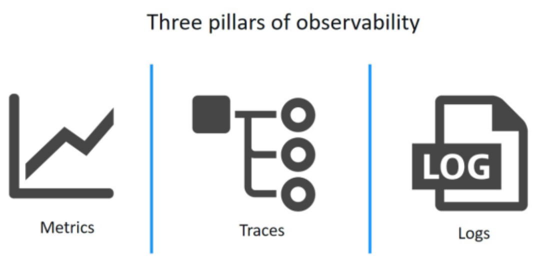
Week 2 — Observability Tools: 25.02.2023

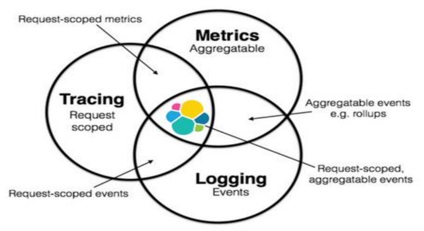
Technical Tasks

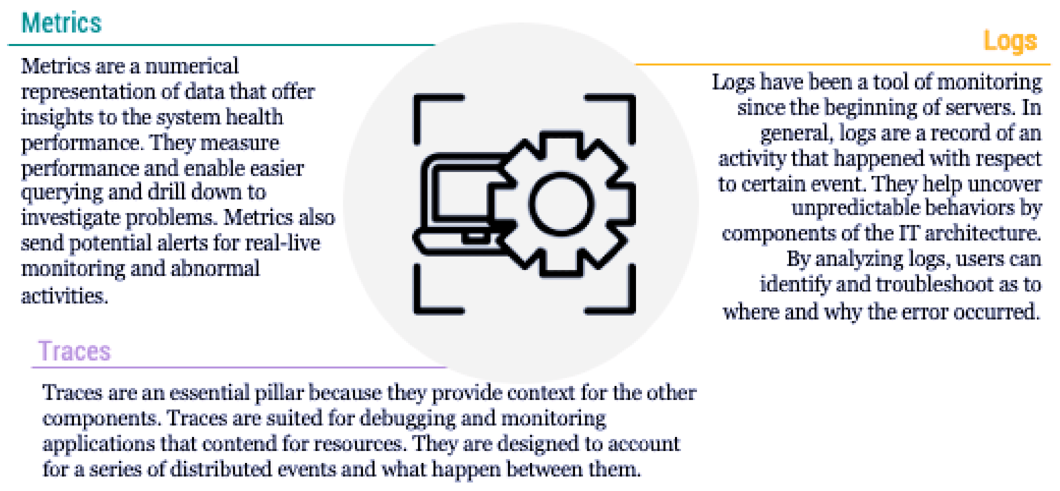
- Instrument our backend flask application to use Open Telemetry (OTEL) with Honeycomb.io as the provider
- Run queries to explore traces within Honeycomb.io
- Instrument AWS X-Ray into backend flask application
- Configure and provision X-Ray daemon within docker-compose and send data back to X-Ray API
- Observe X-Ray traces within the AWS Console
- Integrate Rollbar for Error Logging
- Trigger an error an observe an error with Rollbar
- Install Watchtower and write a custom logger to send application log data to CloudWatch Log group

Learnings

3 Pillars of Observability







Metrics

Metrics are a numerical representation of data that offer insights to the system health performance. They measure performance and enable easier querying and drill down to investigate problems. Metrics also send potential alerts for real-live monitoring and abnormal activities.

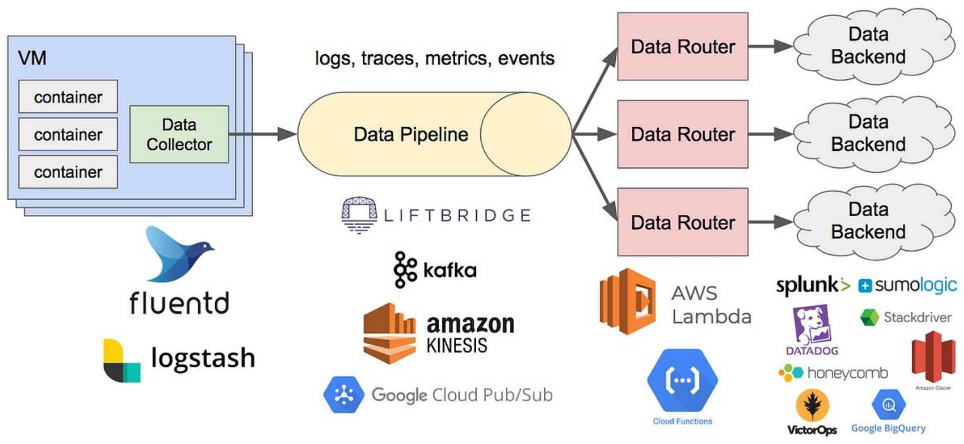
Logs

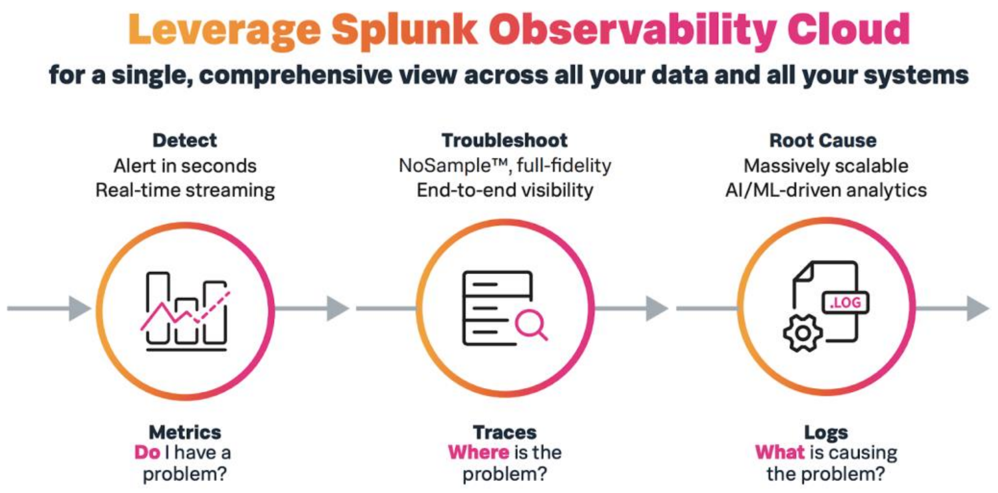
Logs have been a tool of monitoring since the beginning of servers. In general, logs are a record of an activity that happened with respect to certain event. They help uncover unpredictable behaviors by components of the IT architecture. By analyzing logs, users can identify and troubleshoot as to where and why the error occurred.

Traces

Traces are an essential pillar because they provide context for the other components. Traces are suited for debugging and monitoring applications that contend for resources. They are designed to account for a series of distributed events and what happen between them.

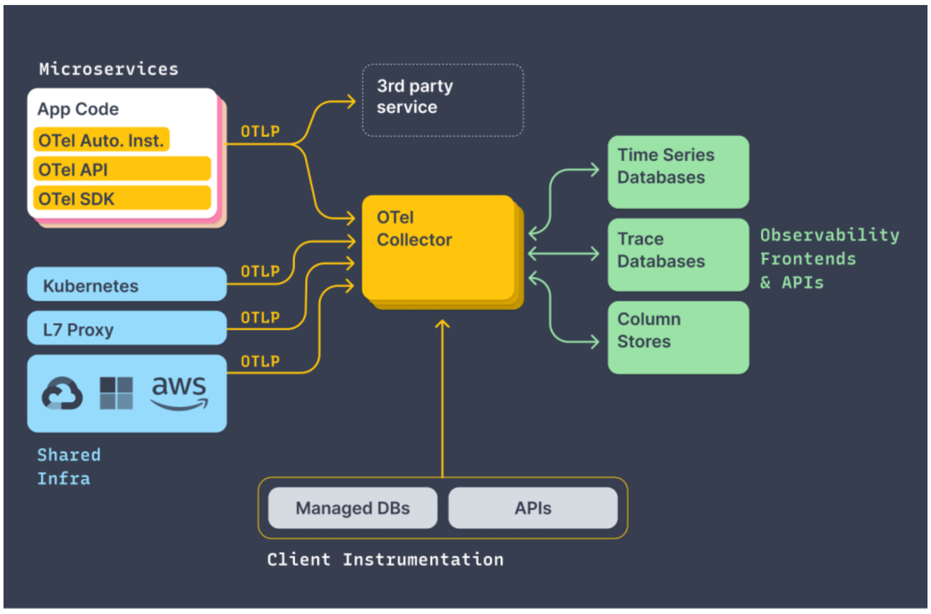
Traceability – i.e. Distributed Traceability is used more for backend, where the requests from front end moves between different systems at the backend. Traceability at the front end is more for tracing the latency between frontend and backend.





- Use Glitch (glitch.com) – Good for hosting small websites/Node App.
- Datadog is a more comprehensive Observability solution than Honeycomb.io, but for container tracing it is best.
- Use Observability Tools exposure to Resume – good skill to have.

Open Telemetry, also known as **OTel** for short, is a vendor-neutral open-source Observability framework for instrumenting, generating, collecting, and exporting telemetry data such as traces, metrics, logs. As an industry-standard it is natively supported by a number of vendors.



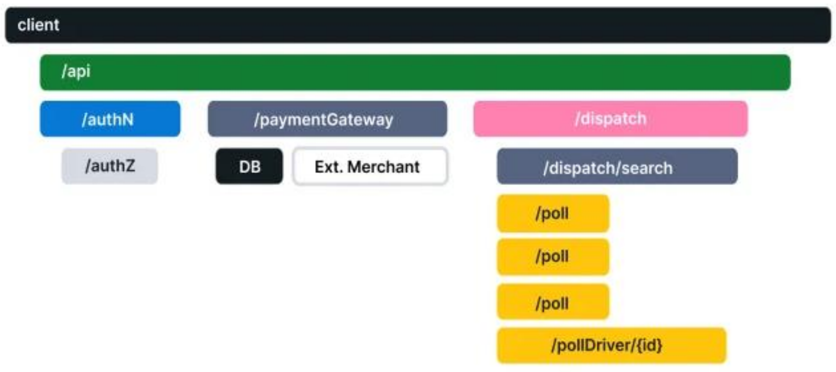
Open Telemetry is not an observability back-end like Jaeger or Prometheus. Instead, it supports exporting data to a variety of open source and commercial back-ends. It provides a pluggable architecture so additional technology protocols and formats can be easily added.

Tracing makes debugging and understanding distributed systems less daunting by breaking down what happens within a request as it flows through a distributed system.

A Trace is made of one or more Spans. The first Span represents the Root Span. Each Root Span represents a request from start to finish. The Spans underneath the parent provide a more in-depth context of what occurs during a request (or what steps make up a request).

Many Observability back-ends visualize Traces as waterfall diagrams that may look something like this:

https://opentelemetry.io/docs/concepts/observability-primer/



Observability vs Monitoring Explained in AWS – cloud security podcast

Approach for Building Security Metrics, Logs for Tracing

- The Application landscape: different applications (monolith, microservice, legacy
- 0used and deployed on VM, Container etc
- Threat Modelling Session for each application to identify the different attack vectors such as Email Phishing, DDOS, or Malware to avoid detection in Anti-Virus solution
- Map the Attack vectors to Industry know Attack Patterns/Techniques in ATACK MITRE framework
- Identify instrumentation agents to create tracing (cloud watch agents, 3rd Party Agents, fire lens)
- AWS Services like AWS Distro for Telemetry (ADOT) for metrics and traces
- Dashboard for Practical Attack vectors for each application.