**0) Założenia i decyzje wstępne (60–90 min)**

* **Standard danych:** ECS (Elastic Common Schema) dla wszystkich usług.
* **Nazewnictwo data streams:** logs-<app>-<env> (np. logs-api-prod).
* **Wariant shipowania (wybierz jeden na całą organizację):**  
  **A. Direct sink → Elasticsearch (data streams)** – prościej, minimalny narzut w runtime.  
  **B. Plik NDJSON (ECS) → Elastic Agent/Filebeat** – większa niezawodność (bufor na dysku).
* **Korelacja z APM:** wymagane; service.name w APM = service.name w logach.
* **Retencja/koszty:** ILM hot→warm→cold→delete (np. 7/30/90/365 dni – dostosuj do potrzeb).

**Artefakt:** krótka notka architektoniczna (1 strona) z wyborem A lub B + naming + ILM.

**1) Inwentaryzacja bieżącej konfiguracji SEQ (~2 h)**

1. Zrób listę usług i ich **Serilog enrichers**, **minimum level**, **tematy alertów** w SEQ.
2. Wyszczególnij własne pola i zaplanuj mapowanie do **ECS**:
   * UserId → user.id
   * OrderId → labels.order\_id (albo transaction.id jeżeli to faktycznie transakcja)
   * ElapsedMs → event.duration (w ns) lub labels.elapsed\_ms
   * HttpMethod → http.request.method
   * StatusCode → http.response.status\_code
   * Path → url.path
   * ClientIp → client.ip
   * Wyjątki → error.type, error.message, error.stack\_trace
3. Zanotuj **kwerendy/filtry SEQ**, które zespół używa najczęściej (będą przepisane do KQL).

**Artefakt:** tabela „SEQ → ECS” + lista 10–15 kwerend do przepisania.

**2) Przygotowanie klastra i przestrzeni w Kibanie (1–2 h, jednorazowo)**

1. **Spaces/RBAC:**
   * Utwórz Space „Observability” + (opcjonalnie) osobne Space per zespół.
   * Role: „Logs Reader”, „SRE”, „App Owner” (z różnymi uprawnieniami do logs-\*).
2. **ILM:** utwórz politykę, np. logs-default-ilm
   * hot: 7 dni, rollover: 50 GB/1 dzień
   * warm: 30 dni, index.priority: 50
   * cold: 90 dni, read-only
   * delete: 365 dni (lub wg compliance)
3. **Templates/data streams:** szablon logs-\* (ECS, ILM=logs-default-ilm, priorytet wysoki).
4. **Data View:** logs-\* z time field @timestamp.
5. **Kanały powiadomień:** e-mail/webhook/Slack dla alertów.

**Artefakt:** 3 zrzuty ekranu (ILM, Template, Data View), zapisane w runbooku.

**3) Zmiany w aplikacjach — logowanie w ECS i korelacja APM (2–4 h / usługę)**

**Wspólne (A i B)**

* Pakiety (przykładowo):
  + Elastic.CommonSchema.Serilog (EcsTextFormatter)
  + Elastic.Apm.NetCoreAll (APM) + Elastic.Apm.SerilogEnricher (korelacja)
* Ustawienia wspólne Serilog:
* Log.Logger = new LoggerConfiguration()
* .MinimumLevel.Override("Microsoft", LogEventLevel.Warning)
* .Enrich.FromLogContext()
* .Enrich.WithMachineName()
* .Enrich.WithEnvironmentName()
* .Enrich.WithElasticApmCorrelationInfo() // dopisze trace.id / transaction.id
* .Enrich.WithProperty("service.name", "api") // koniecznie spójne z APM
* .Enrich.WithProperty("service.environment", "prod")
* .CreateLogger();
* **Mapowanie własnych pól do ECS** – przy logowaniu używaj nazw ECS (przynajmniej dla kluczowych):
* logger.ForContext("user.id", userId)
* .ForContext("http.request.method", method)
* .ForContext("http.response.status\_code", status)
* .Information("Handled {Route} in {ElapsedMs}ms", route, elapsed);

**Wariant A — direct sink do Elasticsearch (data streams)**

* Dodaj sink do ES z obsługą data streams (pakiet dostawcy Elastic lub community z DS).
* Minimalny przykład:
* .WriteTo.Elasticsearch(new[] { new Uri("https://es01:9200") }, opts => {
* opts.DataStream = new DataStreamName("logs", "api", "prod"); // logs-api-prod
* opts.BootstrapMethod = BootstrapMethod.Failure; // załóż szablony, jeśli brak
* // opcjonalnie: opts.EmitEventFailure = ...
* })
* **Uwaga:** ustaw certyfikaty/uwierzytelnienie (API key lub user/pw), ServerCertificateValidationCallback jeżeli środowisko testowe ma self-signed.

**Wariant B — plik NDJSON (ECS) + agent**

* Serilog do pliku (rolling daily) z **EcsTextFormatter**:
* .WriteTo.File(new EcsTextFormatter(),
* "logs/api-prod.ndjson",
* rollingInterval: RollingInterval.Day,
* shared: true)
* Elastic Agent (integra **logs**) albo Filebeat log input zbiera \*.ndjson:
  + Ścieżki: C:\apps\api\logs\\*.ndjson
  + Pipeline: target logs-api-prod (data stream), @timestamp z eventu.
* Zadbaj o **rotację i retencję plików** (np. 7–14 dni) i dostęp do katalogu dla Agenta.

**Artefakt:** PR z diffem Program.cs/appsettings.json + checklist TLS/creds.

**4) Testy techniczne po stronie ingestu (30–60 min / usługę)**

1. Uruchom usługę i wygeneruj 10–20 logów (INFO/ERROR z wyjątkiem).
2. W Kibanie (Discover → logs-\*) sprawdź:
   * Widoczność pól ECS (service.name, trace.id, http.\*, user.id, error.\*).
   * Czy @timestamp jest poprawny.
   * Czy eventy trafiają do właściwego data stream (logs-api-prod).
3. W APM → Services → wybierz transakcję → karta **Logs** – czy działa korelacja?

**Artefakt:** krótki raport PASS/FAIL + lista brakujących pól (jeśli są).

**5) Przepisanie kwerend SEQ → KQL + gotowe „snippety” (~1 h)**

| **Cel (SEQ)** | **Odpowiednik KQL (Kibana)** |
| --- | --- |
| @Level = 'Error' | log.level: "error" |
| Tekst zawiera „timeout” | message: "\*timeout\*" |
| Ma wyjątek | exists:error.message or exists:error.stack\_trace |
| Properties.UserId = '123' | user.id: "123" (albo labels.user\_id:"123") |
| HTTP 5xx | http.response.status\_code >= 500 |
| Ścieżka /api/orders | url.path: "/api/orders" |
| Konkretny serwis | service.name: "api" |
| Środowisko prod | service.environment: "prod" |
| Błędy EF | error.message: "\*SqlException\*" |

**Artefakt:** zapisz 10–15 Saved Searches w Kibanie (najczęstsze przypadki).

**6) Dashboardy (Lens) odpowiadające widokom z SEQ (1–2 h)**

Utwórz dashboard „Logs – API (prod)” z 6–8 panelami:

1. **Errors over time**: count() gdzie log.level: error (interval auto).
2. **Top exception types**: terms po error.type.
3. **Top endpoints by 5xx**: filtr http.response.status\_code >= 500, terms: url.path.
4. **Slow endpoints**: percentyle event.duration (lub własne pole elapsed\_ms).
5. **Top users by errors**: terms: user.id z filtrem log.level: error.
6. **Host heatmap**: terms: host.name × czas (jeśli logujesz host.name).
7. **Live tail** (widget linkujący do Logs → Live).
8. **APM correlation**: link parametryczny do APM (service.name/env/time range).

**Artefakt:** 1 dashboard per usługa (sklonuj i podmień filtr service.name).

**7) Alerty (odpowiedniki z SEQ) (1–2 h)**

1. **Wzrost błędów**: typ **Log threshold**
   * Query: log.level: "error" and service.name: "api"
   * Warunek: count > 50 w 5 min vs baseline (możesz dodać Group by: service.version).
2. **HTTP 5xx spike**: **Custom KQL** + próg count > X.
3. **Cisza w logach (deadman)**: brak eventów z service.name: "api" w 10 min.
4. **Konkretny wyjątek**: error.type: "SqlException" count > 5 w 5 min.
5. **Długi czas obsługi**: event.duration > 1s count > 20 w 5 min.

**Kanały:** e-mail/Slack/webhook (zawartość: link do Discover + Saved Search + kontekst).

**Artefakt:** 5 reguł alertów + test wysyłki.

**8) SLO/SI i raporty operacyjne (opcjonalnie, 1–2 h)**

* Jeżeli macie APM: utwórz **SLO** (latencja p95, error rate) i **Error budget**.
* Zestaw miesięczny: liczba błędów per service.version, top exceptiony, 5xx per endpoint.

**9) Równoległy bieg (A/B) i „Definition of Done” (1 tydzień w tle)**

* Uruchom SEQ i Elastic równolegle.
* Sprawdź na realnym incydencie:
  + Czy **znajdujesz** to samo zdarzenie szybciej/nie wolniej?
  + Czy **dashboard** daje równoważny lub lepszy wgląd?
  + Czy **alert** przyszedł w odpowiednim czasie i z poprawnym linkiem?
* Zbierz feedback zespołu (15 min daily): brakujące widoki/kwerendy.

**DoD (migracja gotowa):**

* 100% usług loguje w ECS i ma service.name, service.environment.
* Działa korelacja **APM ↔ Logs**.
* Są wdrożone dashboardy i alerty minimalne.
* ILM i Security/RBAC skonfigurowane.
* Runbook: „Jak znaleźć zdarzenie X” (1 strona) + skróty KQL.

**10) Wyłączenie SEQ (1–2 h)**

1. Zamroź backup SEQ (DB/plik).
2. Ostatni tydzień: tryb tylko-do-odczytu, komunikat w zespole.
3. Po tygodniu: dekomisja usługi/VM (zgodnie z procedurą IT).

**11) Operacje ciągłe (runbook, 1 h)**

* **Checklist przy starcie nowej usługi:**
  1. service.name, service.environment, service.version ustawione.
  2. Korelacja APM (enricher + propagacja tracecontext).
  3. Mapa pól do ECS (min. http.\*, user.id, error.\*).
  4. Dashboard sklonowany + filtry pod nową usługę.
  5. 2–3 alerty aktywne.
* **Higiena indeksów:** obserwuj kardynalność pól (np. labels.\*), unikaj „losowych” wartości jako keyword bez potrzeby.
* **Koszty:** weryfikuj rollover (wielkość shardów 20–50 GB), retencję per env (prod > stage).

**Załączniki – gotowce do skopiowania**

**A) appsettings.json (wariant A – direct sink, z API key)**

{

"Serilog": {

"MinimumLevel": {

"Default": "Information",

"Override": { "Microsoft": "Warning" }

},

"Enrich": [ "FromLogContext", "WithMachineName", "WithEnvironmentName", "WithElasticApmCorrelationInfo" ],

"WriteTo": [

{

"Name": "Elasticsearch",

"Args": {

"NodeUris": "https://es01:9200;https://es02:9200",

"DataStream": { "Type": "logs", "Dataset": "api", "Namespace": "prod" },

"BootstrapMethod": "Failure",

"ModifyConnectionSettings": "IncludeServerCertificateValidation" // wg pakietu – placeholder

}

}

]

},

"Elastic": {

"ApiKey": "<id:secret>",

"Cloud": false

}

}

Jeśli używacie user/pw i TLS z własnym CA — dodajcie sekcję certyfikatu (zgodnie z pakietem, często przez ServerCertificateValidationCallback albo CertificateFingerprint).

**B) appsettings.json (wariant B – plik NDJSON)**

{

"Serilog": {

"MinimumLevel": { "Default": "Information" },

"Using": [ "Elastic.CommonSchema.Serilog" ],

"WriteTo": [

{

"Name": "File",

"Args": {

"path": "logs/api-prod.ndjson",

"formatter": "Elastic.CommonSchema.Serilog.EcsTextFormatter, Elastic.CommonSchema.Serilog",

"rollingInterval": "Day",

"shared": true

}

}

],

"Enrich": [ "FromLogContext", "WithMachineName", "WithEnvironmentName", "WithElasticApmCorrelationInfo" ]

}

}

**C) Elastic Agent – input (wariant B, skrót idei)**

* Policy → Integration **Logs** (custom log files)
* Paths: C:\apps\api\logs\\*.ndjson
* Dataset: api, Namespace: prod (→ logs-api-prod)
* Time field: @timestamp
* Advanced: parsers.ndjson: true

**D) ILM – przykład (skrót)**

* hot: rollover 50 GB lub 1d, min\_prirep: 1
* warm: 30d, forcemerge 1 segment, priorytet 50
* cold: 90d, searchable snapshot (jeśli dostępne)
* delete: 365d

**E) 5 alertów – szybkie definicje**

1. **Errors spike**: KQL service.name:"api" and log.level:"error" → count > 50/5min.
2. **5xx spike**: KQL service.name:"api" and http.response.status\_code >= 500 → count > 20/5min.
3. **Deadman**: brak eventów z service.name:"api" przez 10 min.
4. **SqlException burst**: error.message:\*SqlException\* → count > 5/5min.
5. **Slow ops**: event.duration > 1000000000 (1s) → count > 20/5min.

**Ryzyka & jak je wygasić**

* **Rozjechane nazwy pól:** trzymaj się ECS; dodaj test jednostkowy walidujący kluczowe pola (snapshot JSON).
* **Brak korelacji APM↔Logs:** sprawdź, czy service.name i trace.id są w logach; czy ruch HTTP przechodzi przez middleware APM.
* **Koszty storage:** za duże shardy / wysoka kardynalność – dostosuj rollover i znormalizuj wartości.
* **Utrata danych przy outage ES (wariant A):** rozważ wariant B dla krytycznych usług lub włącz retry/buffer w sinku.

**Harmonogram „1 sprint”**

**Dzień 1–2:** 0–2 (decyzja, ILM, template, Data View, RBAC)  
**Dzień 3–4:** 3–4 (jedna usługa pilotażowa: kod + test ingestu + korelacja APM)  
**Dzień 5:** 5–7 (Saved Searches, dashboard, alerty)  
**Tydzień 2:** rollout na pozostałe usługi + A/B z SEQ, poprawki, DoD  
**Koniec tyg. 2:** zamrożenie SEQ → read-only → plan dekomisji

**Krótkie podsumowanie (co zyskujesz vs SEQ)**

* Standaryzację **ECS** (gotowe panele/alerty + łatwa korelacja z APM).
* **Data streams + ILM** → automatyczna retencja i kontrola kosztów.
* **RBAC/Spaces** zamiast „all access”.
* Wariant B daje **durability** na dysku jak w SEQ (z buforem), wariant A – prostotę.