System Design: Distributed Logging

Let's understand the basics of designing a distributed logging system.

We'll cover the following

Logging

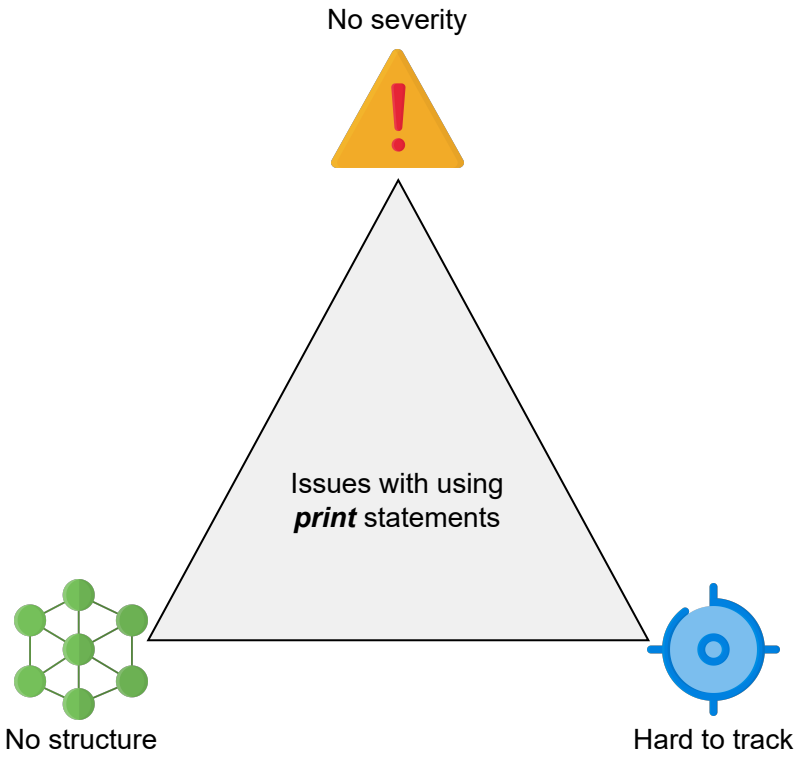
A **log file** records details of events occurring in a software application. The details may consist of microservices, transactions, service actions, or anything helpful to debug the flow of an event in the system. Logging is crucial to monitor the application's flow.

Need for logging

Logging is essential in understanding the flow of an event in a distributed system. It seems like a tedious task, but upon facing a failure or a security breach, logging helps pinpoint when and how the system failed or was compromised. It can also aid in finding out the root cause of the failure or breach. It decreases the meantime to repair a system.

Why don't we simply print out our statements to understand the applicatio flow? It's possible but not ideal. Simple print statements have no way of

tracking the severity of the message. The output of print functions usually goes to the terminal, while our need could be to persist such data on a local or remote store. Moreover, we can have millions of print statements, so it's better to structure and store them properly.



Issues with using print statements as an alternative to logging

Concurrent activity by a service running on many nodes might need causality information to stitch together a correct flow of events properly. We must be careful while dealing with causality in a distributed system. We use a logging service to appropriately manage the diagnostic and exploratory data of our distributed software.

Logging allows us to understand our code, locate unforeseen errors, fix the identified errors, and visualize the application's performance. This way, w are aware of how production works, and we know how processes are running in the system.

Log analysis helps us with the following scenarios:

- To troubleshoot applications, nodes, or network issues.
- To adhere to internal security policies, external regulations, and compliance.
- To recognize and respond to data breaches and other security problems.
- To comprehend users' actions for input to a recommender system.

How will we design a distributed logging system?

We have divided the distributed logging system design into the following two lessons:

1. **Introduction**: We'll discuss how logging works at a distributed level. We'll also show how we can restrict the huge size of a log file, and structure them. This lesson will guide us about the requirements we should consider while logging information about a system.
2. **Design**: In this lesson, we'll define the requirements, API design, and detailed design of our distributed logging system.

Back

Evaluation of a Distributed Search's Design

Next

Introduction to Distributed Logging

Mark as Completed