nginx.\*" OR "iis.\*") AND http.response.status\_code >= 500` — sprawdź piki 5xx i okno czasowe.

• Krok E — Bazy danych: `event.dataset : ("mssql.\*" OR "postgresql.\*" OR "mongodb.\*" OR "redis.\*")` — poszukaj „ERROR/FATAL”, evictions, waits, locks.

• Krok F — Outliers: kliknij Analyze/Log rate analysis → wskaż zakres piku i sprawdź, które pola zmieniły rozkład (change points) i które komunikaty wzrosły.

• Krok G — Korelacja: jeśli rekord ma `trace.id`, otwórz APM trace; jeśli to dotyczy hosta/poda — przejdź do Infrastructure.

• Krok H — Winlogi: przełącz data view na `winlogbeat-\*`/`windows-\*` i odfiltruj istotne kody (patrz sekcja 7).

• Krok I — Notatka i Case: skopiuj link do wyszukiwania, dodaj do Case i przypisz właściciela.

## 5) Saved Queries — KQL dla logów (24h)

• Aplikacyjne błędy (ogólnie)

log.level : ("error","fatal") and @timestamp >= now-24h

• Nginx — HTTP 5xx

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

• IIS — HTTP 5xx

event.dataset : "iis.access" and http.response.status : >= 500 and @timestamp >= now-24h

• API — wyjątki .NET/Node

message : ("Exception" OR "Unhandled" OR "Traceback") and @timestamp >= now-24h

• DB — PostgreSQL błędy

event.dataset : "postgresql.\*" and (postgresql.log.level : ("ERROR","FATAL") or message : ("ERROR" OR "FATAL")) and @timestamp >= now-24h

• DB — MSSQL symptomy

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

• DB — Redis evictions

event.dataset : "redis.\*" and redis.stats.evicted\_keys : >= 1 and @timestamp >= now-24h

• K8s/Docker — restarty

(event.dataset : "kubernetes.\*" and kubernetes.container.restart\_count > 0) or (event.dataset : "docker.\*" and container.status : "restarted") and @timestamp >= now-24h

## 6) Outliers/Anomalie — jak w praktyce

• W Logs/Discover wybierz okno z pikami na histogramie i uruchom „Analyze data / Log rate analysis”.

• Sprawdź „Spikes and dips” oraz „Change points” — które pola (np. service.name, event.dataset, http.response.status\_code) uległy zmianie rozkładu.

• Zerknij na „Frequent message patterns”/kategorie logów: jakie komunikaty dominują w piku?

• Porównaj z poprzednim okresem (Compare to previous) — czy to regresja, czy powrót do normy.

• Klikaj w sugestie filtrów, aby zawęzić do źródła anomalii (konkretny host/pod/endpoint).

## 7) Winlogi — szybki przewodnik (kody i providerzy)

• Application Error: `event.provider : "Application Error" and event.code : 1000` — błąd aplikacji (crash).

• Service Control Manager: `event.provider : "Service Control Manager" and event.code : (7031 or 7034)` — usługa zakończona nieoczekiwanie.

• Schannel/TLS: `event.provider : "Schannel" and event.level : ("Error","Warning")` — problemy z TLS/SSL.

• Security — nieudane logowanie: `event.code : 4625` — koreluj z IP/hostami.

• Windows Defender: `event.provider : "Microsoft-Windows-Windows Defender" and event.level : ("Warning","Error")`.

## 8) Alerty — zestaw startowy

• Log threshold: `log.level:error` liczba > X w 5 min (per service.name).

• Nginx/IIS 5xx spike: liczba zdarzeń ze statusem >= 500 > baseline o 50% w 5 min.

• Winlog — Application Error (1000) lub Service Control Manager (7031/7034) — natychmiastowy alert.

• DB — Redis evictions > 0 w 5 min; Postgres FATAL; MSSQL krytyczne eventy.

## 9) Lab — ćwiczenia praktyczne

• Ćw. 1: W Discover znajdź piki 5xx w Nginx z ostatnich 24 h; zidentyfikuj endpointy i koreluj z APM.

• Ćw. 2: W Logs/Analyze uruchom Log rate analysis dla wybranego piku; wskaż pole powodujące zmianę (np. service.version po deployu).

• Ćw. 3: Odszukaj w winlogach błędy 1000 i 7031/7034; sprawdź, na których hostach wystąpiły i kiedy.

• Ćw. 4: Zbuduj alert „Log threshold” dla `log.level:error` per `service.name` i przetestuj go.

• Ćw. 5: Utwórz Case z wnioskami i przypisz właściciela.

## 10) Pułapki i dobre praktyki

• Data View: upewnij się, że używasz właściwego (np. `logs-\*` vs `winlogbeat-\*`).

• Strefy czasu: Logi z wielu regionów — porównuj w tym samym timezone (Kibana global time).

• Wydajność Discover: ogranicz kolumny i zakres czasu; zapisuj Saved queries i widoki.

• ECS: trzymaj spójne `service.name`/`service.environment`; dzięki temu pivot do APM/Infra działa bezboleśnie.

• Trace correlation: logi z `trace.id` są bezcenne — rozważ wzbogacenie logów po stronie aplikacji.

## 11) Mini-ściąga dla prowadzącego

• Logs → Stream (error/fatal, prod) → szybki rzut oka.

• Discover → filtry i kolumny → aplikacja/proxy/DB w oddzielnych zakładkach.

• Analyze → Log rate analysis → wskaż źródło piku.

• Winlogi → Application Error 1000 / SCM 7031/7034.

• Case → notatka i właściciel.