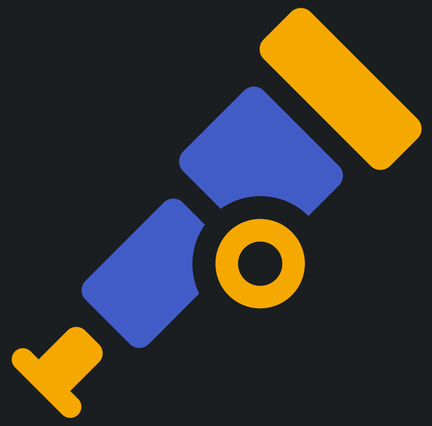


OpenTelemetry

Для самых маленьких

Гольдебаев Александр

Observability

Зачем это нужно?

- Предупреждать возникновение ошибок
- Знать какая часть системы сломалась
- Отслеживать актуальность нового кода
- Сравнивать производительность
- Смотреть в красивые дашборды

Observability

Телеметрия

лишь малая часть

Телеметрия — это процесс сбора данных с ...

данных?

Logs

Traces

Metrics

Signals

Logs

Traces

Metrics

Logs

Logs

Классические логи

1. Простота
2. Чтение

1. Структурирование
2. Парсинг

Logs

Классические логи

1. Простота
2. Чтение

1. Структурирование
2. Парсинг

Структурированные логи

1. Анализ и фильтрация
2. Поиск и агрегирование
3. Интеграция
4. Объём данных

1. Производительность

Logs

Классические логи

Структурированные логи

Logs

Классические логи

```
2024-05-24 10:15:30 INFO: User JohnDoe has logged in
```

Структурированные логи

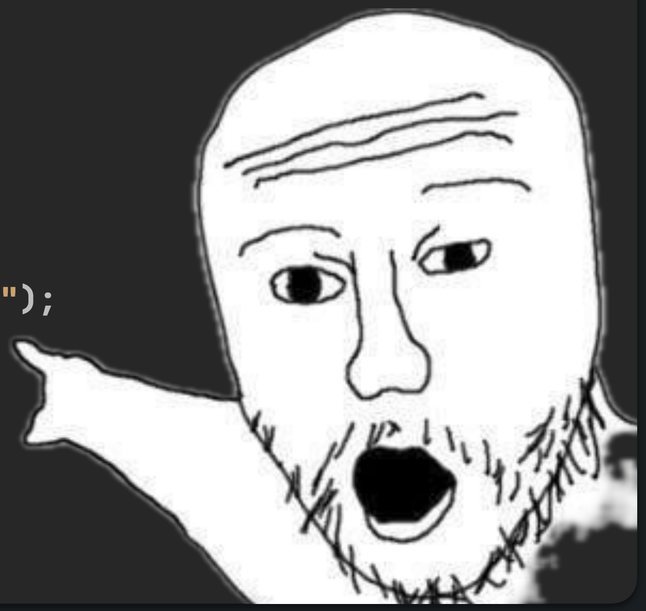
```
{
  "timestamp": "2024-05-24T10:15:30Z",
  "level": "INFO",
  "message": "User has logged in",
  "username": "JohnDoe"
}
```

ILogger

```
public class ExampleController : ControllerBase
{
    private readonly ILogger _logger;

    public ExampleController(ILogger<ExampleController> logger)
    {
        _logger = logger;
    }

    [HttpGet]
    public IResult Index()
    {
        _logger.LogInformation("Index action has been invoked.");
        return Results.Ok();
    }
}
```

ILogger

```
public class ExampleController : ControllerBase
{
    private readonly ILogger _logger;

    public ExampleController(ILogger<ExampleController> logger)
    {
        _logger = logger;
    }

    [HttpGet]
    public IResult Index()
    {
        _logger.Log(LogLevel.Information, "Index action has been invoked.");
        return Results.Ok();
    }
}
```



ILogger

```
info: MainAPI.Controllers.ExampleController[0]
      Index action has been invoked.
```

Serilog

Program.cs

```
Log.Logger = new LoggerConfiguration()
    .Enrich.FromLogContext()
    .WriteTo.Console()
    .CreateLogger();

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddSerilog();
```

- Serilog • 3.1.1
- Serilog.AspNetCore • 8.0.1
- Serilog.Sinks.Console • 5.0.1

Serilog

SettingController.cs

```
[HttpPost]
public IActionResult SetCity(Settings settings)
{
    if (ModelState.IsValid)
    {
        _logger.LogInformation("Setting default city to {City}", settings.City);

        _settingService.SetCity(settings.City);
        return RedirectToAction("Index");
    }

    return View("Index", settings);
}
```

Serilog

SettingController.cs

```
[HttpPost]
public IActionResult SetCity(Settings settings)
{
    if (ModelState.IsValid)
    {
        _logger.LogInformation("Setting default city to {City}", settings.City);

        _settingService.SetCity(settings.City);
        return RedirectToAction("Index");
    }

    return View("Index", settings);
}
```

```
[23:54:25 INF] Setting default city to Moscow
```

Serilog

WeatherForecast.cs

```
public class WeatherForecast
{
    public int Temperature { get; set; }
    public int Humidity { get; set; }
    public int WindSpeed { get; set; }
    public string Summary { get; set; } = string.Empty;
    public string City { get; set; } = string.Empty;

    public override string ToString() => "I'm a weather forecast!";
}
```

Serilog

WeatherForecastController.cs

```
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    _logger.LogInformation(
        "Weather forecast equal to {Forecast}",
        forecast);

    return View(forecast);
}
```

Serilog

WeatherForecastController.cs

```cs
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    _logger.LogInformation(
        "Weather forecast equal to {Forecast}",
        forecast);

    return View(forecast);
}
```

```
[00:21:19 INF] Weather forecast equal to I'm a weather forecast!
```

Serilog

WeatherForecastController.cs

```
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    _logger.LogInformation(
        "Weather forecast equal to {@Forecast}”,
        forecast);

    return View(forecast);
}
```

Serilog

WeatherForecastController.cs

```
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    _logger.LogInformation(
        "Weather forecast equal to {@Forecast}”,
        forecast);

    return View(forecast);
}
```

```
[00:23:37 INF] Weather forecast equal to {"Temperature": 23, "Humidity": 77, "WindSpeed": 10, "Summary": "Normal", "City": "Moscow", "$type": "WeatherForecast"}
```

Message Template Recommendations

https://messagetemplates.org/

Message Template Recommendations

1. Используйте имена свойств в качестве содержимого сообщения

https://messagetemplates.org/

Message Template Recommendations

1. Используйте имена свойств в качестве содержимого сообщения

```
// Not recommend:
Log.Information("Hello, {Person}", user);
```

```
// Do:
Log.Information("Hello, {User}”, user);
```

https://messagetemplates.org/

Message Template Recommendations

1. Используйте имена свойств в качестве содержимого сообщения

2. Избегайте точек и других знаков препинания (Логи ≠ Предложения)

https://messagetemplates.org/

Message Template Recommendations

1. Используйте имена свойств в качестве содержимого сообщения
2. Избегайте точек и других знаков препинания (Логи ≠ Предложения)
3. Используйте свойства шаблона для включения переменных в сообщения

https://messagetemplates.org/

Message Template Recommendations

1. Используйте имена свойств в качестве содержимого сообщения

2. Избегайте точек и других знаков препинания (Логи ≠ Предложения)

3. Используйте свойства шаблона для включения переменных в сообщения

```
// Don't:
Log.Information("The time is " + DateTime.Now);
```

```
// Do:
Log.Information("The time is {Now}", DateTime.Now);
```

https://messagetemplates.org/

Message Template Recommendations

1. Используйте имена свойств в качестве содержимого сообщения
2. Избегайте точек и других знаков препинания (Логи ≠ Предложения)
3. Используйте свойства шаблона для включения переменных в сообщения
4. Именуйте свойства, используя PascalCase

https://messagetemplates.org/

Расширяемся

```
2024-05-24 10:15:30 INFO: User JohnDoe has logged in.
{
  "Timestamp": "2024-05-24T10:15:30Z",
  "Level": "INFO",
  "Message": "User has logged in.",
  "Username": “JohnDoe”
}
```

Расширяемся

```
2024-05-24 10:15:30 INFO: User JohnDoe has logged in.
{
  "StartTime": "2024-05-24T10:15:28Z",
  "EndTime": "2024-05-24T10:15:28Z",
  "Level": "INFO",
  "Message": "User has logged in.",
  "Username": "JohnDoe"
}
```

Расширяемся

```
2024-05-24 10:15:30 INFO: User JohnDoe has logged in.
{
  "StartTime": "2024-05-24T10:15:28Z",
  "EndTime": "2024-05-24T10:15:28Z",
  "Level": "INFO",
  "Message": "User has logged in.",
  "Username": "JohnDoe",
  "Id": 44,
  "ParentId": 12
}
```

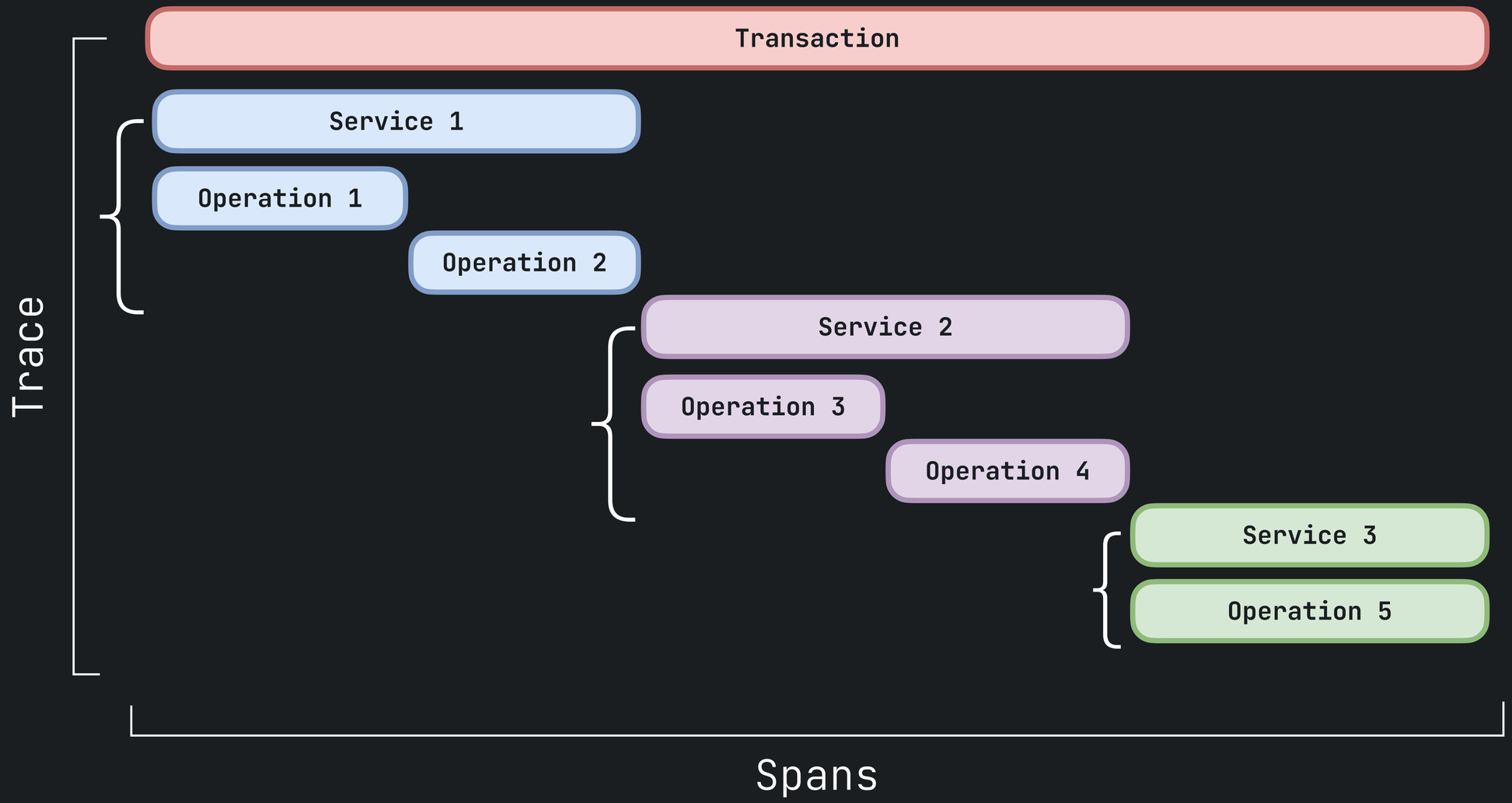
Расширяемся

```
2024-05-24 10:15:30 INFO: User JohnDoe has logged in.
{
  "StartTime": "2024-05-24T10:15:28Z",
  "EndTime": "2024-05-24T10:15:28Z",
  "Level": "INFO",
  "Message": "User has logged in.",
  "Username": "JohnDoe",
  "Id": 44,
  "ParentId": 12,
  "TraceId": 1
}
```

Span

```
2024-05-24 10:15:30 INFO: User JohnDoe has logged in.
{
  "StartTime": "2024-05-24T10:15:28Z",
  "EndTime": "2024-05-24T10:15:28Z",
  "Level": "INFO",
  "Message": "User has logged in.",
  "Username": "JohnDoe",
  "Id": 44,
  "ParentId": 12,
  "TraceId": 1
}
```

Distributed Tracing

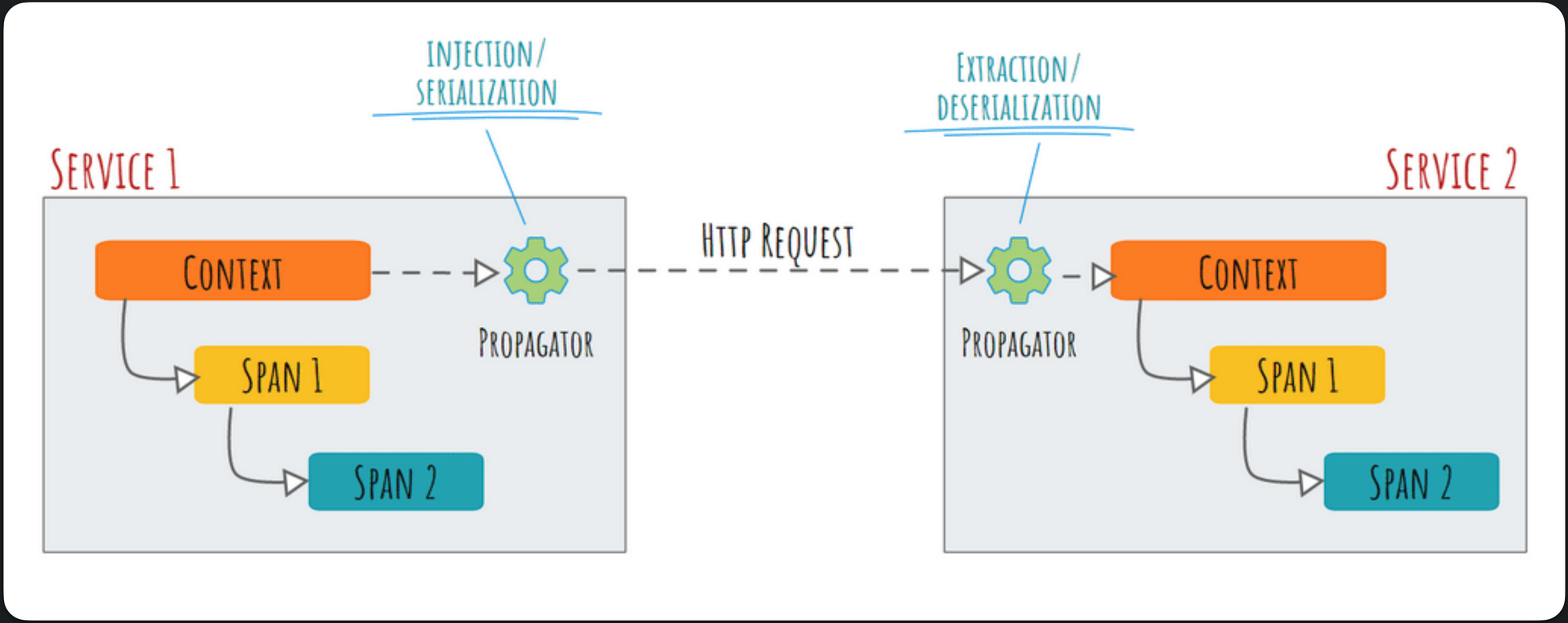


Distributed Tracing

- Отслеживание пути запроса
- Быстрое определение места сбоя
- Продолжительность каждого шага

Context Propagation

Concept that enables Distributed Tracing.

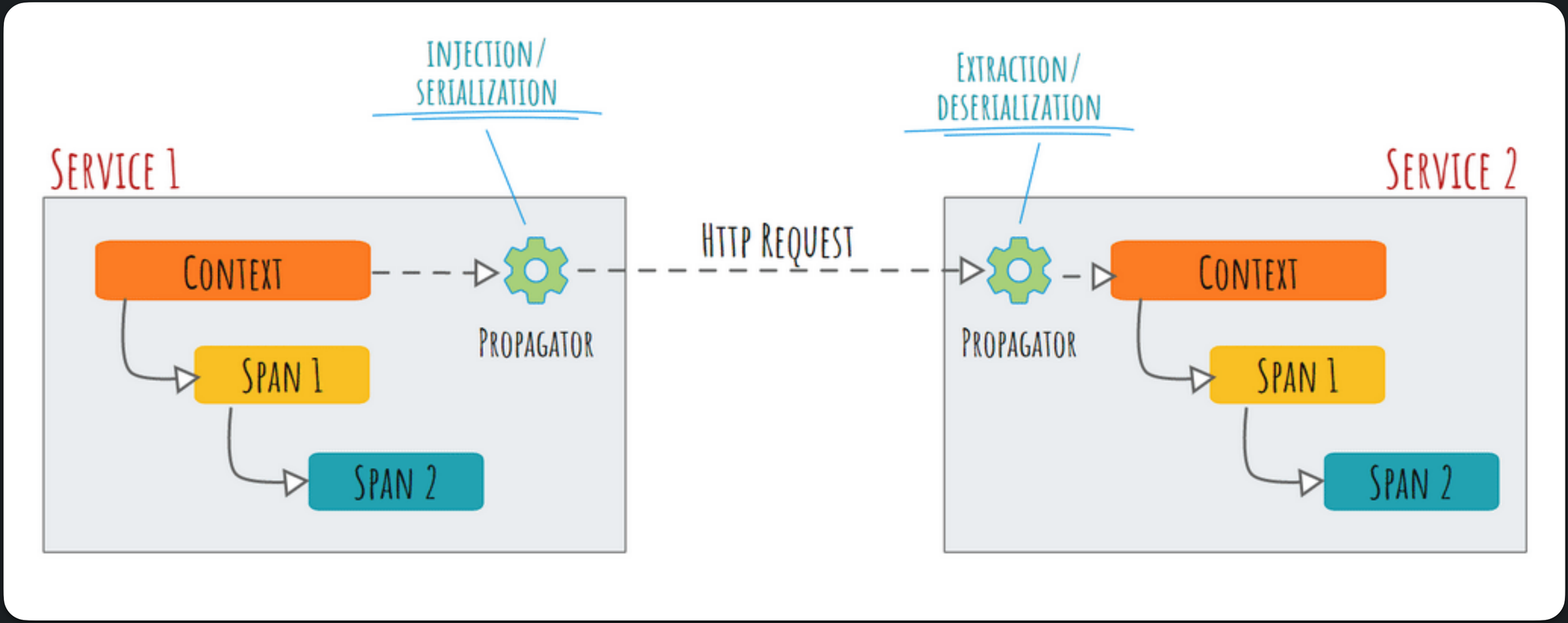


https://opentelemetry.io/docs/specs/otel/context/

https://www.w3.org/TR/trace-context/

Context Propagation

Concept that enables Distributed Tracing.

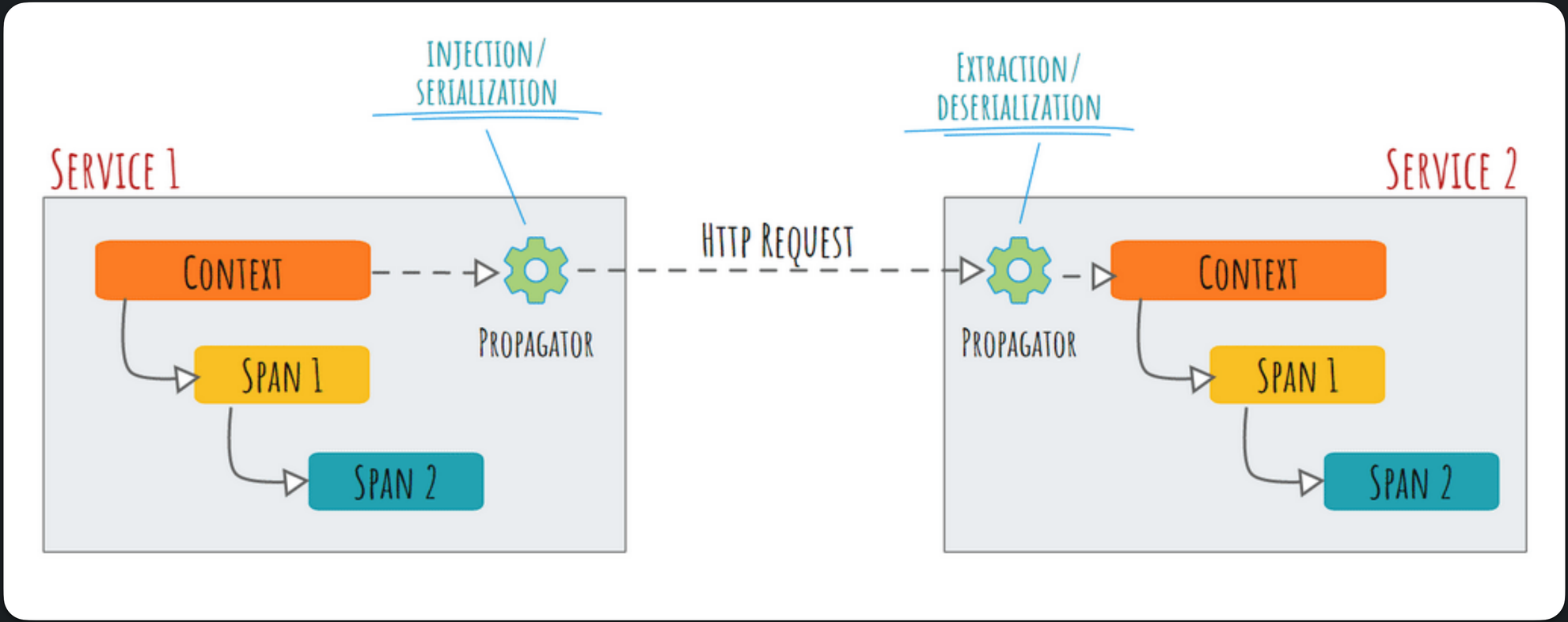


https://opentelemetry.io/docs/specs/otel/context/

https://www.w3.org/TR/trace-context/

Context Propagation

Concept that enables Distributed Tracing.



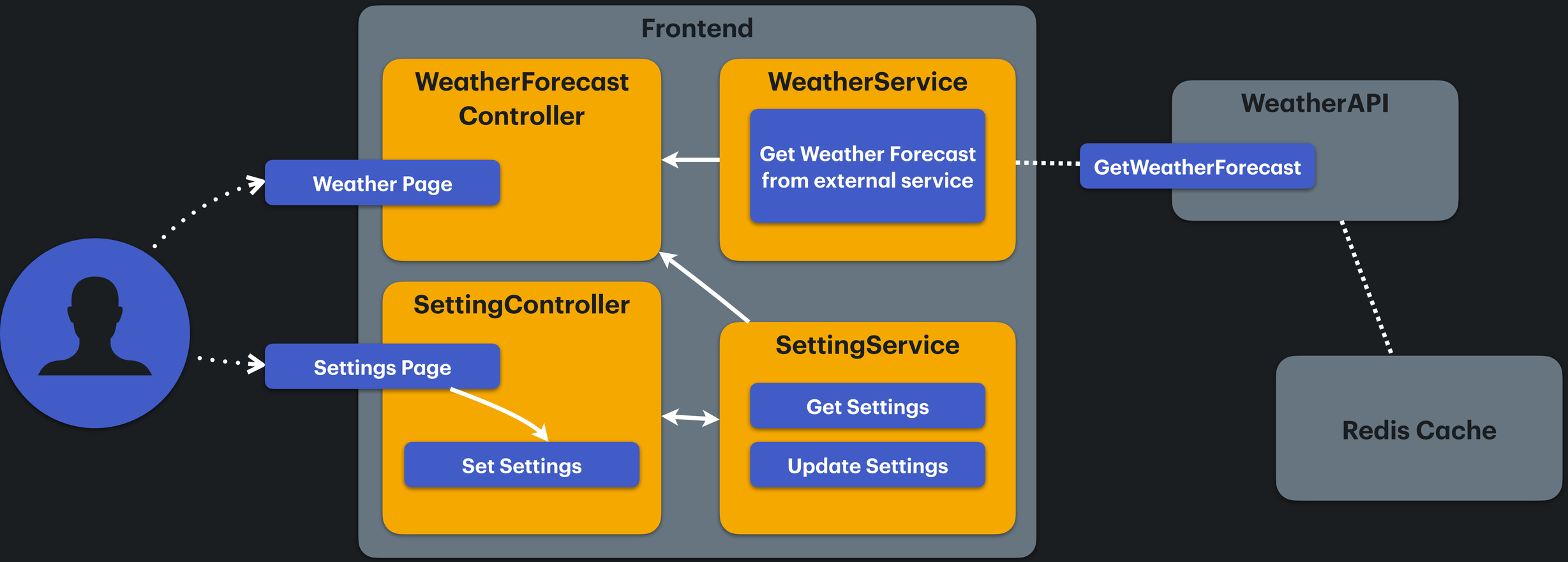
https://opentelemetry.io/docs/specs/otel/context/

https://www.w3.org/TR/trace-context/

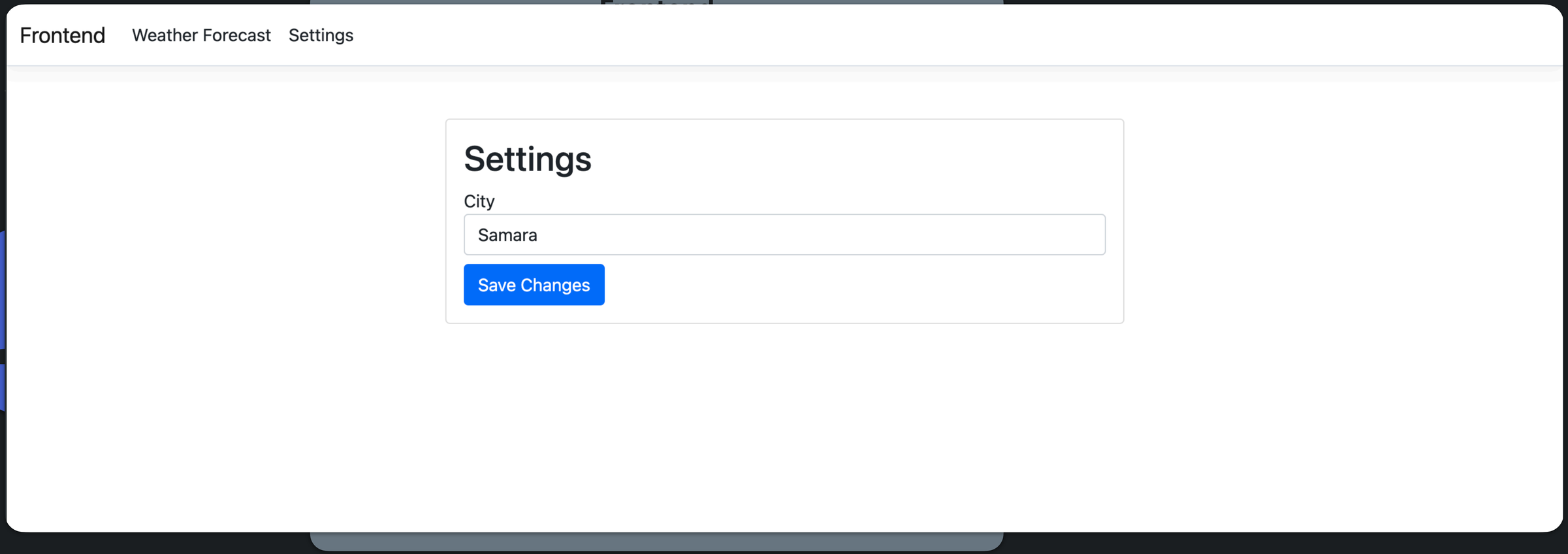
Traces

.NET	OpenTelemetry	Пояснение
Activity	Span	Операция/работа/пролёт
ActivitySource	Tracer	Создатель Activity/Span
Tag	Attribute	Метаданные спана
ActivityKind	SpanKind	Взаимоотношения между зависимыми спанами
ActivityContext	SpanContext	Контекст выполнения
ActivityLink	Link	Ссылка на другой спан

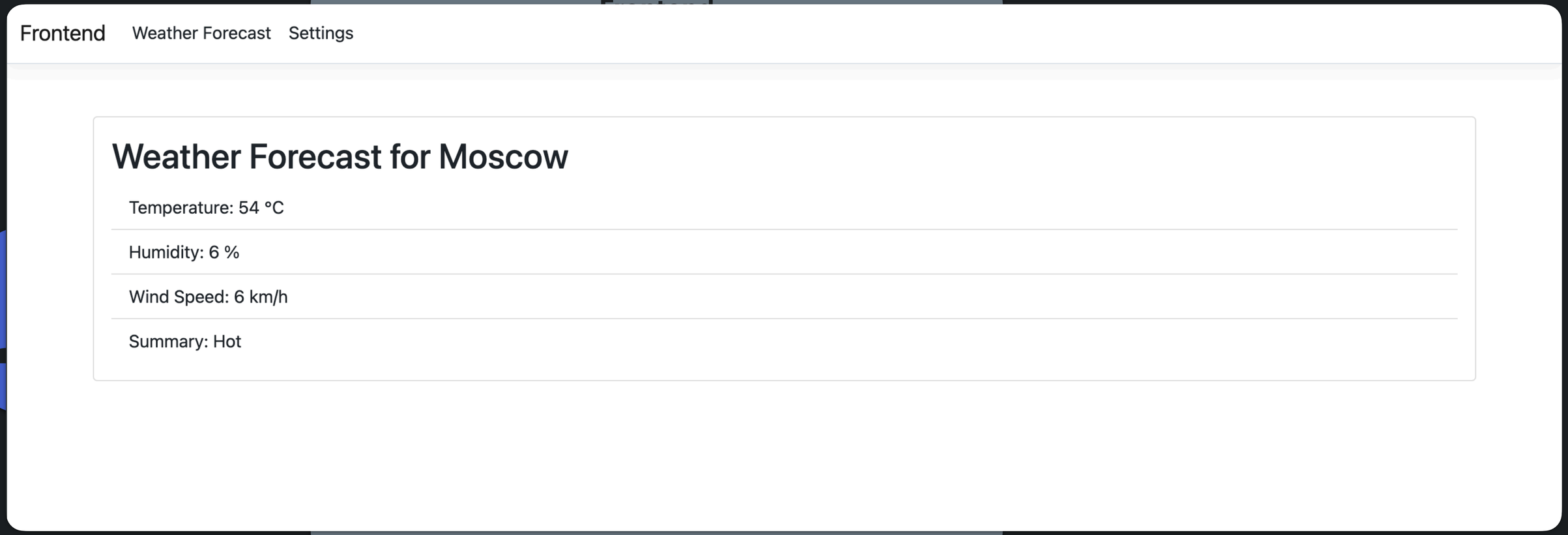
Traces


Frontend
WeatherForecast Controller
WeatherService
Get Weather Forecast from external service
WeatherAPI
GetWeatherForecast
Weather Page
SettingController
SettingService
Settings Page
Get Settings
Set Settings
Update Settings
Redis Cache

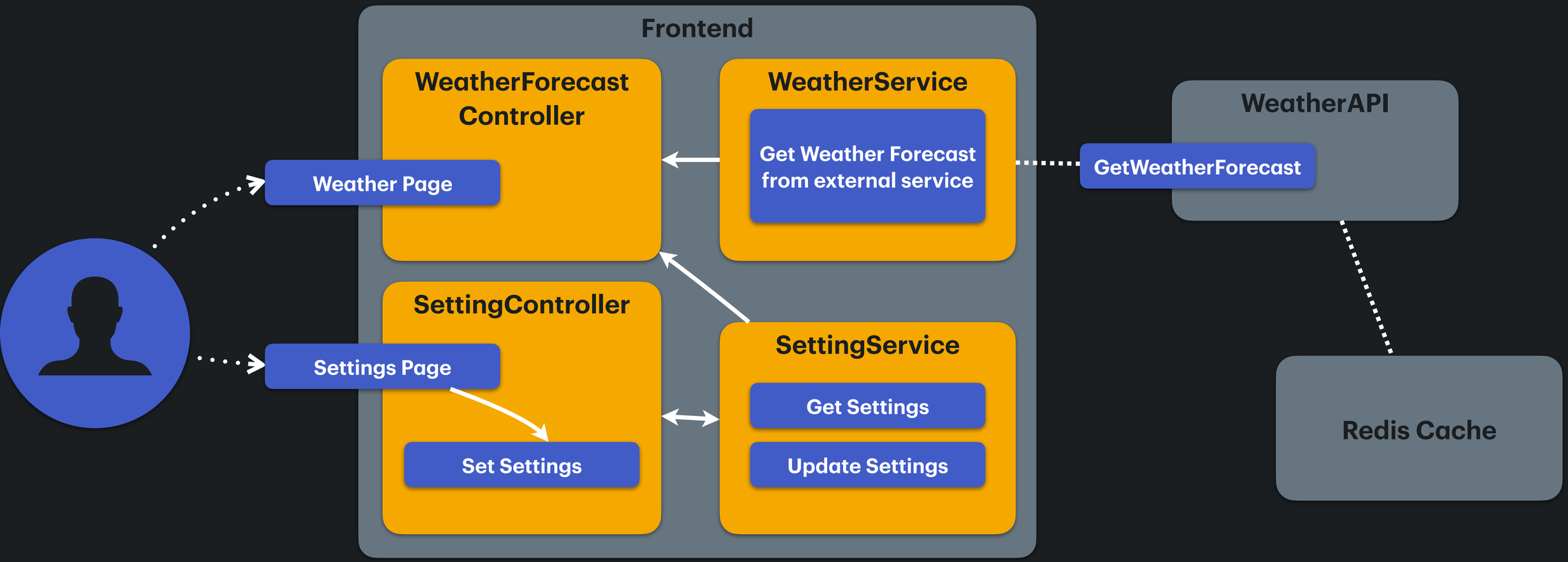
Traces


Frontend
Weather Forecast
Settings
Settings
City
Samara
Save Changes

Traces


Frontend
Weather Forecast
Settings
Weather Forecast for Moscow
Temperature: 54 °C
Humidity: 6 %
Wind Speed: 6 km/h
Summary: Hot

Traces


Frontend
WeatherForecast Controller
WeatherService
Get Weather Forecast from external service
WeatherAPI
GetWeatherForecast
Weather Page
SettingController
SettingService
Settings Page
Get Settings
Set Settings
Update Settings
Redis Cache

Traces

Frontend/SettingController.cs

```
[HttpGet]
public IActionResult Index() => View(new Settings { City = _settingService.GetCity() });

[HttpPost]
public IActionResult SetCity(Settings settings)
{
    if (ModelState.IsValid)
    {
        _logger.LogInformation("Setting default city to {City}", settings.City);

        _settingService.SetCity(settings.City);
        return RedirectToAction("Index");
    }
    return View("Index", settings);
}
```

Traces

Frontend/SettingController.cs

```
private static readonly ActivitySource _activitySource = new(nameof(SettingController), "1.0.0");

[HttpGet]
public IActionResult Index() ⇒ View(new Settings { City = _settingService.GetCity() });

[HttpPost]
public IActionResult SetCity(Settings settings)
{
    using var activity = _activitySource.StartActivity(nameof(SetCity));
    if (ModelState.IsValid)
    {
        _logger.LogInformation("Setting default city to {City}", settings.City);

        _settingService.SetCity(settings.City);
        return RedirectToAction("Index");
    }

    return View("Index", settings);
}
```

Traces

Frontend/SettingController.cs

```
private static readonly ActivitySource _activitySource = new(nameof(SettingController), "1.0.0");

[HttpGet]
public IActionResult Index() ⇒ View(new Settings { City = _settingService.GetCity() });

[HttpPost]
public IActionResult SetCity(Settings settings)
{
    using var activity = _activitySource.StartActivity();
    if (ModelState.IsValid)
    {
        _logger.LogInformation("Setting default city to {City}", settings.City);

        _settingService.SetCity(settings.City);
        return RedirectToAction("Index");
    }

    return View("Index", settings);
}
```

Traces

Frontend/SettingController.cs

```cs
private static readonly ActivitySource _activitySource = new(nameof(SettingController), "1.0.0");

[HttpGet]
public IActionResult Index() => View(new Settings { City = _settingService.GetCity() });

[HttpPost]
public IActionResult SetCity(Settings settings)
{
    using var activity = _activitySource.StartActivity();
    if (ModelState.IsValid)
    {
        _logger.LogInformation("Setting default city to {City}", settings.City);

        _settingService.SetCity(settings.City);
        return RedirectToAction("Index");
    }

    activity?.SetStatus(ActivityStatusCode.Error, "Invalid model state");
    return View("Index", settings);
}
```

Traces

Frontend/WeatherForecastController.cs

```
[HttpGet]
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    _logger.LogInformation(
        "Weather forecast equal to {@Forecast}", forecast);

    return View(forecast);
}
```

Traces

Frontend/WeatherForecastController.cs

```
private static readonly ActivitySource _activitySource = new(nameof(WeatherForecastController), "1.0.0");

[HttpGet]
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    using var activity = _activitySource.StartActivity();
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        activity?.SetStatus(ActivityStatusCode.Error, errorMessage);
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    _logger.LogInformation(
        "Weather forecast equal to {@Forecast}", forecast);

    return View(forecast);
}
```

Traces

Frontend/WeatherForecastController.cs

```cs
private static readonly ActivitySource _activitySource = new(nameof(WeatherForecastController), "1.0.0");

[HttpGet]
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    using var activity = _activitySource.StartActivity();
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        activity?.SetStatus(ActivityStatusCode.Error, errorMessage);
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    activity?.AddEvent(new ActivityEvent($"Weather forecast for {city} is ready"));

    _logger.LogInformation(
        "Weather forecast equal to {@Forecast}", forecast);

    return View(forecast);
}
```

Traces

Frontend/SettingService.cs

```cs
private static readonly ActivitySource _activitySource = new(nameof(SettingService), "1.0.0");
private readonly Settings _settings = new Settings();

public string GetCity()
{
    using var activity = _activitySource.StartActivity();
    activity?.SetTag("city", _settings.City);
    return _settings.City;
}

public void SetCity(string city)
{
    using var activity = _activitySource.StartActivity();
    activity?.SetTag("city", city);
    _settings.City = city;
}
```

Traces Naming Recommendations

Traces Naming Recommendations

1. Имя источника, передаваемое в конструктор, должно быть уникальным

Traces Naming Recommendations

1. Имя источника, передаваемое в конструктор, должно быть уникальным

2. Если в одной сборке есть несколько источников, используйте иерархическое имя (Microsoft.AspNetCore.Hosting)

Traces Naming Recommendations

1. Имя источника, передаваемое в конструктор, должно быть уникальным

2. Если в одной сборке есть несколько источников, используйте иерархическое имя (Microsoft.AspNetCore.Hosting)

3. Имя должно быть основано на сборке, определяющей ActivitySource, а не на сборке, код которой подвергается инструментарию

Signals

Logs

Traces

Metrics

Metrics

- Отслеживание состояния системы на протяжении времени
- Оценка актуальности той или иной функциональности
- Сравнение с прошлыми версиями системы

System.Diagnostics.Metrics

System.Diagnostics.Metrics

```
+—MeterProvider
```

System.Diagnostics.Metrics

```
+—MeterProvider
  |
  +—Meter(name, version)
```

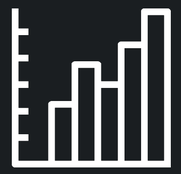
System.Diagnostics.Metrics

```
+—MeterProvider
  |
  +—Meter(name, version)
    |
    +—Instrument<TInstrument, TValue>(name, unit, description)
```

System.Diagnostics.Metrics

```
+—MeterProvider
  |
  +—Meter(name, version)
    |
    +—Instrument<TInstrument, TValue>(name, unit, description)
      |
      +—Measurement<TValue>(value, tags)
```

Instruments

Synchronous	Asynchronous
Counter 	ObservableCounter 
UpDownCounter 	ObservableUpDownCounter 
Histogram 	ObservableGauge 

Instruments

WeatherAPI/MyMetrics.cs

```
public class MyMetrics
{
    public static readonly string GlobalSystemName = Environment.MachineName;
    public static readonly string ApplicationName = AppDomain.CurrentDomain.FriendlyName;
    public static readonly string InstrumentsSourceName = nameof(MyMetrics);

    private int _temperature;

    public Counter<int> SummaryRequestsCounter { get; }
```

Instruments

WeatherAPI/MyMetrics.cs

```
public MyMetrics(IMeterFactory meterFactory)
{
    var meter = meterFactory
        .Create(InstrumentsSourceName, "1.0.0");

    SummaryRequestsCounter = meter
        .CreateCounter<int>(name: "weather.get.requests",
            unit: "Requests",
            description: "The number of requests to get weather for the city");

    meter.CreateObservableGauge<int>(name: "weather.forecast.temperature",
        observeValue: () => new Measurement<int>(_temperature),
        unit: "Celsius",
        description: "The temperature today");
}
```

Instruments

WeatherAPI/MyMetrics.cs

```
    public void SetWeather(WeatherForecast forecast)
        => _temperature = forecast.Temperature;
}
```

Instruments

Frontend/WeatherForecastController.cs

```
public class WeatherForecastController(
    WeatherService _weatherService,
    SettingService _settingService,
    ILogger<WeatherForecastController> _logger,
    MyMetrics myMetrics)
    : Controller
{
    private static readonly ActivitySource _activitySource = new(nameof(WeatherForecastController), "1.0.0");
```

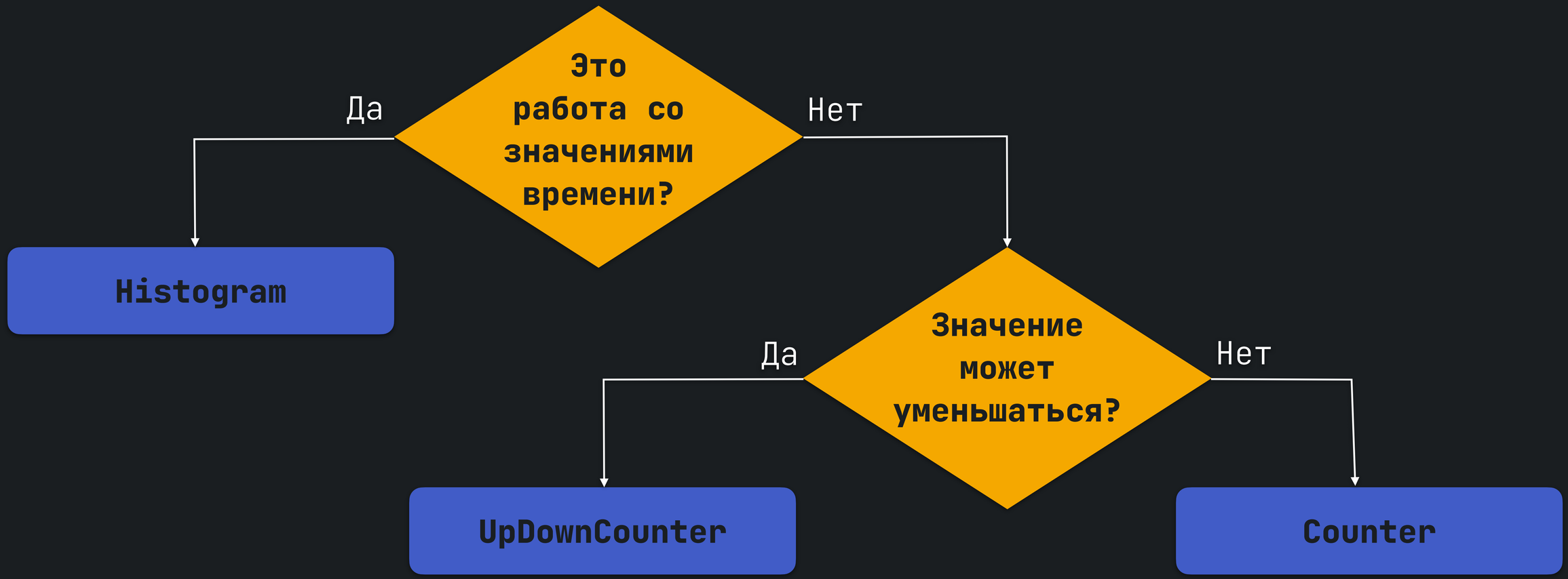
Instruments

Frontend/WeatherForecastController.cs

```
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    using var activity = _activitySource.StartActivity();
    var city = _settingService.GetCity();

    var (isSuccess, forecast, errorMessage) = await _weatherService.GetForecastAsync(city);
    if (!isSuccess)
    {
        activity?.SetStatus(ActivityStatusCode.Error, errorMessage);
        ViewBag.ErrorMessage = errorMessage!;
        return View("Error");
    }

    activity?.AddEvent(new ActivityEvent($"Weather forecast for {city} is ready"));

    myMetrics.SummaryRequestsCounter.Add(1, new KeyValuePair<string, object?>("city", city));
    myMetrics.SetWeather(forecast!);

    _logger.LogInformation(
        "Weather forecast equal to {@Forecast}", forecast);

    return View(forecast);
}
```

Instruments

Рекомендации



dotnet-counters

```
dotnet tool update -g dotnet-counters
```

```
dotnet-counters monitor -n Frontend --counters MyMetrics
```

```
Name                                                    Current Value
[MyMetrics]
    weather.forecast.temperature                                   20
    weather.summary.requests (Requests)
        city=Moscow                                                 3
        city=Saint Petersburg                                       6
        city=Samara                                                 5
```

dotnet-counters

```
dotnet tool update -g dotnet-counters
```

```
dotnet-counters monitor -n MainAPI
```

```
Name                                                    Current Value
[System.Runtime]
    % Time in GC since last GC (%)                              0
    Allocation Rate (B / 1 sec)                                 0
    CPU Usage (%)                                               0.007
    Exception Count (Count / 1 sec)                             0
    GC Committed Bytes (MB)                                     0
    GC Fragmentation (%)                                        0
    GC Heap Size (MB)                                           6.085
```

Metrics Naming Recommendations

https://ucum.org/ucum

Metrics Naming Recommendations

1. Разделение уровней иерархии - ‘.’

https://ucum.org/ucum

Metrics Naming Recommendations

1. Разделение уровней иерархии - ‘.’
2. Разделение слов - ‘_’

https://ucum.org/ucum

Metrics Naming Recommendations

1. Разделение уровней иерархии - ‘.’
2. Разделение слов - ‘_’

```
"contoso.ticket_queue.duration"
"contoso.reserved_tickets"
"contoso.purchased_tickets"
```

https://ucum.org/ucum

Metrics Naming Recommendations

1. Разделение уровней иерархии - ‘.’
2. Разделение слов - ‘_’
3. Именование единиц измерений - UCUM

https://ucum.org/ucum

Metrics ~~Naming~~ Recommendations

1. Разделение уровней иерархии - ‘.’
2. Разделение слов - ‘_’
3. Именование единиц измерений - UCUM

*. Для отдельного логического блока - отдельный класс метрик

https://ucum.org/ucum

> Телеметрия — это процесс сбора данных с удаленных или труднодоступных систем или объектов, передачи этих данных на расстояние, их мониторинга, визуализации и хранения для дальнейшего анализа и использования.

© Jason Statham

Телеметрия — это процесс

сбора

передачи

визуализации

мониторинга

хранения

анализа

использования

Телеметрия — это процесс

сбора

передачи

визуализации

мониторинга

хранения

Телеметрия — это процесс

Сбор данных

Сбор данных

OpenTelemetry

- Объединяет всё единым стандартом
- Предоставляет инструменты для экспорта данных телеметрии
- Имеет открытый исходный код и большое сообщество

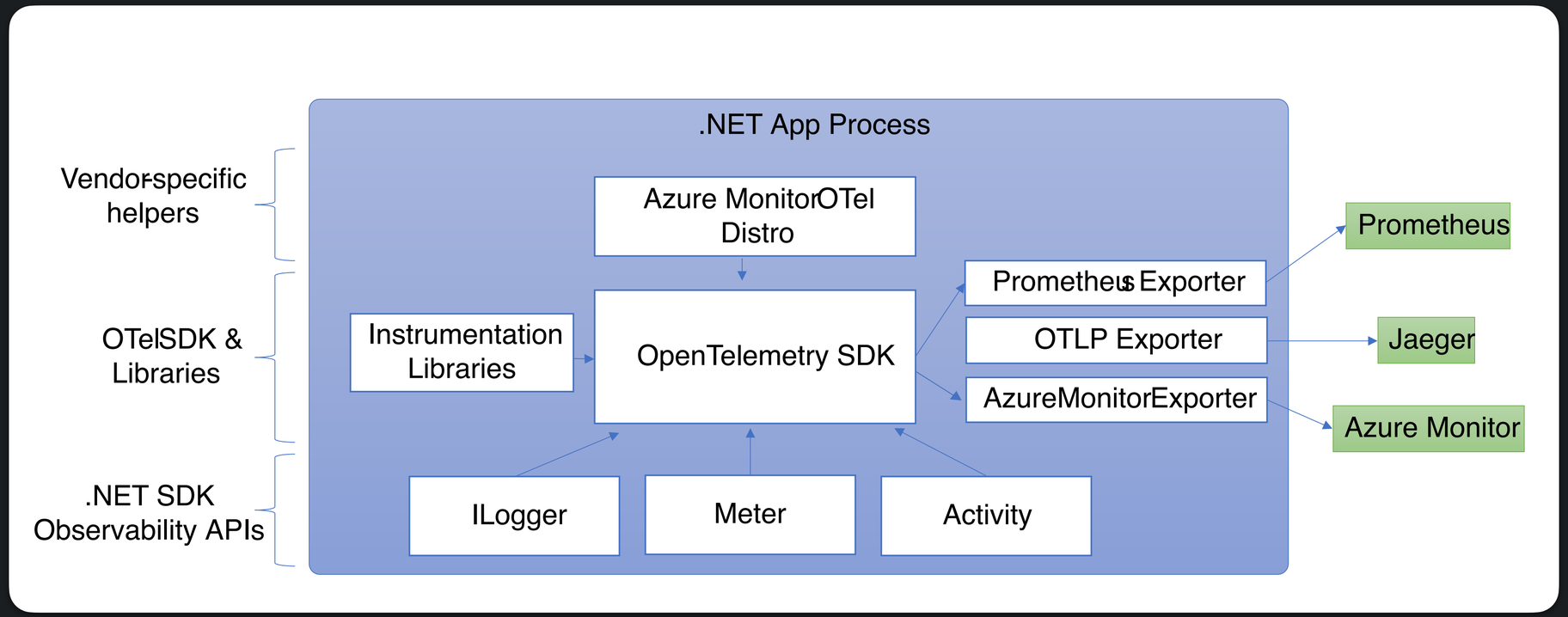
OpenTelemetry

Components

- APIs for libraries
- APIs for app developers
- Semantic conventions
- Interface for exporters
- OTLP

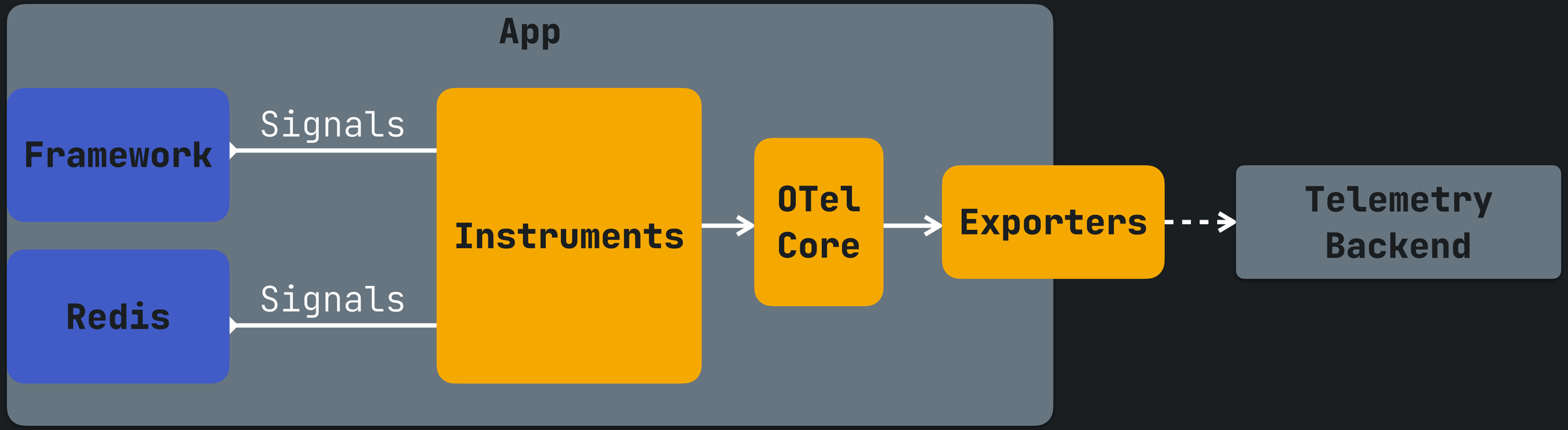
OpenTelemetry

In .NET



OpenTelemetry

Packages



OpenTelemetry packages

Package Name	Description
OpenTelemetry	Main library that provides the core OTEL functionality
OpenTelemetry.Instrumentation.AspNetCore	Instrumentation for ASP.NET Core and Kestrel
OpenTelemetry.Instrumentation.Http	Instrumentation for HttpClient and HttpWebRequest to track outbound HTTP calls
OpenTelemetry.Instrumentation.SqlClient	Instrumentation for SqlClient used to trace database operations
OpenTelemetry.Exporter.Console	Exporter for the console, commonly used to diagnose what telemetry is being exported
OpenTelemetry.Exporter.OpenTelemetryProtocol	Exporter using the OTLP protocol
OpenTelemetry.Exporter.Prometheus.AspNetCore	Exporter for Prometheus implemented using an ASP.NET Core endpoint

OpenTelemetry distributions

1. opentelemetry-dotnet ★ 3k

   https://github.com/open-telemetry/opentelemetry-dotnet

2. opentelemetry-dotnet-contrib ★ 411

   https://github.com/open-telemetry/opentelemetry-dotnet-contrib

Сбор данных

Exporters

Logs

Metrics

Traces

Сбор данных

Exporters

Экспорт данных

Data Collector

Internet

Экспорт данных

Telemetry

Data Collector

Internet

Экспорт данных

OTLP

Data Collector

Internet

OTLP

OpenTelemetry Protocol

https://opentelemetry.io/docs/specs/otlp/

OTLP

OpenTelemetry Protocol

- Объединяет всё единым стандартом

https://opentelemetry.io/docs/specs/otlp/

OTLP

OpenTelemetry Protocol

- Объединяет всё единым стандартом
- Поддерживается почти всеми сервисами для телеметрии

https://opentelemetry.io/docs/specs/otlp/

OTLP

OpenTelemetry Protocol

- Объединяет всё единым стандартом
- Поддерживается почти всеми сервисами для телеметрии
- Надстройка над gRPC или HTTP 1.1

https://opentelemetry.io/docs/specs/otlp/

OTLP

OpenTelemetry Protocol

- Объединяет всё единым стандартом
- Поддерживается почти всеми сервисами для телеметрии
- Надстройка над gRPC или HTTP 1.1
- Запрос-ответ

https://opentelemetry.io/docs/specs/otlp/

OTLP

Design Goals

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Suitable for use of all node types
  (applications, telemetry backends, agents, collectors)

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Suitable for use of all node types
  (applications, telemetry backends, agents, collectors)
- High reliability of data delivery

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Suitable for use of all node types
  (applications, telemetry backends, agents, collectors)
- High reliability of data delivery
- Visibility when the data cannot be delivered

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Suitable for use of all node types
  (applications, telemetry backends, agents, collectors)
- High reliability of data delivery
- Visibility when the data cannot be delivered
- Low CPU usage for serialization and deserialization

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Ability to efficiently modify deserialized data and serialize again

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Ability to efficiently modify deserialized data and serialize again
- High throughput

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Ability to efficiently modify deserialized data and serialize again
- High throughput
- Backpressure signalling

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

OTLP

Design Goals

- Ability to efficiently modify deserialized data and serialize again
- High throughput
- Backpressure signalling
- Load-balancer friendly

https://github.com/open-telemetry/opentelemetry-proto/blob/main/docs/design-goals.md

Storage

Data Collectors / Telemetry backends

	Logs	Traces	Metrics
Grafana Tempo	✗	✓	✗
Grafana Loki	✓	✗	✗
Prometheus	✗	✗	✓
Jaeger	✗	✓	✗
ELK (Elasticsearch)	✓	✓	✓
Aspire Dashboard	✓	✓	✓
Seq	✓	✓	✗

Storage

Data Collectors / Telemetry backends

	Logs	Traces	Metrics	UI
Grafana Tempo	✗	✓	✗	✗
Grafana Loki	✓	✗	✗	✗
Prometheus	✗	✗	✓	✓
Jaeger	✗	✓	✗	✓
ELK (Elasticsearch)	✓	✓	✓	✓
Aspire Dashboard	✓	✓	✓	✓
Seq	✓	✓	✗	✓

Storage

Data Collectors / Telemetry backends

	Logs	Traces	Metrics	UI
Grafana Tempo	✗	✓	✗	✗
Grafana Loki	✓	✗	✗	✗
Prometheus	✗	✗	✓	✓
Jaeger	✗	✓	✗	✓
ELK (Elasticsearch)	✓	✓	✓	✓
Aspire Dashboard	✓	✓	✓	✓
Seq	✓	✓	✗	✓

Export

Prometheus

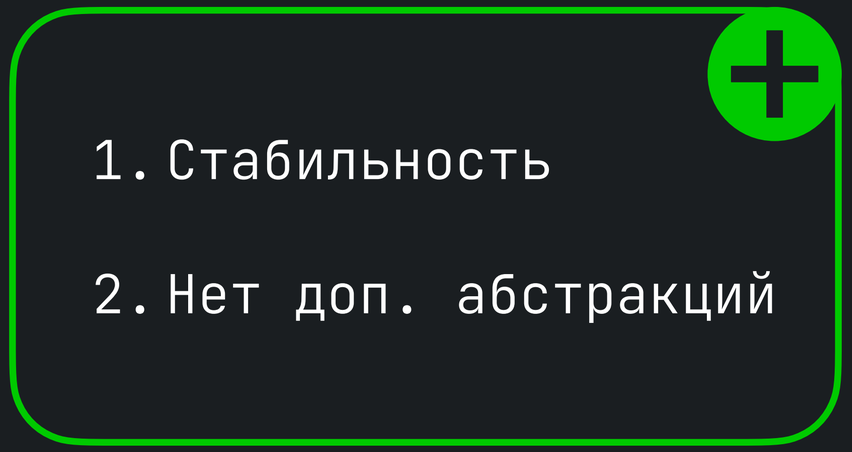
`prometheus-net`

`OpenTelemetry.Exporter.Prometheus`

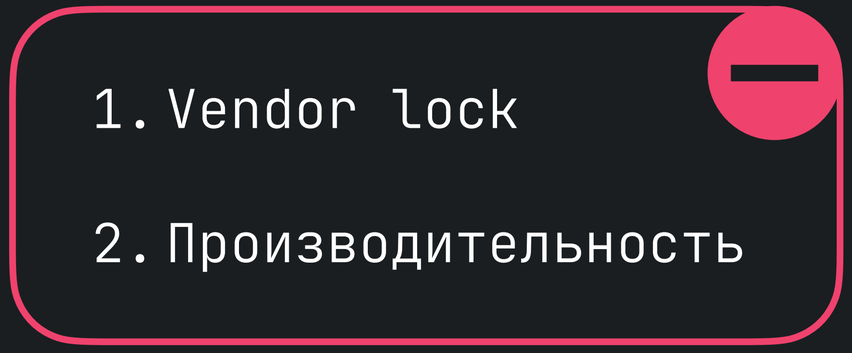
Export

Prometheus

prometheus-net

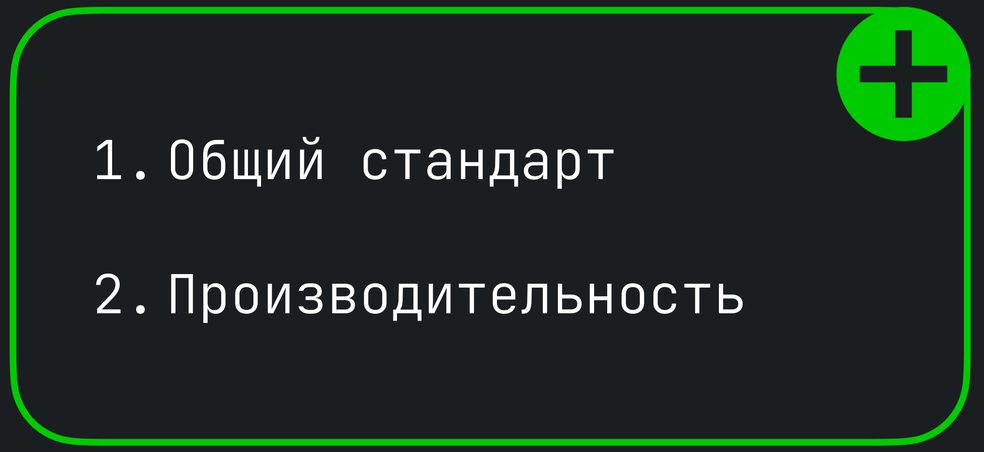


1. Стабильность
2. Нет доп. абстракций

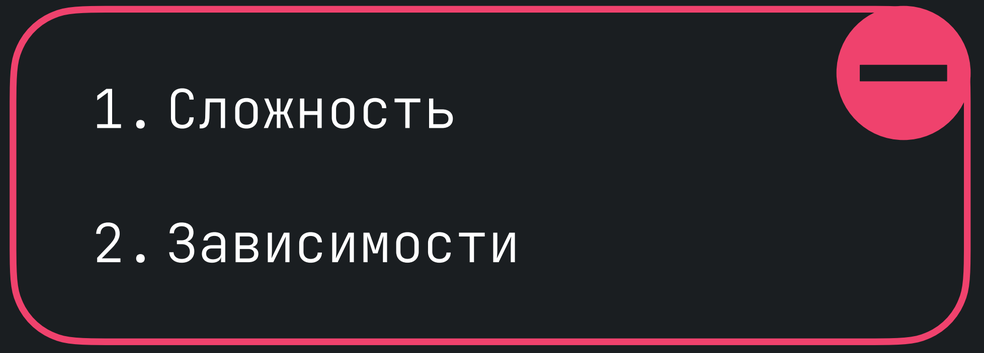


1. Vendor lock
2. Производительность

OpenTelemetry.Exporter.Prometheus



1. Общий стандарт
2. Производительность



1. Сложность
2. Зависимости

Export

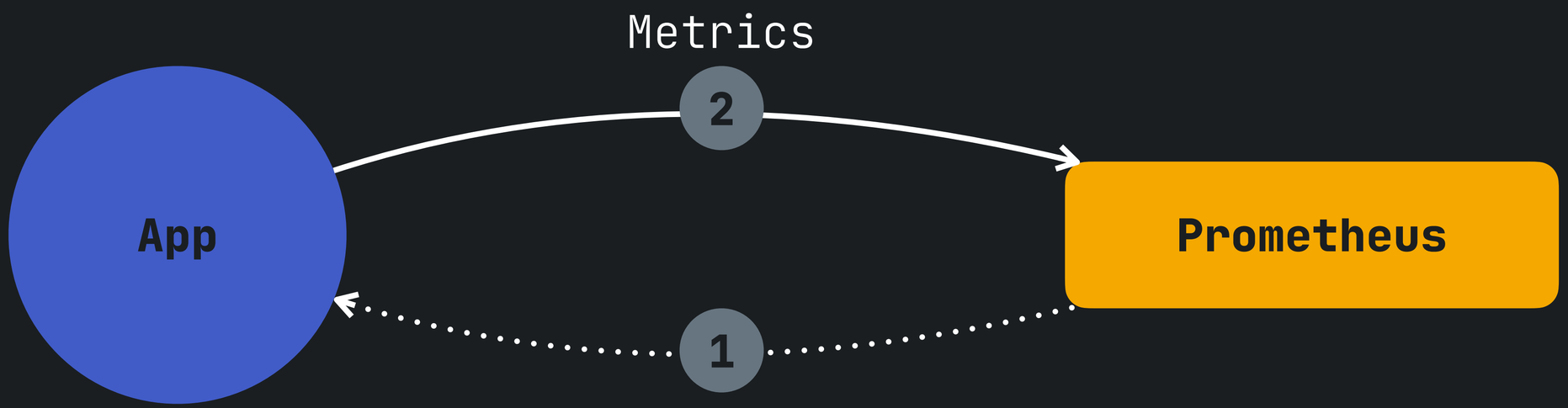
Prometheus

`prometheus-net`

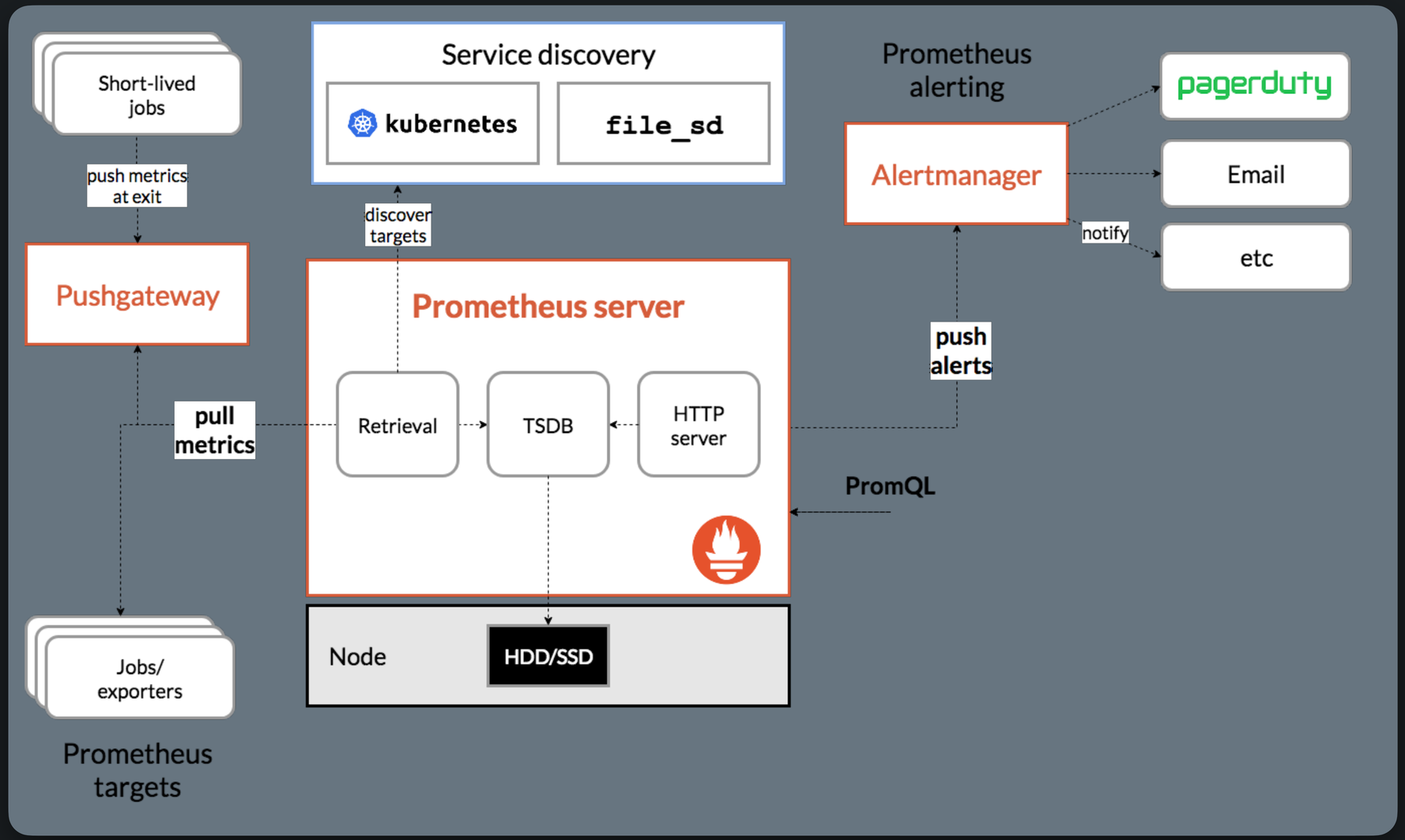
`OpenTelemetry.Exporter.Prometheus`

Prometheus

Pull Model



Prometheus



- by SoundCloud
- Monitoring
- Alerting
- Store in TSDB
- Querying by PromQL

Prometheus

Configuration

prometheus.yml

```yaml
global:
    scrape_interval: 10s
scrape_configs:
    -   job_name: 'prometheus'
        static_configs:
            -   targets: [ 'prometheus:9090' ]

    -   job_name: ‘frontend'
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'host.docker.internal:9184' ]

    -   job_name: 'weather_api'
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'host.docker.internal:9185' ]
```

Export

Prometheus

Frontend/Program.cs

```
services.AddOpenTelemetry() // OpenTelemetry && OpenTelemetry.Extensions.Hosting
    .ConfigureResource(resourceBuilder ⇒ resourceBuilder
        .AddService(MyMetrics.ApplicationName, serviceInstanceId: Environment.MachineName)
        .AddAttributes(new Dictionary<string, object>
        {
            ["EnvironmentName"] =
                MyMetrics.GlobalSystemName // That attribute is visible in the target_info separate metric.
        }))
    .WithMetrics(meterProviderBuilder ⇒ meterProviderBuilder
        .AddMeter(MyMetrics.InstrumentsSourceName)
        .AddAspNetCoreInstrumentation() // OpenTelemetry.Instrumentation.AspNetCore
        .AddHttpClientInstrumentation() // OpenTelemetry.Instrumentation.Http
        .AddRuntimeInstrumentation() // OpenTelemetry.Instrumentation.Runtime
        .AddProcessInstrumentation() // OpenTelemetry.Instrumentation.Process
        .AddPrometheusExporter(opt ⇒
        {
            opt.DisableTotalNameSuffixForCounters = false; // Ability to disable total name suffix for counters
        }));; //OpenTelemetry.Exporter.Prometheus.AspNetCore
```

Prometheus

Exporters

Prometheus

Exporters

Databases

- Aerospike exporter
- AWS RDS exporter
- ClickHouse exporter
- Consul exporter (official)
- Couchbase exporter
- CouchDB exporter
- Druid Exporter
- Elasticsearch exporter
- EventStore exporter
- IoTDB exporter
- KDB+ exporter
- Memcached exporter (official)
- MongoDB exporter
- MongoDB query exporter
- MongoDB Node.js Driver exporter
- MSSQL server exporter
- MySQL router exporter
- MySQL server exporter (official)
- OpenTSDB Exporter
- Oracle DB Exporter
- PgBouncer exporter
- PostgreSQL exporter
- Presto exporter
- ProxySQL exporter
- RavenDB exporter
- Redis exporter
- RethinkDB exporter
- SQL exporter
- Tarantool metric library
- Twemproxy

FinOps

- AWS Cost Exporter
- Azure Cost Exporter
- Kubernetes Cost Exporter

Other monitoring systems

- Akamai Cloudmonitor exporter
- Alibaba Cloudmonitor exporter
- AWS CloudWatch exporter (official)
- Azure Monitor exporter
- Cloud Foundry Firehose exporter
- Collectd exporter (official)
- Google Stackdriver exporter
- Graphite exporter (official)
- Heka dashboard exporter
- Heka exporter
- Huawei Cloudeye exporter
- InfluxDB exporter (official)
- ITM exporter
- Java GC exporter
- JavaMelody exporter
- JMX exporter (official)
- Munin exporter
- Nagios / Naemon exporter
- New Relic exporter
- NRPE exporter
- Osquery exporter
- OTC CloudEye exporter
- Pingdom exporter
- Promitor (Azure Monitor)
- scollector exporter
- Sensu exporter
- site24x7_exporter
- SNMP exporter (official)
- StatsD exporter (official)
- TencentCloud monitor exporter
- ThousandEyes exporter
- StatusPage exporter

Storage

- Ceph exporter
- Ceph RADOSGW exporter
- Gluster exporter
- GPFS exporter
- Hadoop HDFS FSImage exporter
- HPE CSI info metrics provider
- HPE storage array exporter
- Lustre exporter
- NetApp E-Series exporter
- Pure Storage exporter
- ScaleIO exporter
- Tivoli Storage Manager/IBM Spectrum Protect exporter

HTTP

- Apache exporter
- HAProxy exporter (official)
- Nginx metric library
- Nginx VTS exporter
- Passenger exporter
- Squid exporter
- Tinyproxy exporter
- Varnish exporter
- WebDriver exporter

APIs

- AWS ECS exporter
- AWS Health exporter
- AWS SQS exporter
- Azure Health exporter
- BigBlueButton
- Cloudflare exporter
- Cryptowat exporter
- DigitalOcean exporter
- Docker Cloud exporter
- Docker Hub exporter
- Fastly exporter
- GitHub exporter
- Gmail exporter
- InstaClustr exporter
- Mozilla Observatory exporter
- OpenWeatherMap exporter
- Pagespeed exporter
- Rancher exporter
- Speedtest exporter
- Tankerkönig API Exporter

Logging

- Fluentd exporter
- Google's mtail log data extractor
- Grok exporter

Issue trackers and continuous integration

- Bamboo exporter
- Bitbucket exporter
- Confluence exporter
- Jenkins exporter
- JIRA exporter

Messaging systems

- Beanstalkd exporter
- EMQ exporter
- Gearman exporter
- IBM MQ exporter
- Kafka exporter
- NATS exporter
- NSQ exporter
- Mirth Connect exporter
- MQTT blackbox exporter
- MQTT2Prometheus
- RabbitMQ exporter
- RabbitMQ Management Plugin exporter
- RocketMQ exporter
- Solace exporter

Hardware related

- apcupsd exporter
- BIG-IP exporter
- Bosch Sensortec BMP/BME exporter
- Collins exporter
- Dell Hardware OMSA exporter
- Disk usage exporter
- Fortigate exporter
- IBM Z HMC exporter
- IoT Edison exporter
- InfiniBand exporter
- IPMI exporter
- knxd exporter
- Modbus exporter
- Netgear Cable Modem Exporter
- Netgear Router exporter
- Network UPS Tools (NUT) exporter
- Node/system metrics exporter (official)
- NVIDIA GPU exporter
- ProSAFE exporter
- Waveplus Radon Sensor Exporter
- Weathergoose Climate Monitor Exporter
- Windows exporter
- Intel® Optane™ Persistent Memory Controller Exporter

Miscellaneous

- ACT Fibernet Exporter
- BIND exporter
- BIND query exporter
- Bitcoind exporter
- Blackbox exporter (official)
- Bungeecord exporter
- BOSH exporter
- cAdvisor
- Cachet exporter
- ccache exporter
- c-lightning exporter
- DHCPD leases exporter
- Dovecot exporter
- Dnsmasq exporter
- eBPF exporter
- Ethereum Client exporter
- File statistics exporter
- JFrog Artifactory Exporter
- Hostapd Exporter
- IPsec exporter
- IRCd exporter
- Linux HA ClusterLabs exporter
- JMeter plugin
- JSON exporter
- Kannel exporter
- Kemp LoadBalancer exporter
- Kibana Exporter
- kube-state-metrics
- Locust Exporter
- Meteor JS web framework exporter
- Minecraft exporter module
- Minecraft exporter
- Nomad exporter
- nftables exporter
- OpenStack exporter
- OpenStack blackbox exporter
- oVirt exporter
- Pact Broker exporter
- PHP-FPM exporter
- PowerDNS exporter
- Podman exporter
- Prefect2 exporter
- Process exporter
- rTorrent exporter
- Rundeck exporter
- SABnzbd exporter
- SAML exporter
- Script exporter
- Shield exporter
- Smokeping prober
- SMTP/Maildir MDA blackbox prober
- SoftEther exporter
- SSH exporter
- Teamspeak3 exporter
- Transmission exporter
- Unbound exporter
- WireGuard exporter
- Xen exporter

Prometheus

Redis Exporter

prometheus.yml

```yaml
global:
    scrape_interval: 10s
scrape_configs:
    -   job_name: 'prometheus'
        static_configs:
            -   targets: [ 'prometheus:9090' ]

    -   job_name: 'main_api'
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'host.docker.internal:9184' ]

    -   job_name: 'weather_api'
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'host.docker.internal:9185' ]
```

Prometheus

Redis Exporter

prometheus.yml

```
    scrape_interval: 1s
    static_configs:
        -   targets: [ 'host.docker.internal:9184' ]

-   job_name: 'weather_api'
    scrape_interval: 1s
    static_configs:
        -   targets: [ 'host.docker.internal:9185' ]

-   job_name: 'redis_exporter'
    scrape_interval: 30s
    scrape_timeout: 30s
    tls_config:
        insecure_skip_verify: true
    static_configs:
        -   targets: [ "redis-exporter:9121" ]
```

Prometheus

Redis Exporter

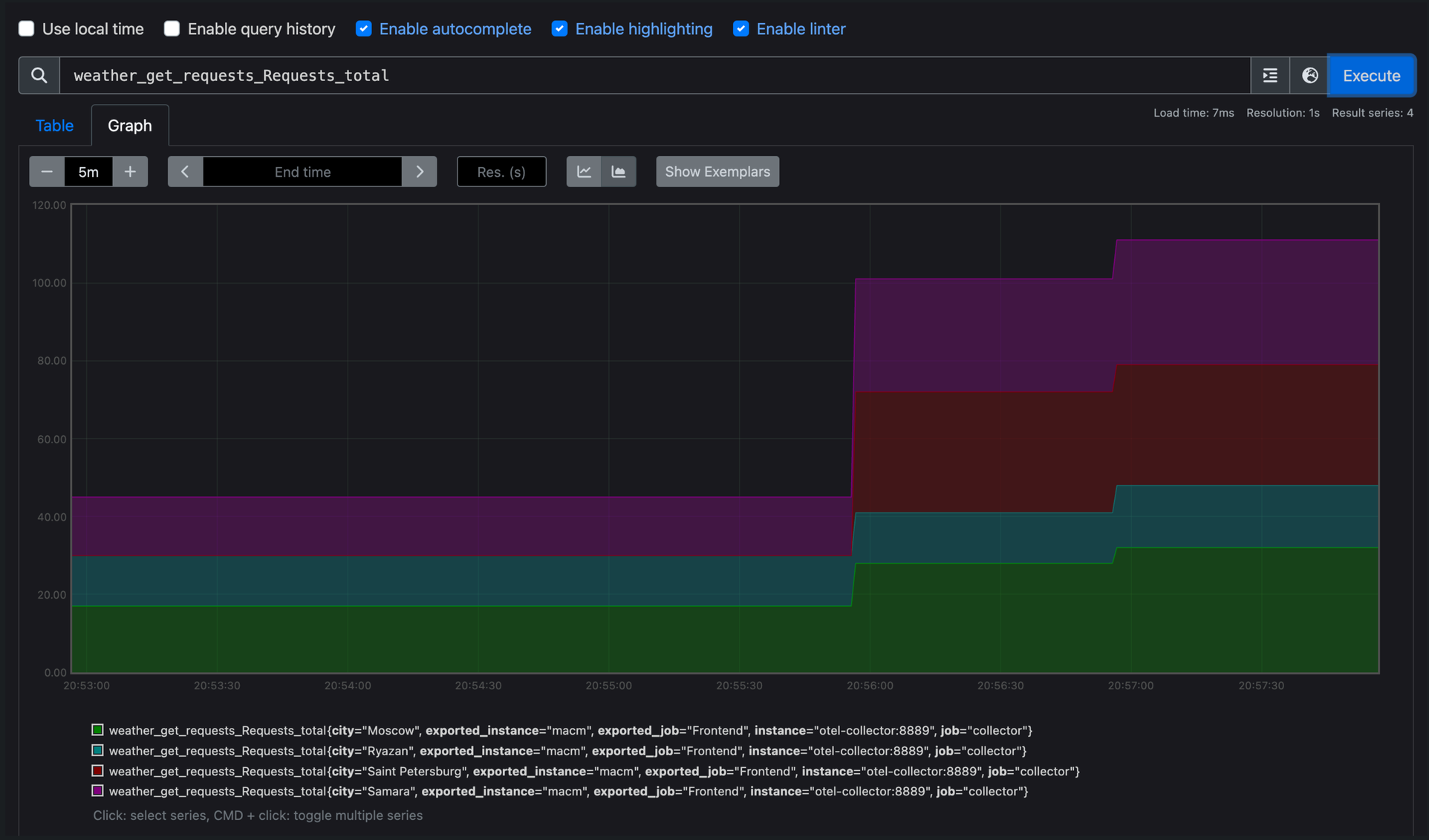
docker-compose.yaml

```yaml
services:
    redis:
        image: redis
        ports:
            - "6379:6379"
    prometheus:
        image: prom/prometheus
        ports:
            - "9090:9090"
        volumes:
            - ./config/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
            - prometheus:/prometheus
    redis-exporter:
        image: bitnami/redis-exporter
        environment:
            - REDIS_ADDR=redis://redis:6379
volumes:
    prometheus: ~
```

```yaml
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'host.docker.

    -   job_name: 'weather_api'
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'host.docker.

    -   job_name: 'redis_exporter'
        scrape_interval: 30s
        scrape_timeout: 30s
        tls_config:
            insecure_skip_verify: true
        static_configs:
            -   targets: [ "redis-export
```

Prometheus

UI Graphs



Prometheus

UI Targets

main_api (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://host.docker.internal:9184/metrics	UP	instance="host.docker.internal:9184" job="main_api"	890.000ms ago	6.613ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://prometheus:9090/metrics	UP	instance="prometheus:9090" job="prometheus"	2.679s ago	8.673ms	

redis_exporter (1/1 up) show less

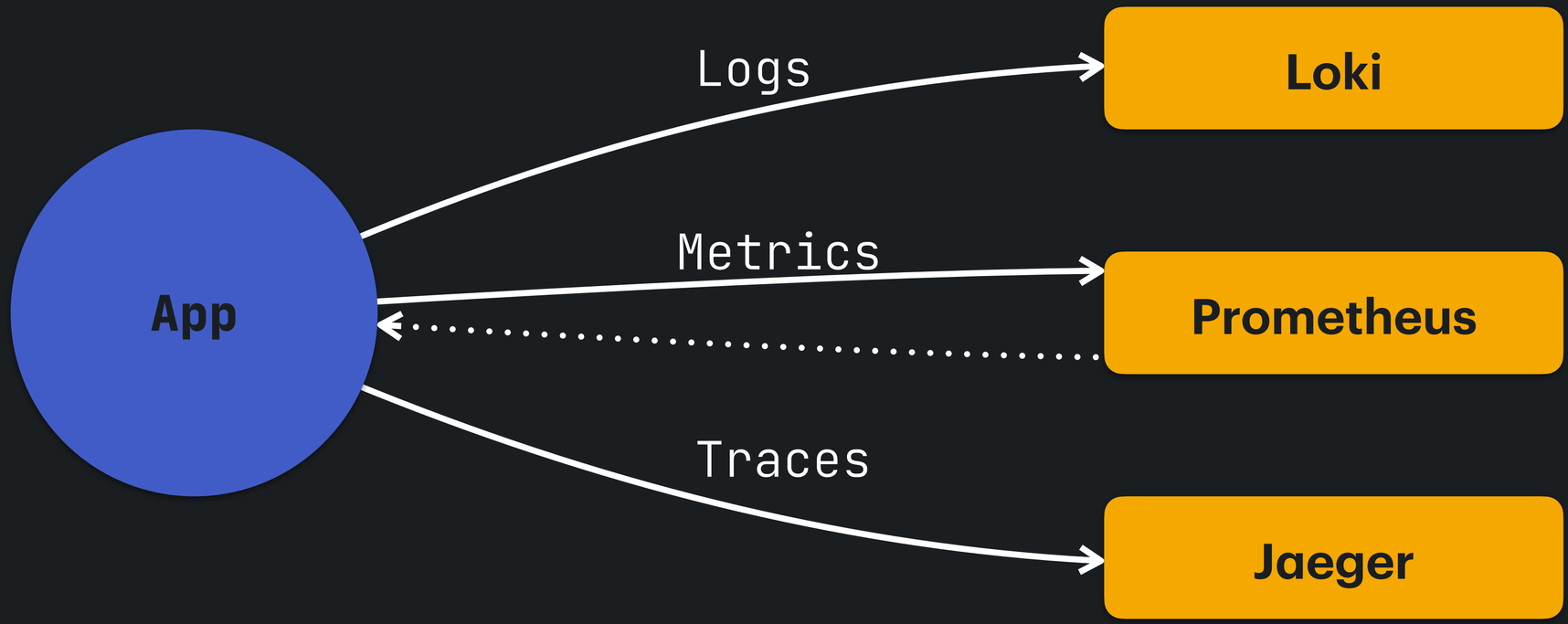
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://redis-exporter:9121/metrics	UP	instance="redis-exporter:9121" job="redis_exporter"	7.943s ago	21.487ms	

weather_api (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://host.docker.internal:9185/metrics	UP	instance="host.docker.internal:9185" job="weather_api"	280.000ms ago	4.572ms	

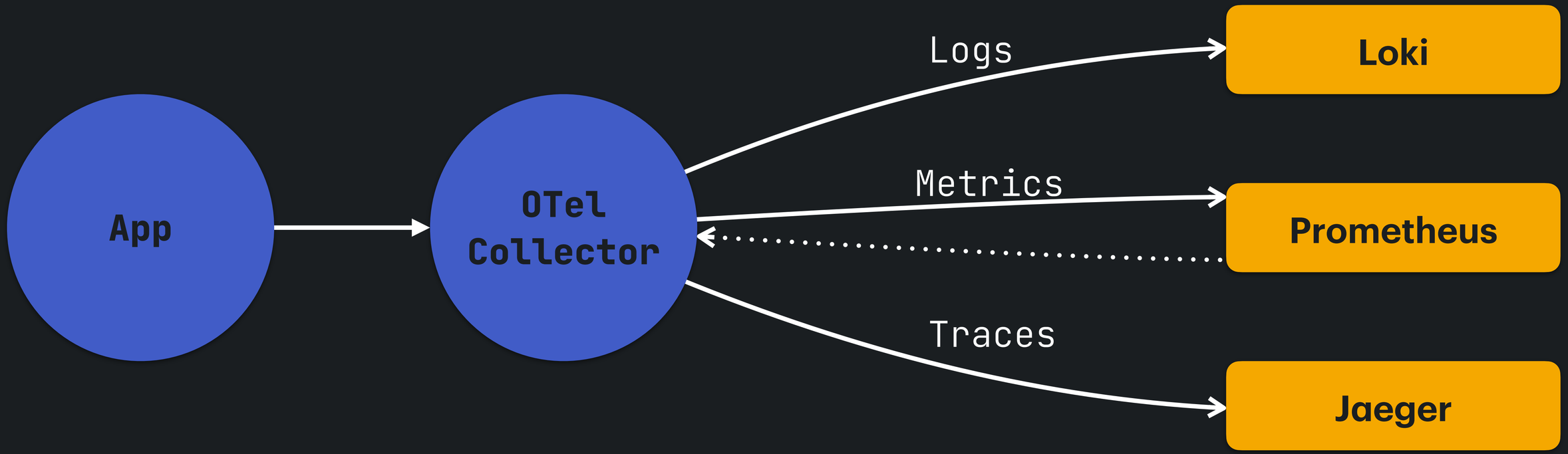
Export

Old but not gold



Export

Old but not gold

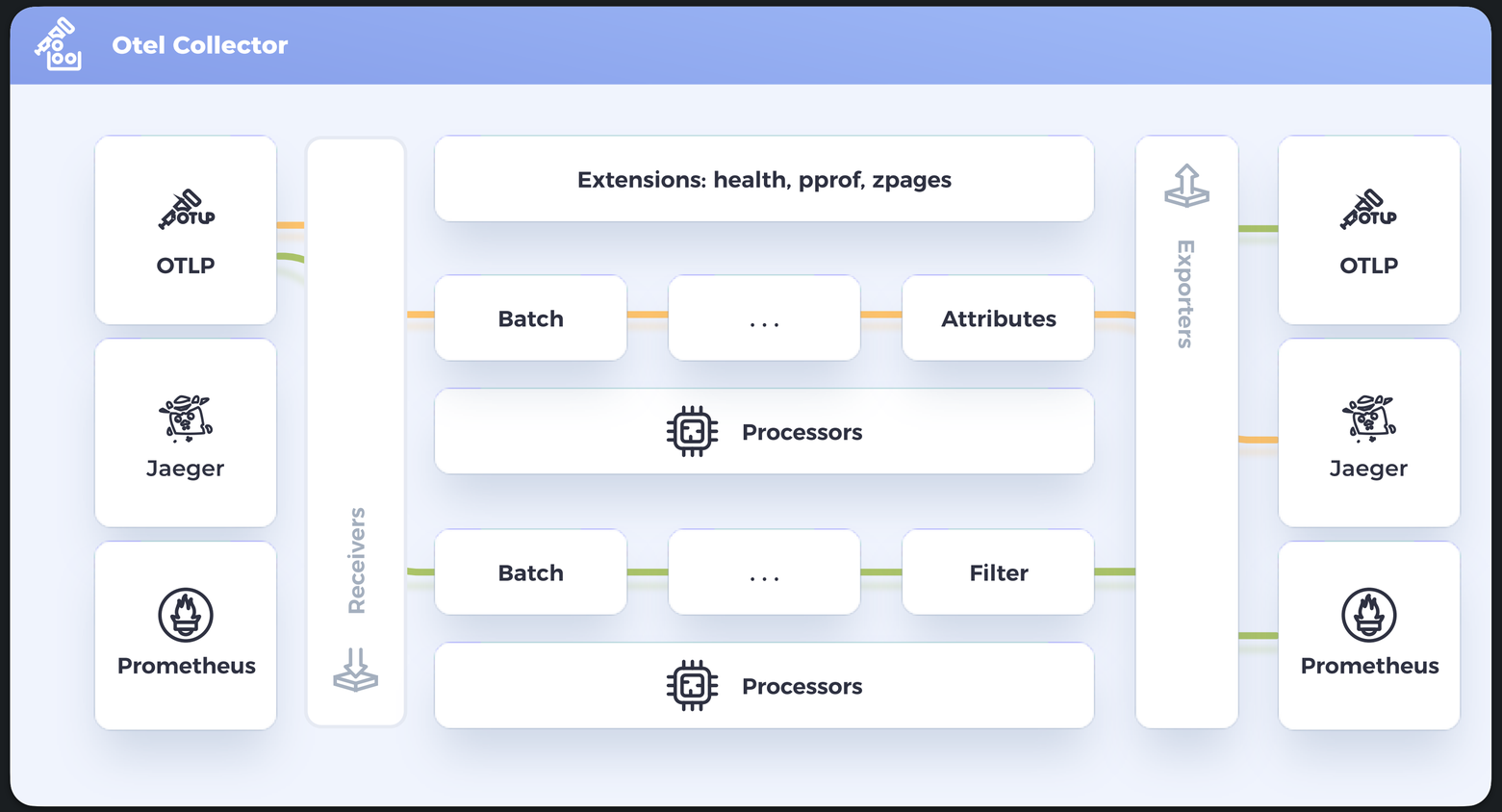


OTel Collector

Objectives

- Usability
- Performance
- Observability
- Extensibility
- Unification

OTel Collector



OTel Collector

1. opentelemetry-collector ★ 4k

   https://github.com/open-telemetry/opentelemetry-collector

2. opentelemetry-collector-contrib ★ 2.7k

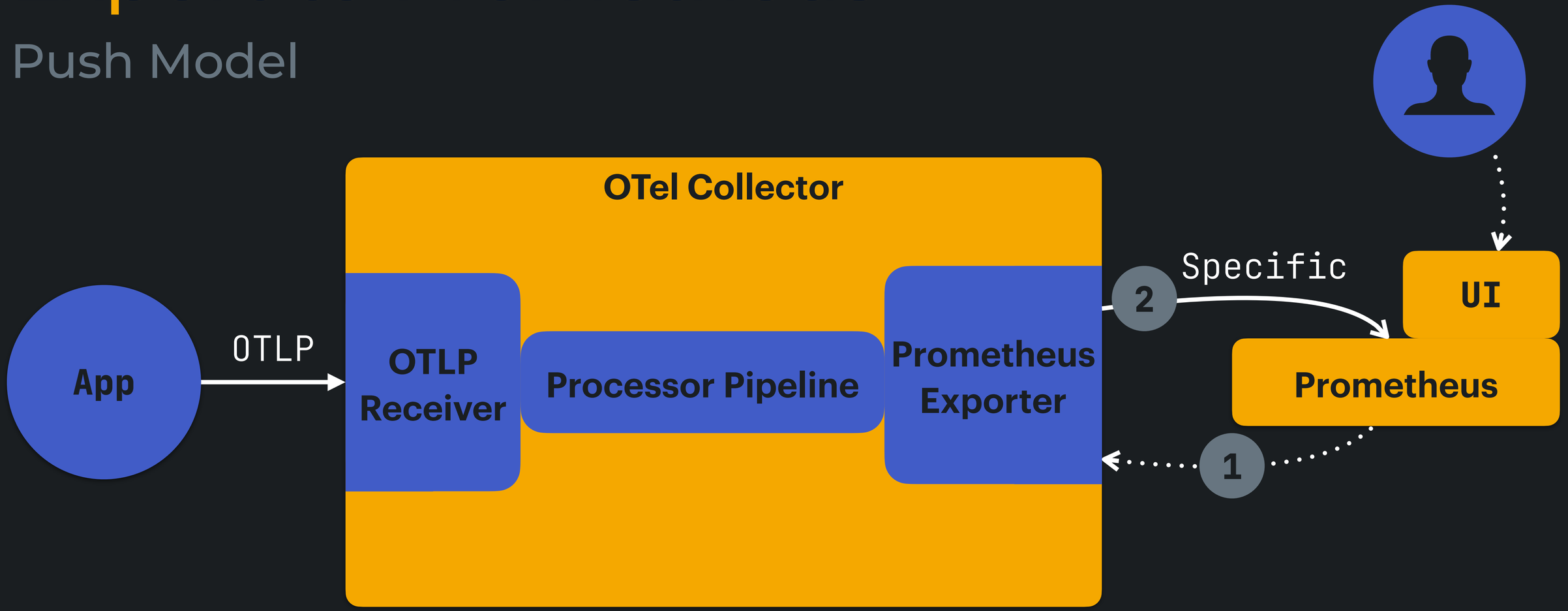
   https://github.com/open-telemetry/opentelemetry-collector-contrib

X. Custom OTel Collector

   https://opentelemetry.io/docs/collector/custom-collector/

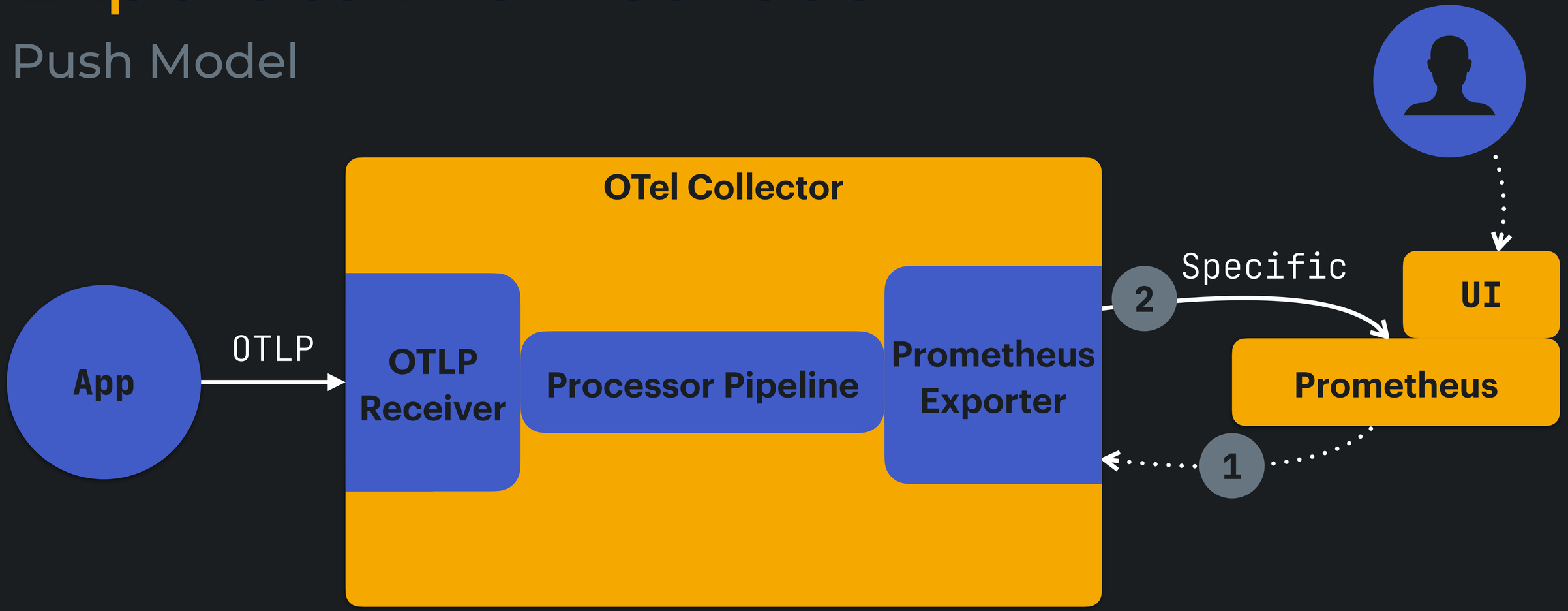
Export to Prometheus

Push Model



Export to Prometheus

Push Model



Export to Prometheus

Application Configuration

Frontend/Program.cs

```
services.AddOpenTelemetry() // OpenTelemetry && OpenTelemetry.Extensions.Hosting
    .ConfigureResource(resourceBuilder ⇒ resourceBuilder
        .AddService(MyMetrics.ApplicationName, serviceInstanceId: Environment.MachineName)
        .AddAttributes(new Dictionary<string, object>
        {
            ["EnvironmentName"] =
                MyMetrics.GlobalSystemName // That attribute is visible in the target_info separate metric.
        }))
    .WithMetrics(meterProviderBuilder ⇒ meterProviderBuilder
        .AddMeter(MyMetrics.InstrumentsSourceName)
        .AddAspNetCoreInstrumentation() // OpenTelemetry.Instrumentation.AspNetCore
        .AddHttpClientInstrumentation() // OpenTelemetry.Instrumentation.Http
        .AddRuntimeInstrumentation() // OpenTelemetry.Instrumentation.Runtime
        .AddProcessInstrumentation() // OpenTelemetry.Instrumentation.Process
        .AddPrometheusExporter(opt ⇒
        {
            opt.DisableTotalNameSuffixForCounters = false; // Ability to disable total name suffix for counters
        }));; //OpenTelemetry.Exporter.Prometheus.AspNetCore
```

Export to Prometheus

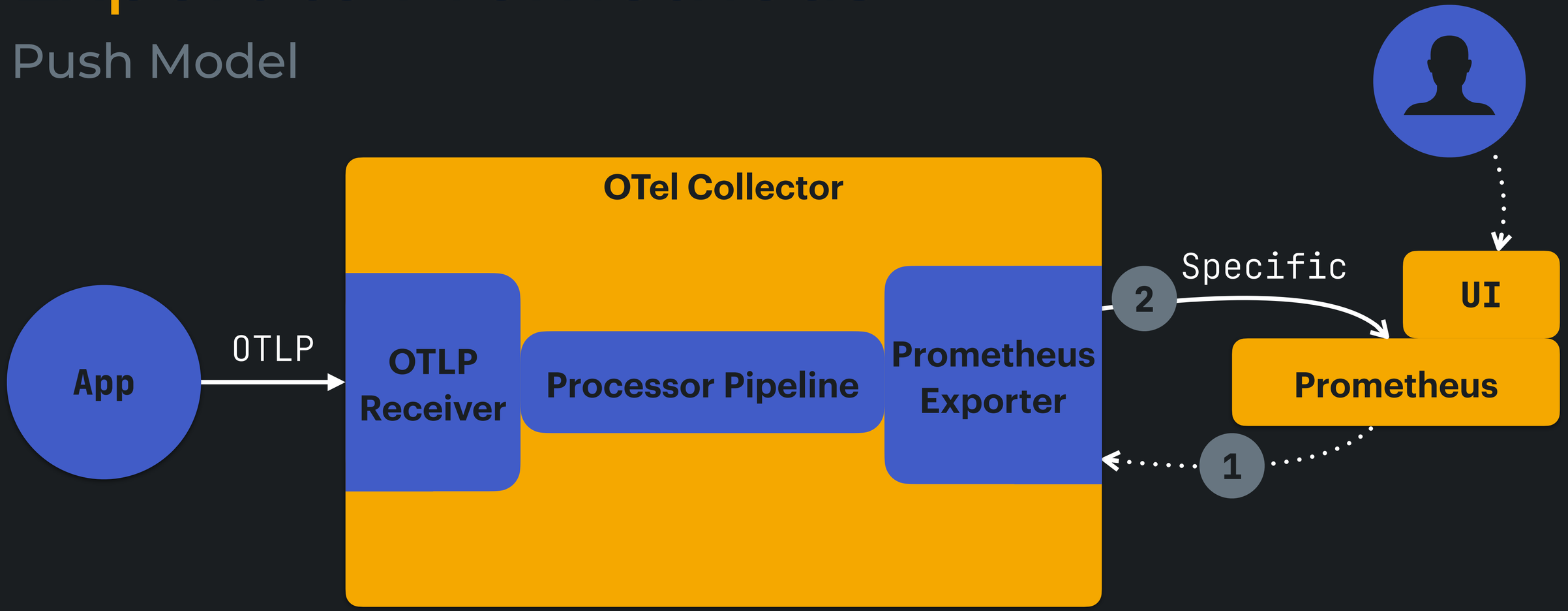
Application Configuration

Frontend/Program.cs

```
services.AddOpenTelemetry() // OpenTelemetry && OpenTelemetry.Extensions.Hosting
    .ConfigureResource(resourceBuilder ⇒ resourceBuilder
        .AddService(MyMetrics.ApplicationName, serviceInstanceId: Environment.MachineName)
        .AddAttributes(new Dictionary<string, object>
        {
            ["EnvironmentName"] =
                MyMetrics.GlobalSystemName // That attribute is visible in the target_info separate metric.
        }))
    .WithMetrics(meterProviderBuilder ⇒ meterProviderBuilder
        .AddMeter(MyMetrics.InstrumentsSourceName)
        .AddAspNetCoreInstrumentation() // OpenTelemetry.Instrumentation.AspNetCore
        .AddHttpClientInstrumentation() // OpenTelemetry.Instrumentation.Http
        .AddRuntimeInstrumentation() // OpenTelemetry.Instrumentation.Runtime
        .AddProcessInstrumentation() // OpenTelemetry.Instrumentation.Process
        .AddOtlpExporter(options ⇒
        {
            options.ExportProcessorType = ExportProcessorType.Batch;
            options.Protocol = OpenTelemetry.Exporter.OtlpExportProtocol.Grpc;
        })); // OpenTelemetry.Exporter.OpenTelemetryProtocol (default port: 4317)
```

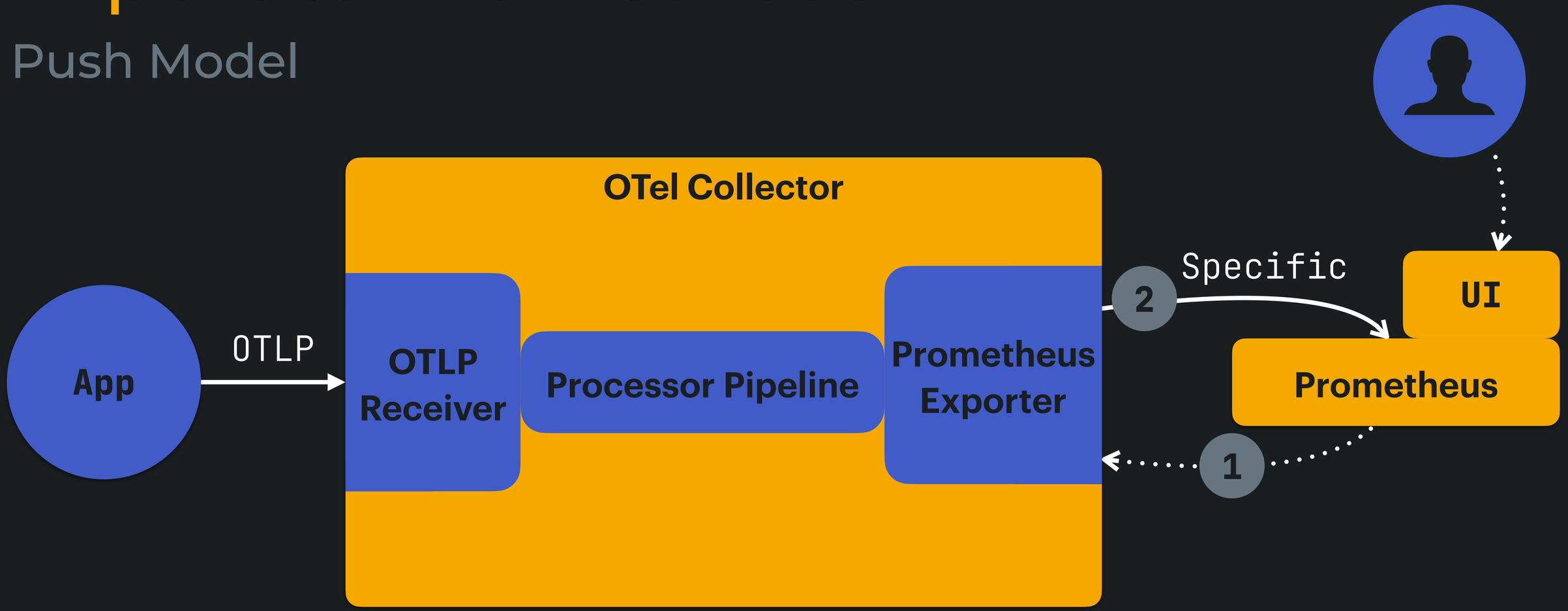
Export to Prometheus

Push Model



Export to Prometheus

Push Model



Export to Prometheus

OTel Collector configuration

otel-collector.yaml

```yaml
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: 0.0.0.0:4317

processors:
  batch:
    timeout: 1s

exporters:
  prometheus:
    endpoint: 0.0.0.0:8889
```

Export to Prometheus

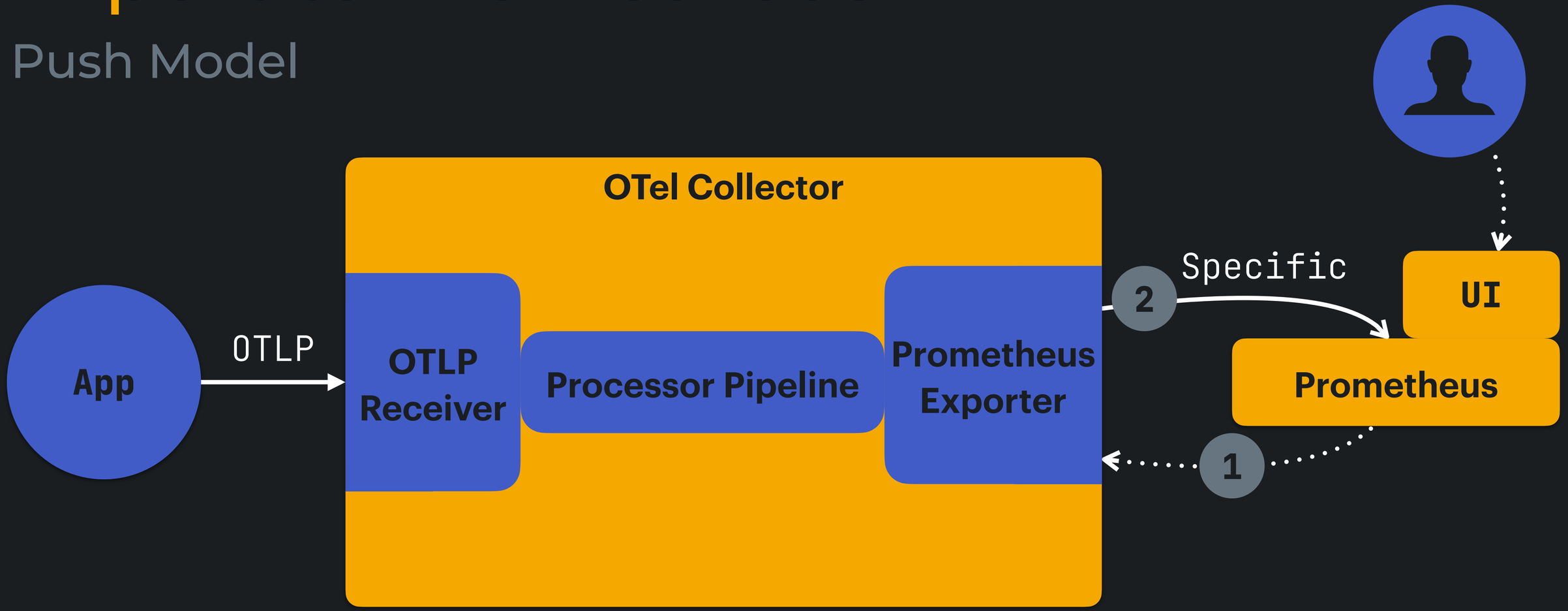
OTel Collector configuration

otel-collector.yaml

```yaml
extensions:
  health_check:
  pprof:
    endpoint: :1777
  zpages:
    endpoint: :55679

service:
  extensions: [ pprof, zpages, health_check ]
  pipelines:
    metrics:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ prometheus ]
```

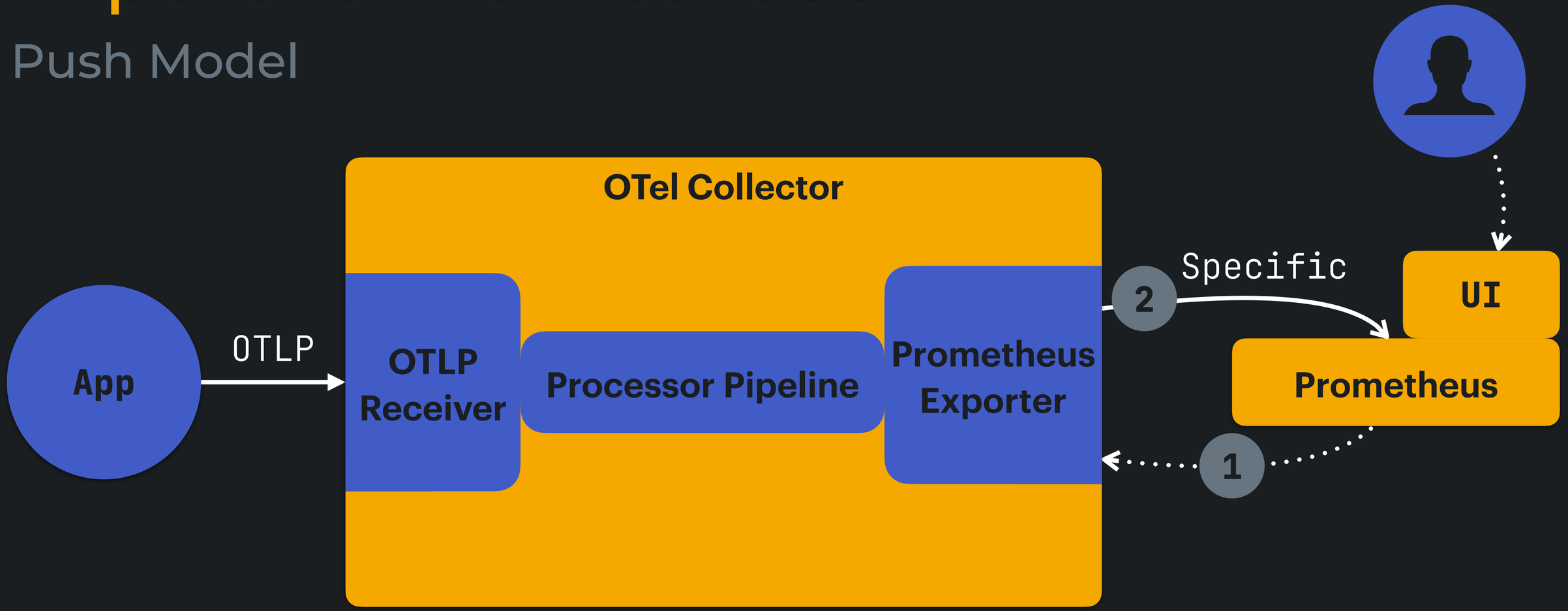
Export to Prometheus

Push Model



Export to Prometheus

Push Model



Export to Prometheus

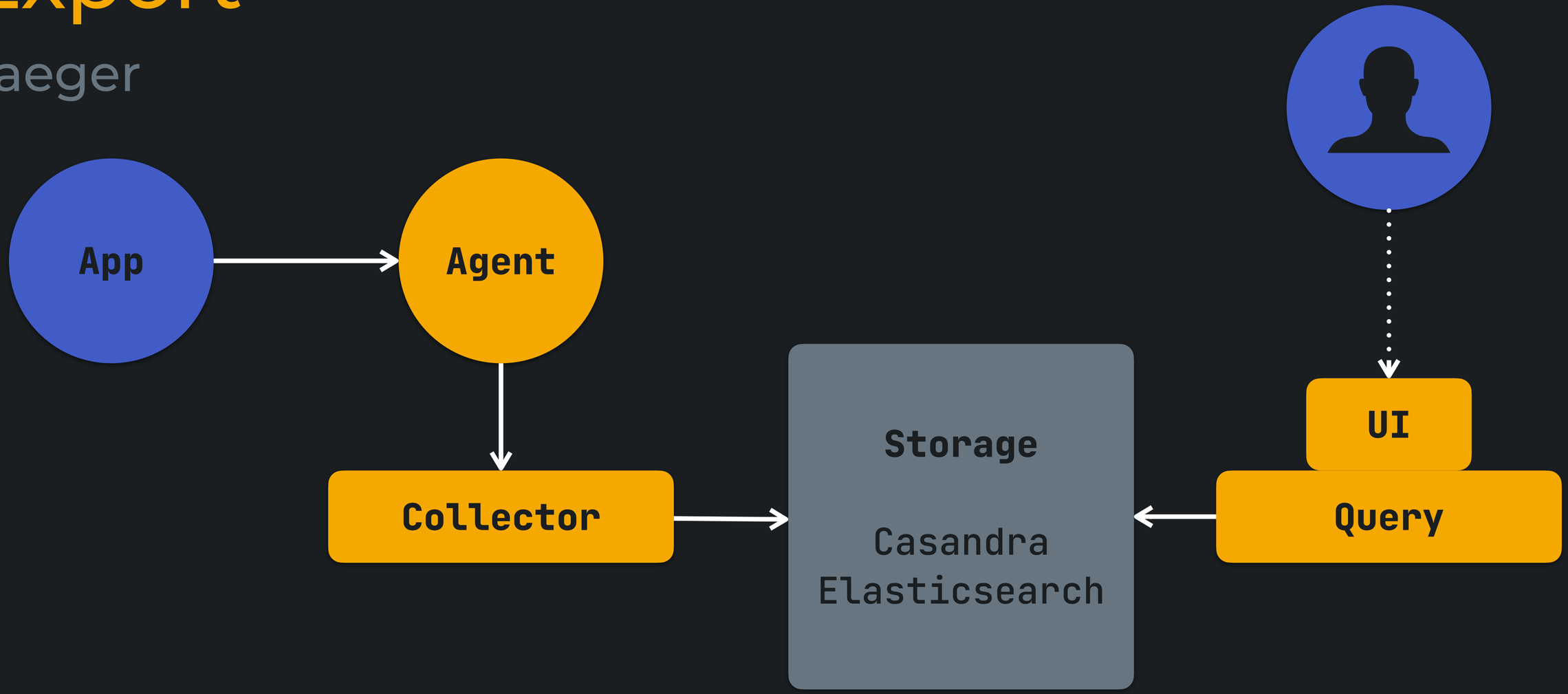
Prometheus configuration

prometheus.yml

```yaml
global:
    scrape_interval: 10s
scrape_configs:
    -   job_name: 'prometheus'
        static_configs:
            -   targets: [ 'prometheus:9090' ]

    -   job_name: 'collector'
        scrape_interval: 1s
        static_configs:
            -   targets: [ 'otel-collector:8889' ]

    -   job_name: 'redis_exporter'
        scrape_interval: 30s
        scrape_timeout: 30s
        tls_config:
            insecure_skip_verify: true
        static_configs:
            -   targets: [ "redis-exporter:9121" ]
```

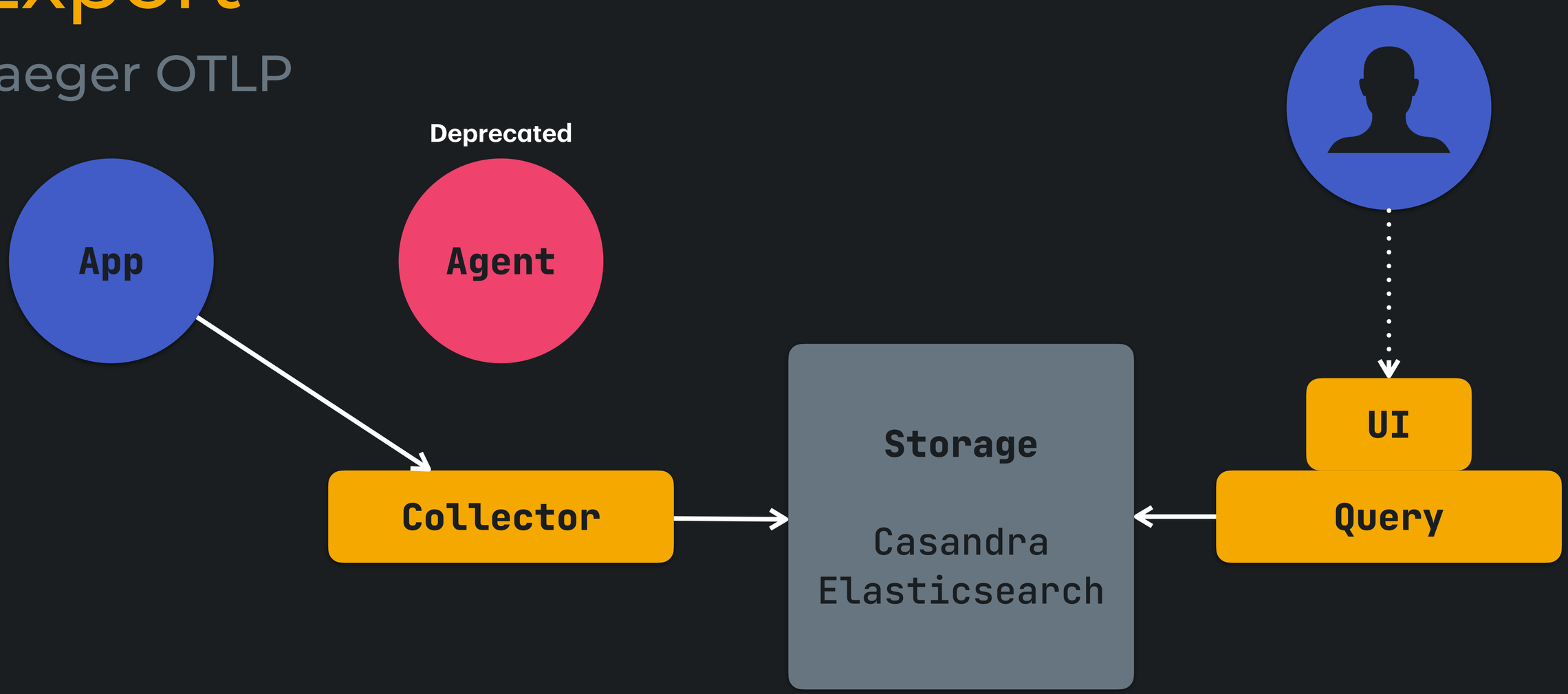
Export

Jaeger



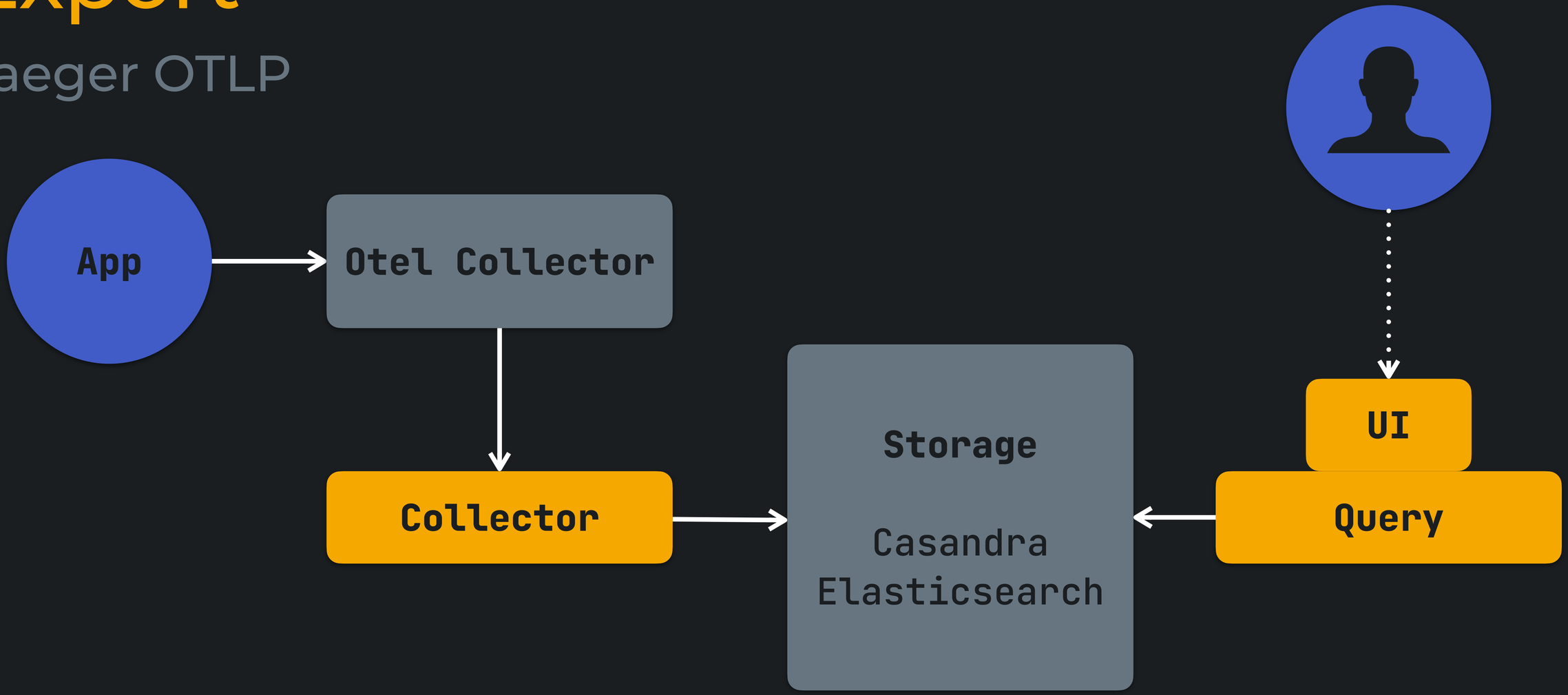
Export

Jaeger OTLP



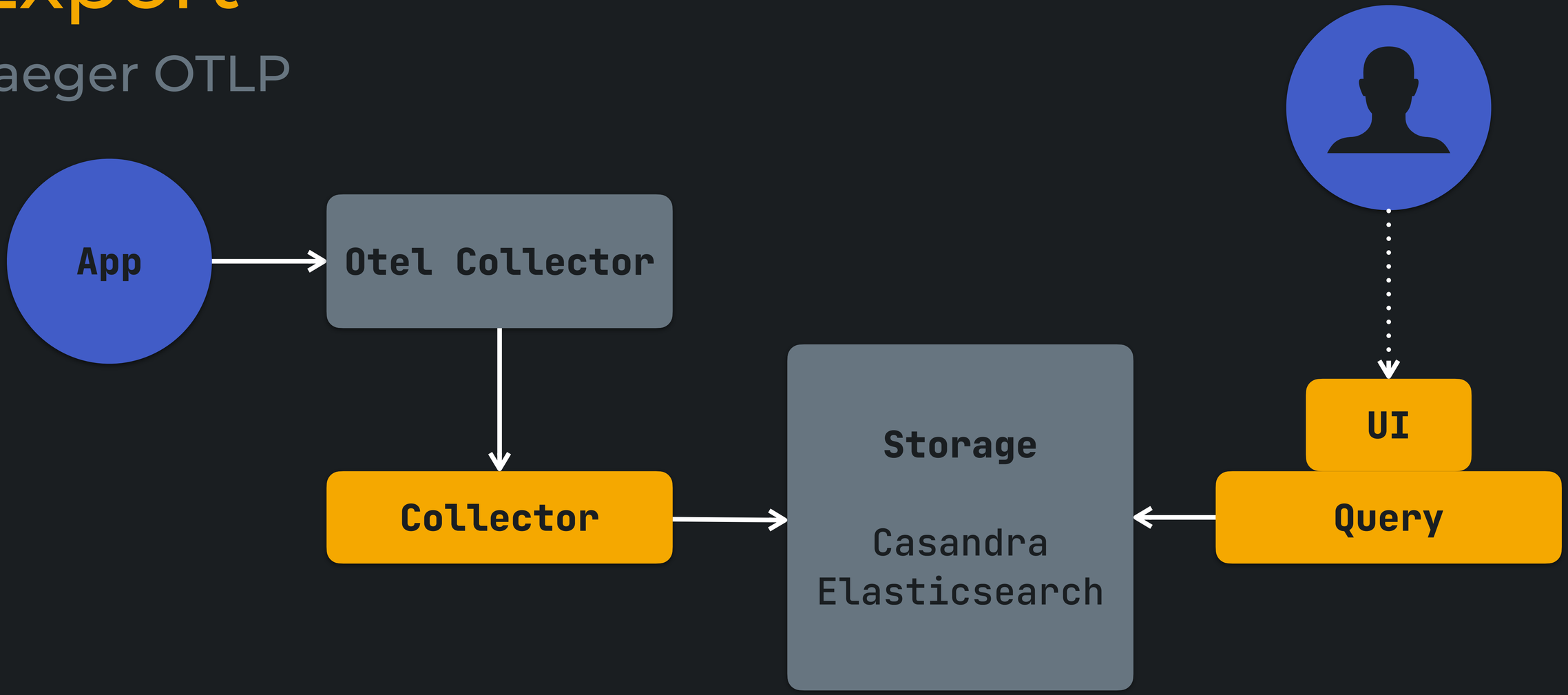
Export

Jaeger OTLP



Export

Jaeger OTLP



Export to Jaeger

Application Configuration

Frontend/Program.cs

```
services.AddOpenTelemetry() // OpenTelemetry && OpenTelemetry.Extensions.Hosting
    .OtherOurConfigurationMetricsAndResource()
    .WithTracing(tracerProviderBuilder => tracerProviderBuilder
        .AddSource(nameof(WeatherForecastController))
        .AddSource(nameof(SettingController))
        .AddSource(nameof(WeatherService))
        .AddSource(nameof(SettingService))
        .SetErrorStatusOnException()
        .SetSampler(new AlwaysOnSampler())
        .AddHttpClientInstrumentation()
        .AddAspNetCoreInstrumentation(options => { options.RecordException = true; })
        .AddOtlpExporter(options =>
        {
            options.ExportProcessorType = ExportProcessorType.Batch;
            options.Protocol = OpenTelemetry.Exporter.OtlpExportProtocol.Grpc;
        })
    )
```

Export to Jaeger

Application Configuration

Frontend/Program.cs

```
services.AddOpenTelemetry() // OpenTelemetry && OpenTelemetry.Extensions.Hosting
    .OtherOurConfigurationMetricsAndResource()
    .WithTracing(tracerProviderBuilder => tracerProviderBuilder
        .AddSource(nameof(WeatherService))
        .AddSource(nameof(SettingService))
        .SetErrorStatusOnException()
        .SetSampler(new AlwaysOnSampler())
        .AddHttpClientInstrumentation()
        .AddAspNetCoreInstrumentation(options => { options.RecordException = true; })
        .AddOtlpExporter(options =>
        {
            options.ExportProcessorType = ExportProcessorType.Batch;
            options.Protocol = OpenTelemetry.Exporter.OtlpExportProtocol.Grpc;
        })
    )
```

Export to Jaeger

Application Configuration

Frontend/WeatherForecastController.cs

```
private static readonly ActivitySource _activitySource = new(nameof(WeatherForecastController), "1.0.0");

[HttpGet]
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    using var activity = _activitySource.StartActivity();
    var city = _settingService.GetCity();

    // other logic...
}
```

Export to Jaeger

Application Configuration

Frontend/WeatherForecastController.cs

```
[HttpGet]
public async Task<ActionResult<WeatherForecast>> GetForecast()
{
    using var activity = Activity.Current;
    var city = _settingService.GetCity();

    // other logic...
}
```

Export to Jaeger

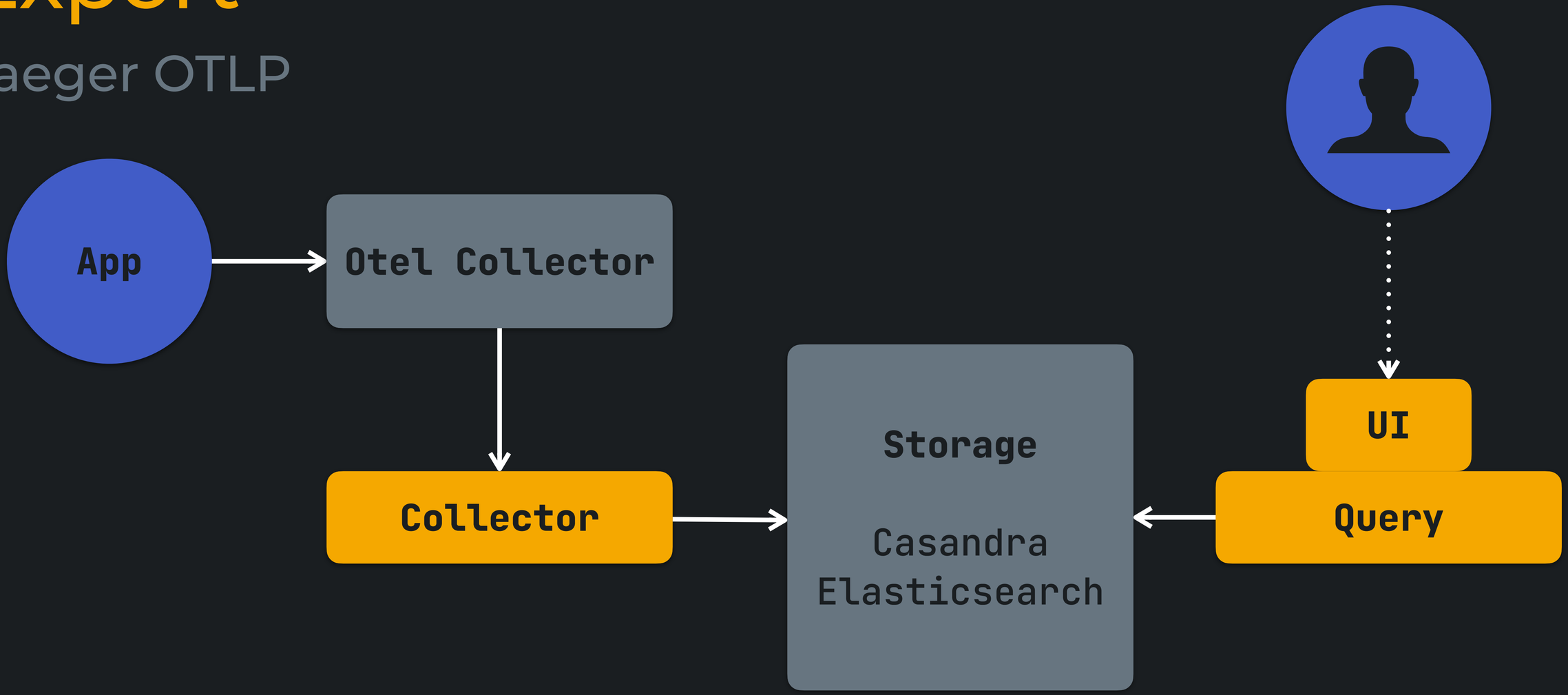
Application Configuration

WeatherAPI/Program.cs

```
services.AddOpenTelemetry()
    .OtherOurConfigurationMetricsAndResource()
    .WithTracing(tracerProviderBuilder ⇒ tracerProviderBuilder
        .AddSource(AppDomain.CurrentDomain.FriendlyName)
        .SetErrorStatusOnException()
        .SetSampler(new AlwaysOnSampler())
        .AddHttpClientInstrumentation()
        .AddAspNetCoreInstrumentation(options ⇒ { options.RecordException = true; })
        .AddRedisInstrumentation(_redisConnection,
            opt ⇒
            {
                opt.Enrich = (activity, eventName) ⇒ activity.SetTag("redis.connection", "localhost:6379");
                opt.Enrich = (activity, eventName) ⇒ activity.SetTag("peer.service", "redis");
                opt.FlushInterval = TimeSpan.FromSeconds(1);
                opt.EnrichActivityWithTimingEvents = true;
            })
        .AddOtlpExporter(options ⇒
        {
            options.ExportProcessorType = ExportProcessorType.Batch;
            options.Protocol = OpenTelemetry.Exporter.OtlpExportProtocol.Grpc;
        }))
```

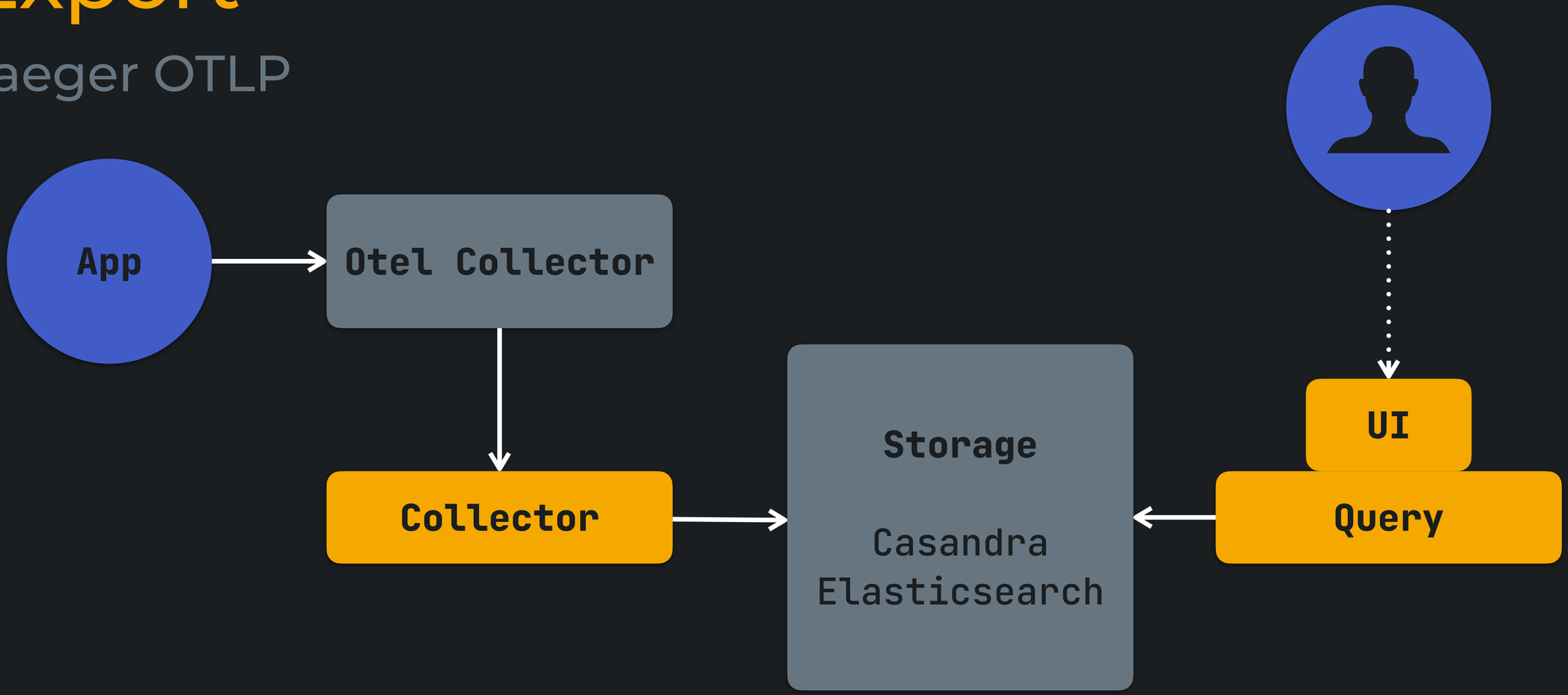
Export

Jaeger OTLP



Export

Jaeger OTLP



Export to Jaeger

OTel Collector configuration

otel-collector.yaml

```yaml
exporters:
  prometheus:
    endpoint: 0.0.0.0:8889

service:
  extensions: [ pprof, zpages, health_check ]
  pipelines:
    metrics:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ prometheus ]
```

Export to Jaeger

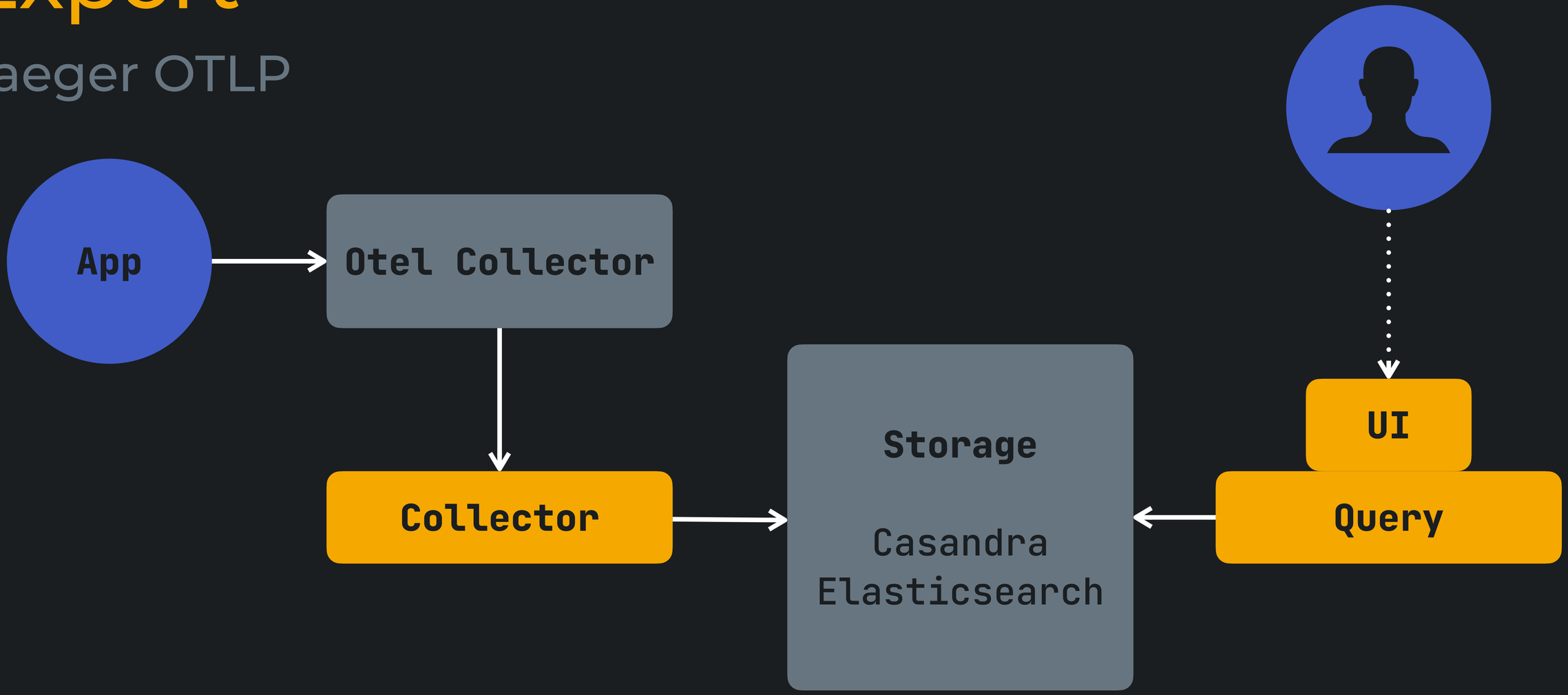
OTel Collector configuration

otel-collector.yaml

```yaml
exporters:
  prometheus:
    endpoint: 0.0.0.0:8889
  otlp/jaeger:
    endpoint: http://jaeger:4317

service:
  extensions: [ pprof, zpages, health_check ]
  pipelines:
    metrics:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ prometheus ]
    traces:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ otlp/jaeger ]
```

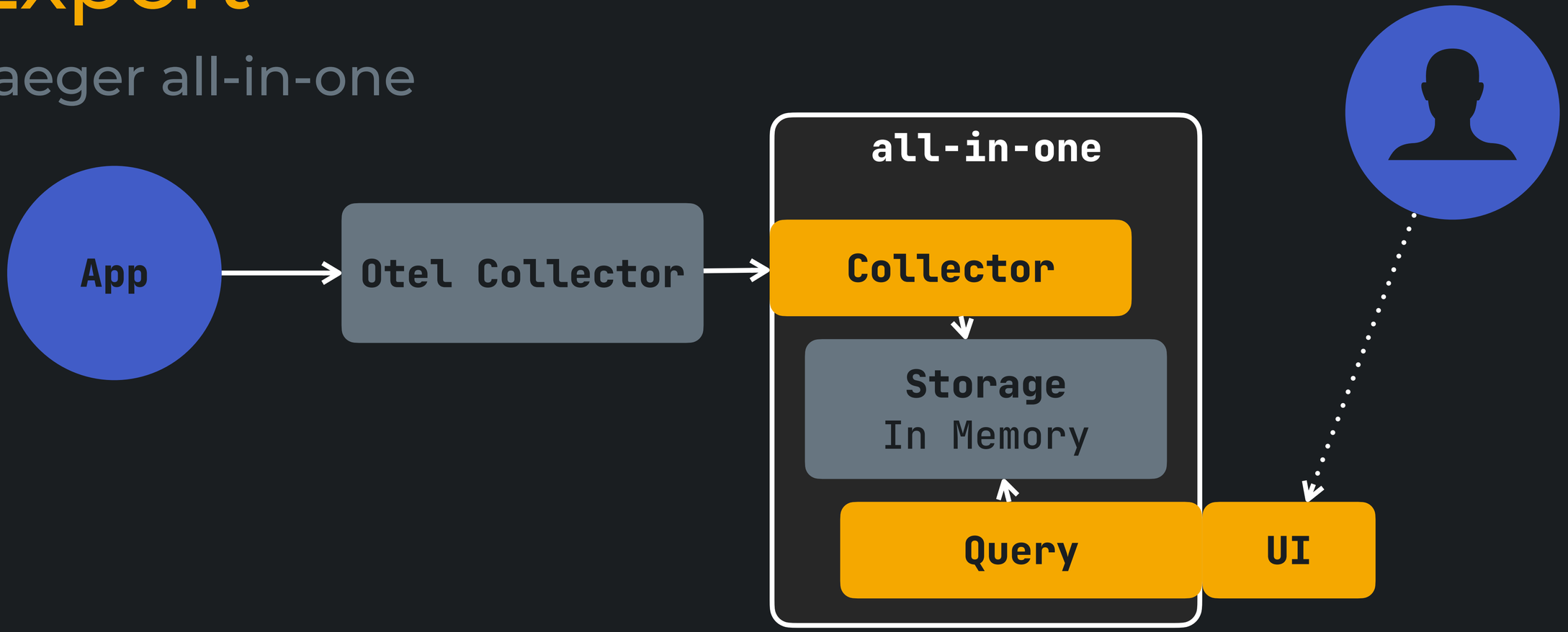
Export

Jaeger OTLP

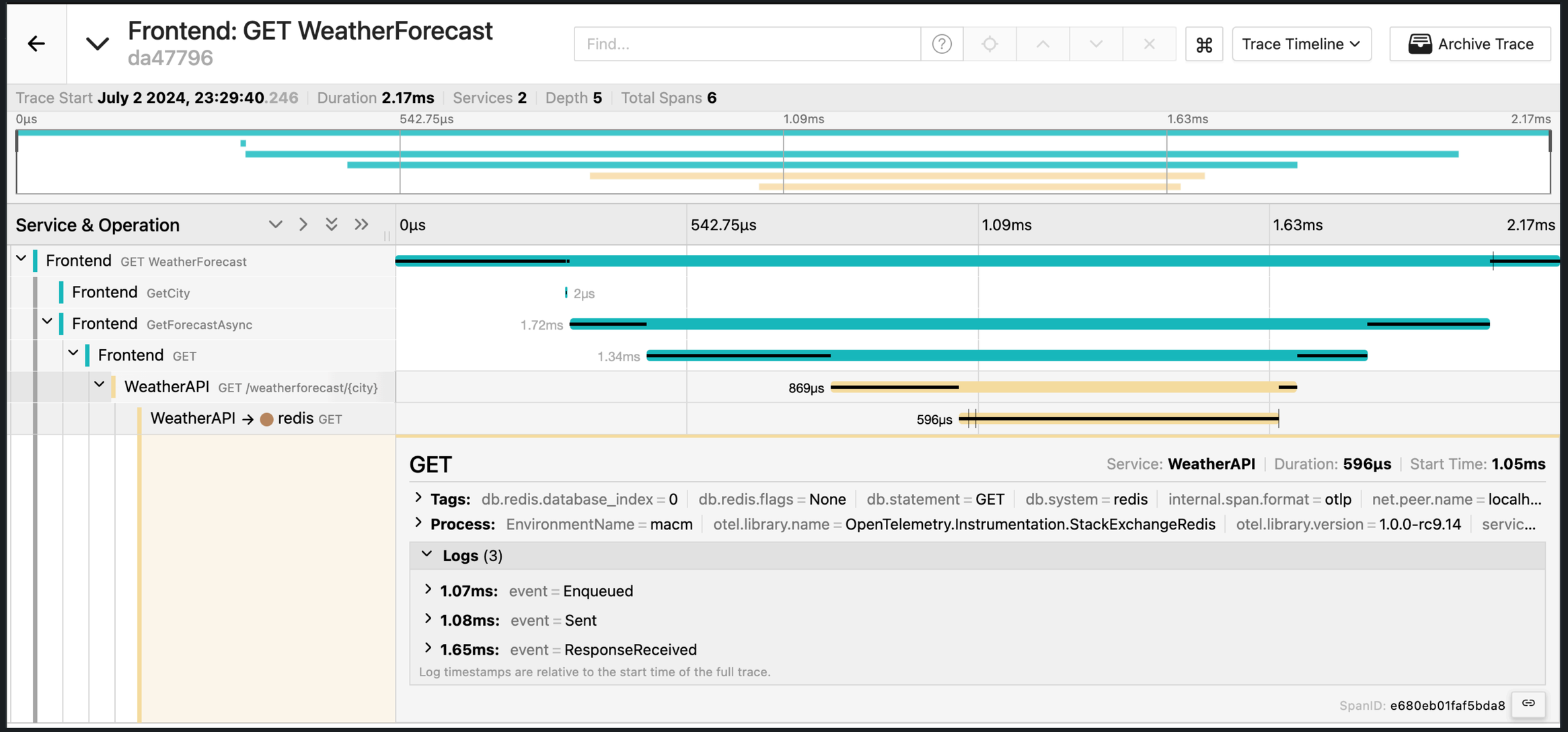


Export

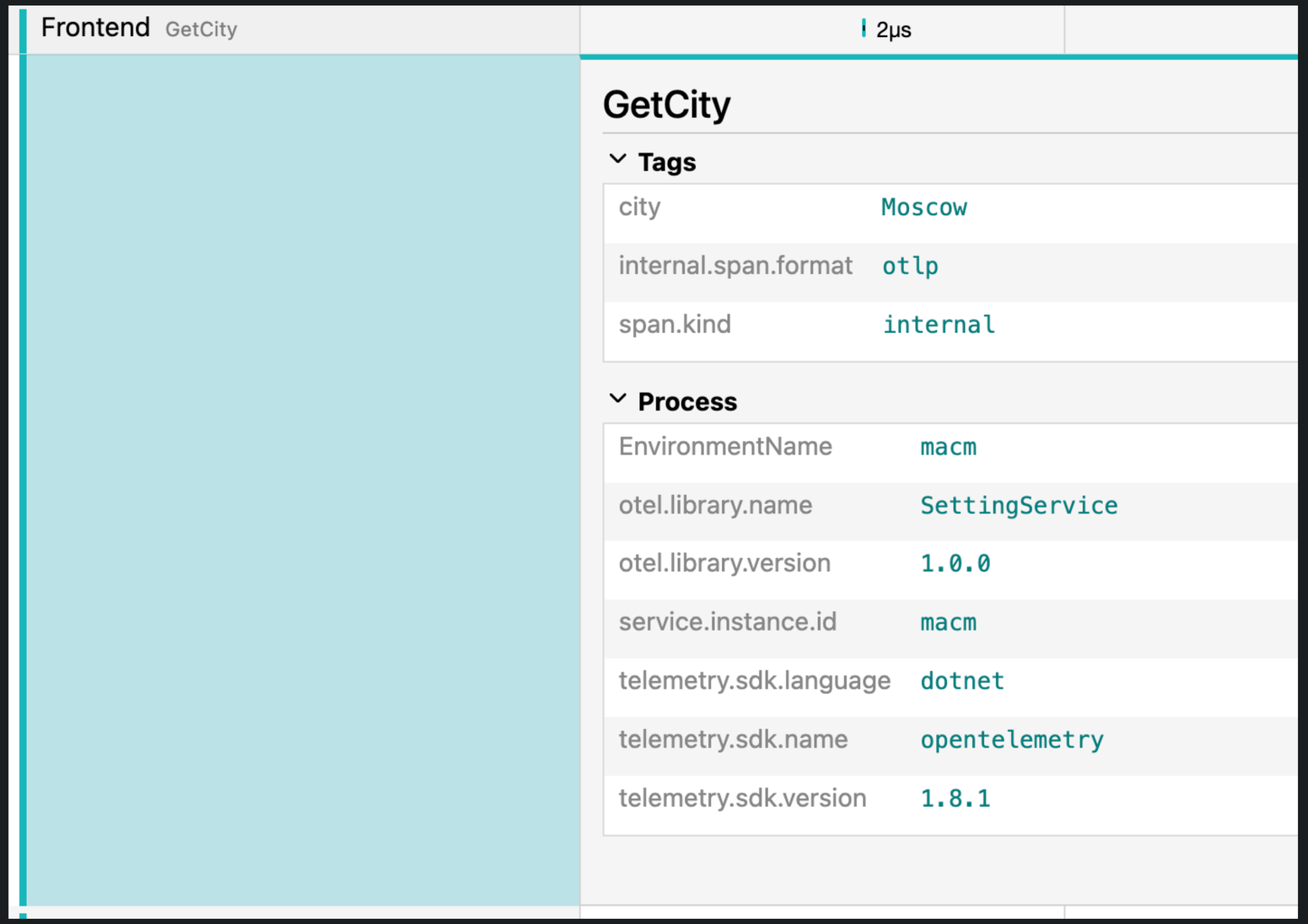
Jaeger all-in-one



Jaeger

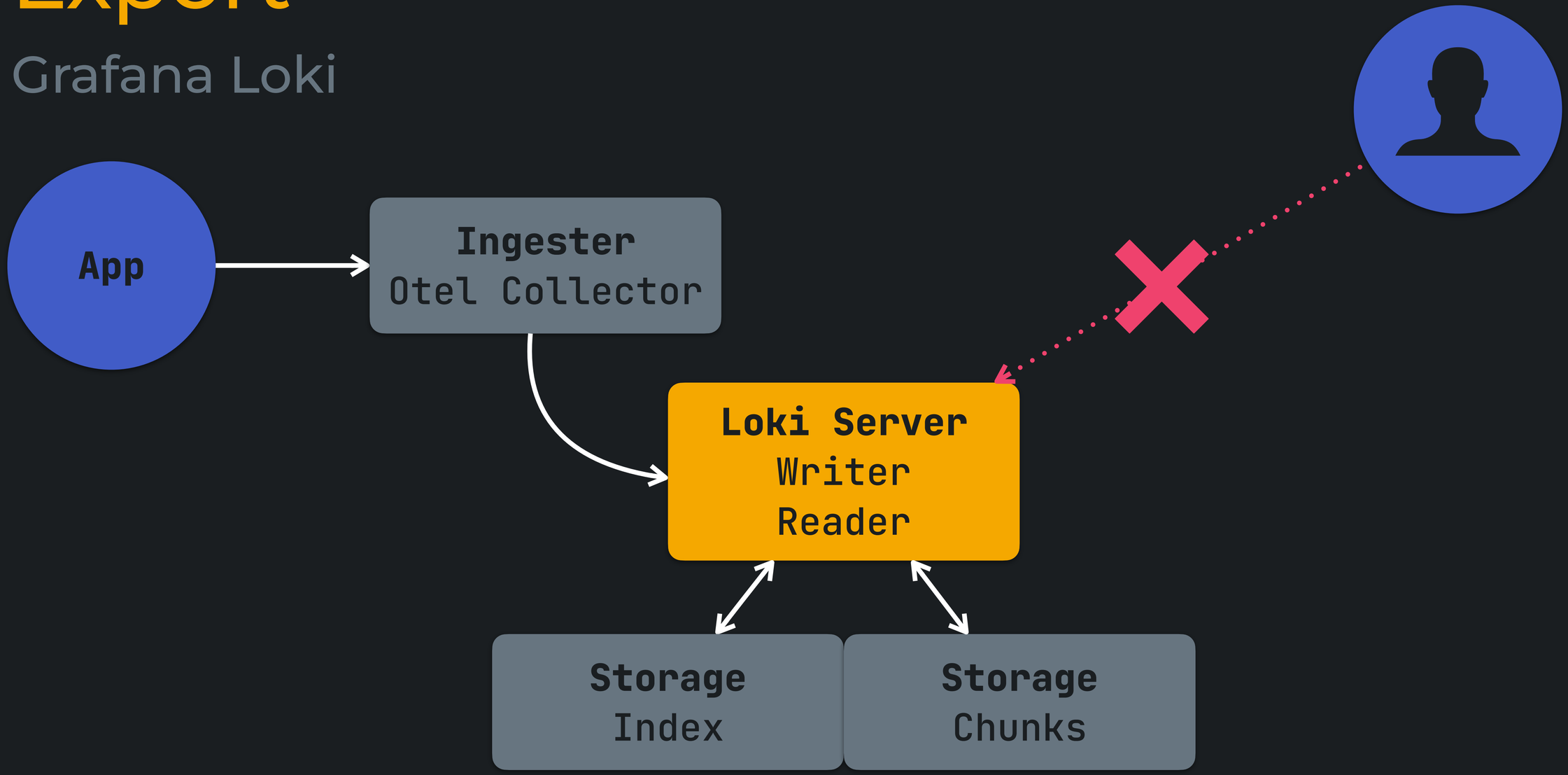

Frontend: GET WeatherForecast
da47796
Find...
Trace Timeline
Archive Trace
Trace Start July 2 2024, 23:29:40.246 | Duration 2.17ms | Services 2 | Depth 5 | Total Spans 6
0µs
542.75µs
1.09ms
1.63ms
2.17ms
Service & Operation
Frontend GET WeatherForecast
Frontend GetCity
2µs
Frontend GetForecastAsync
1.72ms
Frontend GET
1.34ms
WeatherAPI GET /weatherforecast/{city}
869µs
WeatherAPI → redis GET
596µs
GET
Service: WeatherAPI | Duration: 596µs | Start Time: 1.05ms
Tags: db.redis.database_index = 0 | db.redis.flags = None | db.statement = GET | db.system = redis | internal.span.format = otlp | net.peer.name = localh...
Process: EnvironmentName = macm | otel.library.name = OpenTelemetry.Instrumentation.StackExchangeRedis | otel.library.version = 1.0.0-rc9.14 | servic...
Logs (3)
1.07ms: event = Enqueued
1.08ms: event = Sent
1.65ms: event = ResponseReceived
Log timestamps are relative to the start time of the full trace.
SpanID: e680eb01faf5bda8

Jaeger


Frontend GetCity
2µs
GetCity
Tags
city Moscow
internal.span.format otlp
span.kind internal
Process
EnvironmentName macm
otel.library.name SettingService
otel.library.version 1.0.0
service.instance.id macm
telemetry.sdk.language dotnet
telemetry.sdk.name opentelemetry
telemetry.sdk.version 1.8.1

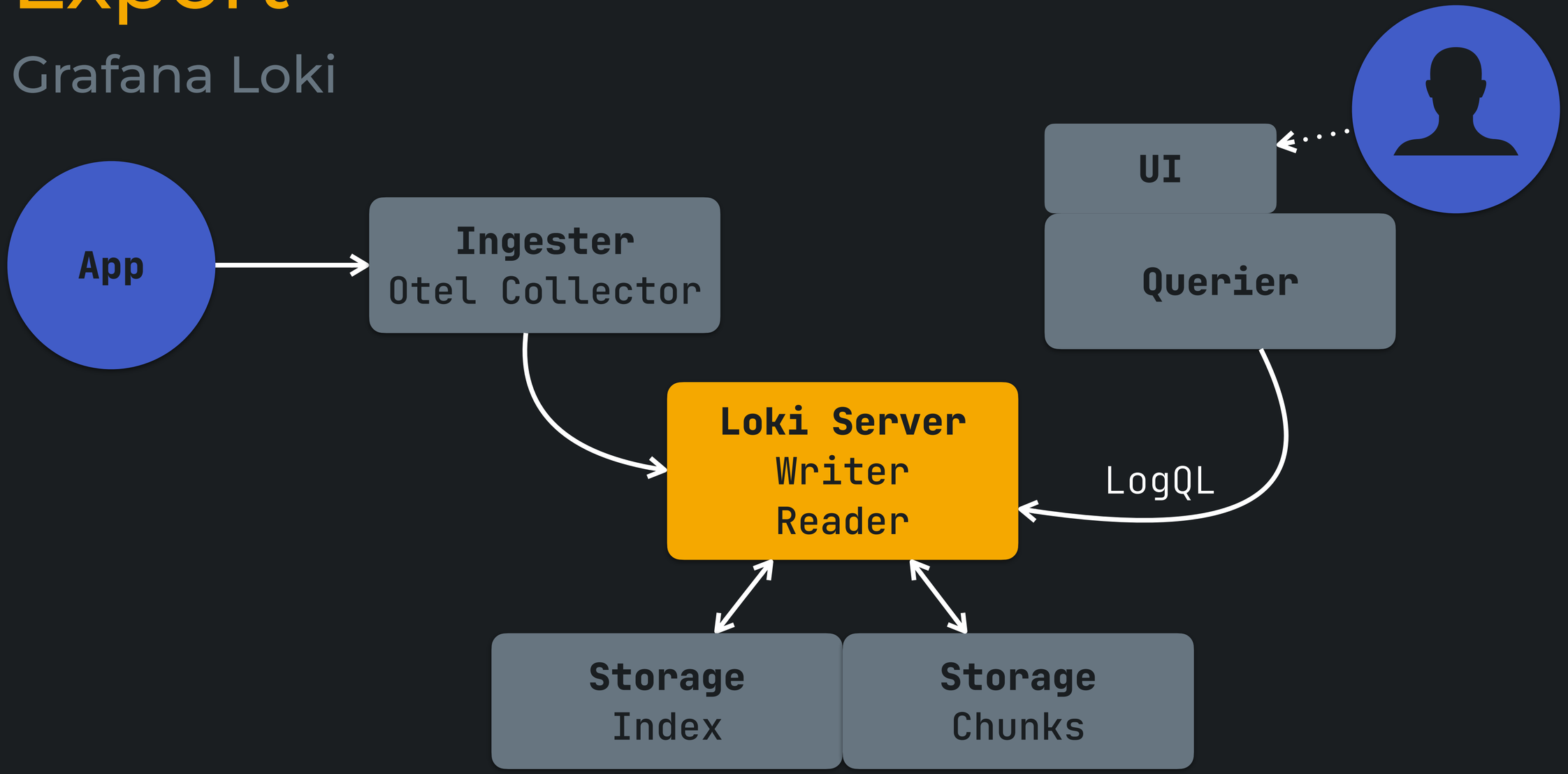
Export

Grafana Loki



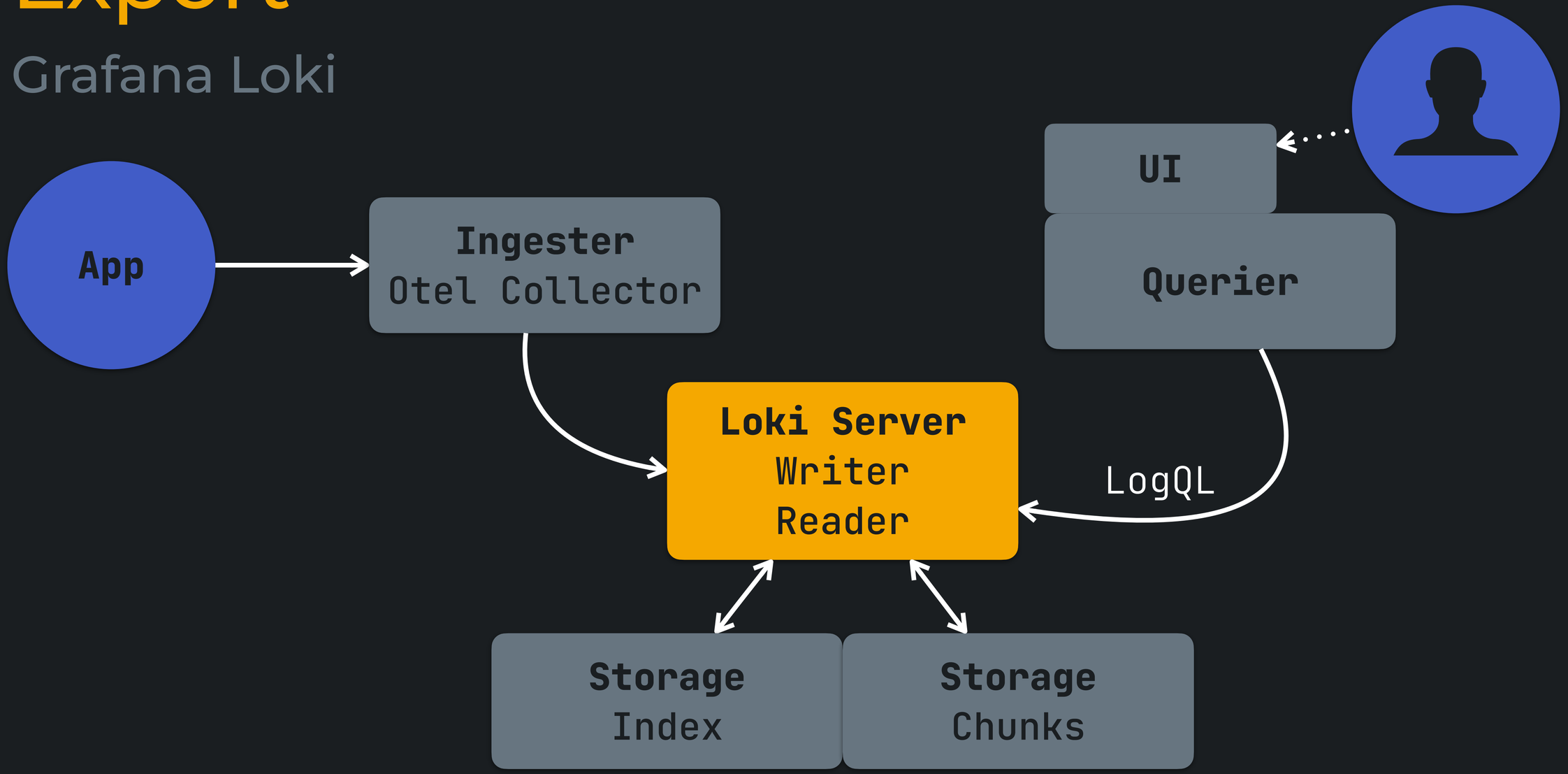
Export

Grafana Loki



Export

Grafana Loki



Export to Loki

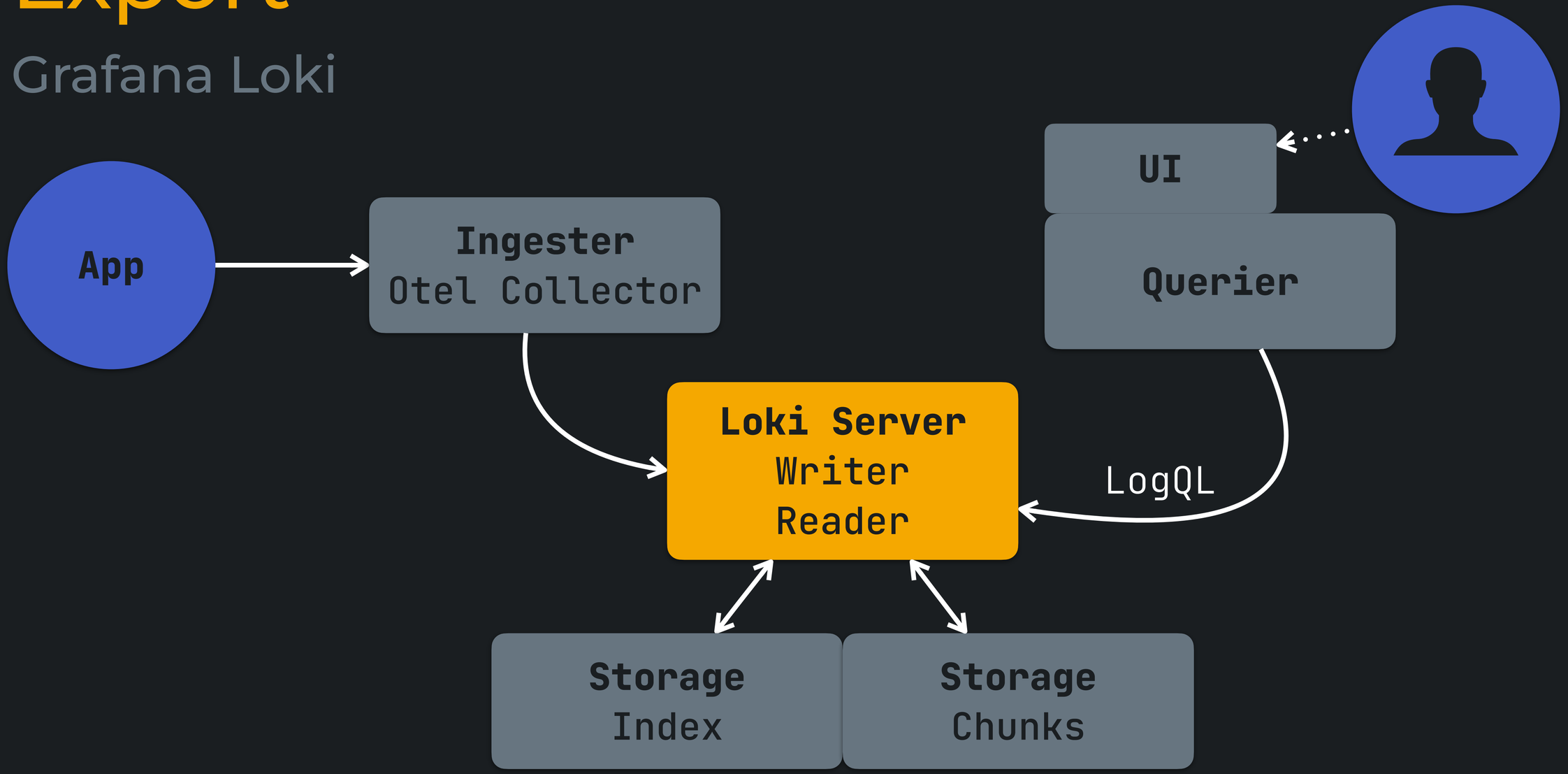
Application Configuration

Frontend/Program.cs

```
host.UseSerilog((ctx, cfg) ⇒
{
    cfg.Enrich.FromLogContext()
        .ReadFrom.Configuration(ctx.Configuration)
        .Enrich.WithProperty("EnvironmentName", ctx.HostingEnvironment.EnvironmentName)
        .WriteTo.Console()
        .WriteTo.OpenTelemetry(config ⇒
        {
            config.IncludedData = IncludedData.TraceIdField | IncludedData.SpanIdField;
            config.Protocol = OtlpProtocol.Grpc;
            config.Endpoint = "http://localhost:4317/otlp/v1/logs";
            config.ResourceAttributes = new Dictionary<string, object>
            {
                { "service.name", ctx.HostingEnvironment.ApplicationName }
            };
        });
});
```

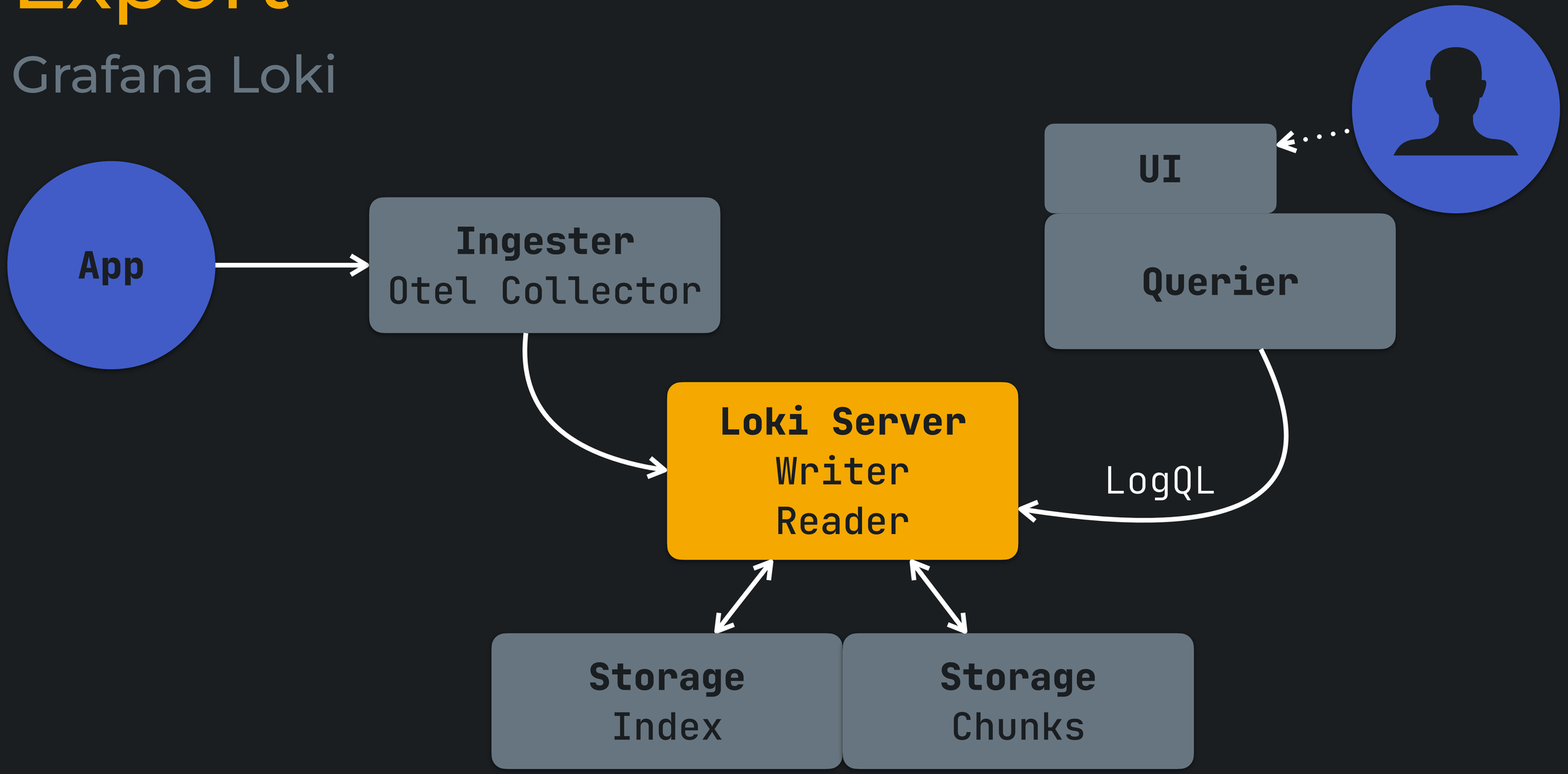
Export

Grafana Loki



Export

Grafana Loki



Export to Loki

OTel Collector configuration

otel-collector.yaml

```yaml
exporters:
  prometheus:
    endpoint: 0.0.0.0:8889
  otlp/jaeger:
    endpoint: http://jaeger:4317

service:
  extensions: [ pprof, zpages, health_check ]
  pipelines:
    metrics:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ prometheus ]
    traces:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ otlp/jaeger ]
```

Export to Loki

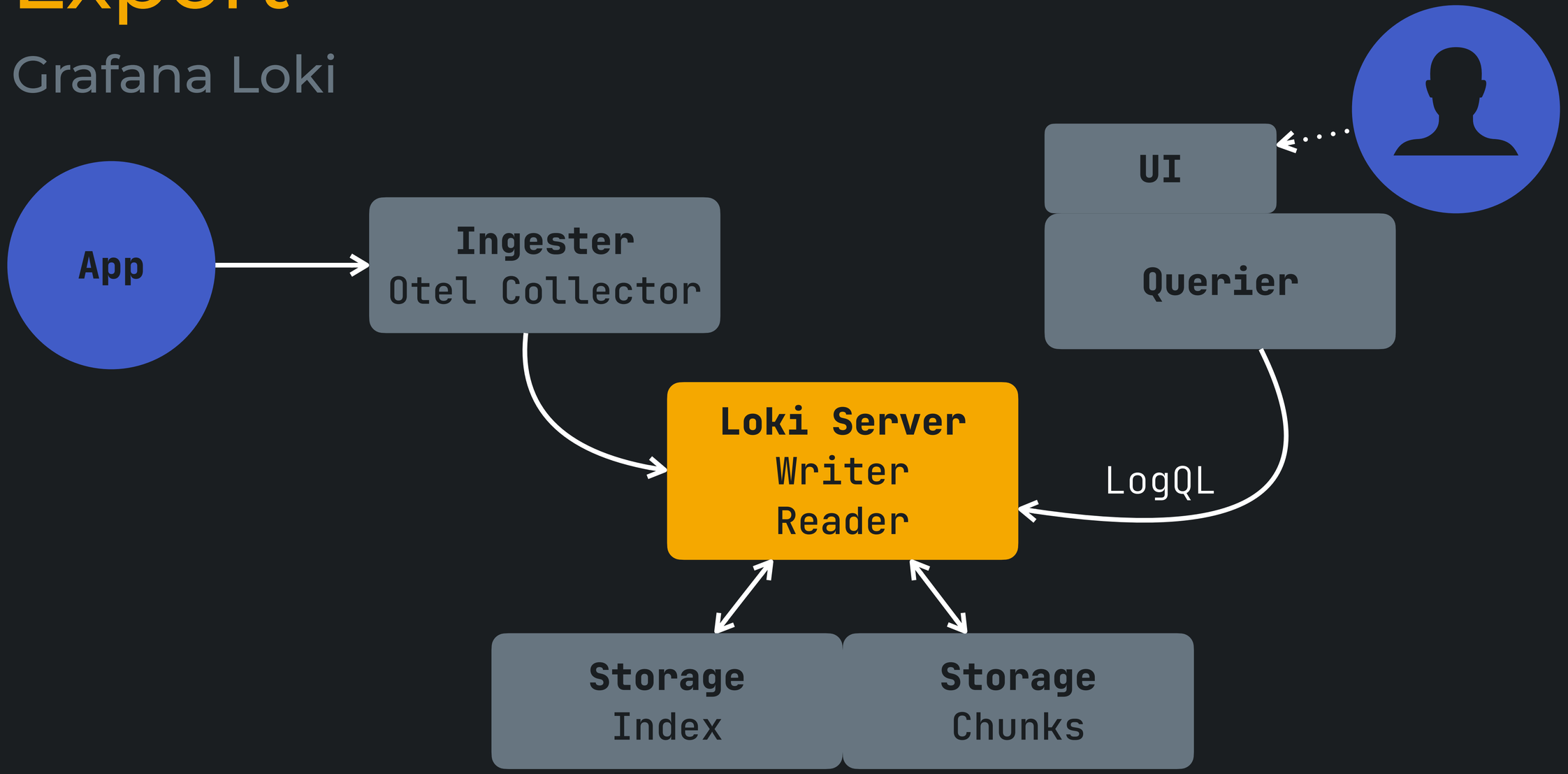
OTel Collector configuration

otel-collector.yaml

```yaml
exporters:
  prometheus:
    endpoint: 0.0.0.0:8889
  otlp/jaeger:
    endpoint: http://jaeger:4317
  loki:
    endpoint: http://loki:3100/loki/api/v1/push

service:
  extensions: [ pprof, zpages, health_check ]
  pipelines:
    #other pipelines
    logs:
      receivers: [ otlp ]
      processors: [ batch ]
      exporters: [ loki ]
```

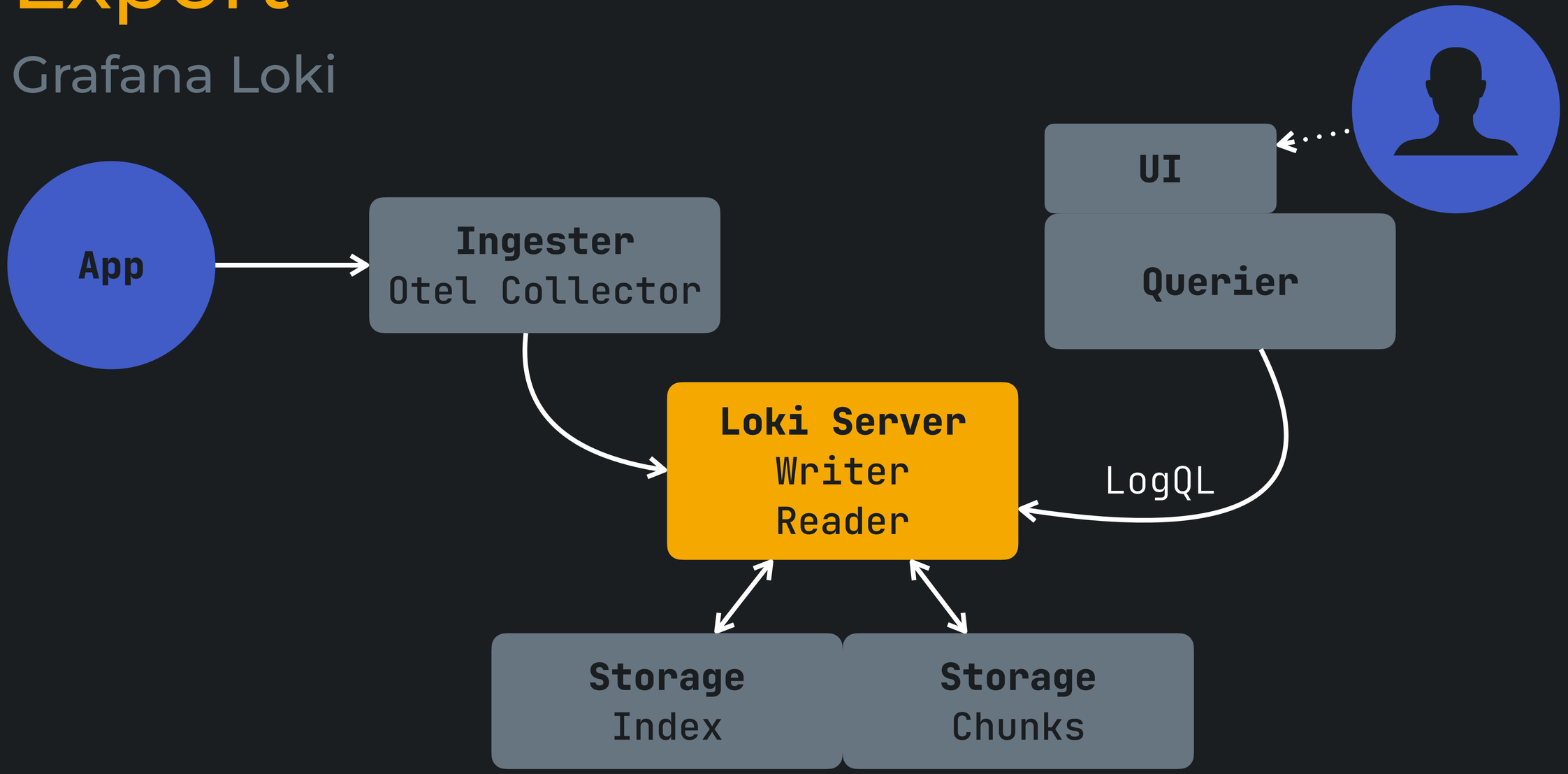
Export

Grafana Loki



Export

Grafana Loki



Export to Loki

Loki Configuration

loki.yaml

```yaml
auth_enabled: false

server:
    http_listen_port: 3100

common:
    ring:
        instance_addr: 127.0.0.1
        kvstore:
            store: inmemory
    replication_factor: 1
    path_prefix: /tmp/loki
```

Export to Loki

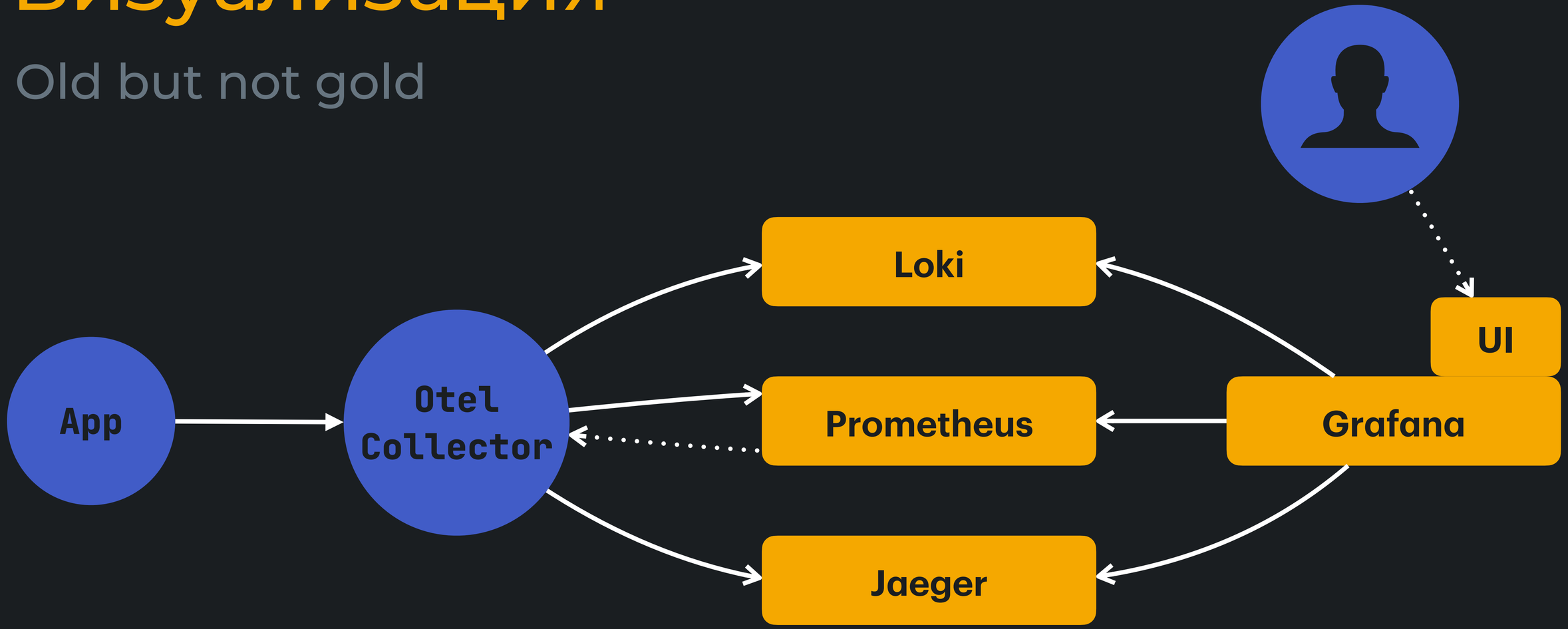
Loki Configuration

loki.yaml

```yaml
schema_config:
    configs:
        -   from: 2020-05-15
            store: tsdb
            object_store: filesystem
            schema: v13
            index:
                prefix: index_
                period: 24h

storage_config:
    filesystem:
        directory: /tmp/loki/chunks
```

Визуализация

Old but not gold



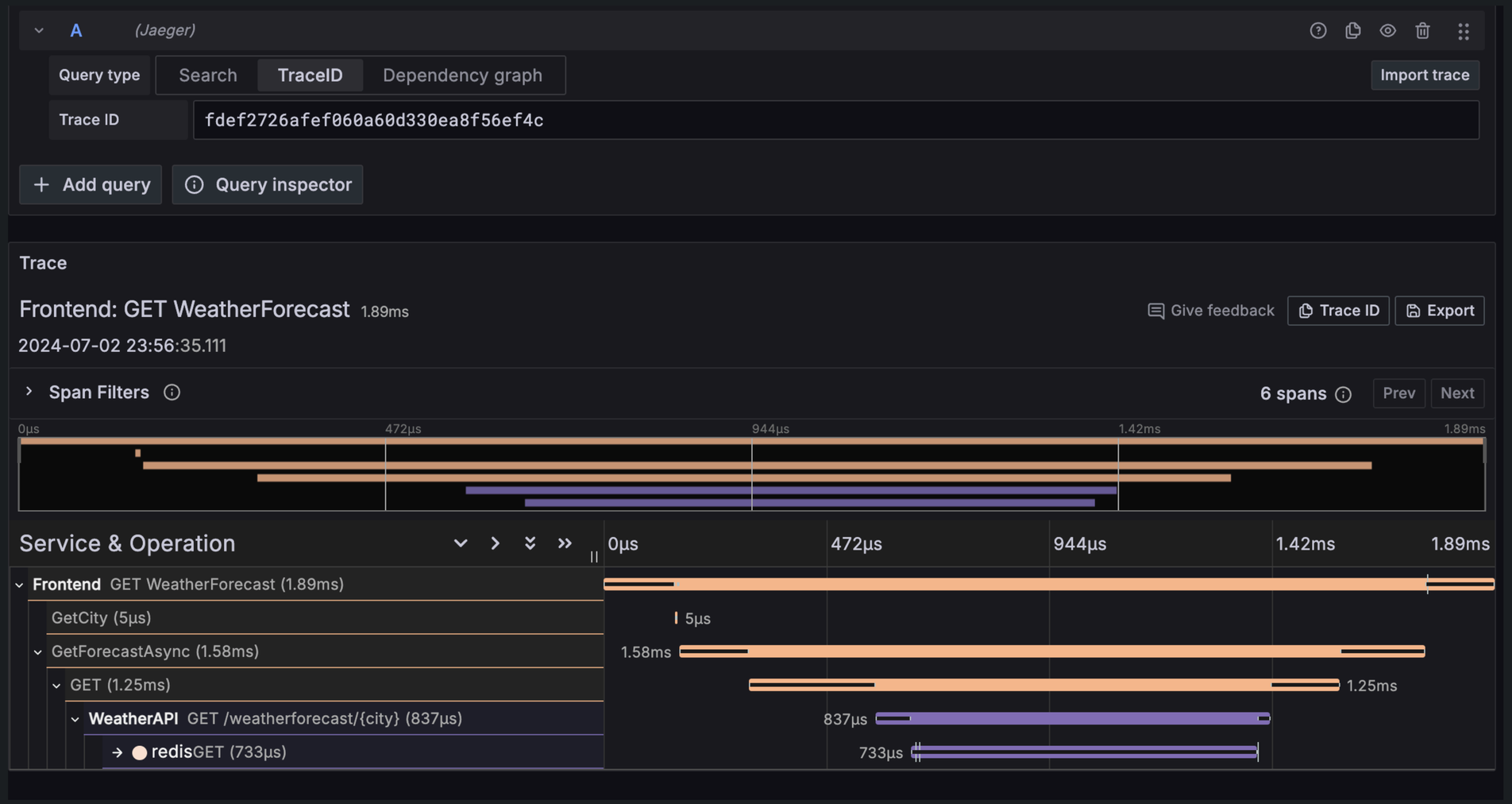
Grafana

Logs


```
› 2024-07-02 23:56:35.113 {
                            "body": "Weather forecast equal to {\"Temperature\":34,\"Humidity\":91,\"WindSpeed\":6,\"Summary\":\"Hot\",\"City\":\"Ryazan\",\"$type\":\"WeatherFo
                          recast\"}",
                            "traceid": "fdef2726afef060a60d330ea8f56ef4c",
                            "spanid": "a538e7cfe0ccf1fd",
                            "severity": "Information",
                            "attributes": {
                              "ActionId": "1de76f14-8350-4ea2-8487-4dec0737597f",
                              "ActionName": "Frontend.Controllers.WeatherForecastController.GetForecast (Frontend)",
                              "ConnectionId": "0HN4QR37ETINU",
                              "EnvironmentName": "Development",
                              "Forecast": {
                                "City": "Ryazan",
                                "Humidity": 91,
                                "Summary": "Hot",
                                "Temperature": 34,
                                "WindSpeed": 6
                              },
                              "RequestId": "0HN4QR37ETINU:00000002",
                              "RequestPath": "/WeatherForecast"
                            },
                            "resources": {
                              "service.name": "Frontend"
                            },
                            "instrumentation_scope": {
                              "name": "Frontend.Controllers.WeatherForecastController"
                            }
                          }
```

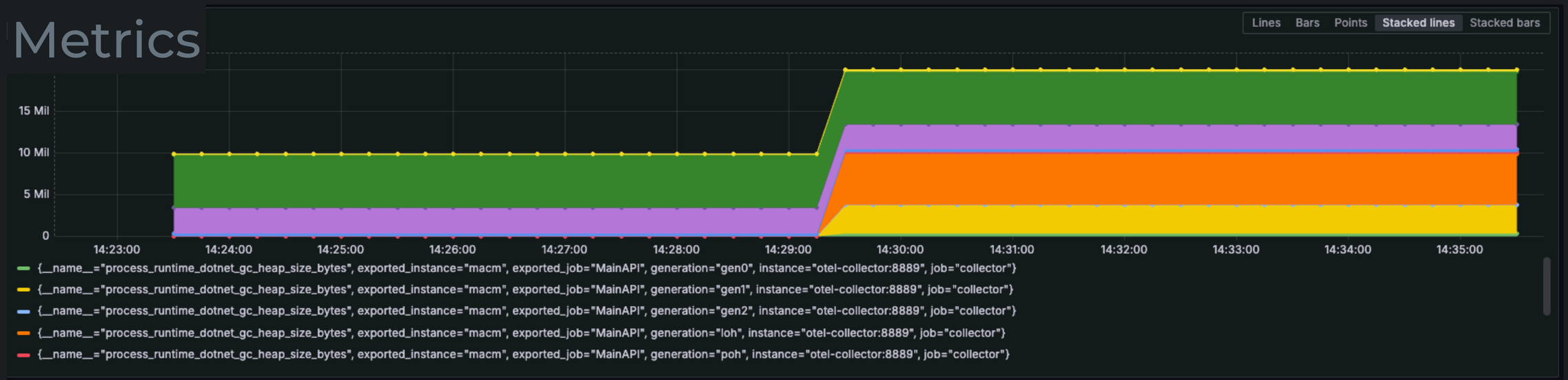
Grafana

Traces



Grafana

Metrics



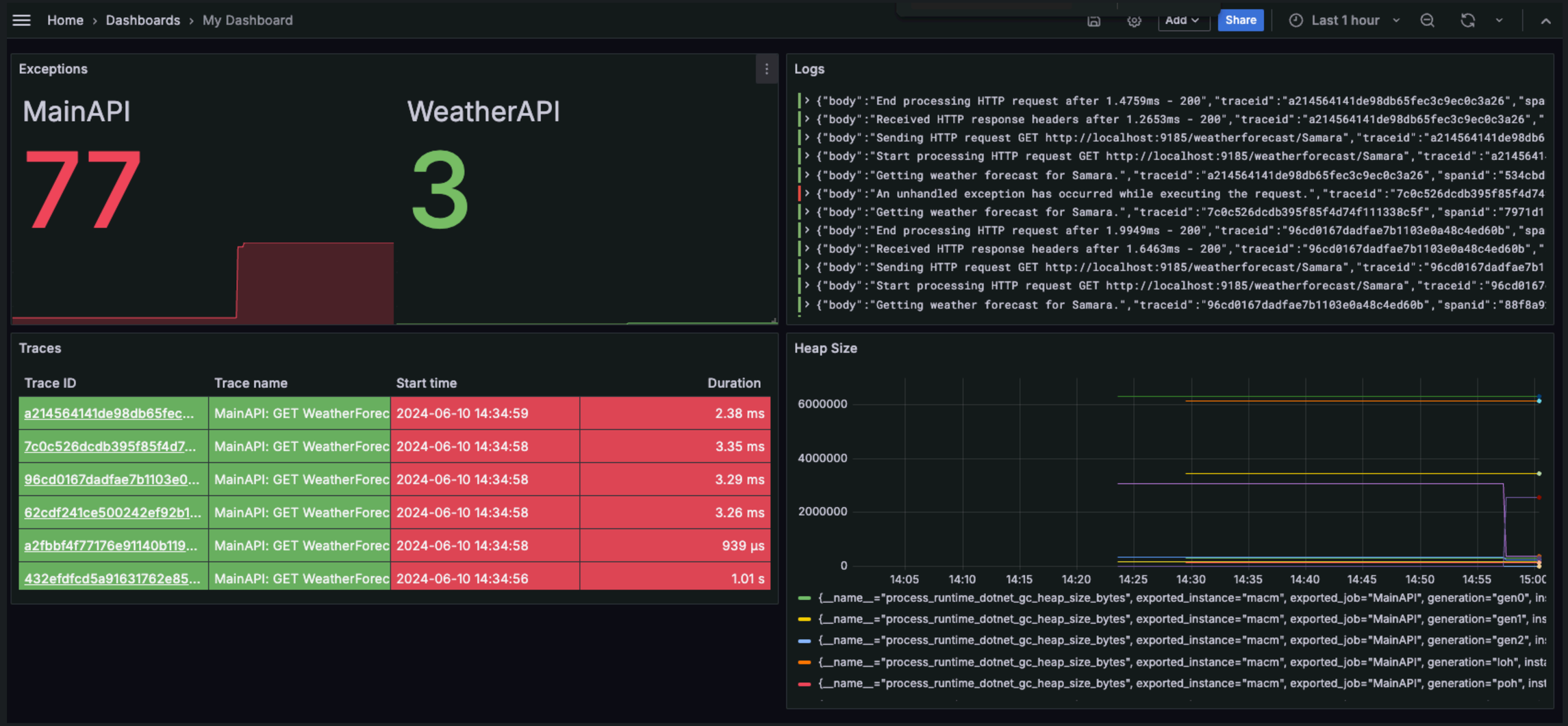
Table

Table | Raw

Time	__name__	exported_instance	exported_job	generation	instance	job	Value
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	WeatherAPI	gen0	otel-collector:8889	collector	341176
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	WeatherAPI	gen1	otel-collector:8889	collector	3063936
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	WeatherAPI	gen2	otel-collector:8889	collector	264
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	WeatherAPI	loh	otel-collector:8889	collector	6325512
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	WeatherAPI	poh	otel-collector:8889	collector	156712
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	MainAPI	gen0	otel-collector:8889	collector	316176
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	MainAPI	gen1	otel-collector:8889	collector	3457112
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	MainAPI	gen2	otel-collector:8889	collector	264
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	MainAPI	loh	otel-collector:8889	collector	6155048
2024-06-10 14:35:44.548	process_runtime_dotnet_gc_h	macm	MainAPI	poh	otel-collector:8889	collector	148472

Grafana

Dashboards



Grafana

Ready Dashboards

Filters:

Data Source

All

Panel

All

Collector Types

All

Sort by

Downloads

Category

All

Dashboards with screenshots

Share your dashboards

Export any dashboard from Grafana 3.1 or greater and share your creations with the community.

Upload from user portal

Search dashboards

Prometheus

Node Exporter Full

Prometheus

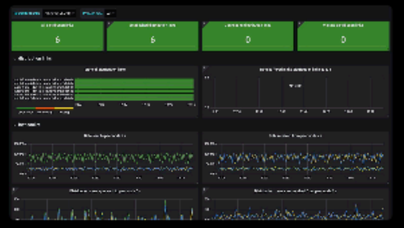
RabbitMQ-Overview

Prometheus

RabbitMQ-Quorum-Queues-Raft

Prometheus

Erlang-Memory-Allocators



Prometheus

Erlang-Distribution

Prometheus

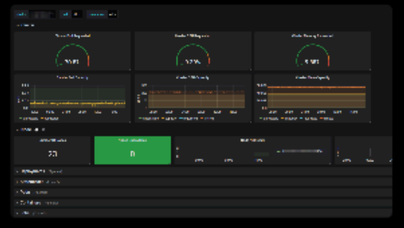
Erlang-Distributions-Compare

Prometheus

RabbitMQ-Stream

Prometheus

CoreDNS

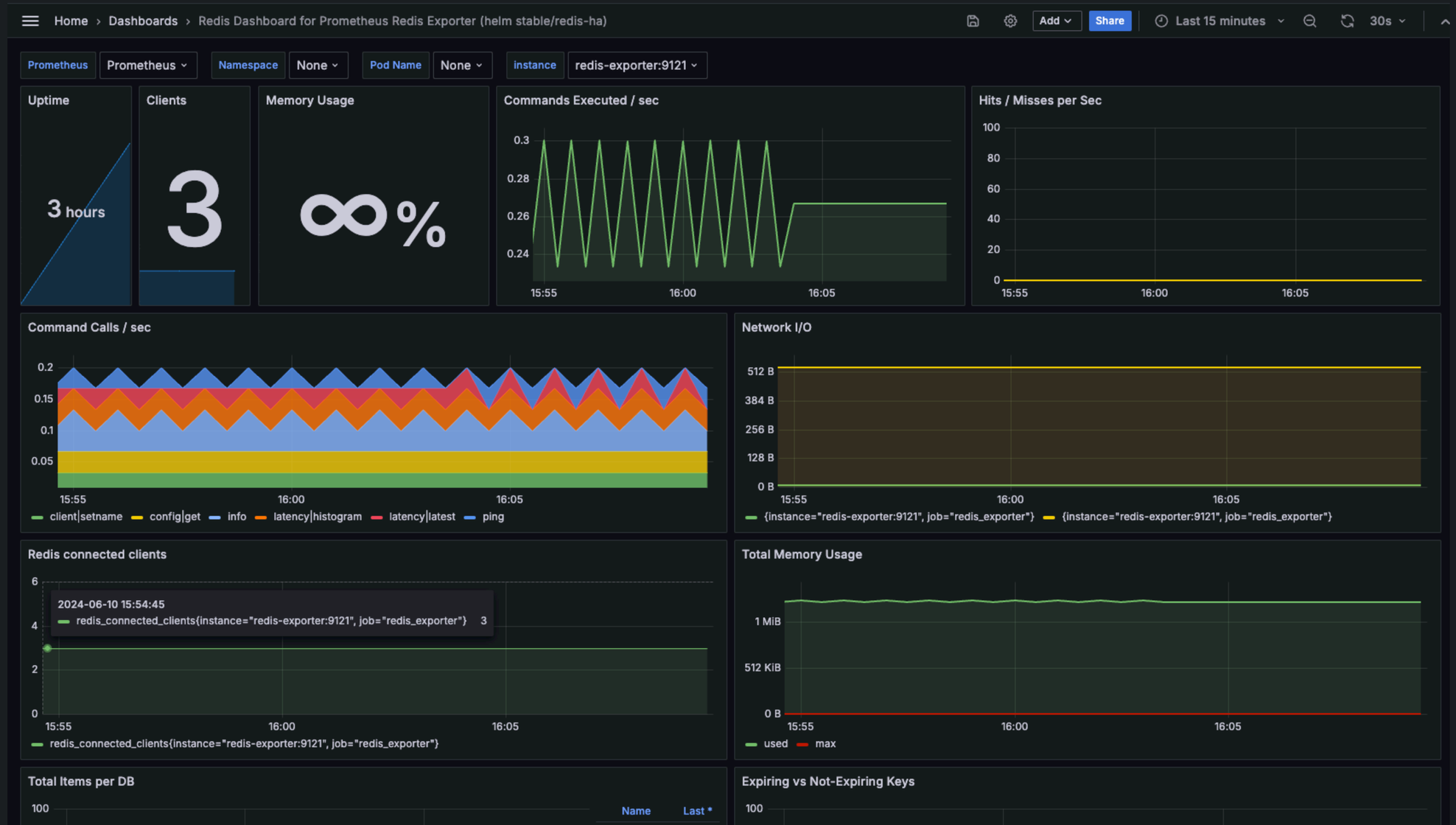


Prometheus

kube-state-metrics-v2

Grafana

Redis Dashboards



Запариваемся

docker-compose.yaml

```yaml
version: "3.9"

volumes:
  loki:
  grafana:
  prometheus:

services:
  prometheus:
    image: prom/prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./config/prometheus/prometheus-collector.yml:/etc/prometheus/prometheus.yml
      - prometheus:/prometheus
    environment:
      - config.file=/etc/prometheus/prometheus-collector.yml
  otel-collector:
    image: otel/opentelemetry-collector-contrib
    command: --config /etc/otel-collector.yaml
    volumes:
      - ./config/otel-collector/otel-collector.yaml:/etc/otel-collector.yaml
    ports:
      - "4317:4317" # OTLP gRPC receiver
      - "13133:13133" # health_check extension
```

Запариваемся

docker-compose.yaml

```yaml
jaeger:
  image: jaegertracing/all-in-one
  ports:
    - "16686:16686"
loki:
  image: grafana/loki
  command: -config.file=/etc/loki/local-config.yaml
  volumes:
    - loki:/loki
grafana:
  image: grafana/grafana-oss
  ports:
    - "3000:3000" # UI
  volumes:
    - grafana:/var/lib/grafana
    - ./config/grafana/grafana.yaml:/etc/grafana/provisioning/datasources/datasources.yaml
  environment:
    - GF_PATHS_PROVISIONING=/etc/grafana/provisioning
    - GF_AUTH_ANONYMOUS_ENABLED=true
    - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
    - GF_AUTH_DISABLE_LOGIN_FORM=true
  depends_on:
    - loki
    - prometheus
    - jaeger
```

Не запариваемся

docker-compose.yaml

```yaml
version: "3.9"

volumes:
    loki: ~
    grafana: ~
    prometheus: ~

services:
    redis:
        image: redis
        ports:
            - "6379:6379"
    aspire-dashboard:
        image: mcr.microsoft.com/dotnet/nightly/aspire-dashboard
        ports:
            - "4317:18889"
            - "18888:18888"
```

Не запариваемся

Не запариваемся

Выводы

Автор не любит YAML

До чего не дошли руки

- OTel Collector Processors
- Alerts
- Health checks
- Deployment & Enterprise configuring
- Scalability
- Data security

Полезные ссылки

OpenTelemetry

https://learn.microsoft.com/ru-ru/dotnet/core/diagnostics/observability-with-otel

https://dev.to/kim-ch/observability-net-opentelemetry-collector-25g1

https://www.youtube.com/watch?v=gviWKCXwyvY

https://www.youtube.com/watch?v=WzZI_IT6gYo

https://www.youtube.com/watch?v=6f3HH7JZTrE

Logs

https://www.youtube.com/watch?v=PBfKDyNPBug

https://learn.microsoft.com/en-us/dotnet/core/diagnostics/logging-tracing

https://learn.microsoft.com/en-us/aspnet/core/fundamentals/logging

Metrics

https://www.youtube.com/watch?v=yAja9Q2KyDM

https://www.youtube.com/watch?v=SvOLXRwe6YQ&t=109s

https://learn.microsoft.com/en-us/dotnet/core/diagnostics/metrics

Traces

https://habr.com/ru/articles/742450/

https://www.youtube.com/watch?v=bYc7u9HFG0s

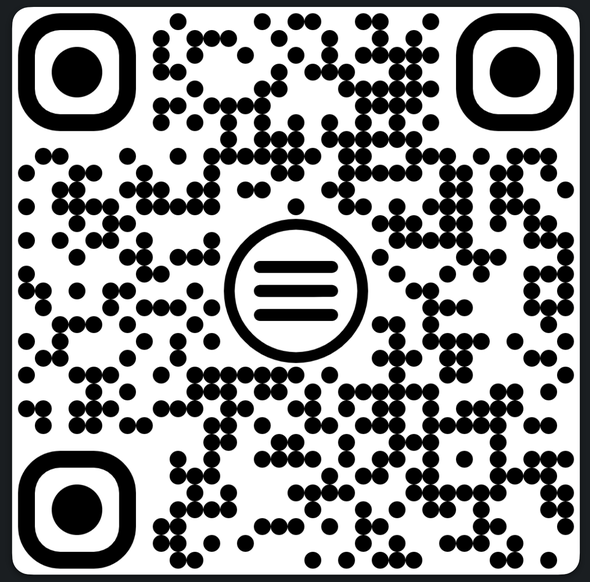
https://www.youtube.com/watch?v=d1gaXhNIsGw

https://www.uber.com/blog/distributed-tracing/

https://github.com/serilog-tracing/serilog-tracing

https://learn.microsoft.com/en-us/dotnet/core/diagnostics/distributed-tracing-instrumentation-walkthroughs

Не самые полезные ссылки



Alexander Goldebaev

@bornToWhine

sgoldebaev@gmail.com

Alexanderbtw