Logging and Monitoring in Microservices

A microservices application can be composed of dozens or hundreds of services. Not only that, a single user operation can span multiple of services. Therefore, not only you have to be mindful of these individual services but also of the communication between them. It is also possible that each of these services are on different hardware or even data centers. In order to understand what’s going in your microservices application and to debug it when time comes, we must emit meaningful telemetry such as metrics and logs.

Metrics are meant to be analyzed. You can observe the behavior of the system in real or near real-time or use the information to create trends. Typical metrics includes:

* System metrics: CPU, memory, network, disk usages.
* Application metrics: In essence, any metrics relevant to your application. (i.e. # of requests per second)
* External metrics: If your application depends on any external system, it is good to have metrics on them as well

Logs are textual representation of meaningful events that occur when your application is running.

# Considerations

There are several things to consider when setting up logging and monitoring of microservices. These can have a direct impact on the service offered by your application.

* Management: How will the metrics and logs be gathered?
* Rate: At what rate will we gathered the metrics and logs?
* Cost: What are the costs of gathering and storing the information?
* Accuracy: If the sampling is too low, we might not see problems.
* Latency: How fast do you need the information available?
* Storage: How long in the past do you want to keep the information?
* Visualization: It is great to have the information in raw form but do you need to application any analysis and generate dashboard?

# Distributed Tracing

In a microservices application, a single operation can span multiple services. Consequently, in order to reconstruct the chain of events, we must correlate each service call that were involve. This can be done with the usage of a unique identifier often called a correlation ID. Here are a few things to keep in mind:

* The ID needs to be generated at the beginning of the chain and pass along each call.
* The ID can be passed a HTTP head or as part of the message schema.
* It might be useful to past more meaningful information in the form of a context like caller-called relationship.
* It is important to standardized the correlation IDs in your logs. (i.e. structured logs)

# Technology

There are certain technological options that should be considered for system and container metrics and applications logs. For example, for system metrics, there are time-series database such as Prometheus (a pull-based system) and InfluxDB (a push-based system). Similarly, for dashboard, there are tools such as Kibana and Grafana. Similarly, for application logs, one popular approach is to use Fluentd and Elasticsearch where Fluentd collects logs generated from stdout or stderr and sends them to Elasticsearch which is optimized for search. Much like anything in microservices, we need to make sure it scale with the load of your system.