Elastic — Dokumentacja operacyjna (PL)

Zestawienie: Heartbeat, Metricbeat, Filebeat, Winlogbeat, Packetbeat, APM (.NET & Angular)  
*Data: 2025-08-20 11:42*

# HEARTBEAT — dokumentacja (Windows, offline ZIP)

## Cel komponentu

Monitorowanie dostępności i czasu odpowiedzi usług (HTTP/HTTPS, ICMP, TCP). Szybka identyfikacja awarii i opóźnień.

## Co potrafi zrobić

* Ping HTTP/HTTPS (status, latency, TLS cert info).
* ICMP/TCP ping hostów i portów.
* Etykietowanie monitorów (`tags`, `fields`) i korelacja z hostami.
* Eksport do Kibany: Uptime/Discover/Dashboards; alerty na niedostępność.

## Instalacja na Windows (offline, ZIP)

1) Skopiuj rozpakowany katalog `heartbeat-{ver}-windows-x86\_64` na serwer bez Internetu.

2) Utwórz użytkownika serwisowego (opcjonalnie) i folder logów `C:\Elastic\logs`.

3) Skonfiguruj `heartbeat.yml` oraz pliki w `monitors.d`.

4) Rejestracja jako usługa:

powershell  
 .\install-service-heartbeat.ps1  
 Start-Service heartbeat

## Minimalna konfiguracja (HTTP monitor) — przykład

yaml  
heartbeat.monitors:  
 - type: http  
 id: onet-http  
 name: "Onet.pl"  
 schedule: "@every 30s"  
 urls: ["http://onet.pl"]  
 check.request:  
 method: "GET"  
 check.response:  
 status: 200  
 tags: ["public", "web"]  
 fields:  
 env: "prod"  
  
output.elasticsearch:  
 hosts: ["https://<ES\_HOST>:9200"]  
 username: "<user>"  
 password: "<pass>"  
  
setup.kibana.host: "https://<KIBANA\_HOST>:5601"

## Gdzie to zobaczysz w Kibanie

* \*\*Discover\*\*: indeks `heartbeat-\*`, filtry po `monitor.status`, `monitor.name`, `summary.up`.
* \*\*Uptime\*\*: przegląd monitorów, timeline „up/down”, SLI/SLO dla dostępności.
* \*\*Dashboardy Beats\*\* (po `setup --dashboards`).

## Alerty — szybkostart

* Rule (KQL): `monitor.status : "down"`; warunek „> 0 w 1 min”, Group by `monitor.name`.
* Akcje: e-mail/Slack; w treści uwzględnij `{{context.matchCount}}`, `{{context.tookMs}}`.

## Troubleshooting

* Brak danych: sprawdź Timepicker (Last 15m) i `output.elasticsearch`.
* 4xx/5xx: włącz `ssl.verification\_mode: none` tylko testowo; sprawdź proxy/allowlists.

# METRICBEAT — dokumentacja (Windows, offline ZIP)

## Cel komponentu

Zbieranie metryk systemowych i usług (CPU, RAM, dyski, procesy, sieć, Windows perf counters).

## Co potrafi zrobić

* Metryki hosta (`system`), Windows (`windows`), IIS/SQL Server (moduły).
* Agregacje w Kibanie: Hosts/Infrastructure, Lens/Visualize.
* Alerty: progi CPU, RAM, dysk, wysoka liczba handle/threads, itp.

## Instalacja (offline)

1) Skopiuj ZIP i rozpakuj.

2) W `metricbeat.yml` ustaw wyjście do Elasticsearch i Kibany.

3) Włącz moduły:

powershell  
 .\metricbeat.exe modules enable system windows

4) Rejestracja jako usługa:

powershell  
 .\install-service-metricbeat.ps1  
 Start-Service metricbeat

## Minimalna konfiguracja

yaml  
metricbeat.modules:  
 - module: system  
 metricsets: [cpu, memory, network, diskio, filesystem, process]  
 period: 10s  
 processes: ['.\*']  
  
 - module: windows  
 metricsets: [service]  
 period: 30s  
  
output.elasticsearch:  
 hosts: ["https://<ES\_HOST>:9200"]  
 username: "<user>"  
 password: "<pass>"  
  
setup.kibana.host: "https://<KIBANA\_HOST>:5601"

## Kibana — gdzie patrzeć

* \*\*Analytics → Dashboard\*\*: „Metricbeat System Overview”.
* \*\*Discover\*\*: `metricset.name`, `host.name`, `system.process.\*`, `windows.service.\*`.

## Przykładowe alerty

* CPU > 90% przez 5 min (Group by `host.name`).
* Dysk < 10% wolnego (Fields: `system.filesystem.used.pct`).

## Troubleshooting

* Brak metryk: uprawnienia usługi (WMI/PerfMon), antywirus/EDR.

# FILEBEAT — dokumentacja (Windows, offline ZIP)

## Cel komponentu

Wysyłanie logów aplikacji/systemowych do Elasticsearch. Parsowanie JSON, rozpoznawanie pól, pipeline'y.

## Co potrafi zrobić

* Czytanie plików (rotacje, wzorce, glob).
* Parsowanie JSON i multiline (stacktrace).
* Moduły gotowe (IIS, SQL, system); własne prosessory.

## Instalacja (offline)

1) Rozpakuj ZIP na serwer.

2) Skonfiguruj `filebeat.yml` (paths + output).

3) Rejestracja:

powershell  
 .\install-service-filebeat.ps1  
 Start-Service filebeat

## Minimalna konfiguracja (logi aplikacji z JSON)

yaml  
filebeat.inputs:  
 - type: filestream  
 id: app-json  
 enabled: true  
 paths: ["C:/Logs/MyApp/\*.log"]  
 parsers:  
 - ndjson:  
 target: ""  
 add\_error\_key: true  
  
processors:  
 - drop\_fields:  
 fields: ["log.offset"]  
 - add\_fields:  
 target: "app"  
 fields:  
 name: "myapp"  
 env: "prod"  
  
output.elasticsearch:  
 hosts: ["https://<ES\_HOST>:9200"]  
 username: "<user>"  
 password: "<pass>"  
  
setup.kibana.host: "https://<KIBANA\_HOST>:5601"

## Kibana — szybkie KQL

* Błędy 24h: `log.level : ("error" or "critical") and @timestamp >= now-24h`
* Z jednego hosta: `host.name : "api-01" and log.level : "error"`
* Wyklucz Filebeat: `not agent.type : "filebeat"`

## Troubleshooting

* Multiline (stacktrace): użyj `parsers: - multiline` (jeśli nie NDJSON).
* Zgubione wpisy: sprawdź uprawnienia i `ignore\_older`.

# WINLOGBEAT — dokumentacja (Windows, offline ZIP)

## Cel komponentu

Zbieranie zdarzeń Windows (Security, System, Application, opcjonalnie Sysmon).

## Co potrafi zrobić

* Monitoruje kanały EventLog (`winlog`).
* Wysyła pola z `winlog.event\_id`, `event\_data.\*`.
* Dashboardy „Windows Security” po imporcie.

## Instalacja (offline)

1) Rozpakuj ZIP.

2) W `winlogbeat.yml` określ kanały.

3) Rejestracja:

powershell  
 .\install-service-winlogbeat.ps1  
 Start-Service winlogbeat

## Przykładowa konfiguracja

yaml  
winlogbeat.event\_logs:  
 - name: Security  
 level: Information  
 - name: System  
 - name: Application  
  
output.elasticsearch:  
 hosts: ["https://<ES\_HOST>:9200"]  
 username: "<user>"  
 password: "<pass>"  
  
setup.kibana.host: "https://<KIBANA\_HOST>:5601"

## Szybkie KQL (Security)

* Nieudane logowania (4625): `winlog.event\_id : 4625`
* Udane RDP/interaktywne (4624): `winlog.event\_id : 4624 and winlog.event\_data.LogonType : (2 or 10)`
* Konto/grupy: `winlog.event\_id : (4720 or 4728 or 4732 or 4738)`
* Instalacja usługi: `winlog.event\_id : (4697 or 7045)`

## Alerty (przykłady)

* Brute-force: `4625` > 20/5 min (Group by `host.name`, `TargetUserName`).
* Nowe konto/zmiana grupy: `4720/4728/4732` > 0/1 min.

## Troubleshooting

* Brak Security: sprawdź Auditing Policy (Local/Domain) i rozmiar logu.

# PACKETBEAT — dokumentacja (Windows, offline ZIP)

## Cel komponentu

Analiza ruchu sieciowego na poziomie protokołów (HTTP, DNS, MySQL, MSSQL, TLS), metryki transakcji i opóźnień.

## Co potrafi zrobić

* Przechwytuje metadane transakcji: latency, status, bytes in/out.
* Wgląd w top usługi, hosty, endpointy HTTP.
* Korelacja z problemami aplikacji (opóźnienia, błędy sieci).

## Instalacja (offline)

1) Rozpakuj ZIP i uruchom z uprawnieniami administratora (wymagane do sniffingu).

2) W `packetbeat.yml` ustaw interfejs i protokoły.

3) Rejestracja:

powershell  
 .\install-service-packetbeat.ps1  
 Start-Service packetbeat

## Minimalna konfiguracja

yaml  
packetbeat.interfaces.device: 0 # lub nazwa interfejsu  
  
packetbeat.protocols:  
 - type: http  
 ports: [80, 8080, 8000, 5000, 8002]  
 - type: tls  
 ports: [443, 8443]  
 - type: dns  
 ports: [53]  
  
output.elasticsearch:  
 hosts: ["https://<ES\_HOST>:9200"]  
 username: "<user>"  
 password: "<pass>"  
  
setup.kibana.host: "https://<KIBANA\_HOST>:5601"

## Kibana — na co patrzeć

* „Top HTTP endpoints by latency/status”.
* DNS errors/timeouts; TLS versions/cipher suites (jeśli włączone).

## Uwagi

* W środowiskach o wysokim bezpieczeństwie rozważ filtry BPF (ograniczenie ruchu).
* Pamiętaj o zgodności z politykami prywatności (metadane vs. payload).

# APM .NET — integracja (ASP.NET Core)

## Cel

Traces + metryki dla .NET (API/worker). Widoczność: latency, błędy, dependency map, transakcje.

## Co potrafi zrobić

* Automatyczny tracing ASP.NET Core, HTTP client, SQL client.
* Metryki runtime (GC, threads) i błędy.
* Korelacja z logami (jeśli logujesz `trace.id`/`transaction.id`).

## Instalacja (agent + konfiguracja)

1) Dodaj pakiet Elastic APM .NET lub agent auto-instrumentation.

2) Konfiguracja przez `appsettings.json` lub zmienne środowiskowe.

### appsettings.json (przykład)

json  
{  
 "ElasticApm": {  
 "ServerUrls": "https://<APM\_SERVER>/",  
 "SecretToken": "<token|apikey>",  
 "ServiceName": "MyApi",  
 "Environment": "prod",  
 "CaptureBody": "all",  
 "TransactionSampleRate": 0.3  
 }  
}

### Zmienne środowiskowe (alternatywa)

ELASTIC\_APM\_SERVER\_URLS=https://<APM\_SERVER>/  
ELASTIC\_APM\_SECRET\_TOKEN=<token>  
ELASTIC\_APM\_SERVICE\_NAME=MyApi  
ELASTIC\_APM\_ENVIRONMENT=prod

## Kibana — gdzie patrzeć

* \*\*APM → Services\*\*: Latency, Throughput, Failed transaction rate, Errors, Dependencies, Instances latency distribution.
* \*\*Dashboards\*\*: Service inventory, Top errors.

## Częste pułapki

* 401/403 do APM Servera: sprawdź token/API key i CORS (dla RUM).
* Brak SQL spans: upewnij się, że używasz wspieranych providerów (System.Data.SqlClient/Microsoft.Data.SqlClient).

# APM dla Angular 15 — integracja (RUM)

## Cel

Front-end tracing (RUM): czasy ładowania, routing, błędy JS, requesty XHR/fetch, korelacja z backendem.

## Instalacja (Angular 15)

1) Zainstaluj paczkę RUM (np. `@elastic/apm-rum`).

2) Zainicjalizuj wcześnie (np. w `main.ts` lub w module Core).

### Przykład inicjalizacji

typescript  
import { init as initApm } from '@elastic/apm-rum'  
  
export const apm = initApm({  
 serviceName: 'frontend-angular',  
 serverUrl: 'https://<APM\_SERVER>/',  
 environment: 'prod',  
 pageLoadTraceId: true,  
 distributedTracingOrigins: ['https://api.example.com'],  
 captureHeaders: true  
})

## Wymagane ustawienia APM Server (CORS/Allowed Origins)

* Dodaj `allowed\_origins` lub skonfiguruj „Allowed origins” w Kibanie (APM Integration).
* Jeśli widzisz `401` lub „blocked by CORS”:

- dopisz `distributedTracingOrigins` w RUM,

- ustaw „Allowed origins” na domenę frontendu.

## Kibana — gdzie patrzeć

* \*\*APM → Services → frontend-angular\*\*: Page load, Route change, Errors.
* Korelacja z backendem (trace id w nagłówkach).

## Troubleshooting

* 401 — błędny token/anon ingest wyłączony.
* CORS — brak domeny w allowlist.

# ANEKS — KQL i gotowe dashboardy

### KQL — Filebeat

* Błędy (24h): `log.level : ("error" or "critical") and @timestamp >= now-24h`
* Wykluczenie Filebeat: `not agent.type : "filebeat"`

### KQL — Winlogbeat (Security)

* `winlog.event\_id : 4625` (nieudane logowania)
* `winlog.event\_id : 4624 and winlog.event\_data.LogonType : (2 or 10)` (udane interaktywne/RDP)
* `winlog.event\_id : (4720 or 4728 or 4732 or 4738)` (konta/grupy)
* `winlog.event\_id : (4697 or 7045)` (usługi)

### Dashboardy Beats (po imporcie)

* Filebeat: „[Filebeat] Overview”
* Winlogbeat: „[Winlogbeat] Security Overview”
* Metricbeat: „[Metricbeat] System Overview”