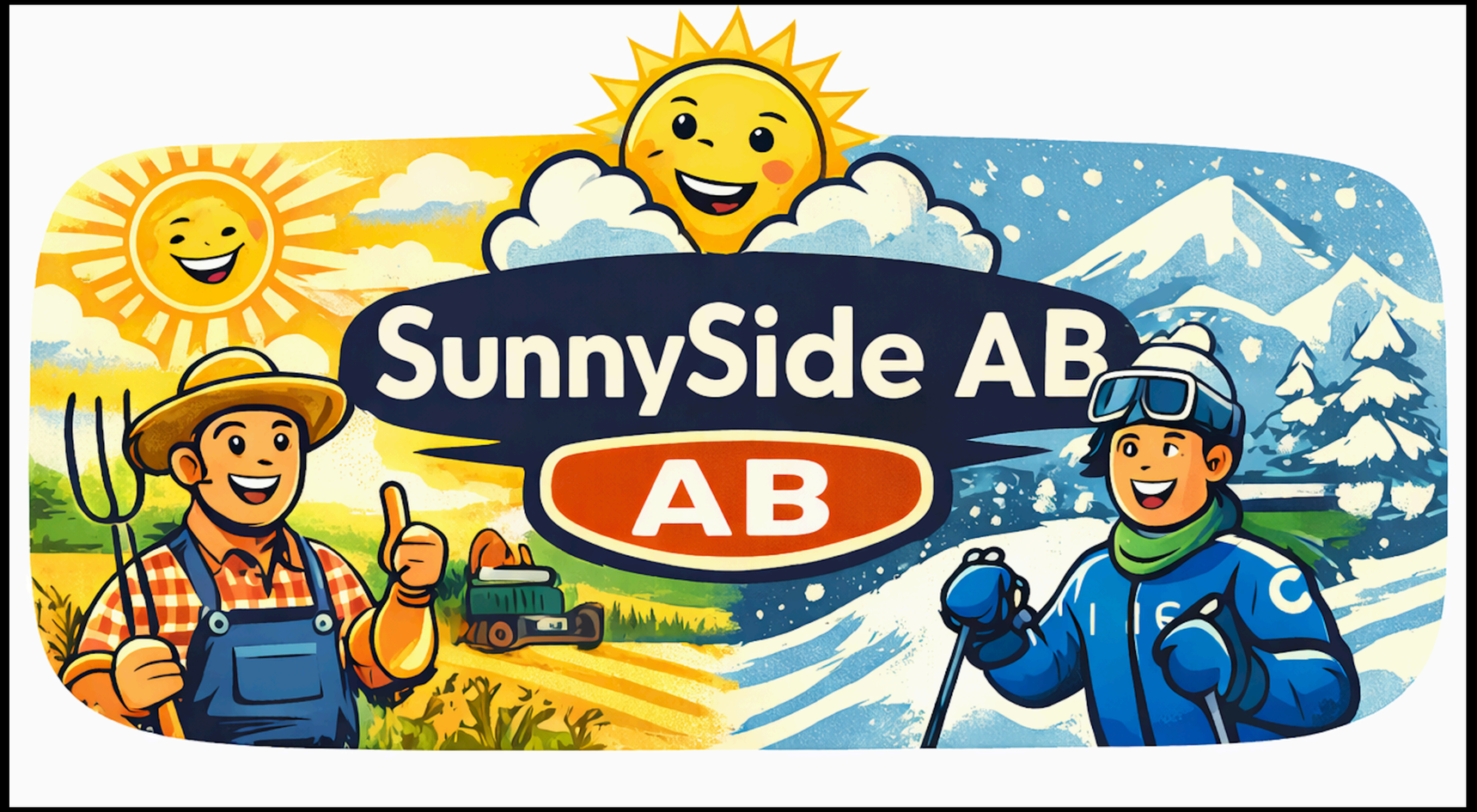
# OBSERVABILITY WITH OPENTELEMETRY

MARTIN HOLT

CADEC 2026.01.22 & 2026.01.28 | CALLISTAENTERPRISE.SE

CALLISTA

# INTRODUCING...

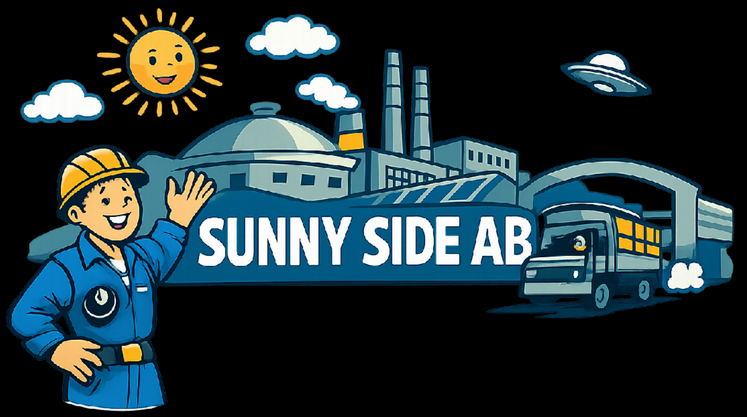


# A NEW ARCHITECTURE!

Team Weathermen

kafka

Team Kafka




Team DreamMachine

## A QUESTION!


# How Do We Know That This Works?

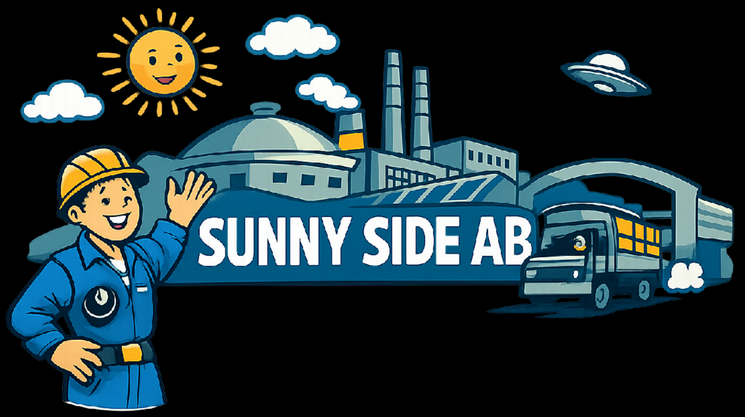
# NOW WITH OBSERVABILITY!

Team Weathermen

kafka

Team Kafka

Team Observability




Team DreamMachine

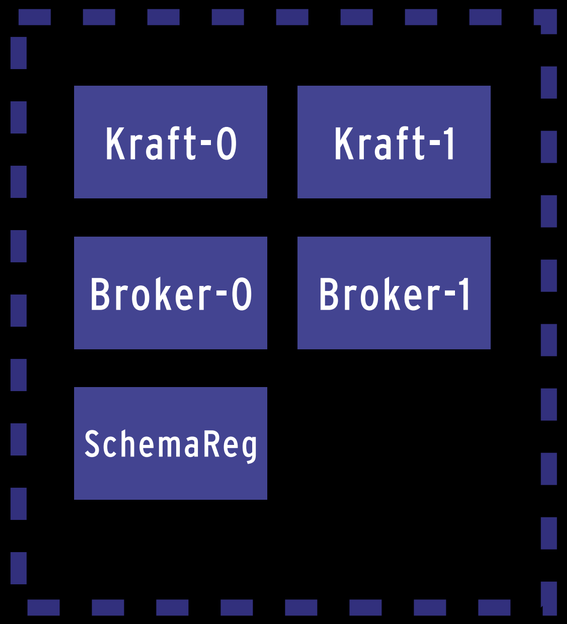
# WHY DO WE OBSERVE?

# FEEDBACK IS GREAT!

OBSERVE

ACTION

Team Kafka




OBSERVE

ACTION

REPORTS

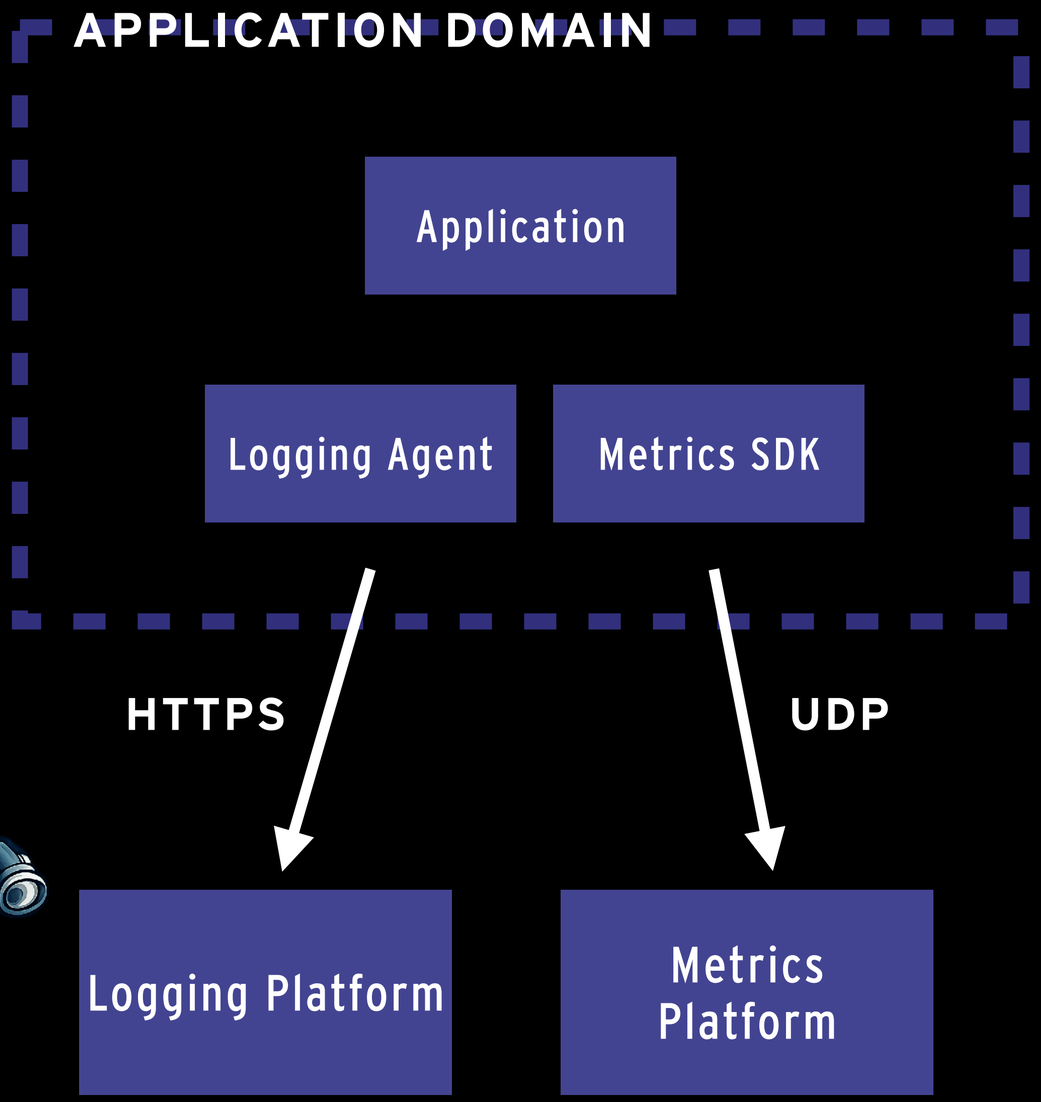
APIS

DASHBOARDS

Team DreamMachine

- Looking in…
  - Incident resolution
  - Meeting SLAs by observing SLIs
  - Target improvements
- Looking out…
  - Understanding customer experience
  - Innovate to meet customer needs
  - Fix before it becomes a problem

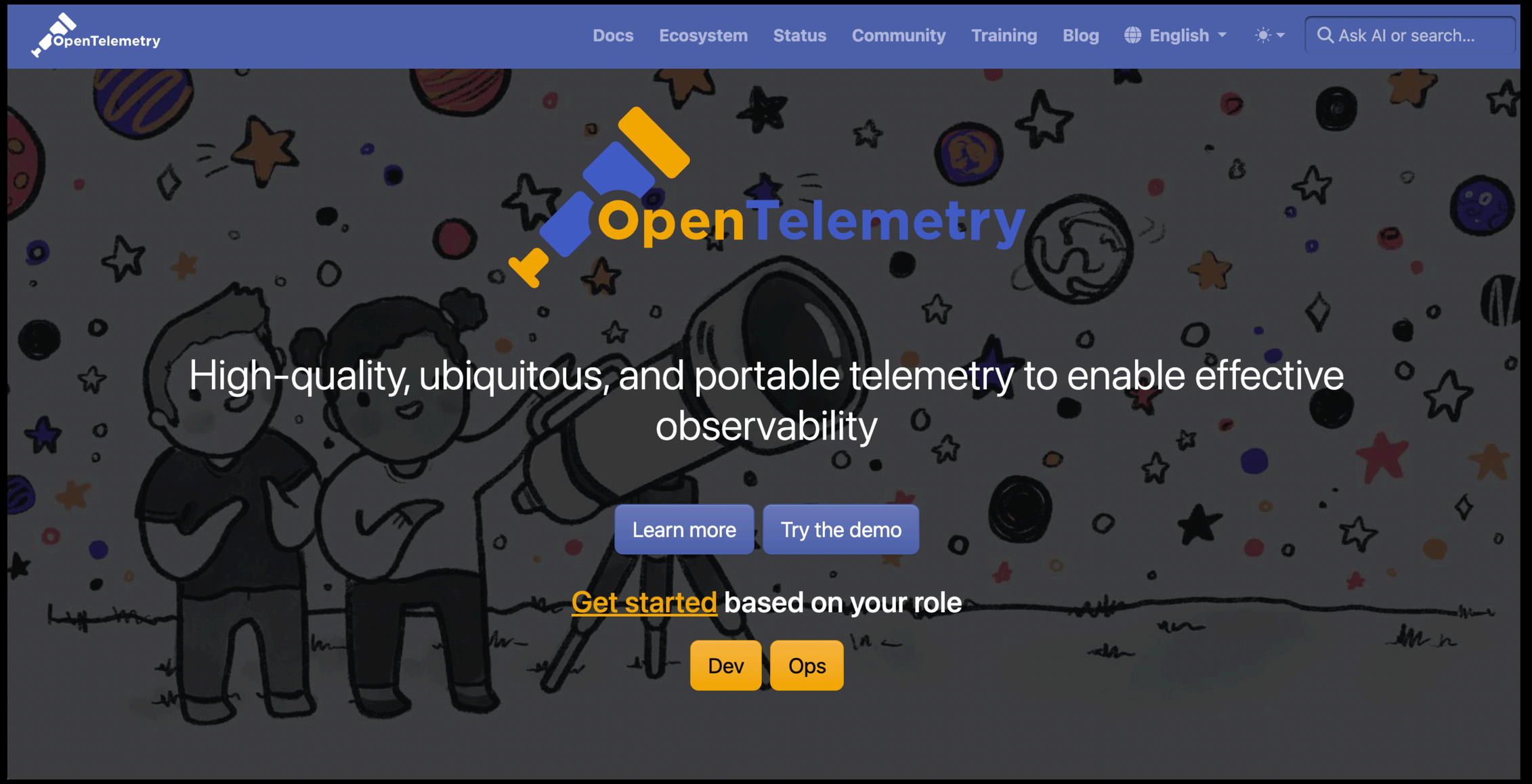
# PLUMBING IS NOT!




- Costly
  - May require multiple vendors
  - License costs
  - Risk of vendor lock-in
- Complexity
  - Configuration explosion
  - Unreasonable skill expectations
  - Challenging to debug
- Reliability
  - Misbehaving agents
  - Brittle infrastructure

# OPEN TELEMETRY

# INTRODUCING OPENTELEMETRY



# OPENTELEMETRY - VISION

*“Telemetry… raw observational data streaming out of software applications”*

Telemetry Should Be Easy

Telemetry Should Be Universal

Telemetry Should Vendor-Neutral

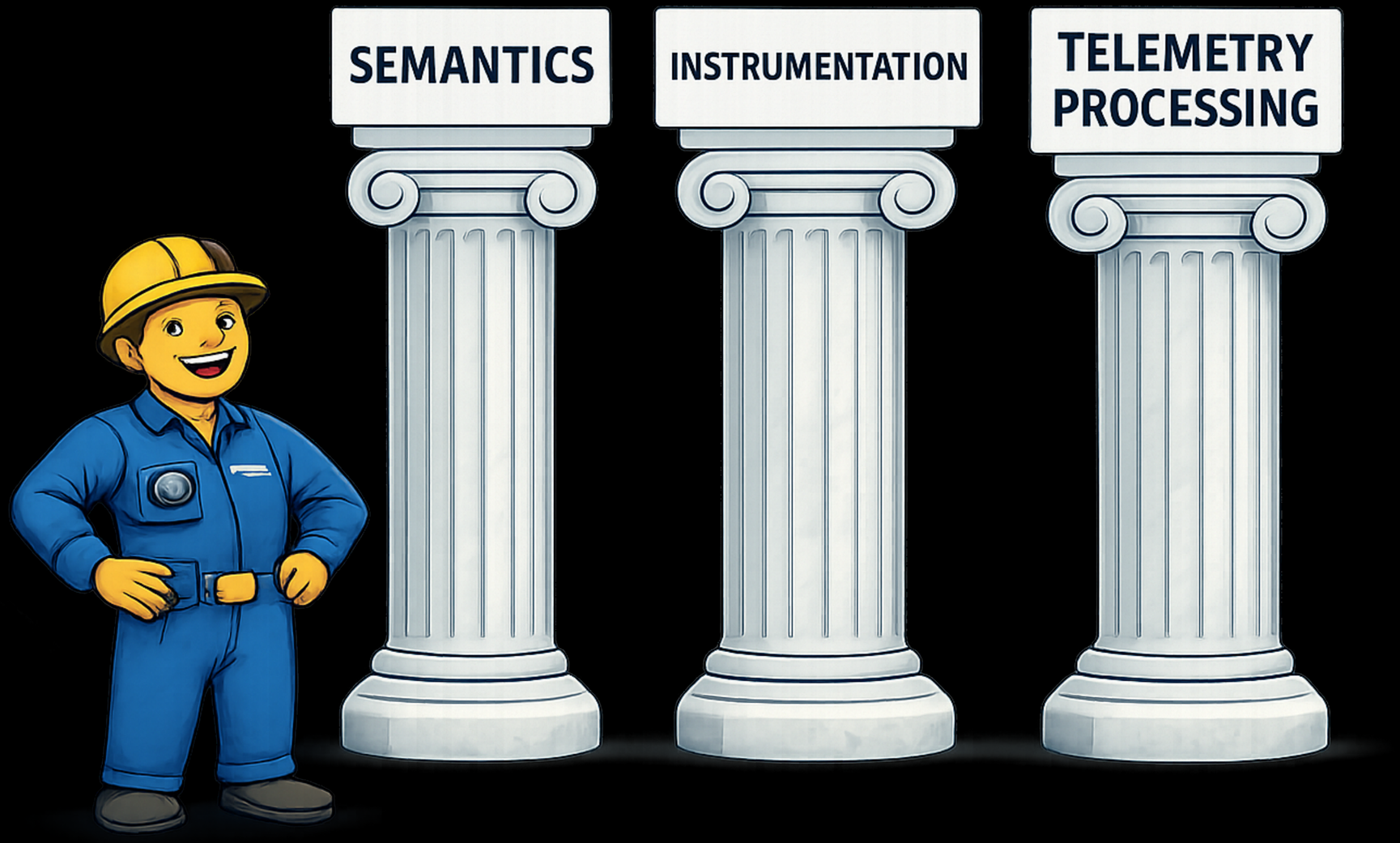
Telemetry Should Loosely Coupled

Telemetry Should Be Built-In

# OPENTELEMETRY CONTRIBUTORS

| Organization | Total contributions |
| --- | --- |
| Splunk Inc. | 34,550 - 17% |
| Elasticsearch Inc | 25,081 - 12% |
| Microsoft Corporation | 23,304 - 11% |
| Hound Technology Inc | 13,921 - 7% |
| Grafana Labs | 13,033 - 6% |
| Datadog, Inc. | 9,440 - 5% |
| Dynatrace, Inc. | 6,590 - 3% |
| New Relic, Inc. | 4,701 - 2% |
| Cloud Native Computing Foundation | 4,485 - 2% |
| Snowflake Inc. | 3,938 - 2% |

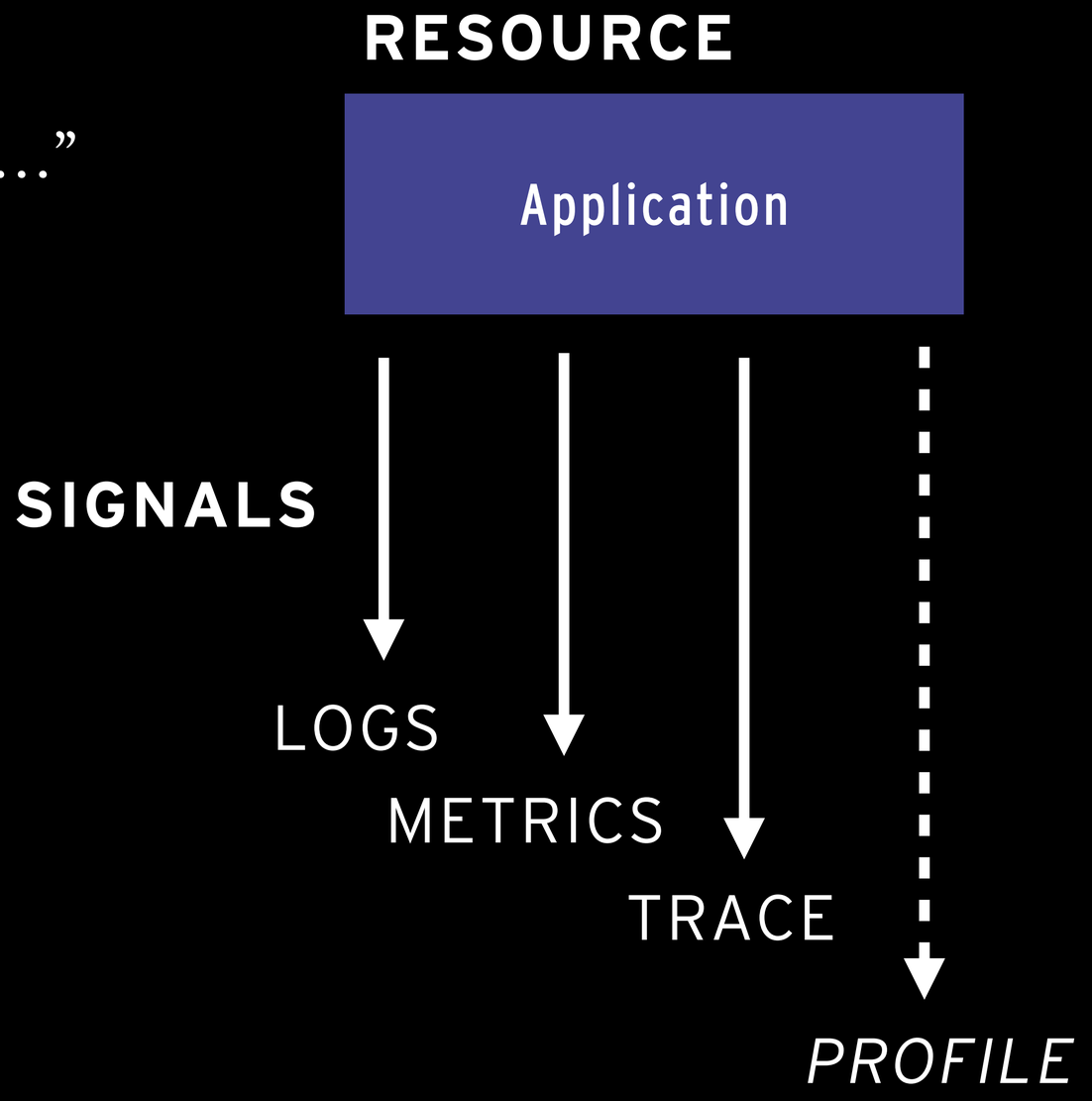
# THE THREE PILLARS OF OPENTELEMETRY



# SEMANTICS

# SEMANTICS - NAMING THINGS MATTERS!

- Signals
  - “Outputs that…describe underlying activity…”
  - Logs, metrics and trace
  - Extendable - baggage, events, profiling
- Resource
  - Entity producing signals
- Resource attributes
  - Identify the resource



# RESOURCE ATTRIBUTES

- “Weather Producer” gathers readings
  - Quarkus application
  - Deployed in clusters
  - Resource attributes identify a resource
- Required:
  - “service.name”
- Recommended:
  - “service.instance.id"
  - “service.version”
  - “deployment.environment”

QUARKUS

```
service.name = weather-producer
service.instance.id = weather-producer-1
location = stockholm
team = weathermen
deployment.environment = prod
```

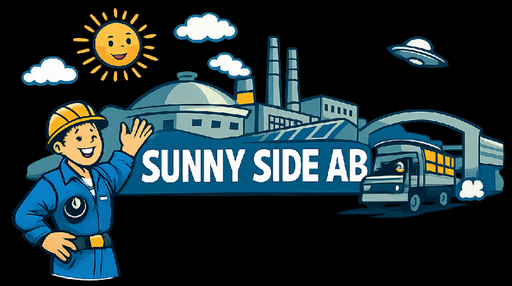
# RESOURCE ATTRIBUTES

Team Weathermen

```
service.name = weather-producer
service.instance.id = weather-producer-2
location = oslo
team = weathermen
```

Team Observability

Grafana




Team DreamMachine

```
service.name = weather-consumer
service.instance.id = weather-consumer-1
team = backendboys
```

kafka

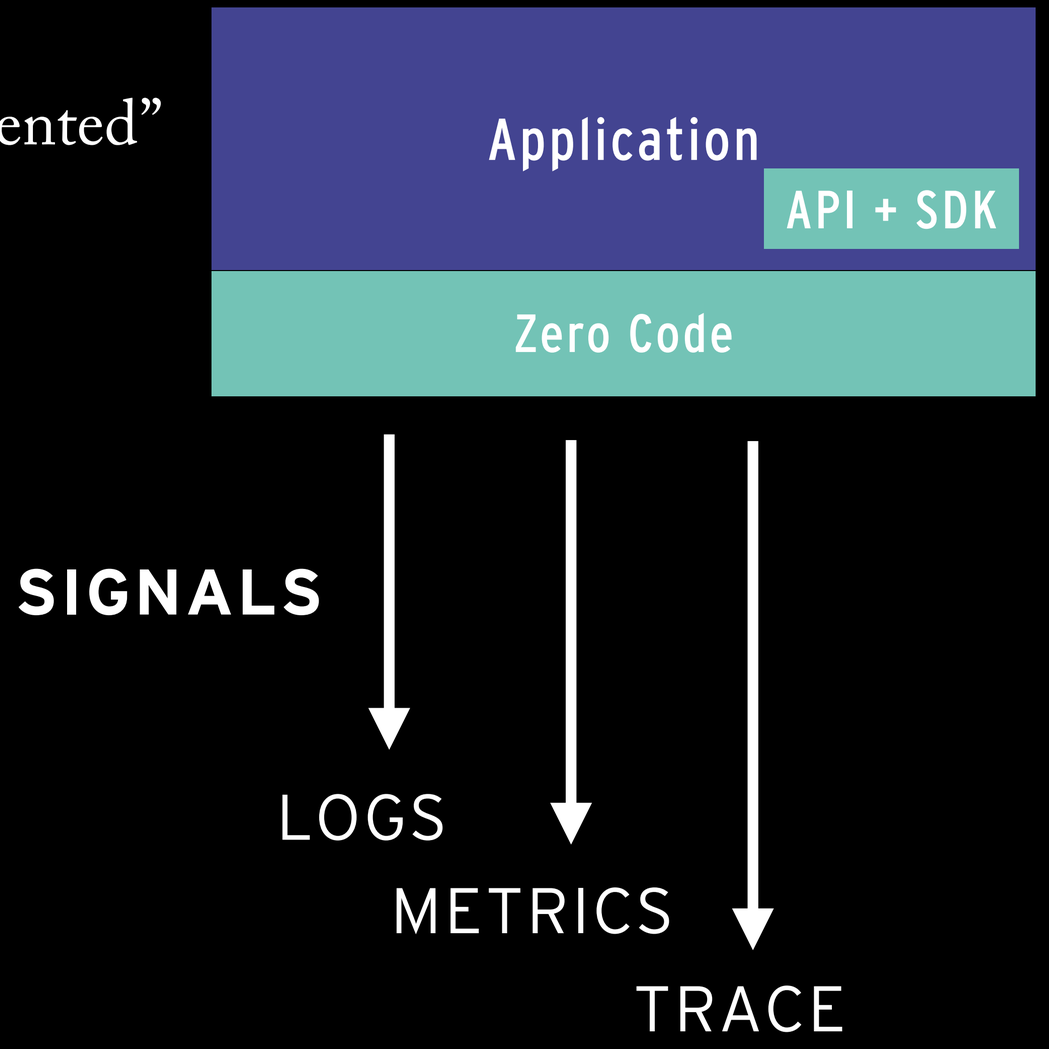
Team Kafka

```
service.name = kraft
service.instance.id = kraft-1
team = kafka
```

# INSTRUMENTATION

# INSTRUMENTATION

- “For a system to be observable, it must be instrumented”
- “Telemetry should be built in”
  - Zero Code instrumentation
- “Telemetry should be easy”
  - Code based solutions
  - APIs and SDKs familiar to developers
- “Telemetry should be universal”
  - Works across languages & frameworks
- This is the difficult bit!



# INSTRUMENTING WEATHER PRODUCERS

- Framework support
  - Add dependencies
  - Activate via configuration

```
dependencies {

    // Quarkus platform
     implementation(…)

     // OpenTelemetry Support
     implementation("io.quarkus:quarkus-opentelemetry")
```


- Code based customisation
  - APIs define signals
  - SDK implementations

```
import io.opentelemetry.api.metrics.LongCounter;
import io.opentelemetry.api.metrics.Meter;

@Inject
public WeatherService(Meter meter) {
    this.weatherEventsCounter =
      meter.counterBuilder(“weather.events")
            .setDescription("Total weather events")
            .setUnit("1") // No physical unit
            .build();
```

# INSTRUMENTING THE KAFKA BROKERS

- Kafka Brokers are Java applicatons
  - Instrument with the OTEL Java Agent
  - Zero Code instrumentation
- Captures telemetry at the “edges”
  - Client requests are observable
  - Internal workings not observable
- Results may be limited
  - Great for logs
  - Not so great for metrics
- Check your support agreements!


**docker-compose.yaml**

```
broker2:
  image: apache/kafka:4.1.0
  environment:
    OTEL_RESOURCE_ATTRIBUTES: service.name=broker,
                              service.instance.id=broker-2…
    KAFKA_OPTS: ”-javaagent:opentelemetry-javaagent.jar,
                 -Dotel.metrics.exporter=otlp
```

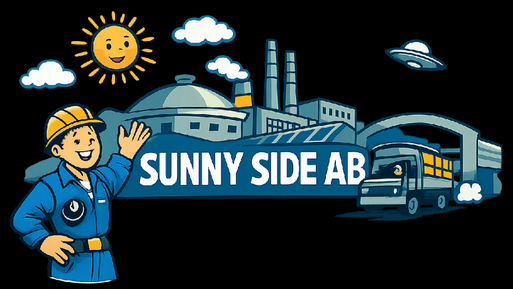
# WHAT IS OBSERVABLE?

QUARKUS

Team Weathermen

Grafana

?




Team DreamMachine

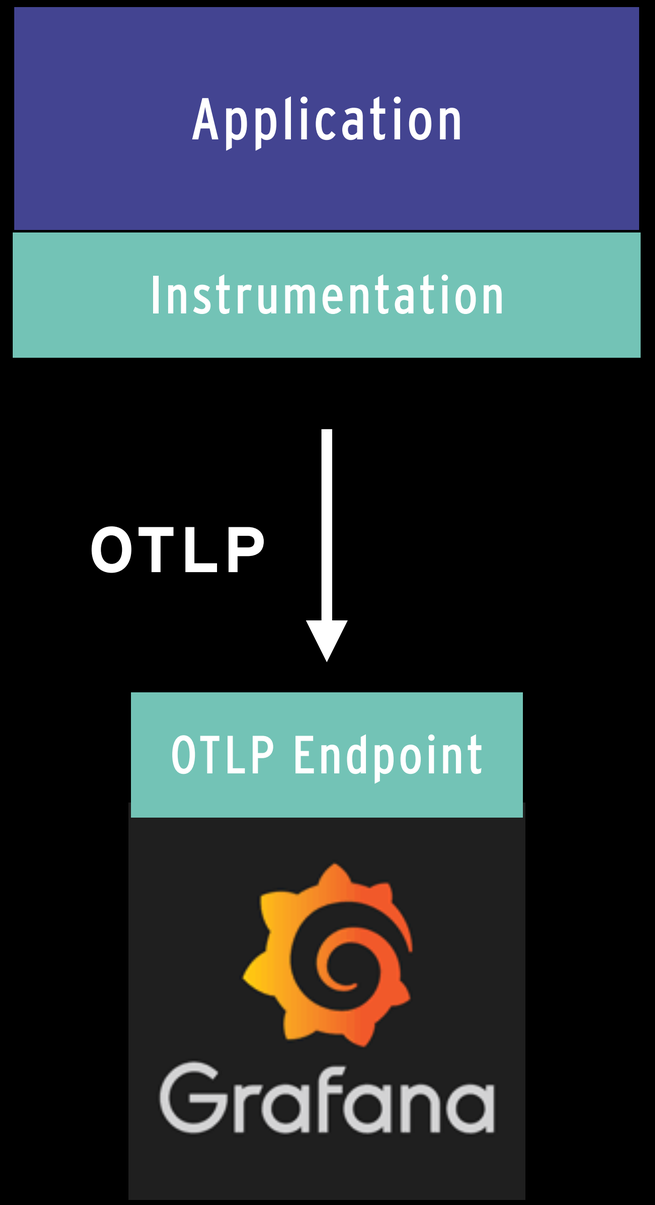
kafka

Team Kafka

# TELEMETRY PROCESSING

# OPEN TELEMETRY PROTOCOL - OTLP

- Standard for transporting signals
  - Schemas for all signal types
  - Payloads endoded with Protobuf
  - Transported over gRPC or Http
- “Telemetry should be universal”
  - Consistent across languages and applications
- “Telemetry should be easy”
  - Eliminates custom adapters for each application
- “Telemetry should be vendor-neutral”
  - Connect any OTLP exporter to any OTLP endpoint

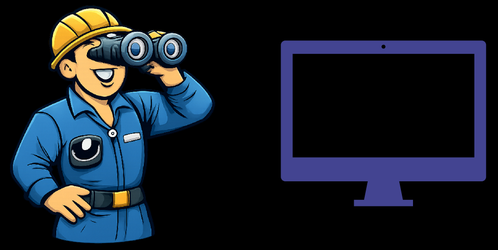


# DIRECT EXPORT PATTERN

QUARKUS

OTLP

Grafana



```
% quarkus dev
```

OTLP

Grafana

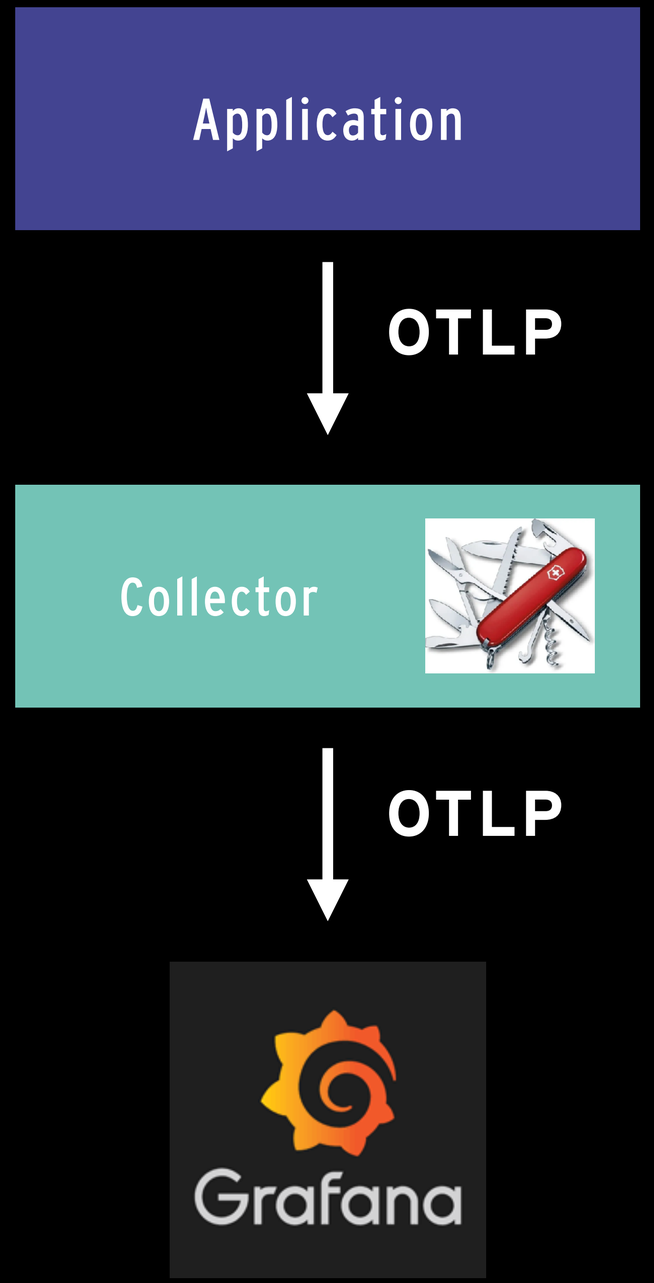
*localhost:62839*

**TESTCONTAINERS**

# INTRODUCING THE COLLECTOR

- Swiss army knife of observability
- Receives, processes and exports signals
- “Telemetry should be easy”
  - Configurable application
- “Telemetry should be universal”
  - Use with any platform
- “Telemetry should be vendor neutral”
  - Choose your distribution (*)
  - Compatible and interchangeable
- “Telemetry should be loosely coupled”
  - Decouple your observability platform

```
receivers:
  - otlp
processors:
  - batch
  - filter
exporters:
  - otlp
  - debug
```



# RECEIVERS - ENABLING YOUR TELEMETRY!


Collector

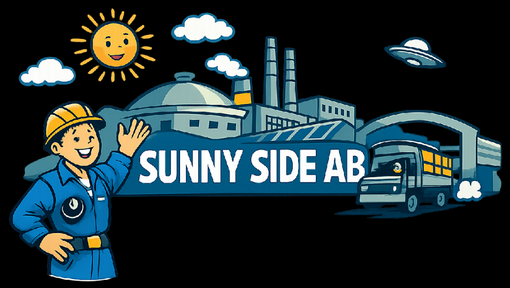
```
receivers:
  - otlp
```

```
receivers:
  - kafkametrics
```

(Kafka Admin Client)

kafka

```
receivers:
  - jmx
```




```
receivers:
  - prometheus
```

```
receivers:
  - filelog
```

LOG4J™

MICROMETER application observability

*/metrics/prometheus*

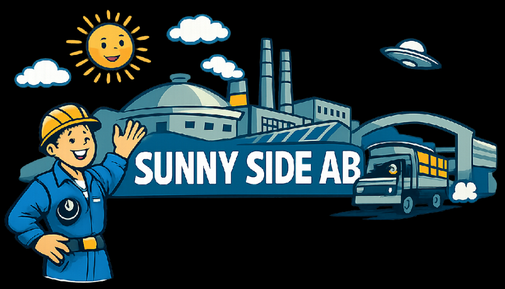
# NOW EVERYTHING IS OBSERVABLE!

QUARKUS
Team Weathermen

OTLP Collector

Grafana

Prometheus Collector




Team DreamMachine

Kafka Collector

Filelog Collector

kafka

Team Kafka

# PROCESSORS - SHAPING YOUR SIGNALS

- Transform and enrich telemetry data
- Enforce standards and governance
- Filter out low-value signals
- Batch and buffer data
- Allow teams to take control of their signals

```
resource:
  attributes:
    - key: team
      value: kafka
      action: upsert

filter:
  metrics:
    metric:
      - 'isMatch(name, "http.*")'

batch:
  send_batch_size: 8192
  send_batch_max_size: 16384
  timeout:5s
```

# AGGREGATOR PATTERN

QUARKUS

QUARKUS

QUARKUS

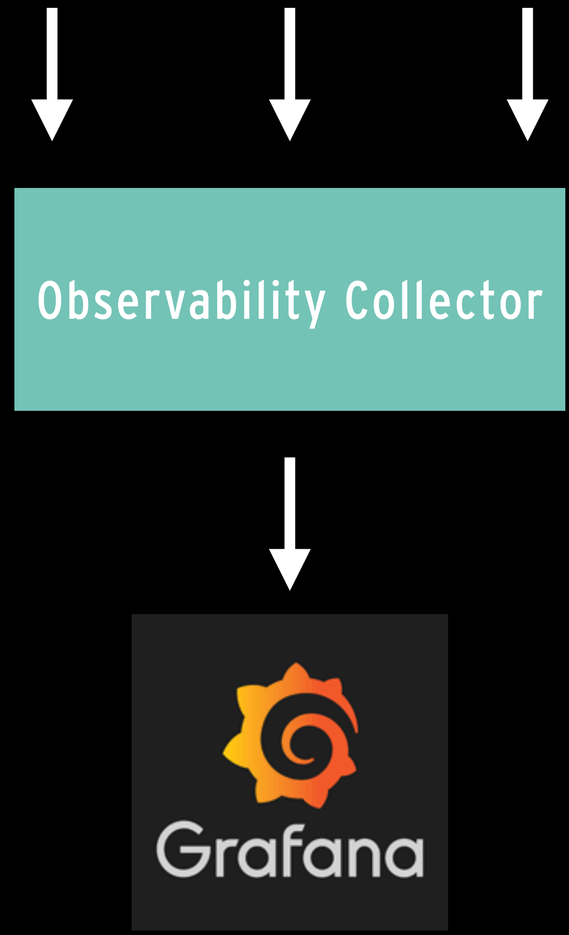
```
processors:
  - resource
  - filter
  - batch
```

OTLP Collector

Team Weathermen

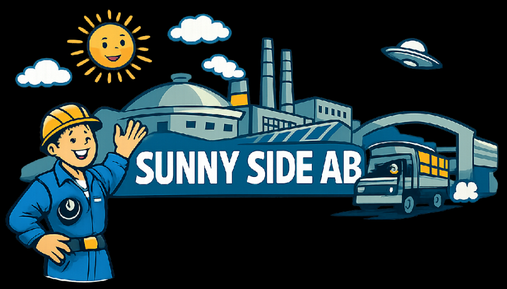
# GATEWAY PATTERN

- All Telemetry Passes Through A Collector
- Enforce Standards
  - Add Missing Resource Attributes
- Protect Observability Platform
  - Identify and throttle high volume users
- Handle Edge Cases
  - Mapping Resource Attributes to Prometheus Labels
- Allows for cost allocation per team



# NOW OBSERVABLE AND LOOSELY COUPLED






OTLP Collector

Observability Collector

Prometheus Collector

Kafka Collector

Filelog Collector

kafka

# WHAT HAVE WE ACHIEVED?

# DID WE SOLVE ANYTHING?

- Have we reduced cost?
  - Significant initial investment
  - No vendor lock-in
- Have we reduced complexity?
  - Common language enables standards
  - Responsibilities assigned
  - Reduced knowledge burden
- Have we improved reliability?
  - Collectors enable ownership of telemetry
  - Collectors are also monitorable

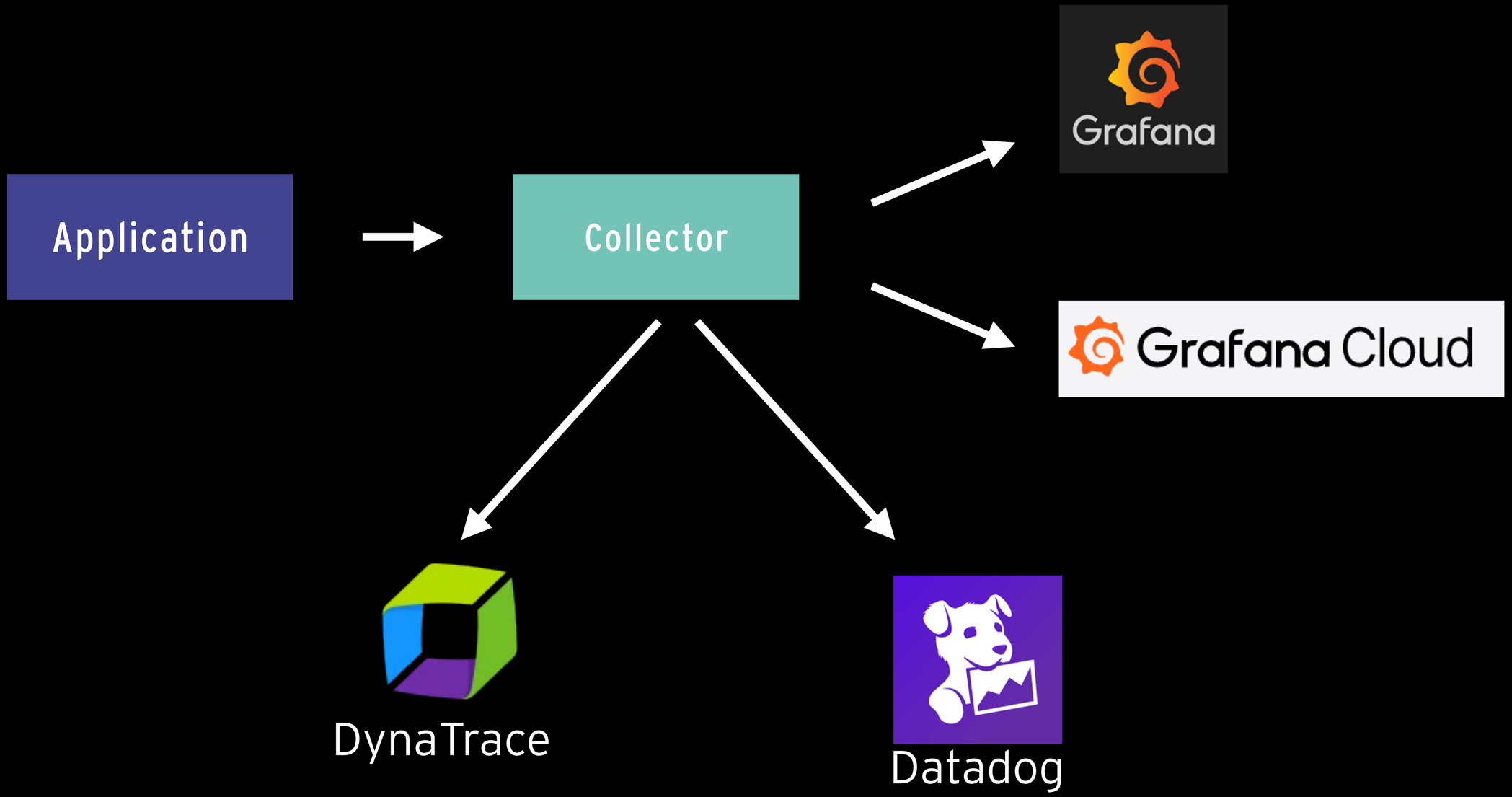

Team Weathermen

OTLP Collector

Observability Collector

Team Observability

# TRY A NEW PLATFORM?



# OBSERVE MORE


Application

Collector



# FURTHER READING

OpenTelemetry

https://opentelemetry.io