Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

Тенденції розвитку інформаційних систем та технологій

Централізовані системи логування. EFK.

Виконав
студент групи ІТ-41ф

Новиков Д. М.

Перевірив:

ас. Цимбал С. І.

Київ 2024

Мета роботи: ознайомлення із централізованими системами логування на прикладі EFK.

Хід роботи:

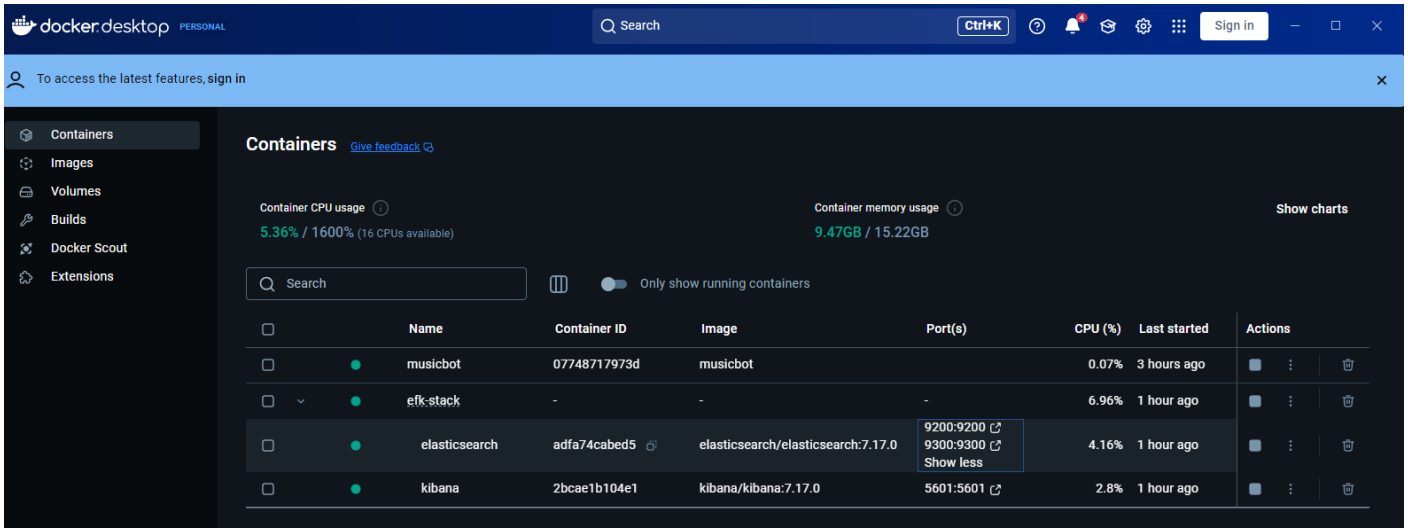
1. Запустити тестовий EFK стек

Оскільки я буду писати Web API на ASP.NET Core, провайдер Serilog взаємодіє з Elasticsearch напряму. Таким чином, зі стеку EFK (Elasticsearch, Fluentd, Kibana) мені не потрібен Fluentd. Його роль збирача та форматувача логів виконує спеціальний компонент Serilog - Sink. Підготуємо Docker-compose.yml для створення ELK:

```yaml
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0
    container_name: elasticsearch
    environment:
      discovery.type: single-node
      xpack.monitoring.enabled: true
      xpack.watcher.enabled: false
    ports:
      - 9200:9200
      - 9300:9300
    volumes: # Stores elasticsearch data locally on the es_data Docker volume
      - es_data:/usr/share/elasticsearch/data

  kibana:
    image: docker.elastic.co/kibana/kibana:7.17.0
    container_name: kibana
    environment:
      ELASTICSEARCH_URL: http://elasticsearch:9200
    ports:
      - 5601:5601
    depends_on:
      - elasticsearch

volumes:
  es_data:
```

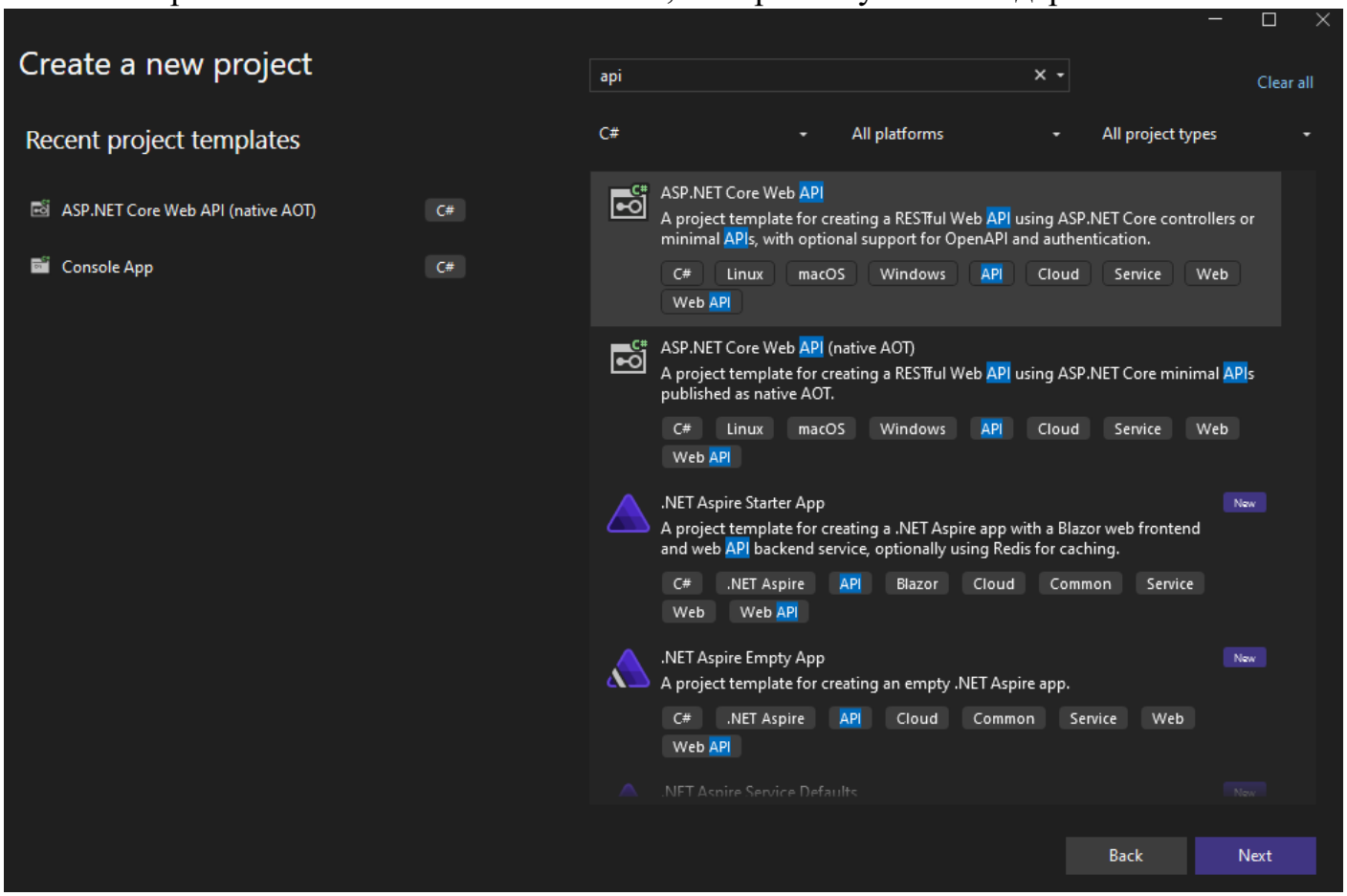
Запустимо стек за допомогою команди docker-compose up:

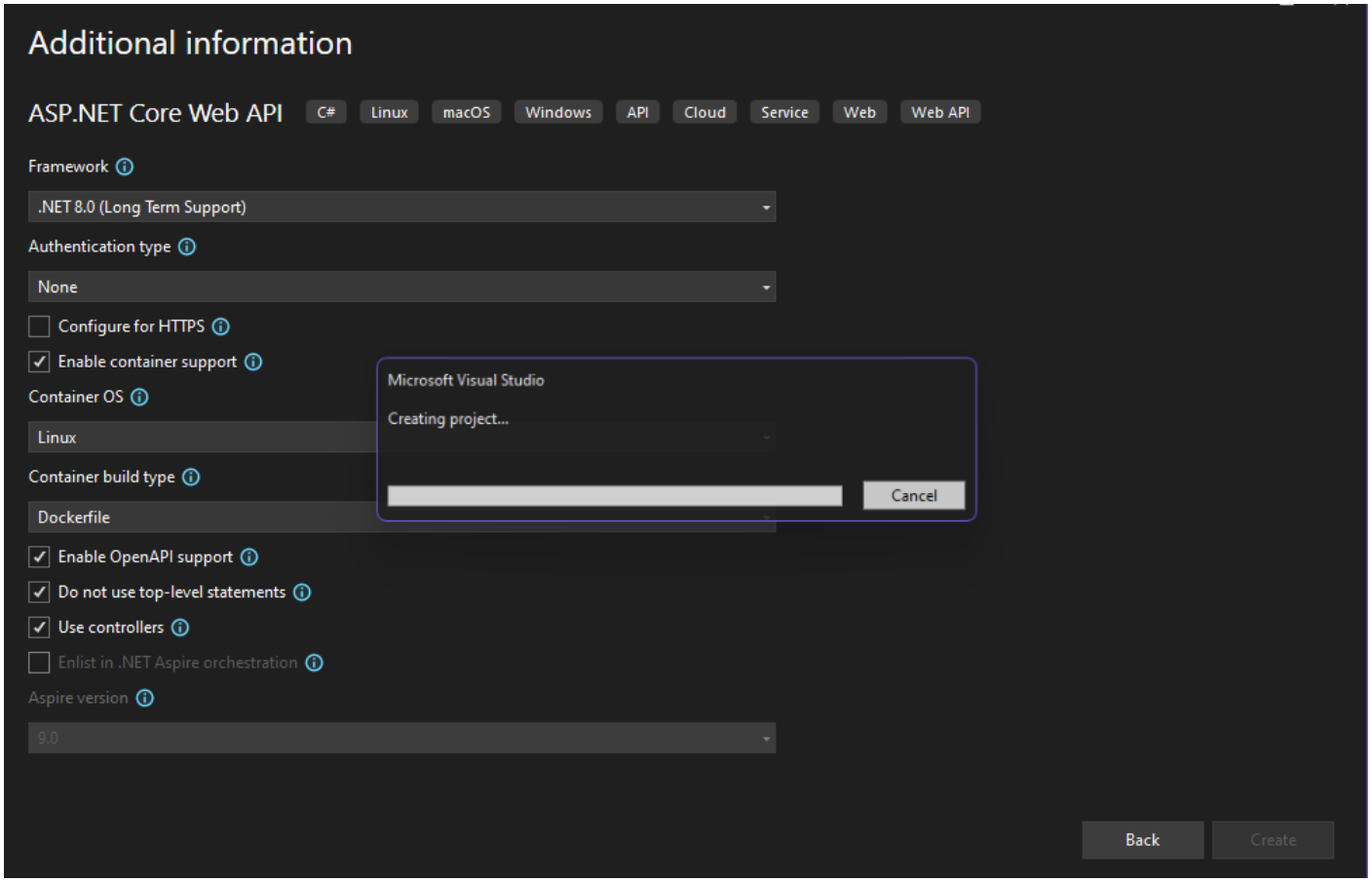
```
PS E:\Univer\Sharaga 13 Term\TRIST\LR3\EFK-stack> docker-compose up -d
[+] Running 3/3
 ✓ Network efk-stack_default  Created
 ✓ Container elasticsearch    Started
 ✓ Container kibana           Started
```



2. Розробити найпростіший застосунок, який буде записувати логи, щоб їх можна було побачити , фільтрувати у Kibana.

Створимо Web API на ASP.NET Core, використовуючи стандартний шаблон:





а) Додамо необхідні залежності:

- Serilog;
- Serilog.AspNetCore;
- Serilog.Sinks.Console;
- Serilog.Sinks.Elasticsearch.

б) У файлі Program.cs налаштуємо конфігурацію для Serilog, а також додамо кілька логів:

```
public class Program
{
	public static void Main(string[] args)
	{
		// Configure Serilog using appsettings.json.
		Log.Logger = new LoggerConfiguration()
			.ReadFrom.Configuration(new ConfigurationBuilder()
				.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
				.Build())
			.CreateLogger();

		try
		{
			Log.Information("Starting the application.");

			var builder = WebApplication.CreateBuilder(args);

			…

			app.Run();
		}
		catch (Exception ex)
		{
			Log.Fatal(ex, "Application startup failed.");
		}
		finally
		{
			Log.CloseAndFlush();
		}
	}
}
```

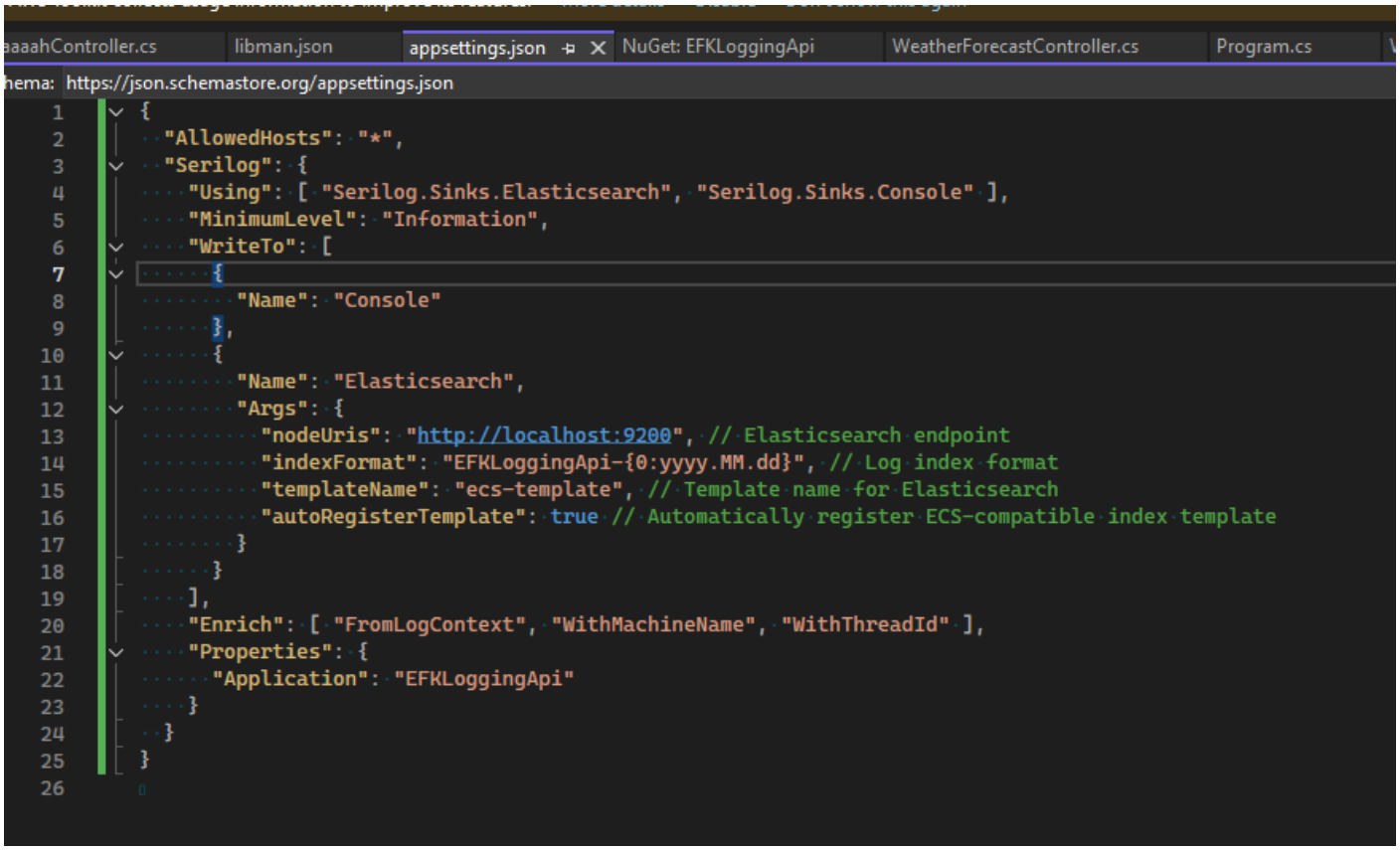
в) Модифікуємо існуючий контролер WeatherForecastController, додавши логування на початку виклику методу та перед поверненням результату:

```
    [HttpGet(Name = "GetWeatherForecast")]
    0 references
    public IEnumerable<WeatherForecast> Get()
    {
        _logger.LogInformation("Weather forecast requested.");

        var forecasts = Enumerable.Range(1, 5).Select(index => new WeatherForecast
        {
            Date = DateOnly.FromDateTime(DateTime.Now.AddDays(index)),
            TemperatureC = Random.Shared.Next(-20, 55),
            Summary = Summaries[Random.Shared.Next(Summaries.Length)]
        })
        .ToArray();

        _logger.LogInformation("Returning {Count} forecasts.", forecasts.Length);
        return forecasts;
    }
}
```

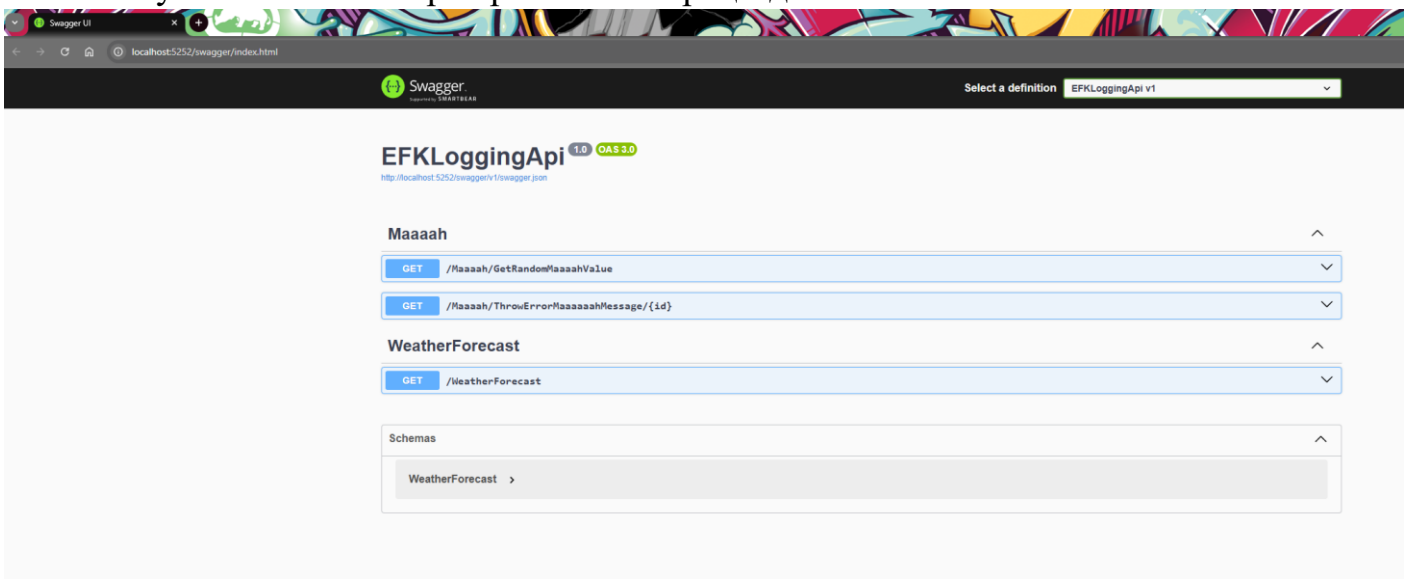
г) Створимо додатковий контролер для тестування:

```
namespace EFKLoggingApi.Controllers
{
    [ApiController]
    [Route("[controller]")]
    3 references
    public class MaaaahController : ControllerBase
    {
        private readonly ILogger<MaaaahController> _logger;
        0 references
        public MaaaahController(ILogger<MaaaahController> logger)
        {
            _logger = logger;
        }

        [HttpGet("GetRandomMaaaahValue", Name = "GetRandomMaaaahValue")]
        0 references
        public int GetRandomMaaaahValue()
        {
            _logger.LogInformation("GetRandomMaaaahValue requested.");

            var randomValue = new Random().Next(0, 100);
            _logger.LogInformation("Maaaaaaaaaah value is [{RandomMaaaahValue}].", randomValue);

            return randomValue;
        }

        [HttpGet("ThrowErrorMaaaaaahMessage/{id}", Name = "ThrowErrorMaaaaaahMessage")]
        0 references
        public string ThrowErrorMaaaaaahMessage(int id)
        {
            _logger.LogInformation("ThrowErrorMaaaaaahMessage requested.");

            try
            {
                if (id <= 0)
                    throw new Exception($"id cannot be less than or equal to 0. Value passed is [{id}].");

                return id.ToString();
            }
            catch (Exception ex)
            {
                _logger.LogError(ex, ex.Message);
            }

            return string.Empty;
        }
    }
}
```

д) Модифікуємо конфігураційний файл appsettings.json, щоб налаштувати Serilog. Вкажемо формат індексу, адресу (endpoint) Elasticsearch і додаткові поля:



Запустимо API та перевіримо його працездатність:



Maaaah

GET /Maaaah/GetRandomMaaaahValue

Parameters | Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5252/Maaaah/GetRandomMaaaahValue' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5252/Maaaah/GetRandomMaaaahValue
```

Server response

Code	Details
200	Response body: `13` (Download)

Response headers

```
content-type: application/json; charset=utf-8
date: Sun,01 Dec 2024 19:52:09 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	OK Media type: text/plain Controls Accept header. Example Value | Schema: `0`	*No links*

WeatherForecast

GET /WeatherForecast

Parameters | Cancel

No parameters

Execute | Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:5252/WeatherForecast' \
  -H 'accept: text/plain'
```

Request URL

```
http://localhost:5252/WeatherForecast
```

Server response

Code	Details
200	Response body (Download)


```
[
  {
    "date": "2024-12-02",
    "temperatureC": -7,
    "temperatureF": 20,
    "summary": "Chilly"
  },
  {
    "date": "2024-12-03",
    "temperatureC": 20,
    "temperatureF": 67,
    "summary": "Scorching"
  },
  {
    "date": "2024-12-04",
    "temperatureC": -10,
    "temperatureF": 15,
    "summary": "Chilly"
  },
  {
    "date": "2024-12-05",
    "temperatureC": -8,
    "temperatureF": 18,
    "summary": "Cool"
  },
  {
    "date": "2024-12-06",
    "temperatureC": -1,
```

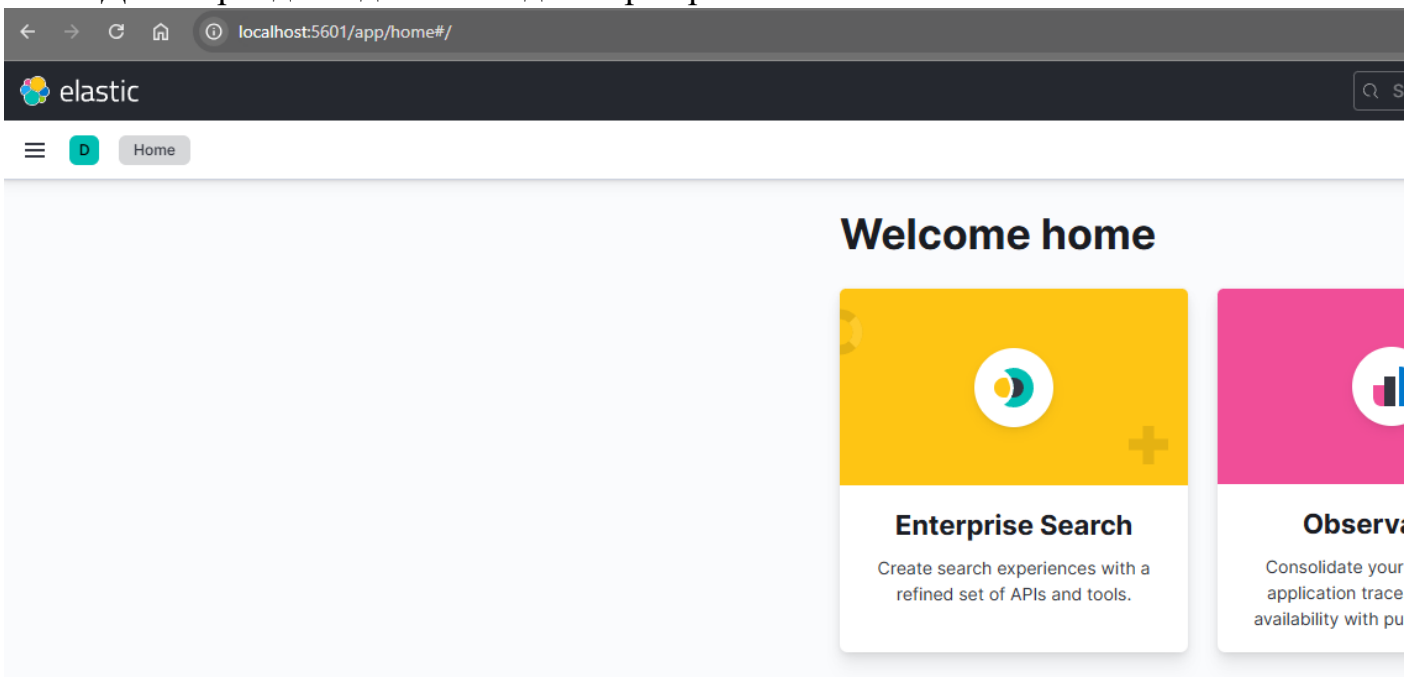

Response headers

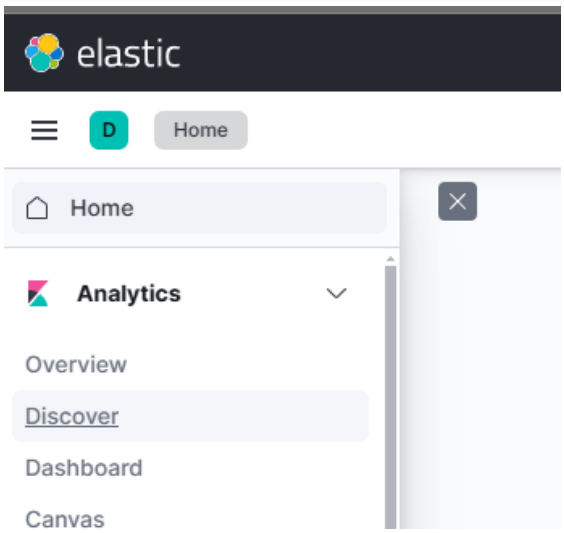
```
content-type: application/json; charset=utf-8
date: Sun,01 Dec 2024 19:52:25 GMT
server: Kestrel
transfer-encoding: chunked
```

Responses

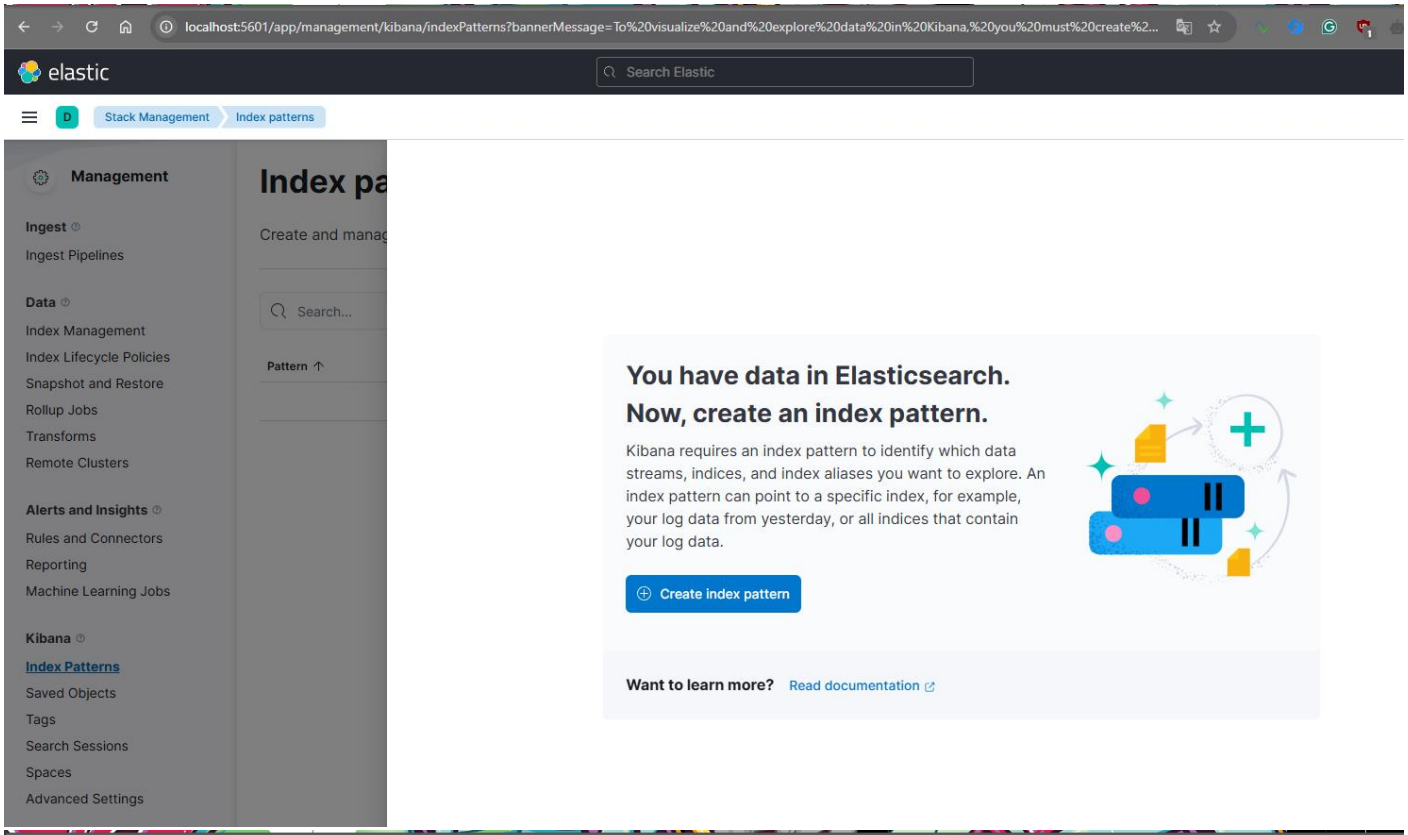
```
[21:52:17 INF] Executed endpoint 'EFKLoggingApi.Controllers.MaaaahController.ThrowErrorMaaaaaahMessage (EFKLoggingApi)'
[21:52:17 INF] Request finished HTTP/1.1 GET http://localhost:5252/Maaaah/ThrowErrorMaaaaaahMessage/5 - 200 null text/pl
ain; charset=utf-8 21.5716ms
[21:52:20 INF] Request starting HTTP/1.1 GET http://localhost:5252/Maaaah/ThrowErrorMaaaaaahMessage/-2 - null null
[21:52:20 INF] Executing endpoint 'EFKLoggingApi.Controllers.MaaaahController.ThrowErrorMaaaaaahMessage (EFKLoggingApi)'
[21:52:20 INF] Route matched with {action = "ThrowErrorMaaaaaahMessage", controller = "Maaaah"}. Executing controller ac
tion with signature System.String ThrowErrorMaaaaaahMessage(Int32) on controller EFKLoggingApi.Controllers.MaaaahControl
ler (EFKLoggingApi).
[21:52:20 INF] ThrowErrorMaaaaaahMessage requested.
[21:52:20 ERR] id cannot be less than or equal to 0. Value passed is [-2].
System.Exception: id cannot be less than or equal to 0. Value passed is [-2].
   at EFKLoggingApi.Controllers.MaaaahController.ThrowErrorMaaaaaahMessage(Int32 id) in E:\Univer\Sharaga 13 Term\TRIST\
LR3\EFKLoggingApi\Controllers\MaaaahController.cs:line 34
[21:52:20 INF] Executing ObjectResult, writing value of type 'System.String'.
[21:52:20 INF] Executed action EFKLoggingApi.Controllers.MaaaahController.ThrowErrorMaaaaaahMessage (EFKLoggingApi) in 9
1.9511ms
[21:52:20 INF] Executed endpoint 'EFKLoggingApi.Controllers.MaaaahController.ThrowErrorMaaaaaahMessage (EFKLoggingApi)'
[21:52:20 INF] Request finished HTTP/1.1 GET http://localhost:5252/Maaaah/ThrowErrorMaaaaaahMessage/-2 - 200 0 text/plai
n; charset=utf-8 101.1145ms
[21:52:23 INF] Request starting HTTP/1.1 GET http://localhost:5252/WeatherForecast - null null
[21:52:23 INF] Executing endpoint 'EFKLoggingApi.Controllers.WeatherForecastController.Get (EFKLoggingApi)'
[21:52:23 INF] Route matched with {action = "Get", controller = "WeatherForecast"}. Executing controller action with sig
nature System.Collections.Generic.IEnumerable`1[EFKLoggingApi.WeatherForecast] Get() on controller EFKLoggingApi.Control
lers.WeatherForecastController (EFKLoggingApi).
[21:52:25 INF] Weather forecast requested.
[21:52:25 INF] Returning 5 forecasts.
[21:52:25 INF] Executing ObjectResult, writing value of type 'EFKLoggingApi.WeatherForecast[]'.
[21:52:25 INF] Executed action EFKLoggingApi.Controllers.WeatherForecastController.Get (EFKLoggingApi) in 1522.0483ms
[21:52:25 INF] Executed endpoint 'EFKLoggingApi.Controllers.WeatherForecastController.Get (EFKLoggingApi)'
[21:52:25 INF] Request finished HTTP/1.1 GET http://localhost:5252/WeatherForecast - 200 null application/json; charset=
utf-8 1530.5107ms
```

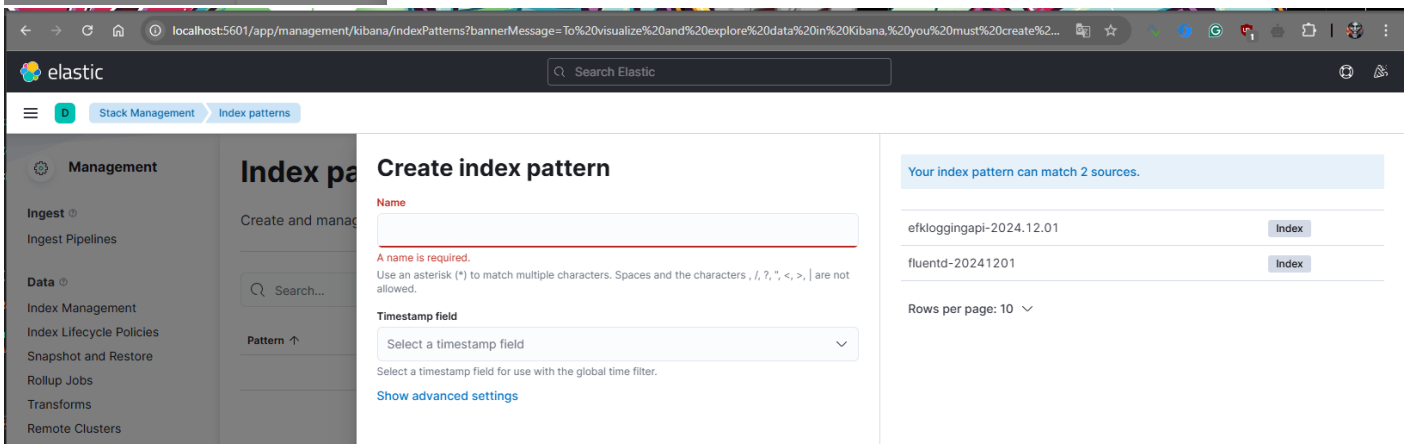
Далі перейдемо до Kibana для перевірки логів:

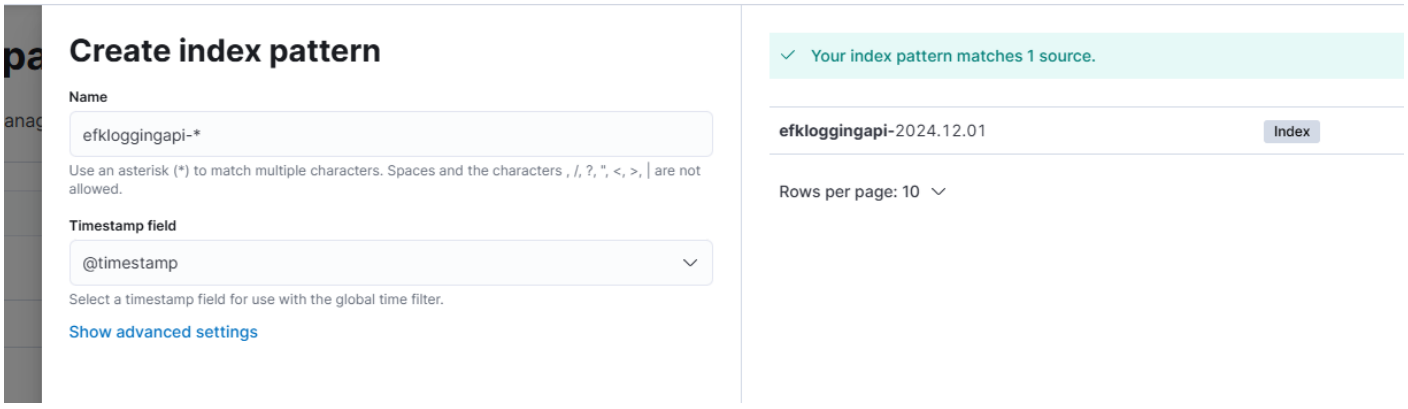


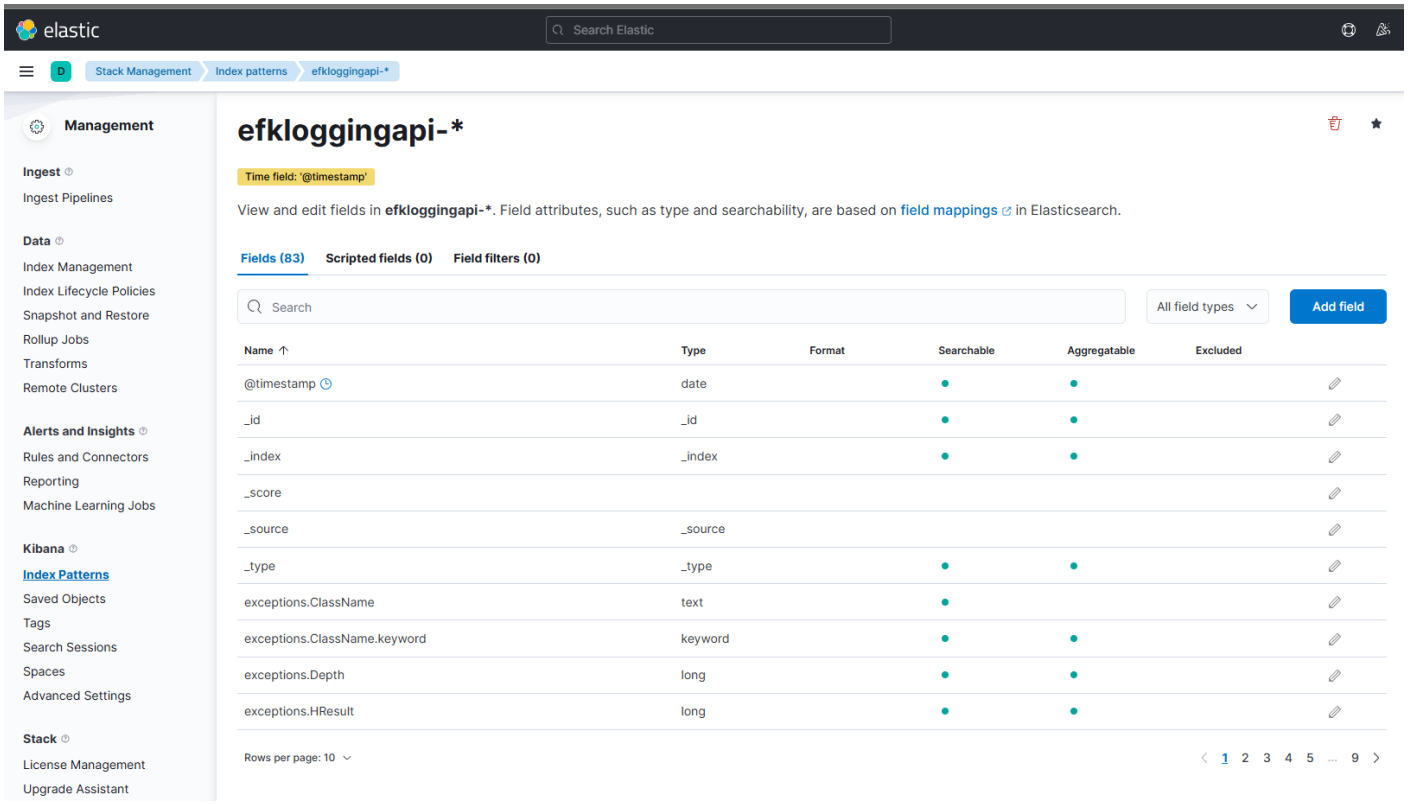


а) Створимо шаблон індексу для даних з нашого API:

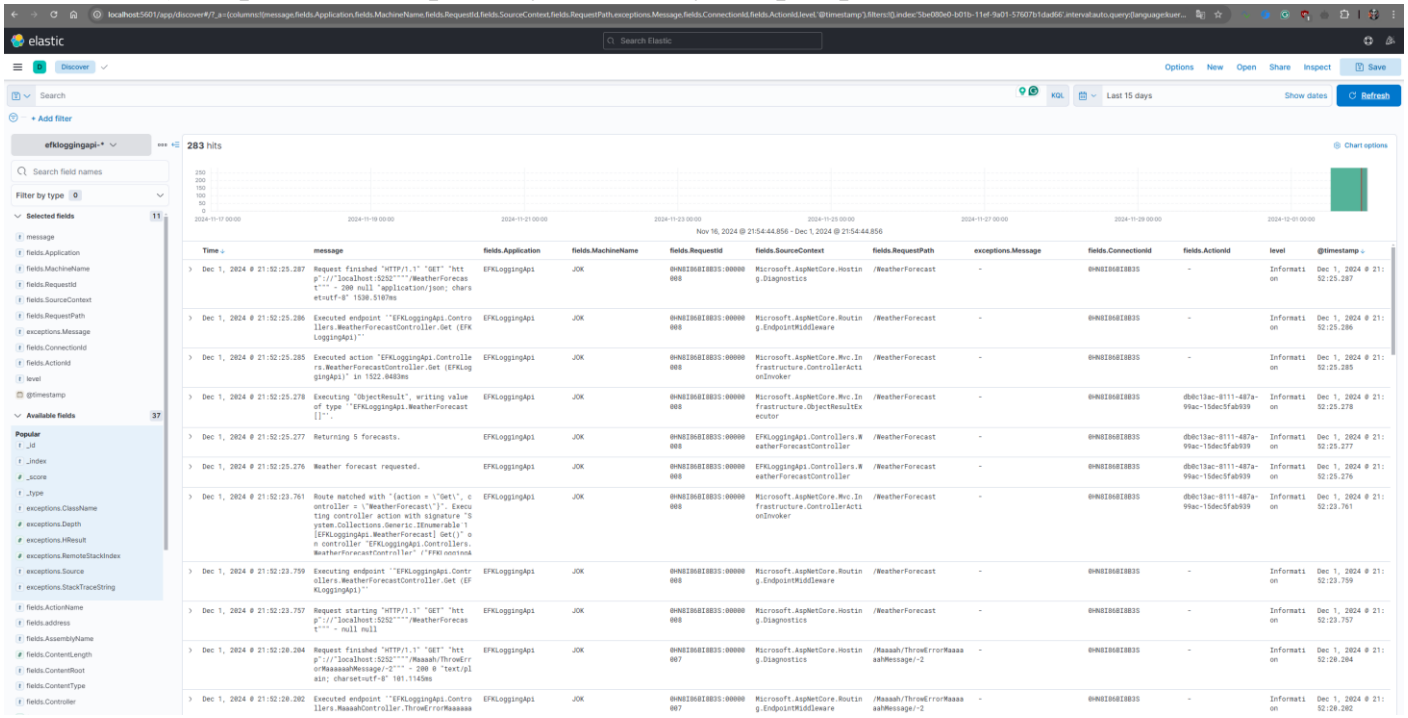

localhost:5601/app/management/kibana/indexPatterns?bannerMessage=To%20visualize%20and%20explore%20data%20in%20Kibana,%20you%20must%20create%2...
elastic
Search Elastic
D
Stack Management
Index patterns
Management
Ingest
Ingest Pipelines
Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters
Alerts and Insights
Rules and Connectors
Reporting
Machine Learning Jobs
Kibana
Index Patterns
Saved Objects
Tags
Search Sessions
Spaces
Advanced Settings
Index pa
Create and manag
Search...
Pattern
You have data in Elasticsearch.
Now, create an index pattern.
Kibana requires an index pattern to identify which data streams, indices, and index aliases you want to explore. An index pattern can point to a specific index, for example, your log data from yesterday, or all indices that contain your log data.
Create index pattern
Want to learn more?
Read documentation


localhost:5601/app/management/kibana/indexPatterns?bannerMessage=To%20visualize%20and%20explore%20data%20in%20Kibana,%20you%20must%20create%2...
elastic
Search Elastic
D
Stack Management
Index patterns
Management
Ingest
Ingest Pipelines
Data
Index Management
Index Lifecycle Policies
Snapshot and Restore
Rollup Jobs
Transforms
Remote Clusters
Index pa
Create and manag
Search...
Pattern
Create index pattern
Name
A name is required.
Use an asterisk (*) to match multiple characters. Spaces and the characters , /, ?, ", <, >, | are not allowed.
Timestamp field
Select a timestamp field
Select a timestamp field for use with the global time filter.
Show advanced settings
Your index pattern can match 2 sources.
efkloggingapi-2024.12.01
Index
fluentd-20241201
Index
Rows per page: 10

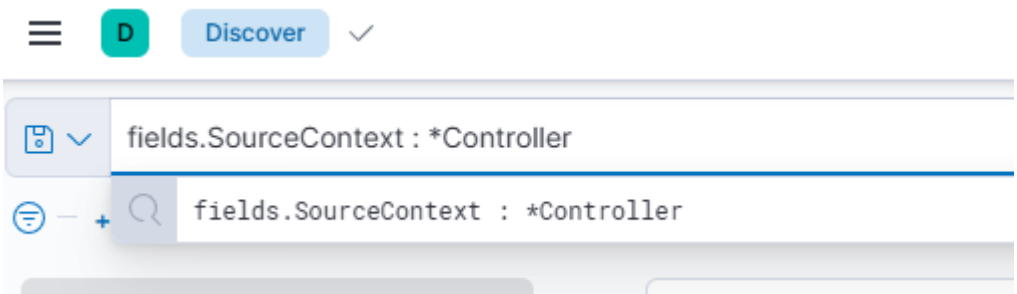

Create index pattern
Name
efkloggingapi-*
Use an asterisk (*) to match multiple characters. Spaces and the characters , /, ?, ", <, >, | are not allowed.
Timestamp field
@timestamp
Select a timestamp field for use with the global time filter.
Show advanced settings
Your index pattern matches 1 source.
efkloggingapi-2024.12.01
Index
Rows per page: 10

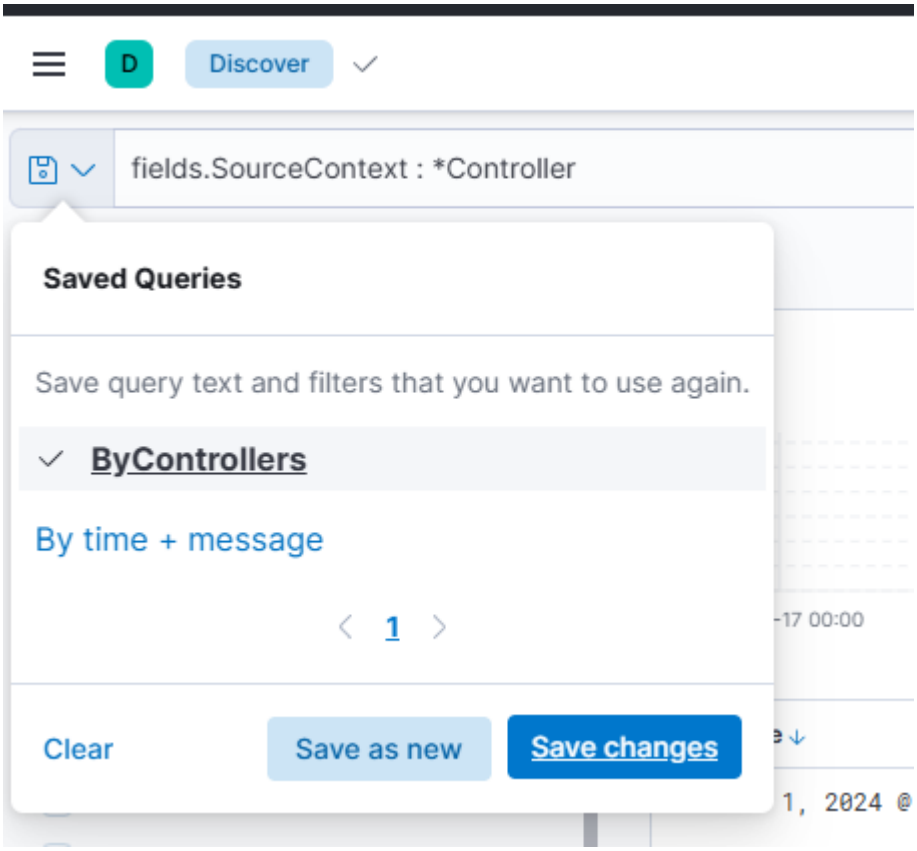


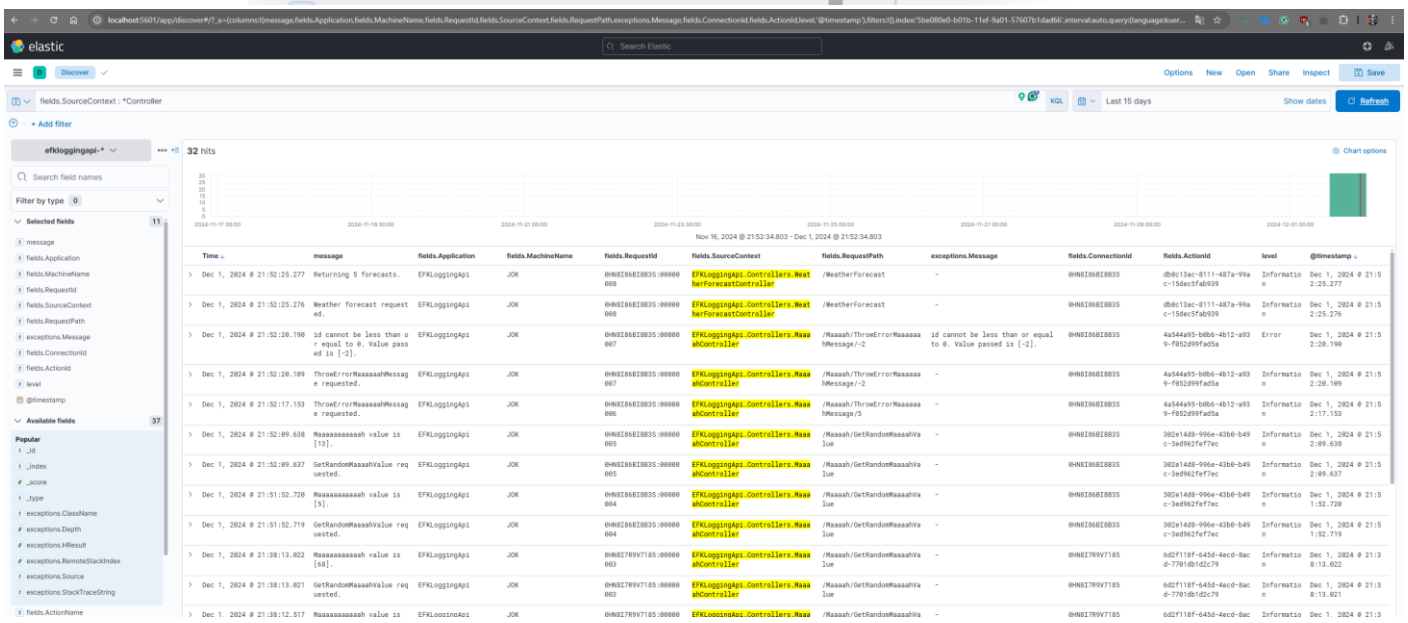
б) Перейдемо до розділу Discovery і перевіримо наші логи:



в) Додамо фільтр за SourceContext, щоб відобразити лише ті логи, які були згенеровані методами контролерів:







Висновки: в результаті виконання цієї лабораторної роботи було ознайомлено з базовими концепціями технологій централізованих систем логування на прикладі EFK. Однак через вибір мови програмування Fluentd не використовувався.

На основі отриманих знань було реалізовано практичну частину, яка полягала у створенні застосунку, що передає логи до централізованої системи для їх збереження, подальшої обробки та аналізу.

Вихідний код застосунку можна знайти за наступним посиланням на GitHub.