Getting Started with OpenTelemetry in Java

Agenda:

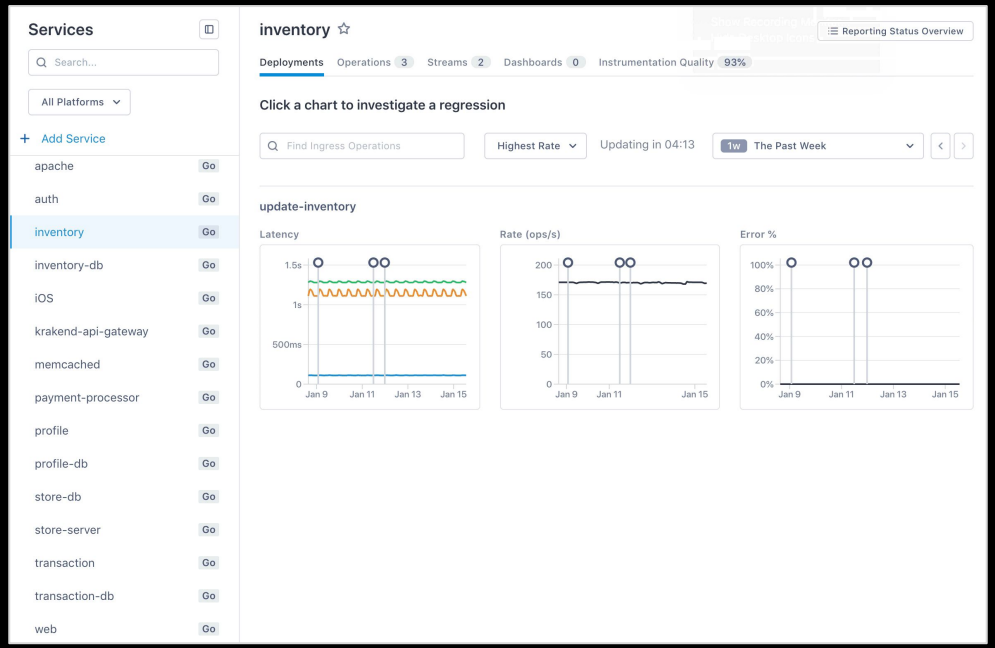
- 9:00am Introductions
  - What brought you to the workshop?
  - Setup LS account
- 9:10am OpenTelemetry Overview
- 9:30am Tutorial / code walkthrough

Handy Links:

- Account setup: https://bit.ly/otel-workshop
- Walkthrough code: https://github.com/tedsuo/otel-java-basics
- Java Launcher: https://github.com/lightstep/otel-launcher-java
- Quickstart Guide: https://opentelemetry.lightstep.com/java

Setup LS account

https://bit.ly/otel-workshop




Getting
Started
~with~
OpenTelemetry

te·lem·e·try
/tə'lem.ə.tri/
noun

The science or process of collecting information about objects that are far away and sending the information somewhere electronically.

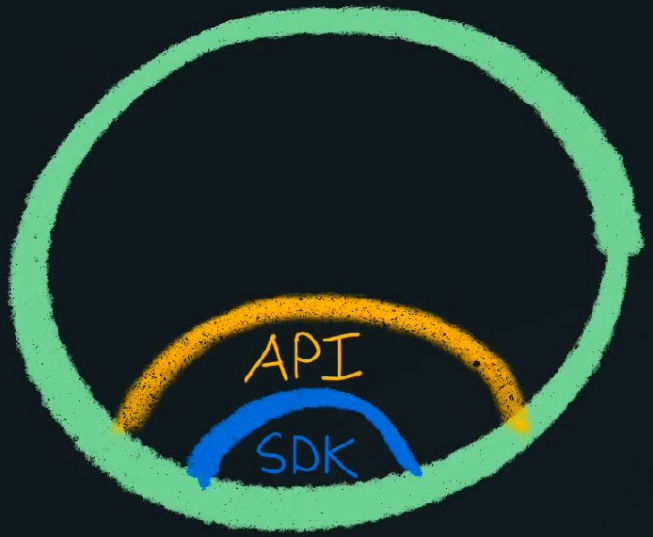
https://dictionary.cambridge.org/us/dictionary/english/telemetry

OpenTelemetry

the BIG pieces

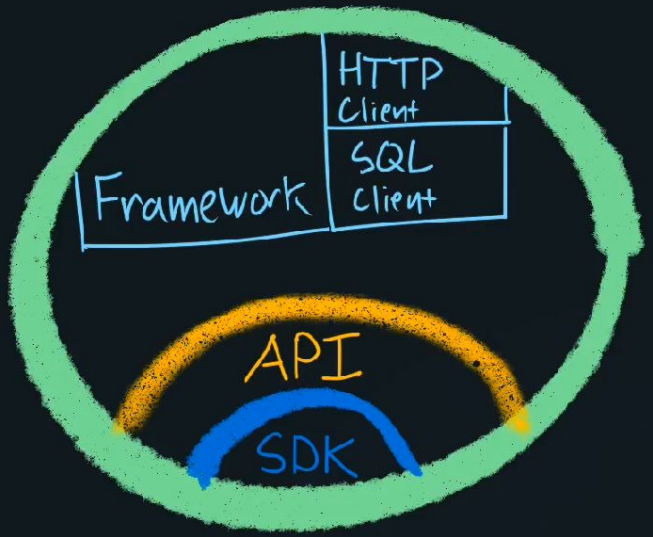
SDK

- Implementation
- Framework
  - Configuration
  - Plugins
  - Lifecycle hooks
- Access during program setup only.
- After setup, the SDK should not be directly accessed by application code.
- The SDK should NEVER be accessed by instrumentation code.



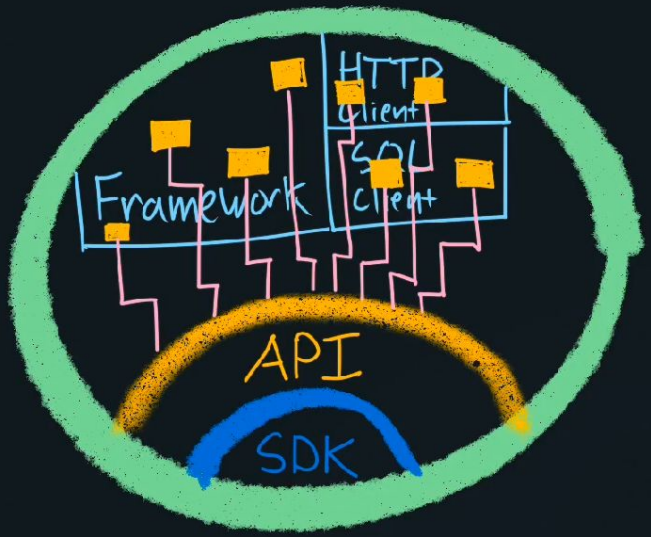
API

- Interfaces
- Data Standards
- Used for instrumentation
- Supports multiple implementations



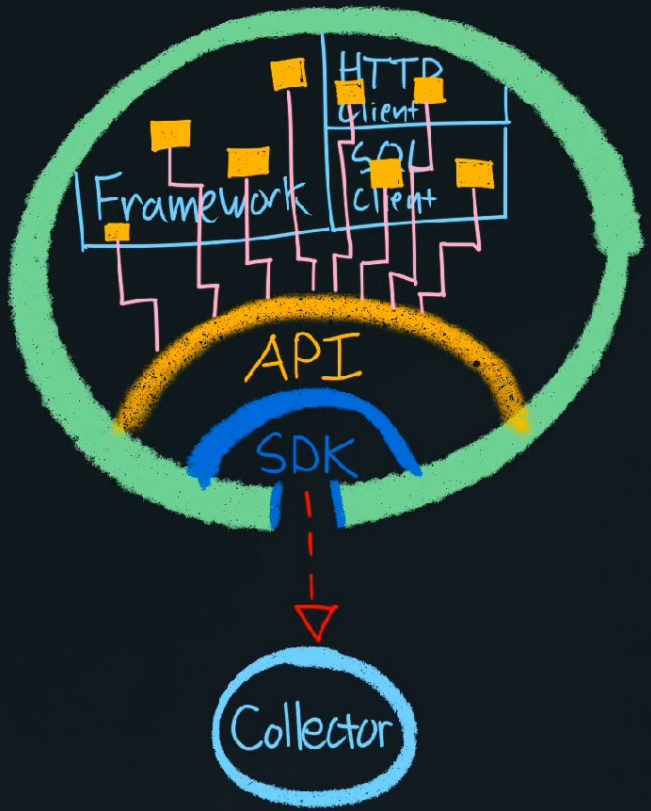
Frameworks and Libraries

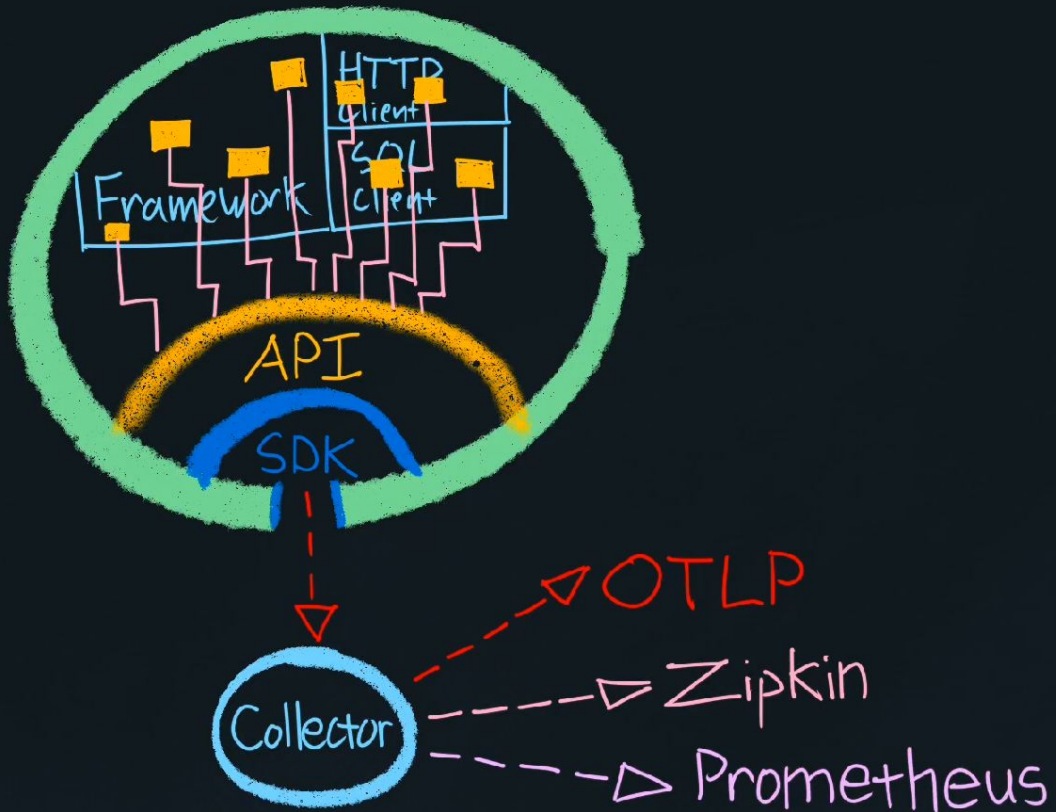
- Provide instrumentation
- Provide context propagation
- Instrumentation can be native or be a plugin installed by OpenTelemetry.

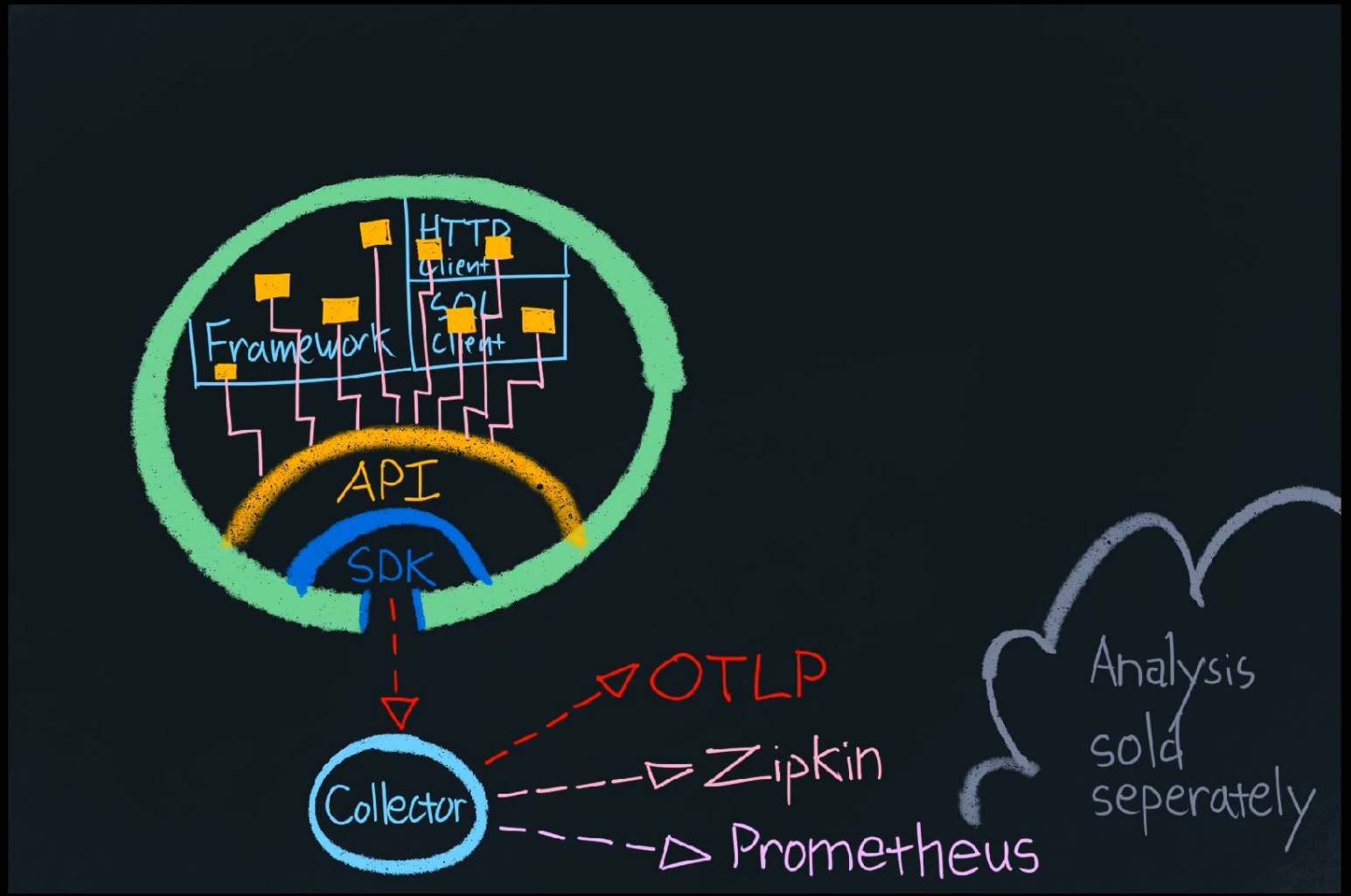


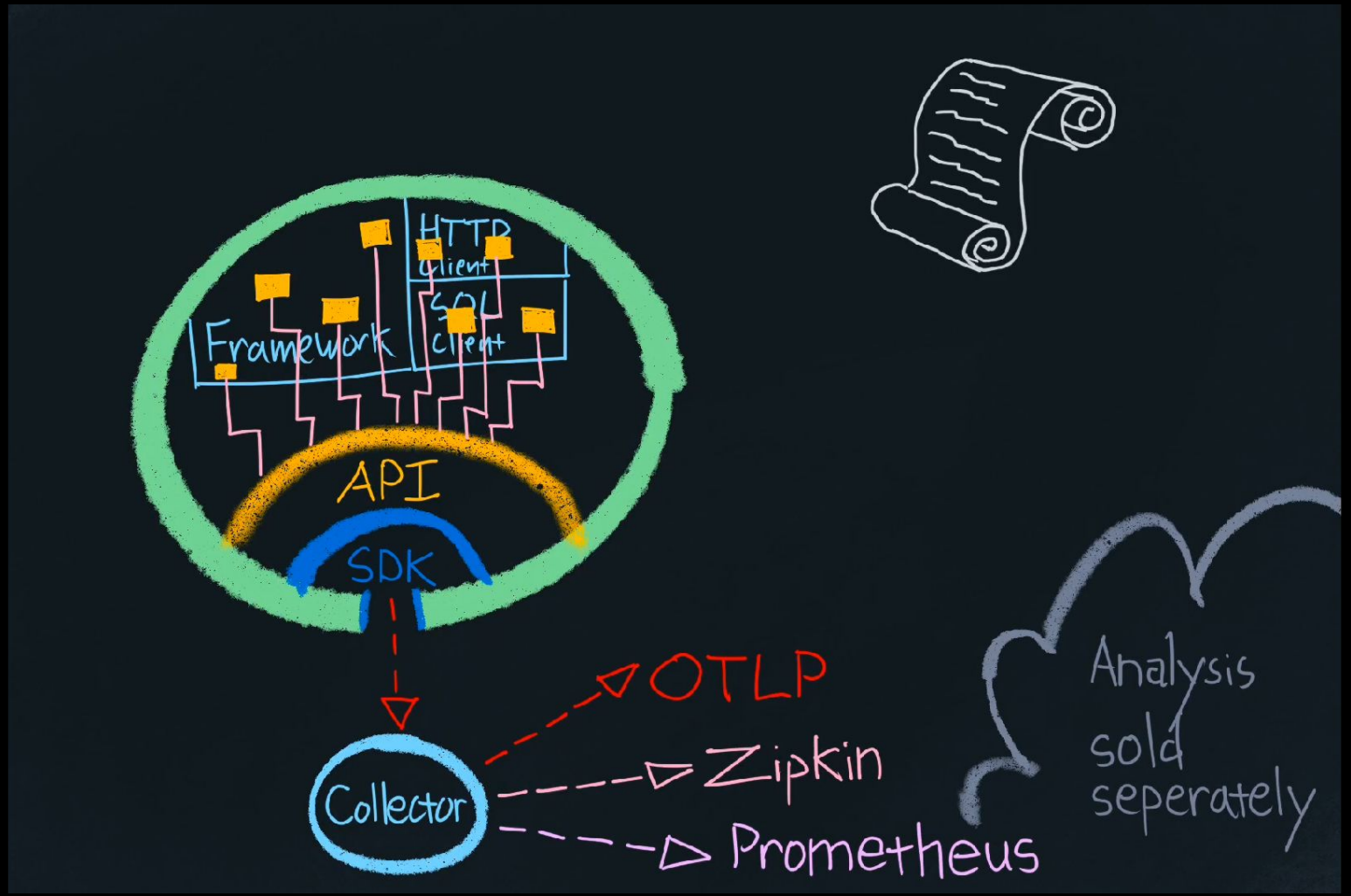
Frameworks and Libraries

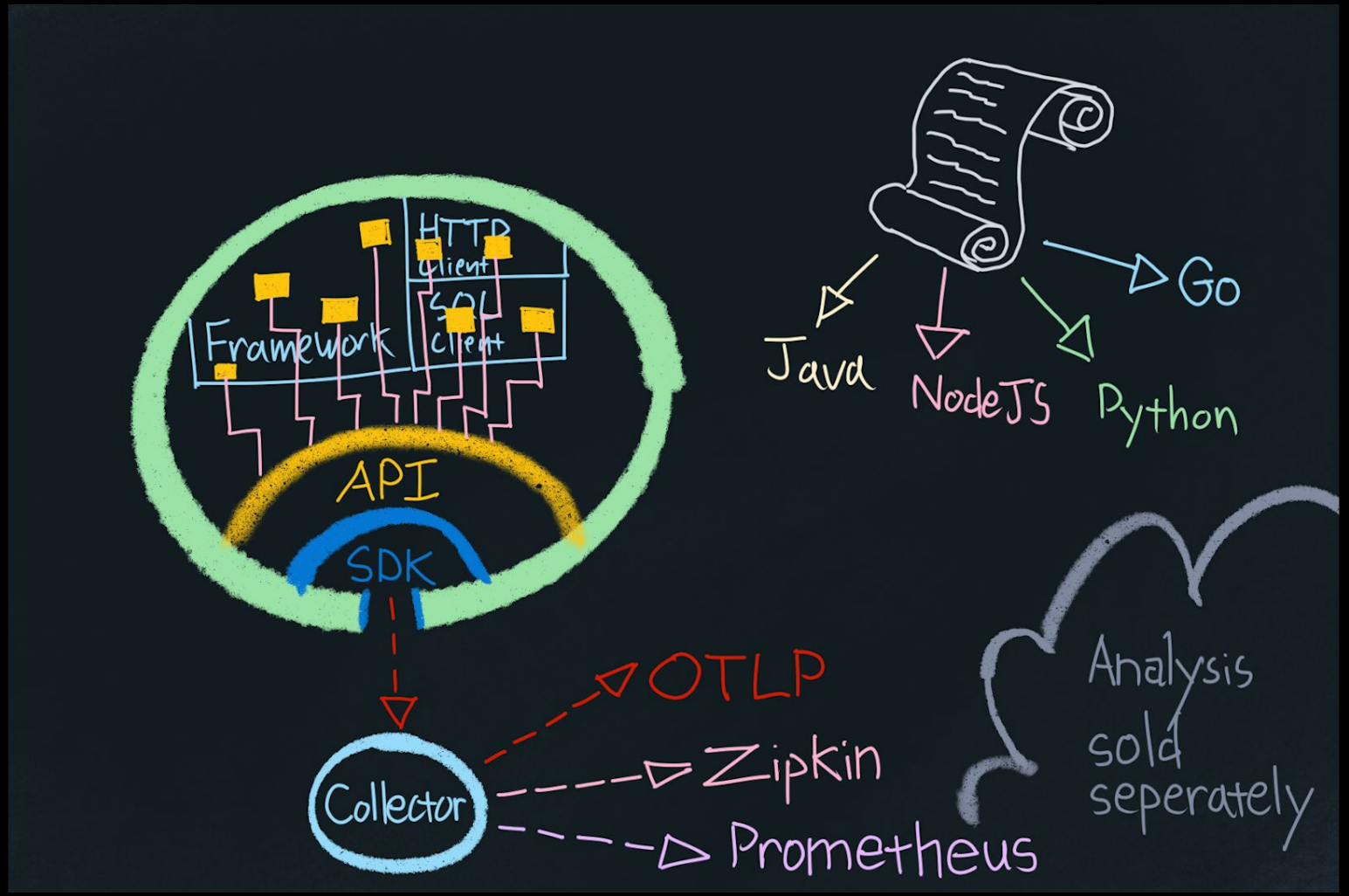
- Provide instrumentation
- Provide context propagation
- Instrumentation can be native or be a plugin installed by OpenTelemetry.


HTTP
client
SQL
client
Framework
API
SDK
Collector


HTTP client
SQL client
Framework
API
SDK
Collector
OTLP
Zipkin
Prometheus


HTTP client
SQL client
Framework
API
SDK
OTLP
Zipkin
Prometheus
Collector
Analysis
sold
seperately


HTTP client
SQL client
Framework
API
SDK
Collector
OTLP
Zipkin
Prometheus
Analysis
sold
seperately

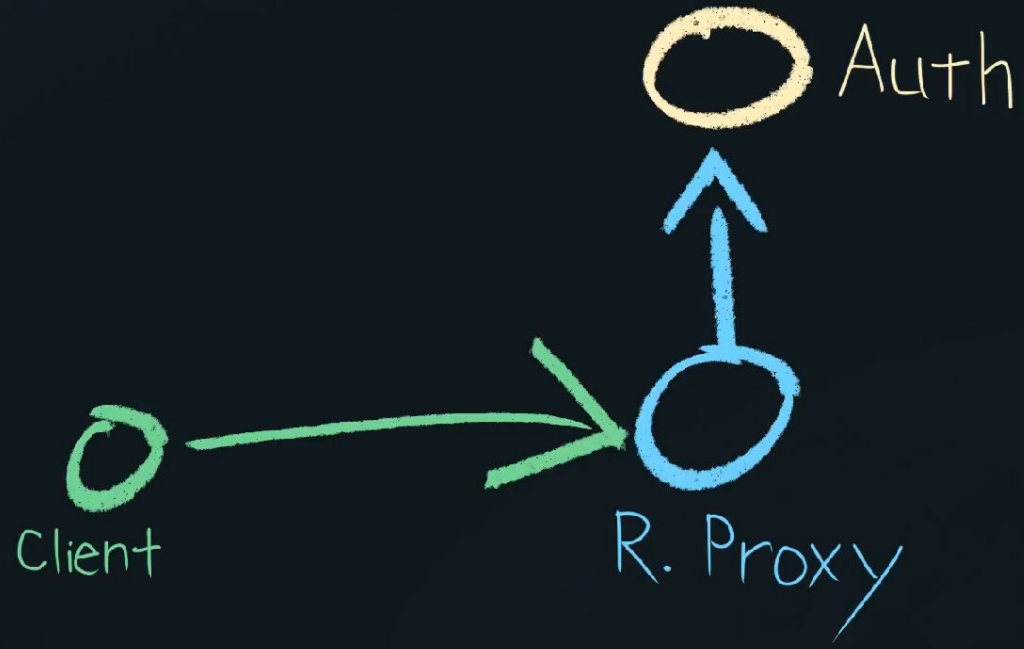

HTTP client
SQL client
Framework
API
SDK
Collector
OTLP
Zipkin
Prometheus
Java
NodeJS
Python
Go
Analysis sold seperately

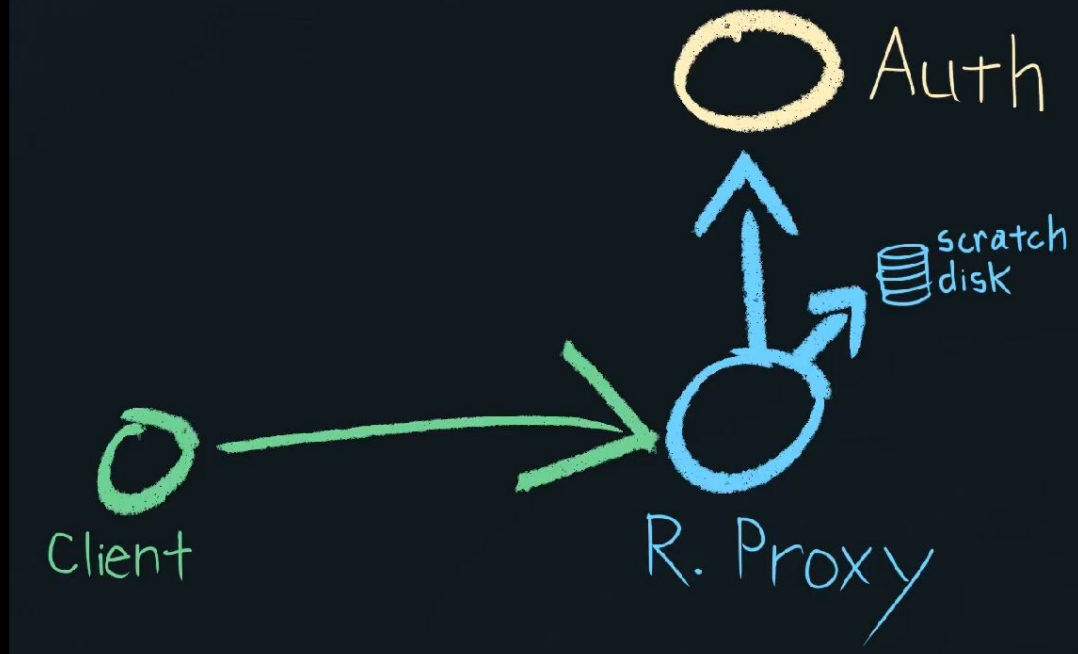
Agenda:

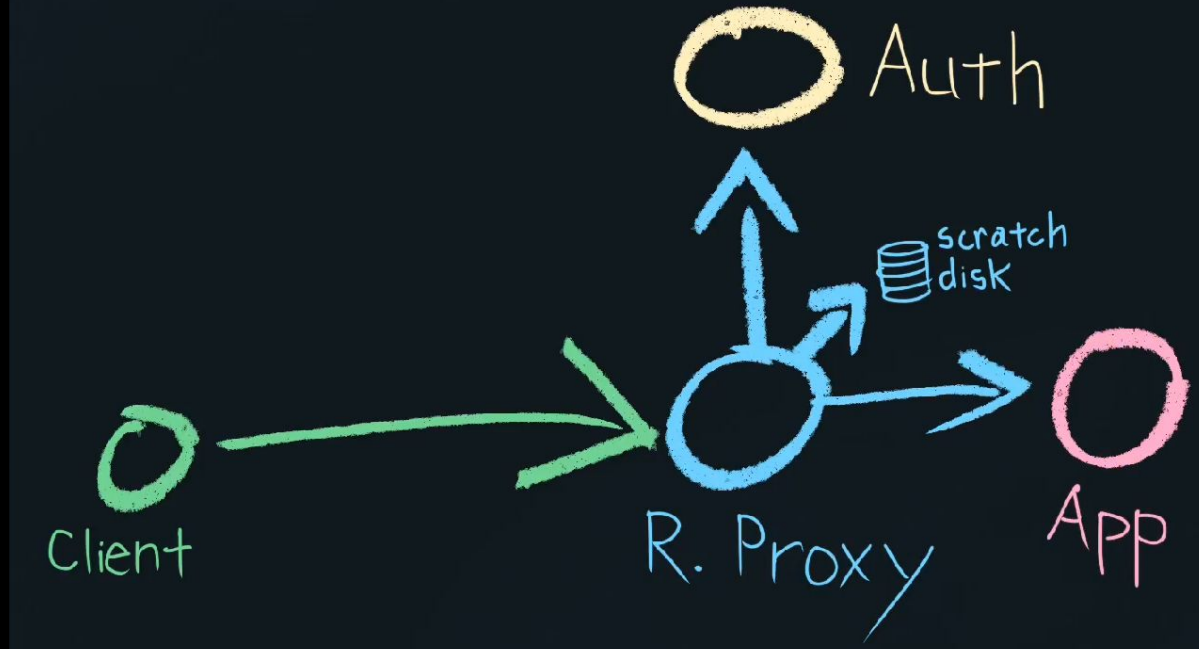
- Transactions
- Core Concepts
- Setup + Deploy

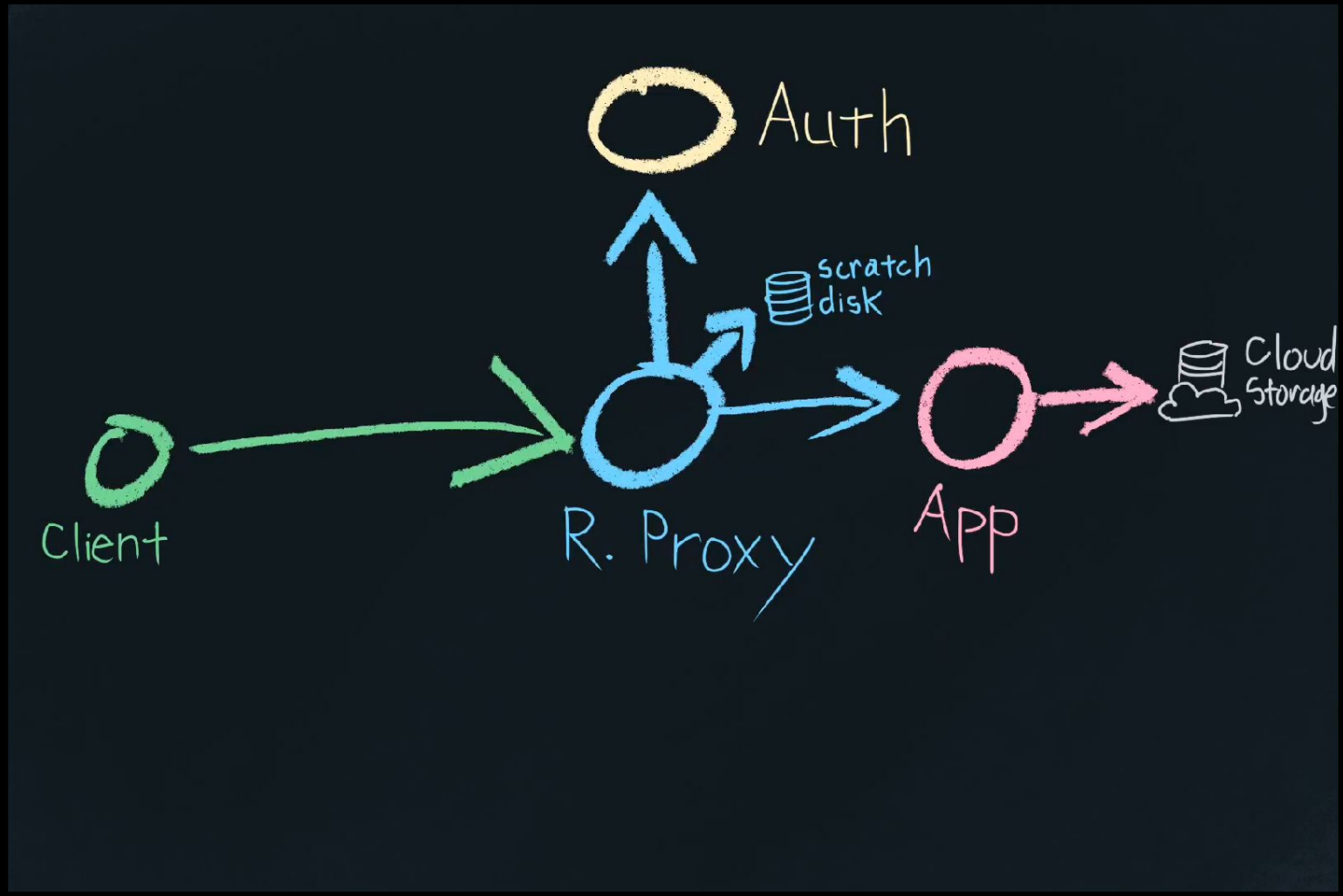

Client

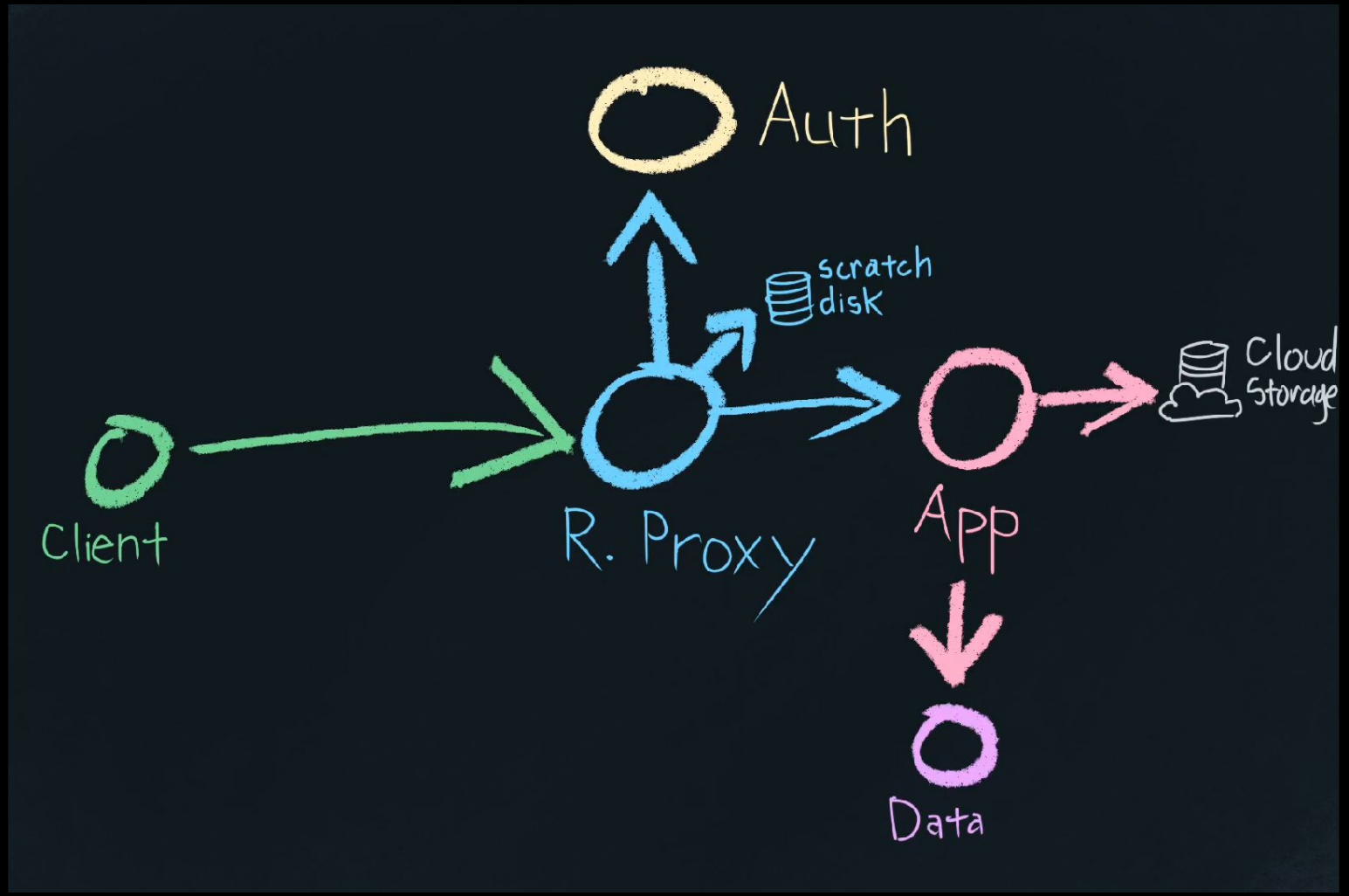

Client

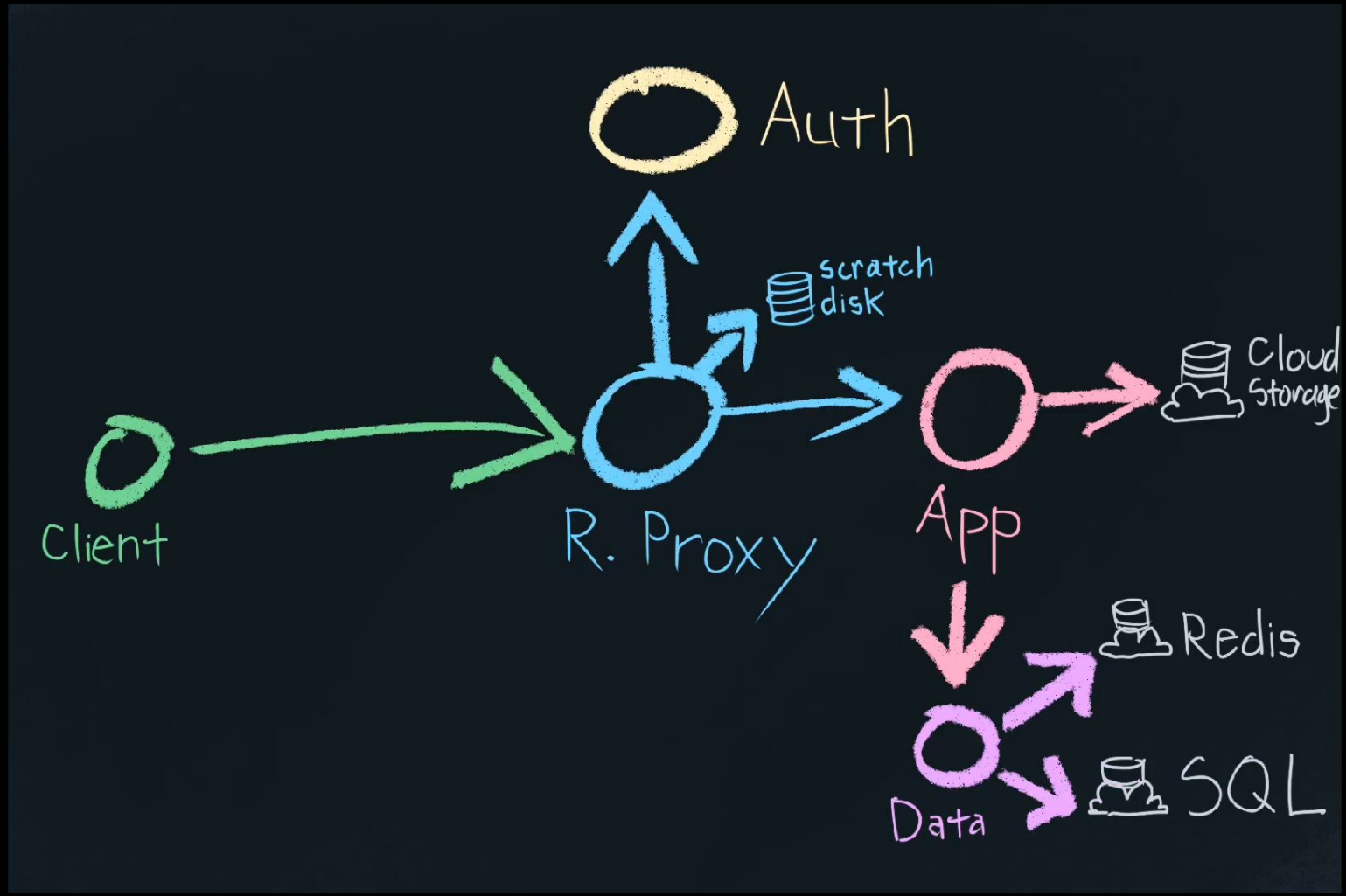

Auth
Client
R. Proxy

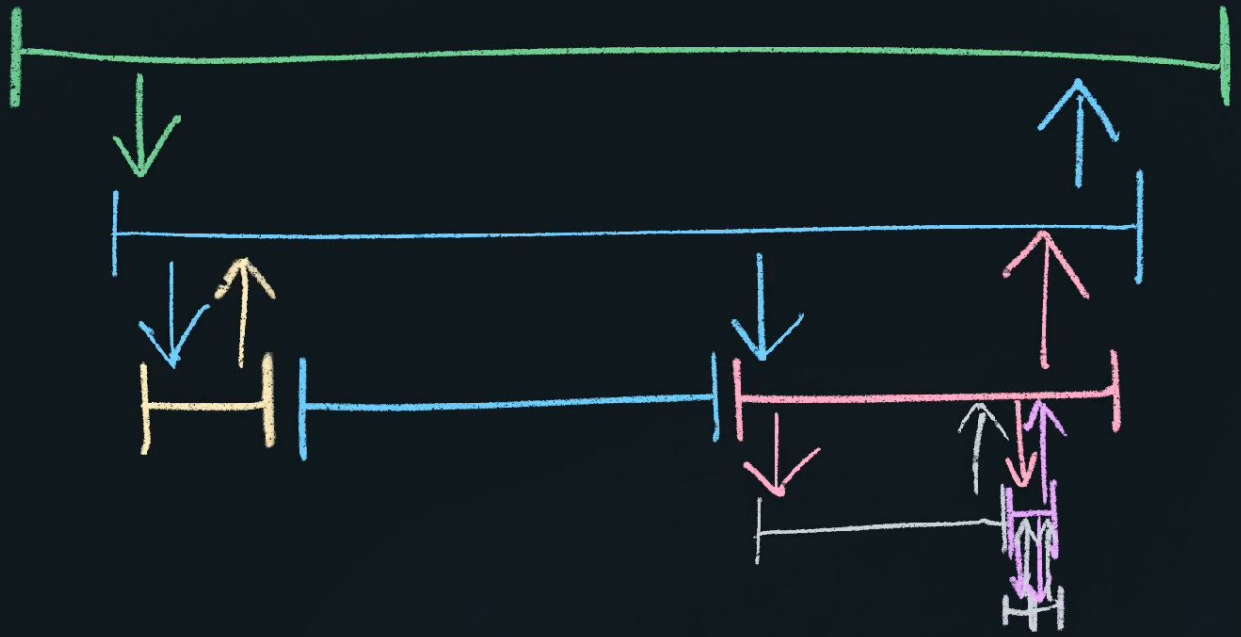

Auth
scratch
disk
Client
R. Proxy

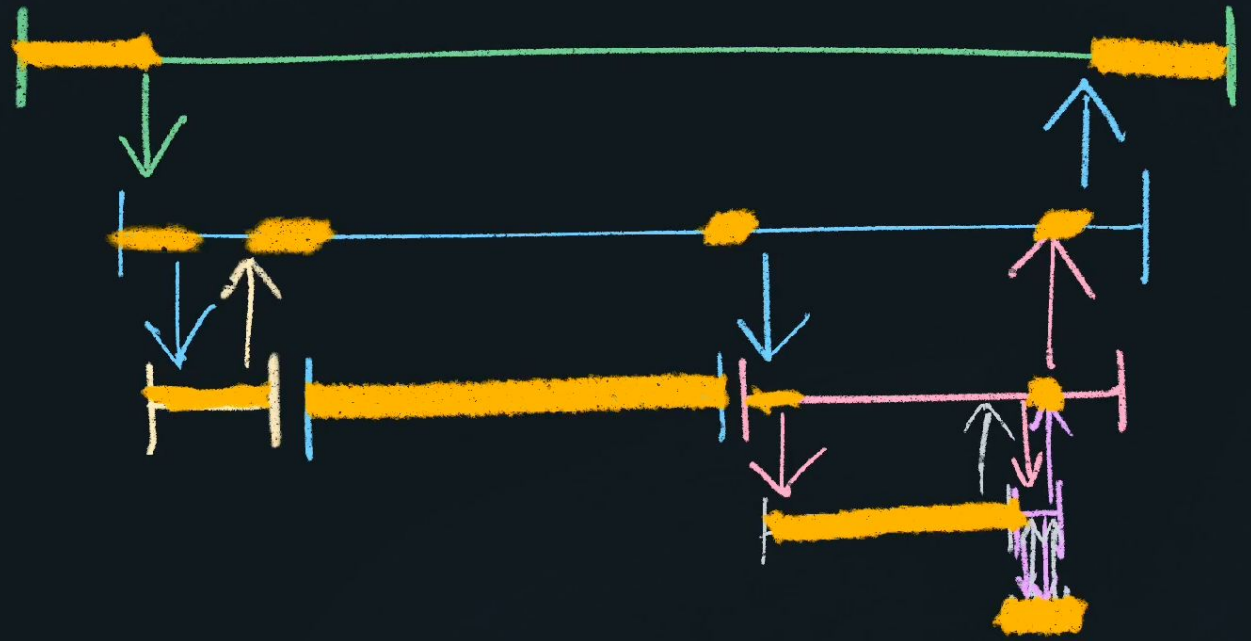

Auth
scratch
disk
Client
R. Proxy
App


Auth
scratch disk
Cloud Storage
Client
R. Proxy
App

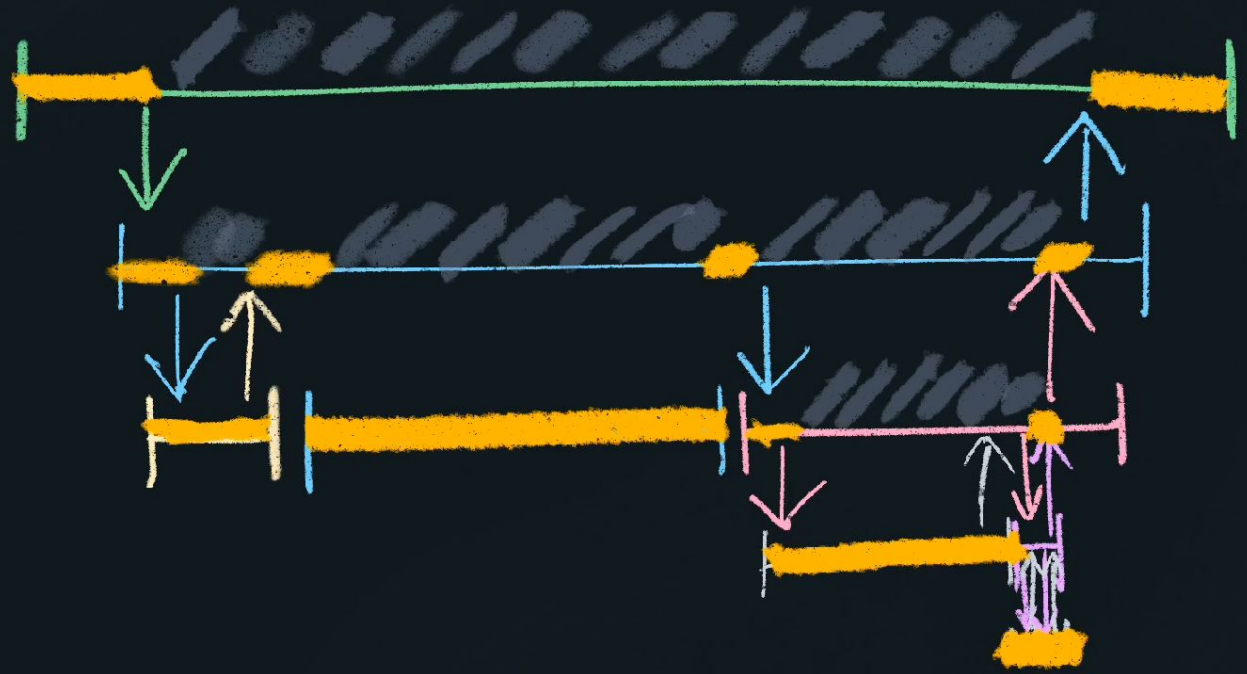

Auth
scratch disk
Cloud Storage
Client
R. Proxy
App
Data


Auth
scratch disk
Cloud Storage
Client
R. Proxy
App
Redis
Data
SQL

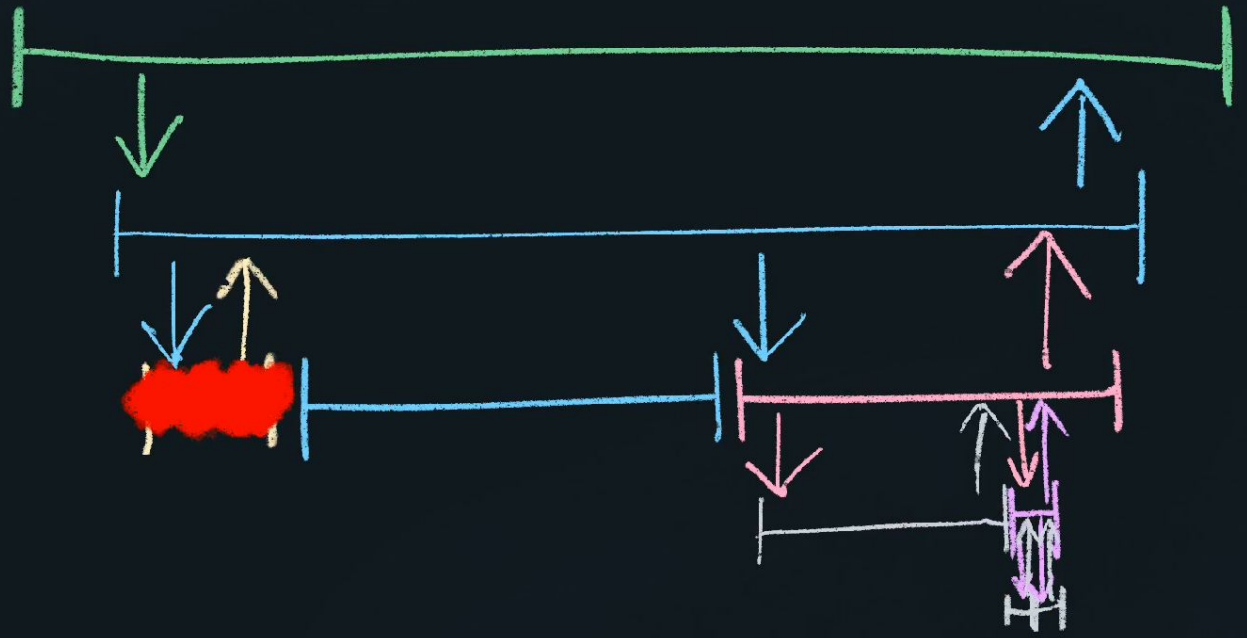




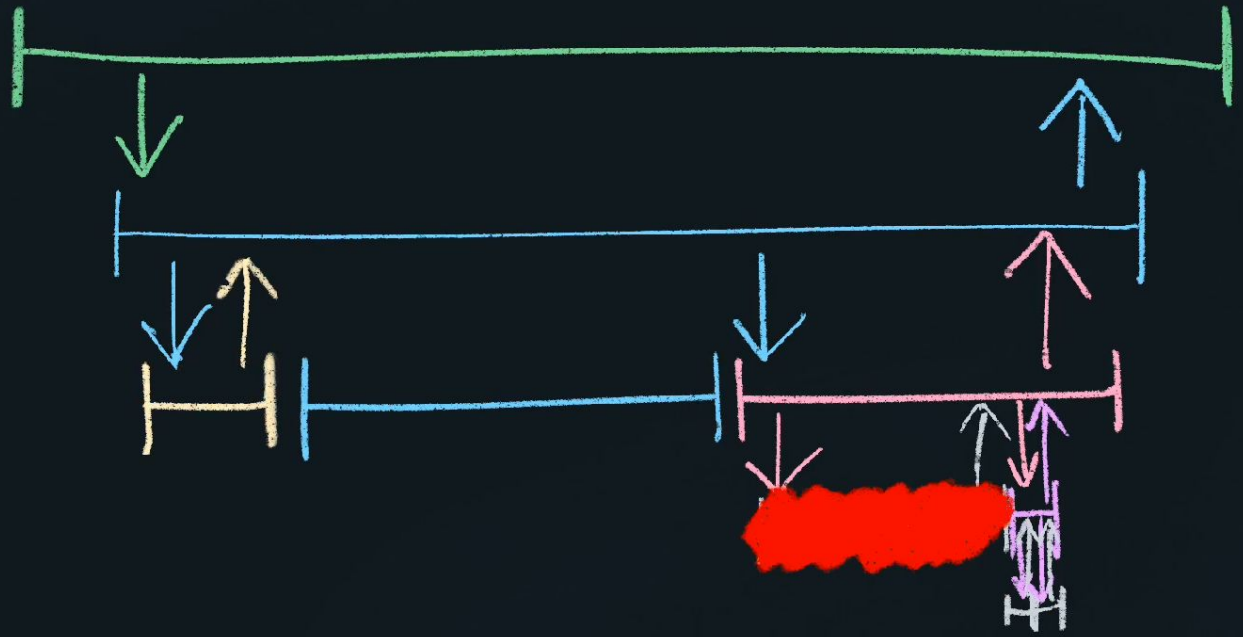
☆ Latency



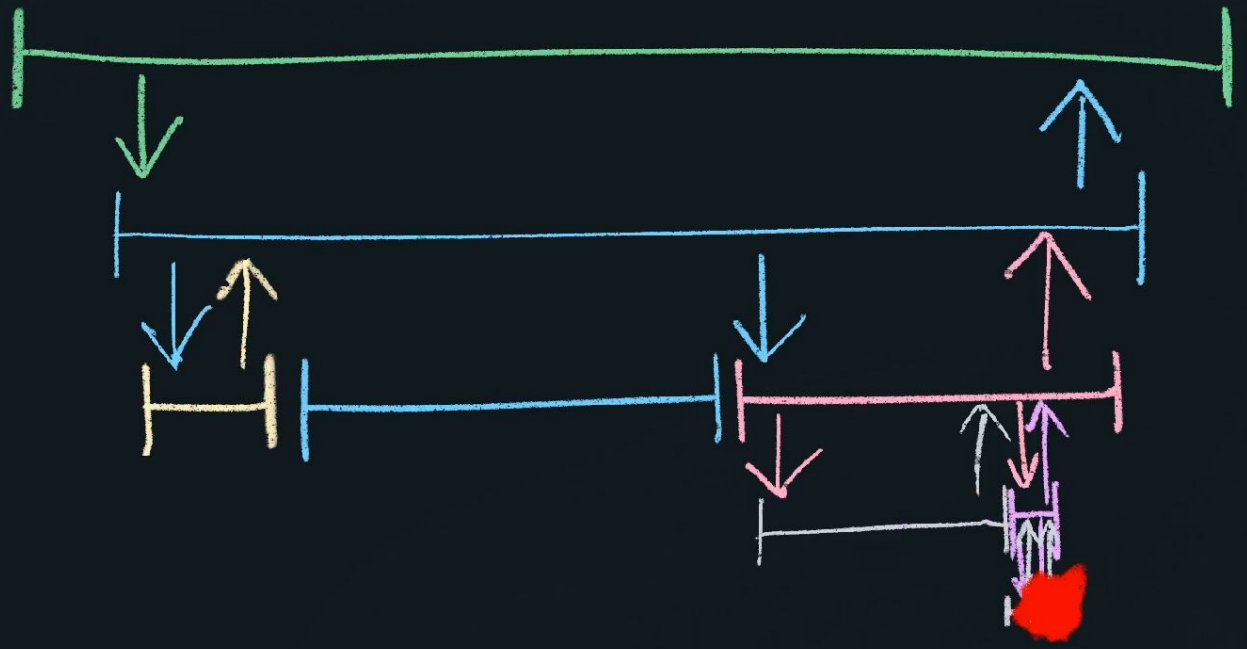
☆ Latency



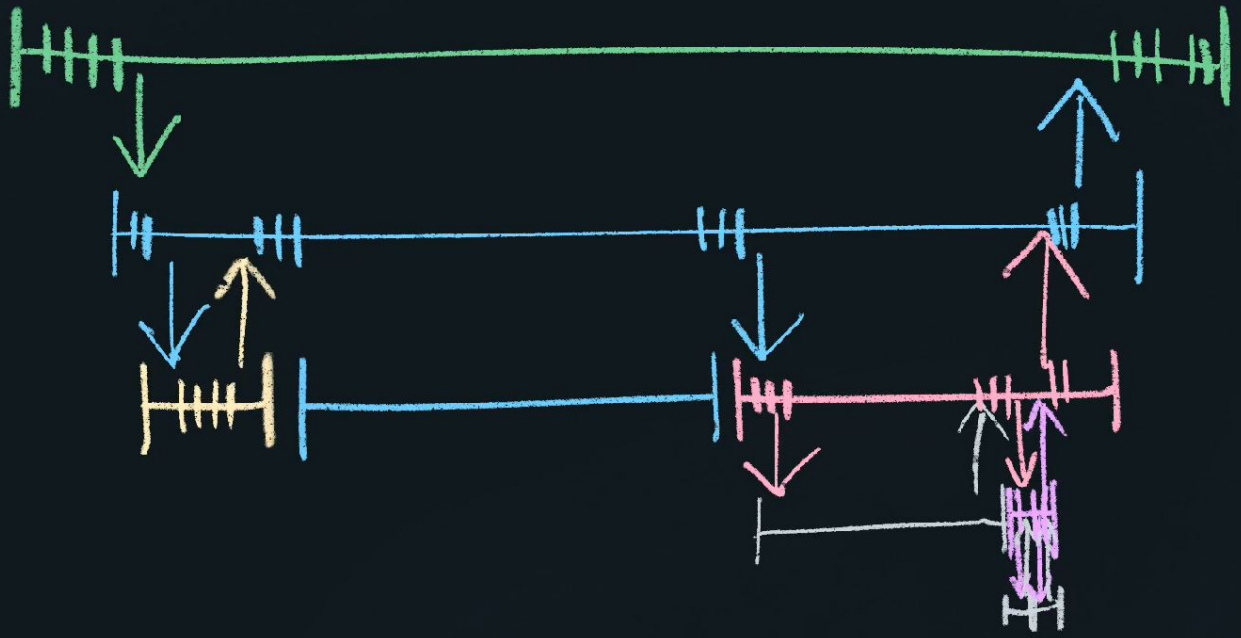
- Latency
- Errors



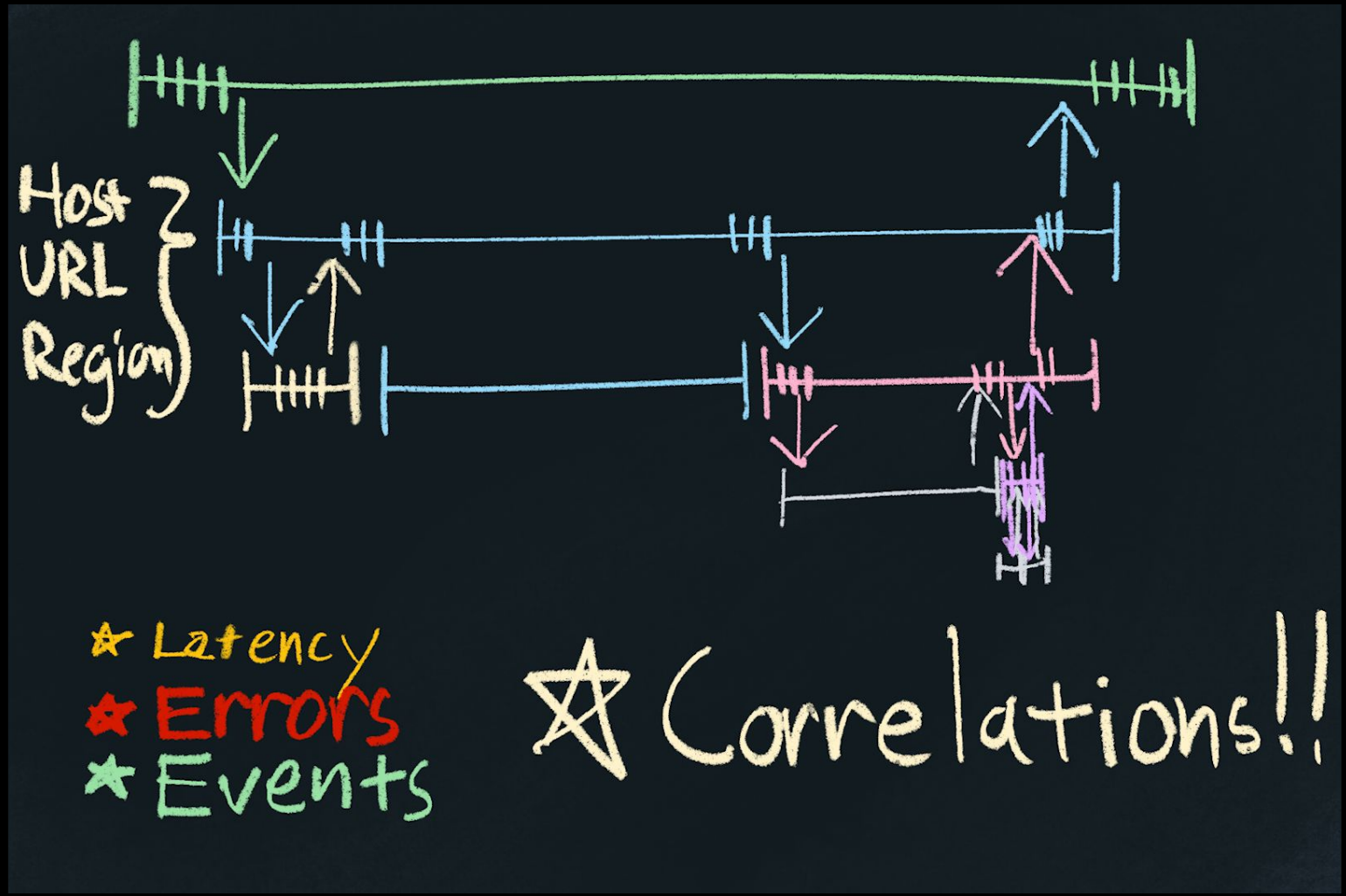
- Latency
- Errors



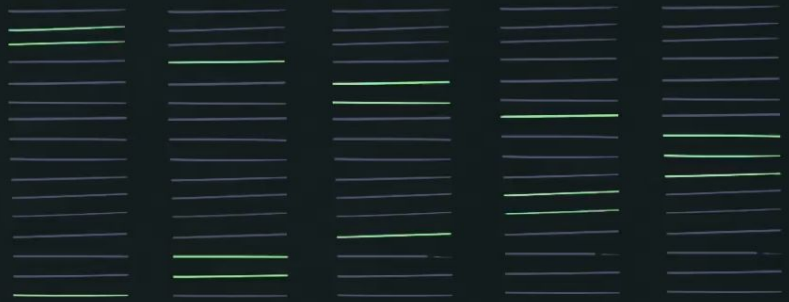
- Latency
- Errors



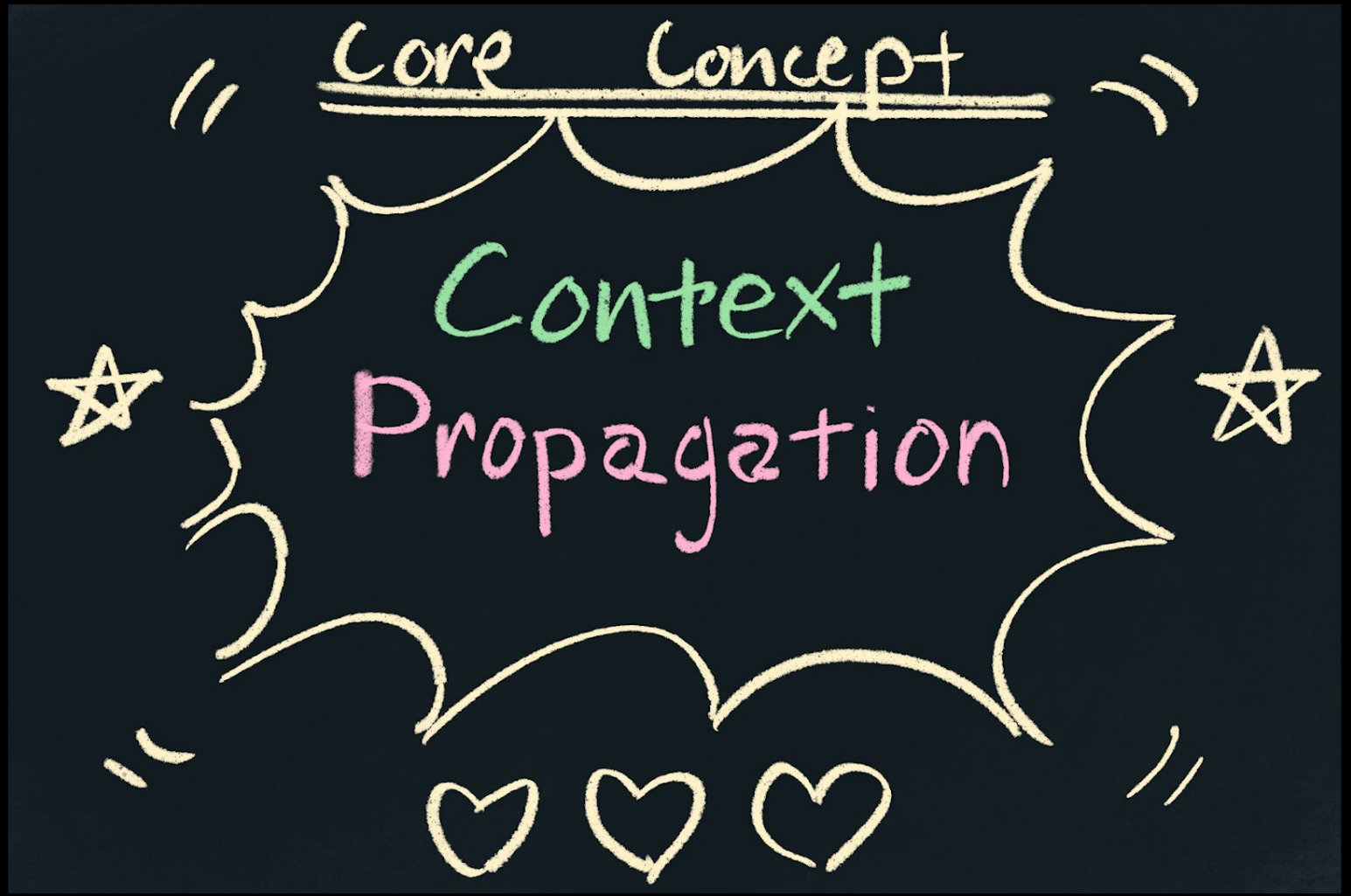
- Latency
- Errors
- Events

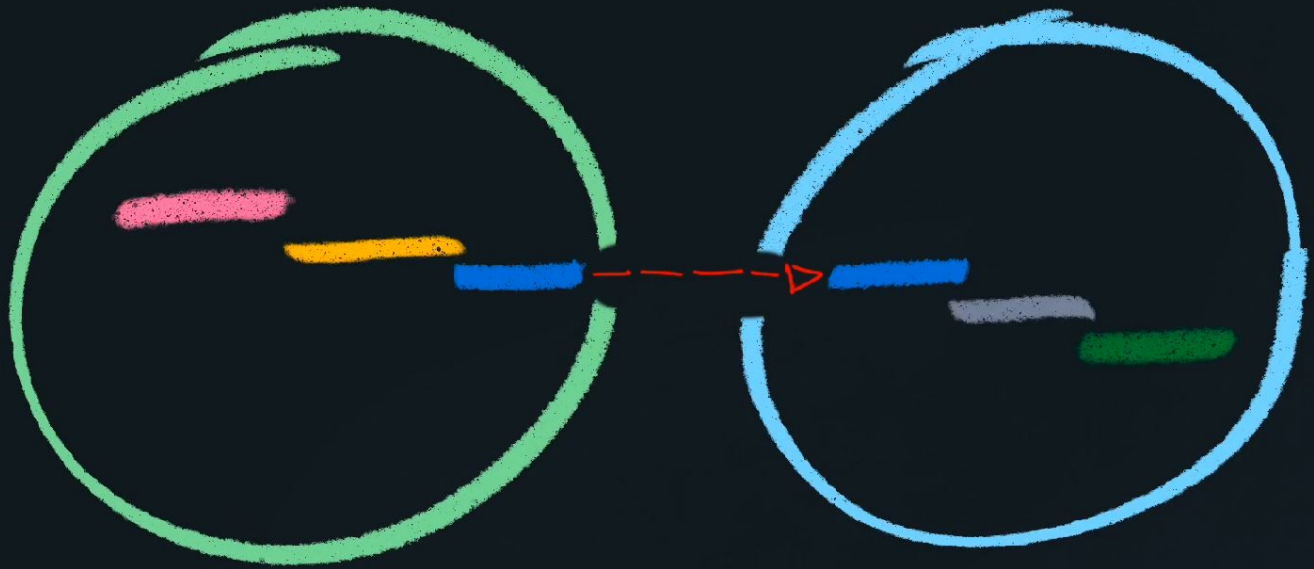

Host
URL
Region
Latency
Errors
Events
Correlations!!

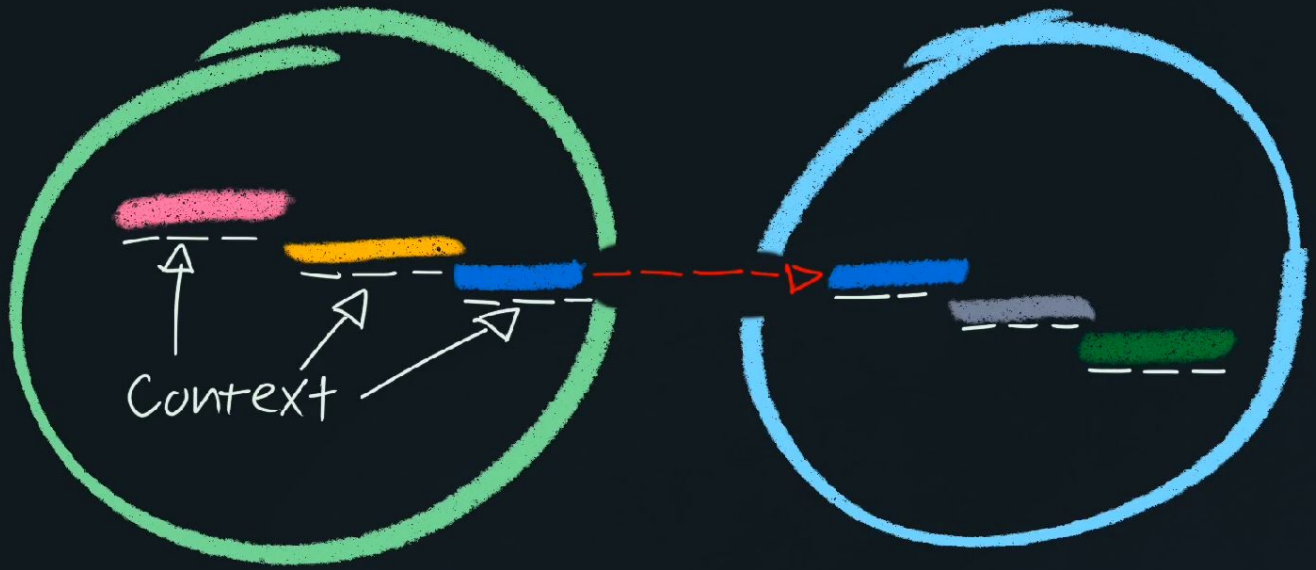
The logs I have

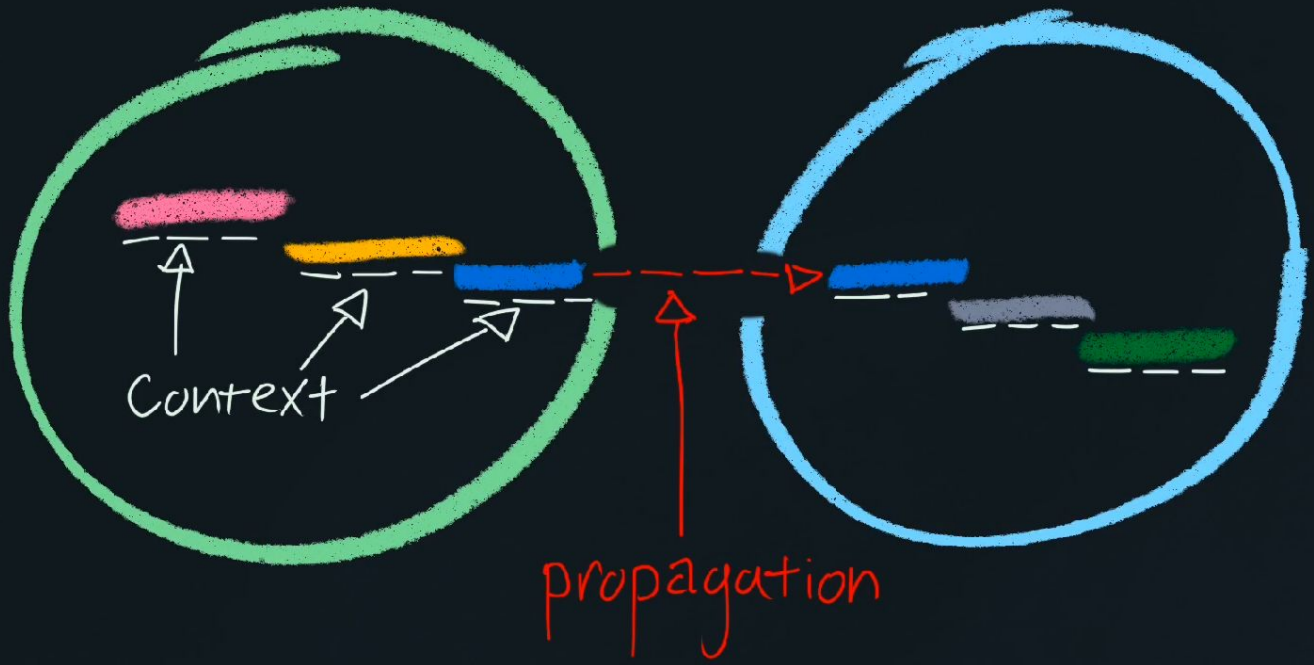


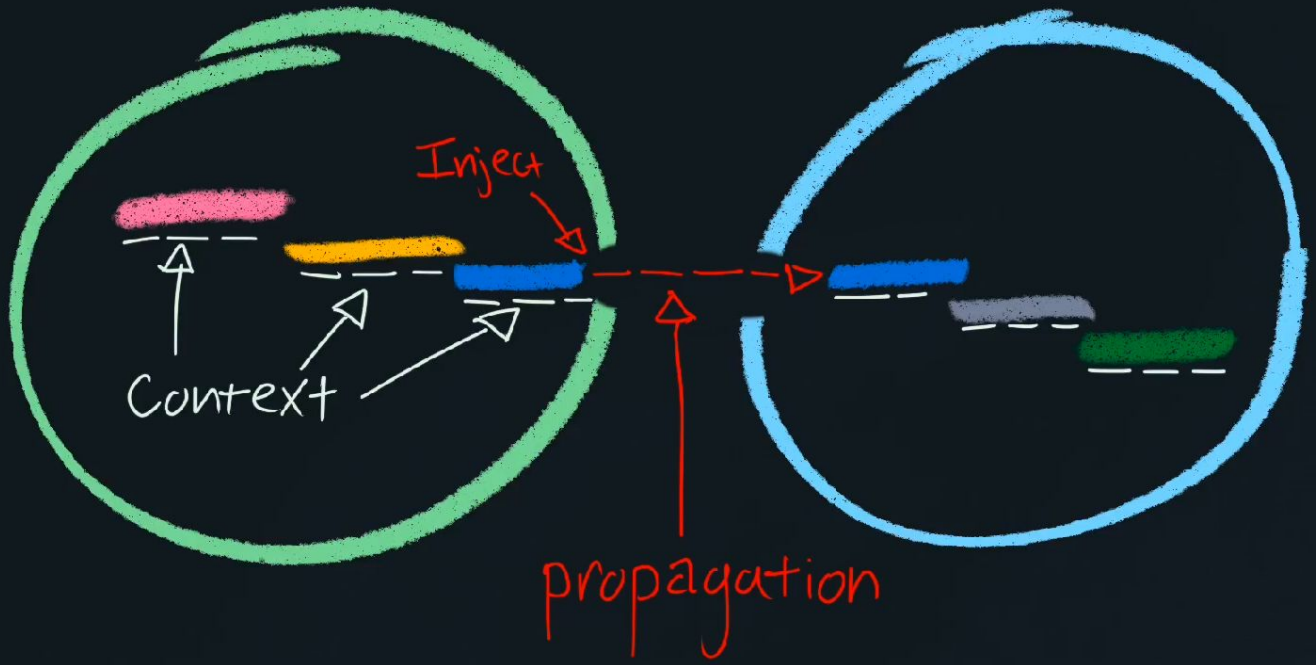
The logs I want

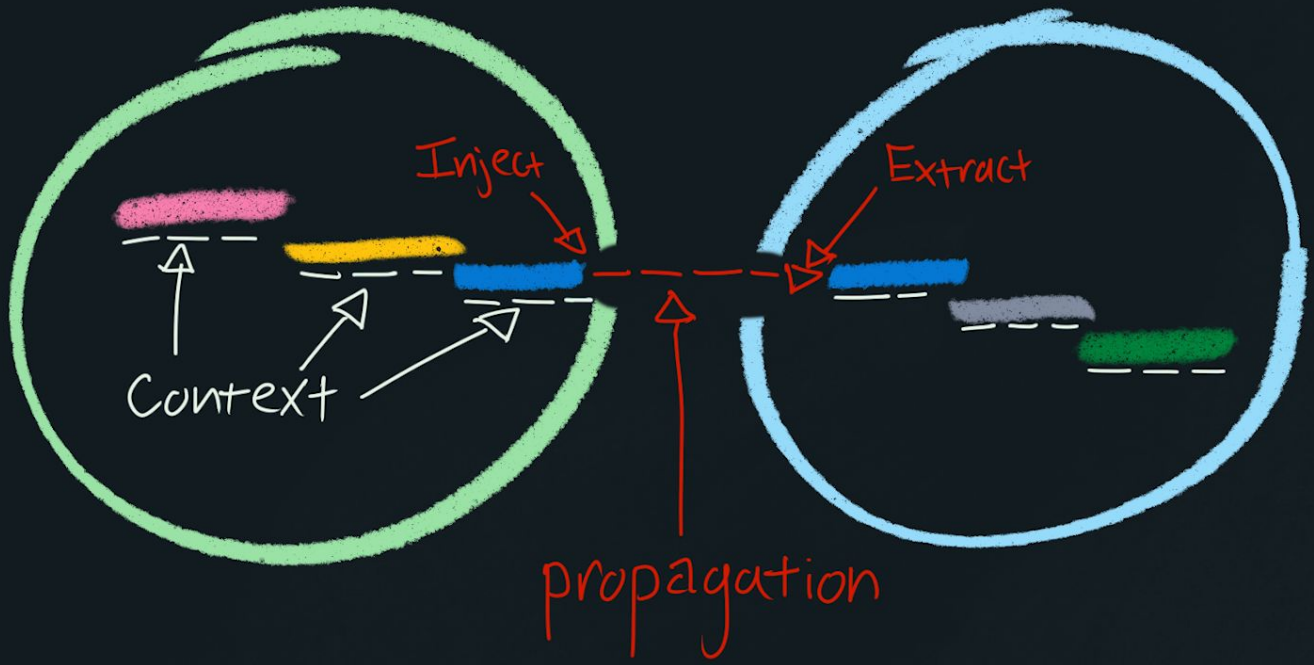

Core Concept
Context
Propagation

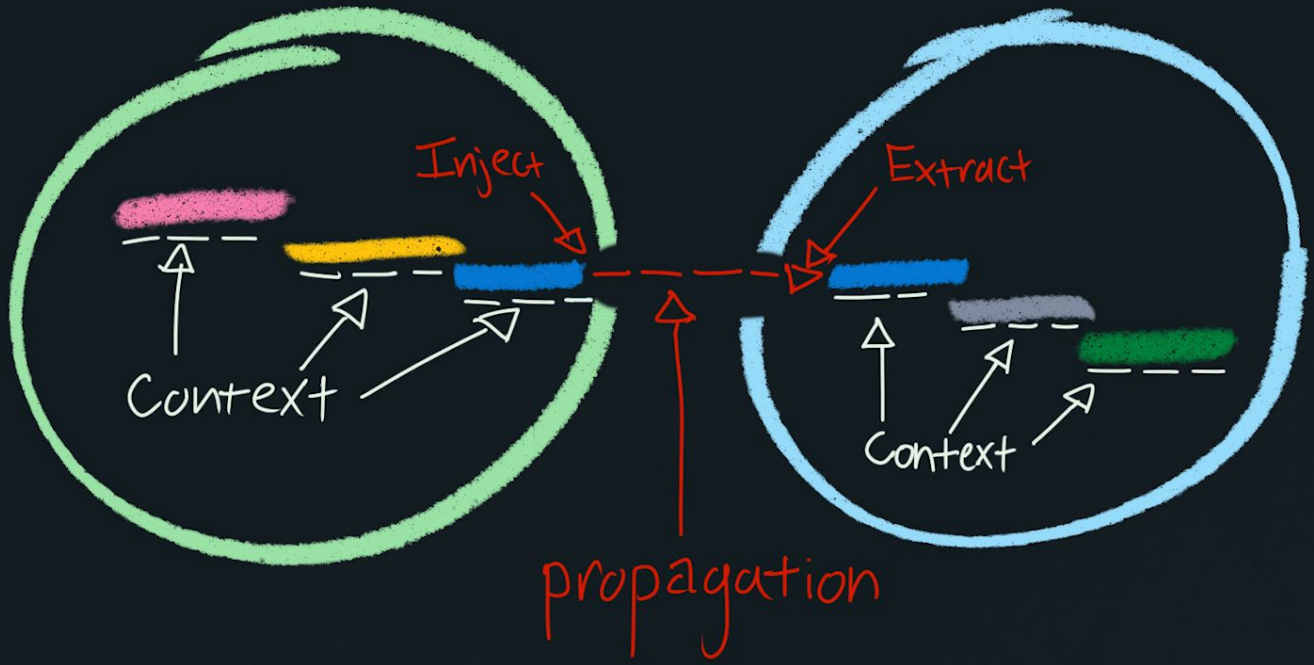



Context


Context
propagation


Inject
Context
propagation


Inject
Extract
Context
propagation


Inject
Extract
Context
Context
propagation

W3C Tracing Header

W3C Tracing Headers

Trace-Context

traceparent

- trace-id
- span-id
- Sampling Flag

tracestate

- internal details

W3C Tracing Headers

Trace-Context

traceparent

- trace-id
- span-id
- sampling Flag

tracestate

- internal details

Baggage

baggage

- arbitrary key/value pairs

SDK Setup

* Verify Framework & Library Support
* Verify Context Propagation
* Verify Data Quality

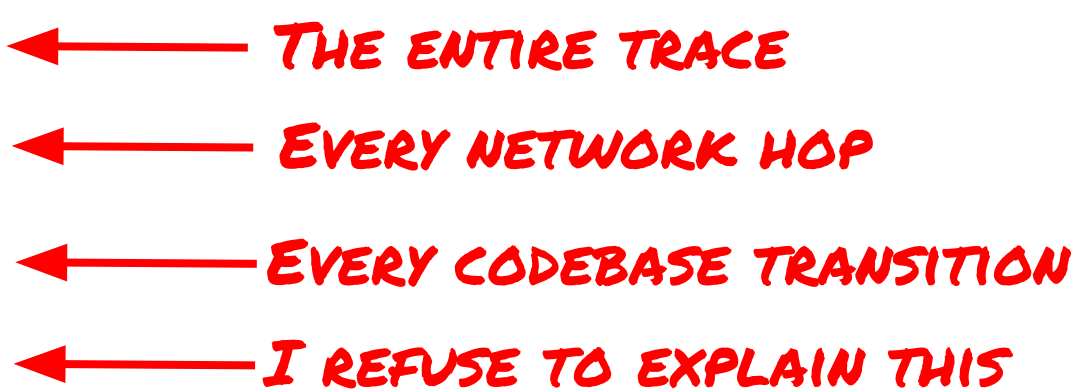
Code Walkthrough

Best Practices

Granularity: How big is a span, exactly?

In tracing, there are several relevant scopes

- Transaction
- Process
- Library
- Function



Which scope represents an “operation name”?

- Transaction
- Process
- Library

Generally too big

Somewhere in here...

- Function

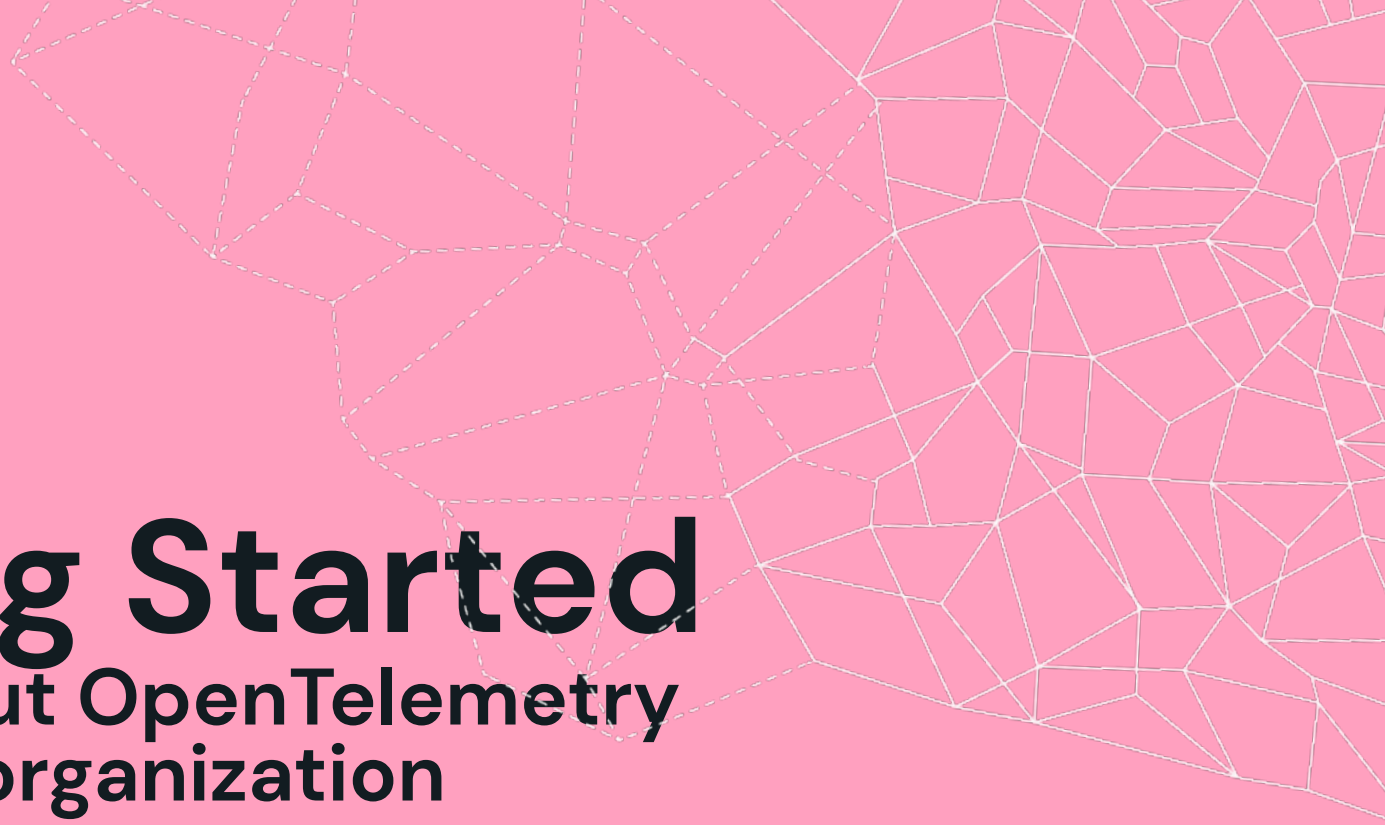
Generally too small

Practical realities around span granularity

- Spans are more expensive than logs
- Starting/Finishing spans also involves juggling Scopes/Contexts
- Trace indexing is “span based” in many tracing systems. Can’t always search for attributes across multiple spans.
- These current practical limitations lean towards having fewer, larger spans.

Practical advice

- Prefer coarsely grained spans, rich with data.
- Keep tags clustered together to improve searchability
- Attach stack trace details to spans via logs
- Centralize start/finish in framework code so that application code does not need to deal with Scope management.

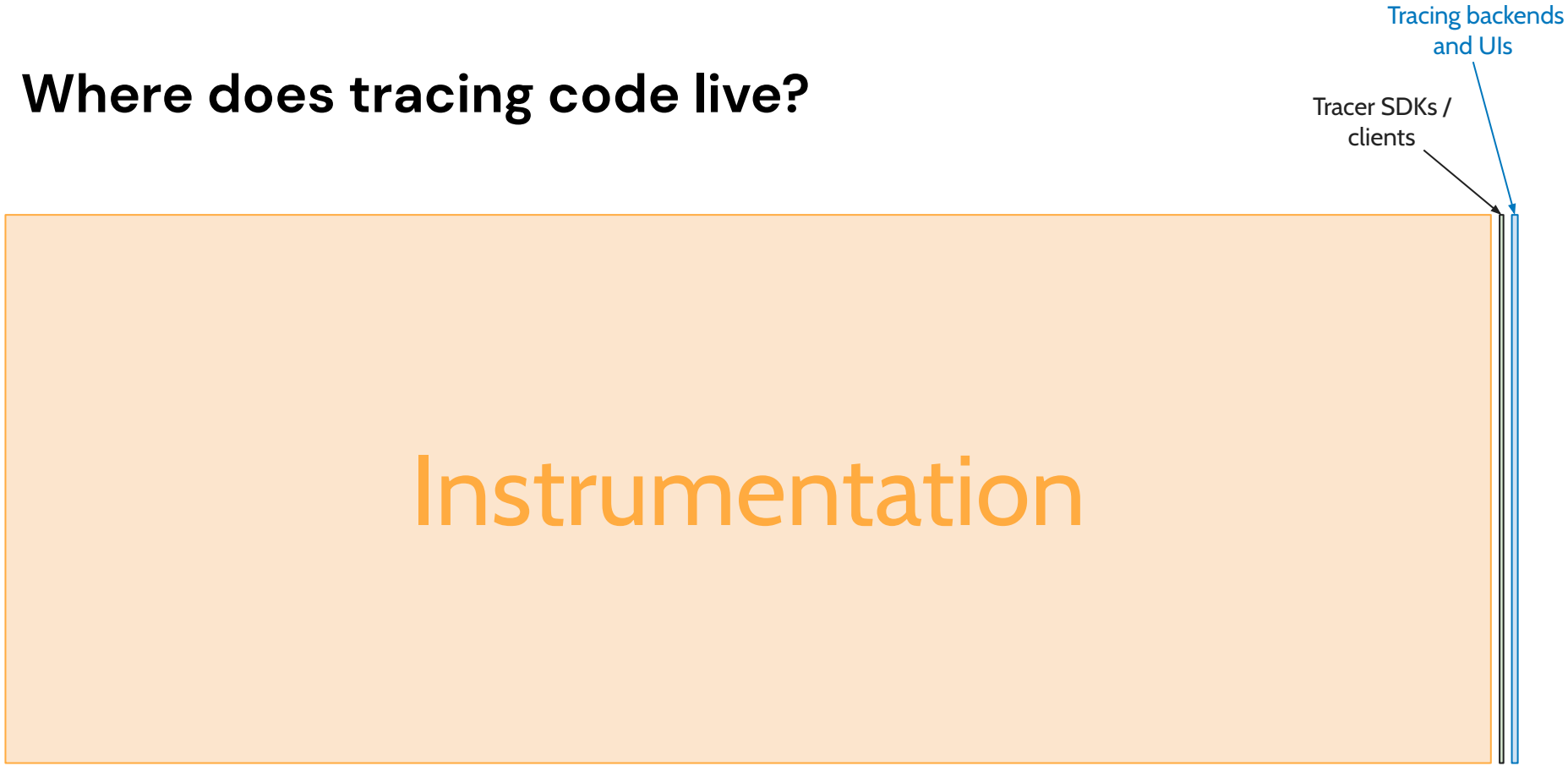


Getting Started

How to roll out OpenTelemetry across your organization

Where to Start:
What are you trying to solve?

Where does tracing code live?



What to trace and how to start

- Identify a high-value business transaction
  - i.e., “discover nearby *x*”, “add to cart”, etc.
- Identify the points of ingress and egress
- Breadth-first, not depth-first
- Get the first end-to-end trace reported

You've got your first trace... now what?

Expand!

- Add detail to your trace
  - Inner functions/calls
  - Meaningful tags and logs
- Add interoperating transactions/service
  - Tracing interplay often can provide new insights

Common Pitfalls:
What could go wrong?

What usually happens?

- Someone is a champion of tracing
- They basically go from group to group in the company and beg them to instrument
- *Maybe* some instrument, inconsistent at best
- Incomplete data, so can’t show value
- Tracing effort ends

It’s difficult to have a great success story with poor initial data quality - essentially, lack of instrumentation.

Pitfalls and Best Practices

- Project Management
  - Not having a project plan can prolong or derail efforts
  - Especially true when working with multiple teams

- Centralized Resources
  - Documentation
  - Shared framework adapter or helper library
  - Standardized tag and naming conventions
    - Which team made this? How do we record “this” ID, etc.

- Incorporate tracing into the service-provisioning process

OpenTelemery Quickstart

Constellation

17 Business Days

$6,000

- Introduction to Distributed Tracing
- OpenTelemetry Instrumentation Workshop
- OpenTelemetry Collector Workshop
- Observability Strategy Workshop
- Two weeks of office hours for Q&A
- Internal Documentation Materials

Contact: support@lightstep.com

OTel Roll Out Cheat Sheet

Production Ready
Beta
- GO
- Python
- Java
- JS

Org buy-in:
- ☆ Pick a known pain point
- ☆ Instrument the transaction
- ☆ Look for outliers and low hanging fruit.

Chat and Help: https://ltstp.run/discord

opentelemetry.io
- ☆ API Docs
- ☆ Project Status
- ☆ calendar, gitter, github, etc

otel.lightstep.com
- ☆ Getting Started Guides
- ☆ Launchers / Installation Helpers
- ☆ cookbooks, deep dives, etc

For OTel updates, follow me @tedsuo