Need of monitoring

- Downtime cost
- Types of monitoring

Let's go over how the failure of a single service can affect the smooth execution of related systems. To avoid cascading failures, monitoring can play a vital role with early warnings or steering us to the root cause of faults.

Downtime cost
There are fault-tolerant system designs that hide most of the failures from the end users, but it's crucial to catch the failures before they snowball into a bigger problem. The unplanned outage in services can be costly. For example, in October 2021, Meta's applications were down for nearly nine hours, resulting in a loss of around $13 million per hour. Such losses emphasize the potential impact of outages.

With the above examples, we can divide our monitoring focus into two broad categories of errors:

Server-side errors: These are errors that are usually visible to monitoring services as they occur on servers. Such errors are reported as error 5xx in HTTP response codes.

Client-side errors: These are errors whose root cause is on the client-side. Such errors are reported as error 4xx in HTTP response codes. Some client-side errors are invisible to the service when client requests fail to reach the service.

Pre requisites of a monitoring System

The two conventional approaches to handling IT infrastructure failures are the reactive and proactive approaches.

In the reactive approach, corrective action is taken after the failure occurs. In this approach, even if DevOps takes quick action to find the cause of the error and promptly handle the failures, it causes downtime. As a result, there will be system downtime in the reactive approach, which is generally undesirable for continuously running applications.

In a proactive approach, proactive actions are taken before failure occurs. Therefore, it prevents downtimes and associated losses. The proactive approach works on predicting system failures to take corrective action to avoid the failure. This approach offers better reliability by preventing downtime.

In modern services, completely avoiding problems is not possible. Something is always failing inside huge data centers and network deployments. The goal is to find the impending problems early on and design systems in such a way that service faults are invisible to the end users.

Metrics

Metrics objectively define what we should measure and what units will be appropriate. Metric values provide an insight into the system at any point in time For example, a web server's ability to handle a certain amount of traffic per second or its ability to join a pool of web servers are examples of high-level data correlated with a component's specific purpose or activity.