Jun 2024

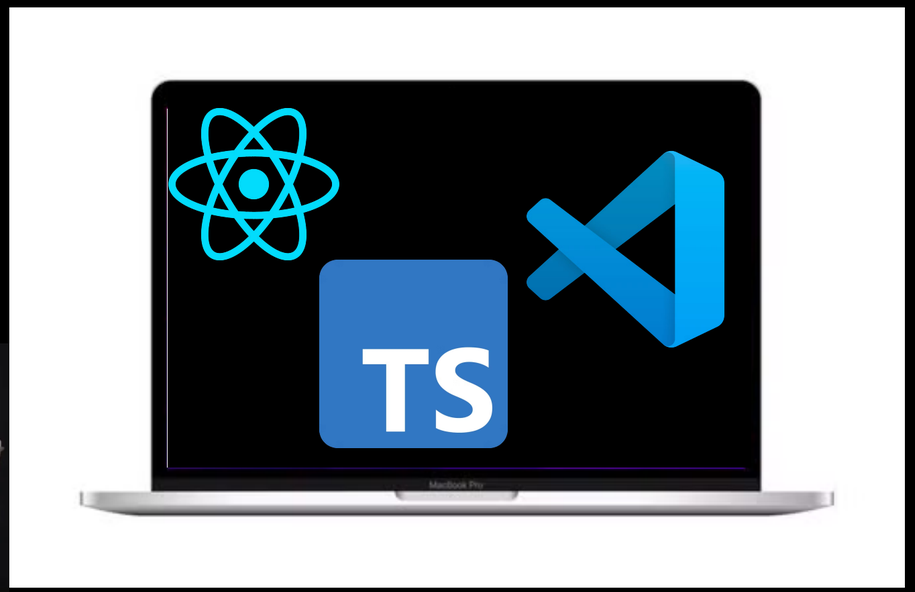
INTRO TO OBSERVABILITY

By Manik Rana

Try Pitch

HI, I'M MANIK

- Background:
  - Mostly a web dev (React, Nextjs, Python, Go)
  - Student
  - Dogs are awesome



Try Pitch

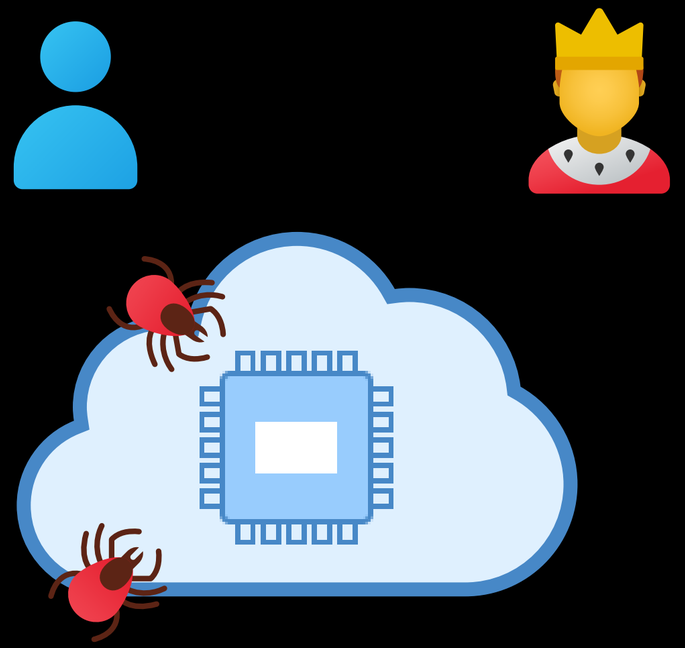
AGENDA

1. **OBSERVABILITY**
    1. What
    2. Components
    3. Best Practices
2. **MONITORING**
    1. What
    2. Components
    3. Best Practices
3. **OBSERVABILITY + MONITORING**
4. **DEMO**

Try Pitch

BUGS ARE A RACE BETWEEN THE CUSTOMER AND YOU

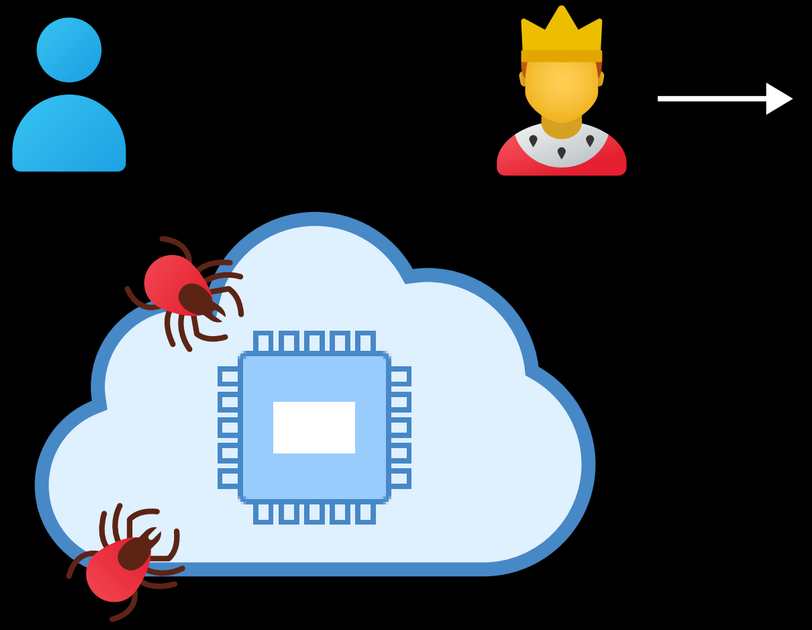
- Who discovers the bug first?



Try Pitch

BUGS ARE A RACE BETWEEN THE CUSTOMER AND YOU

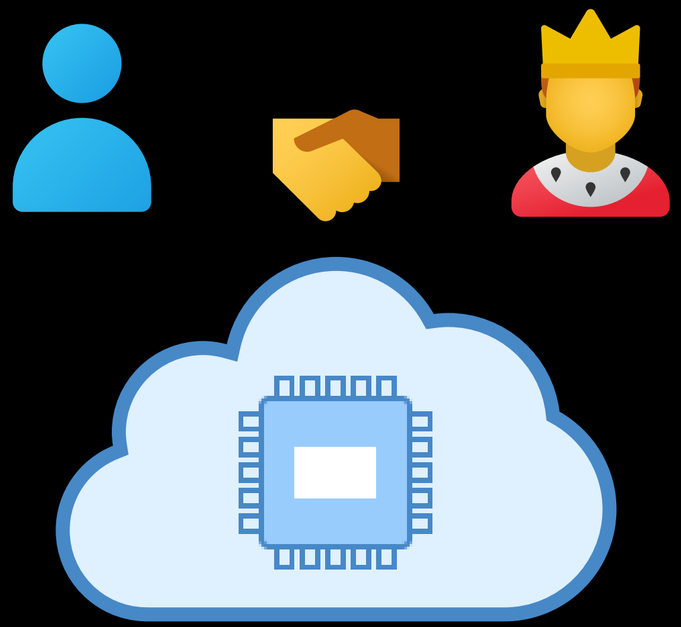
- Who discovers the bug first?
  - If customer → your app is unreliable



Try Pitch

BUGS ARE A RACE BETWEEN THE CUSTOMER AND YOU

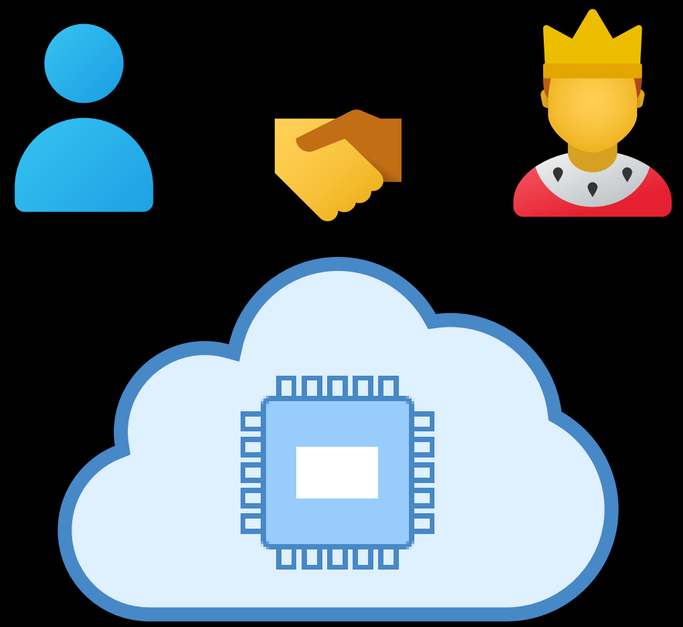
- Who discovers the bug first?
  - If customer → your app is unreliable
  - If you → you provide great service!



Try Pitch

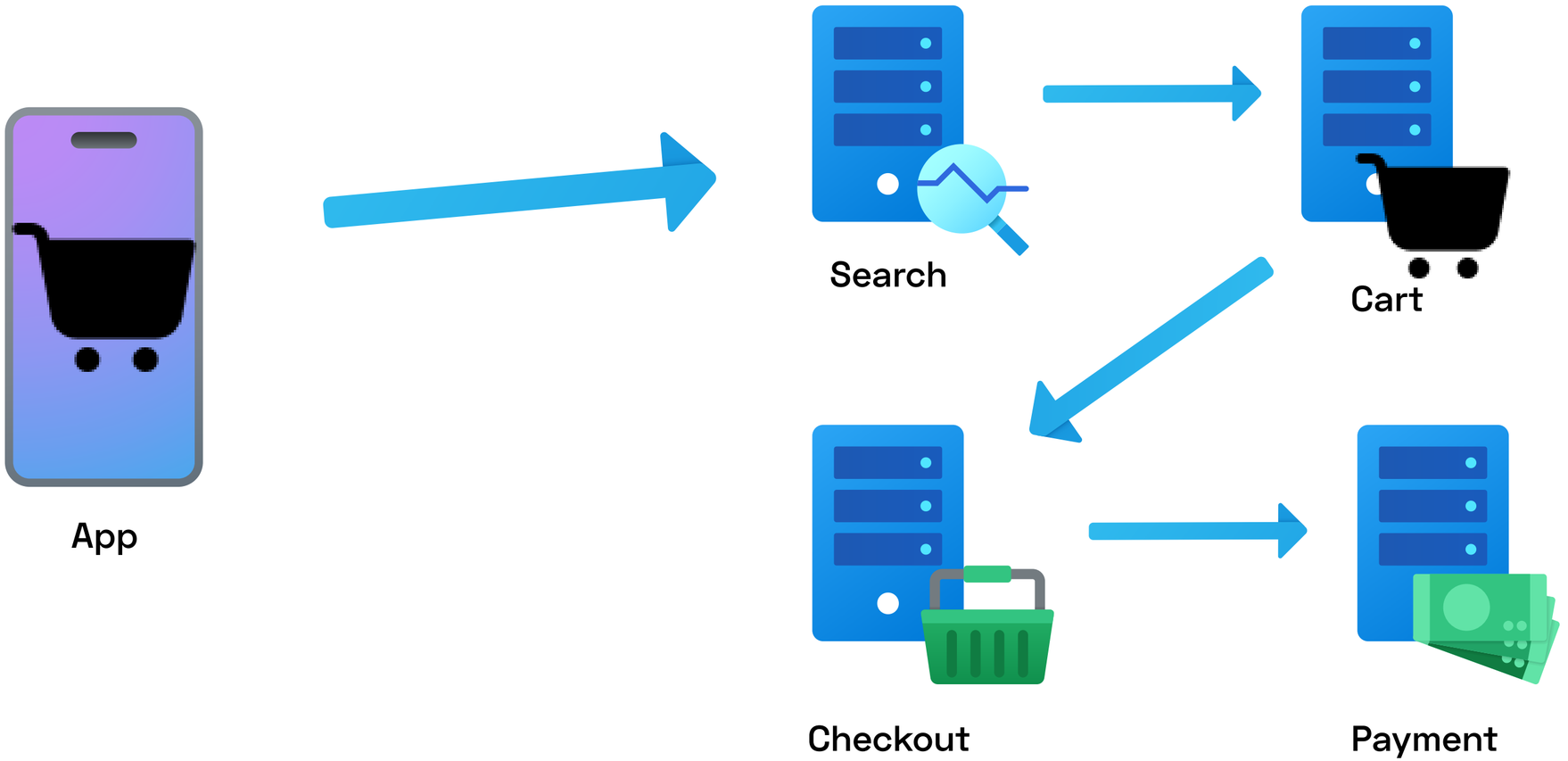
BUGS ARE A RACE BETWEEN THE CUSTOMER AND YOU

- Who discovers the bug first?
  - If customer → your app is unreliable
  - If you → you provide great service!
- How do you stay ready to catch bugs?
  - Observability



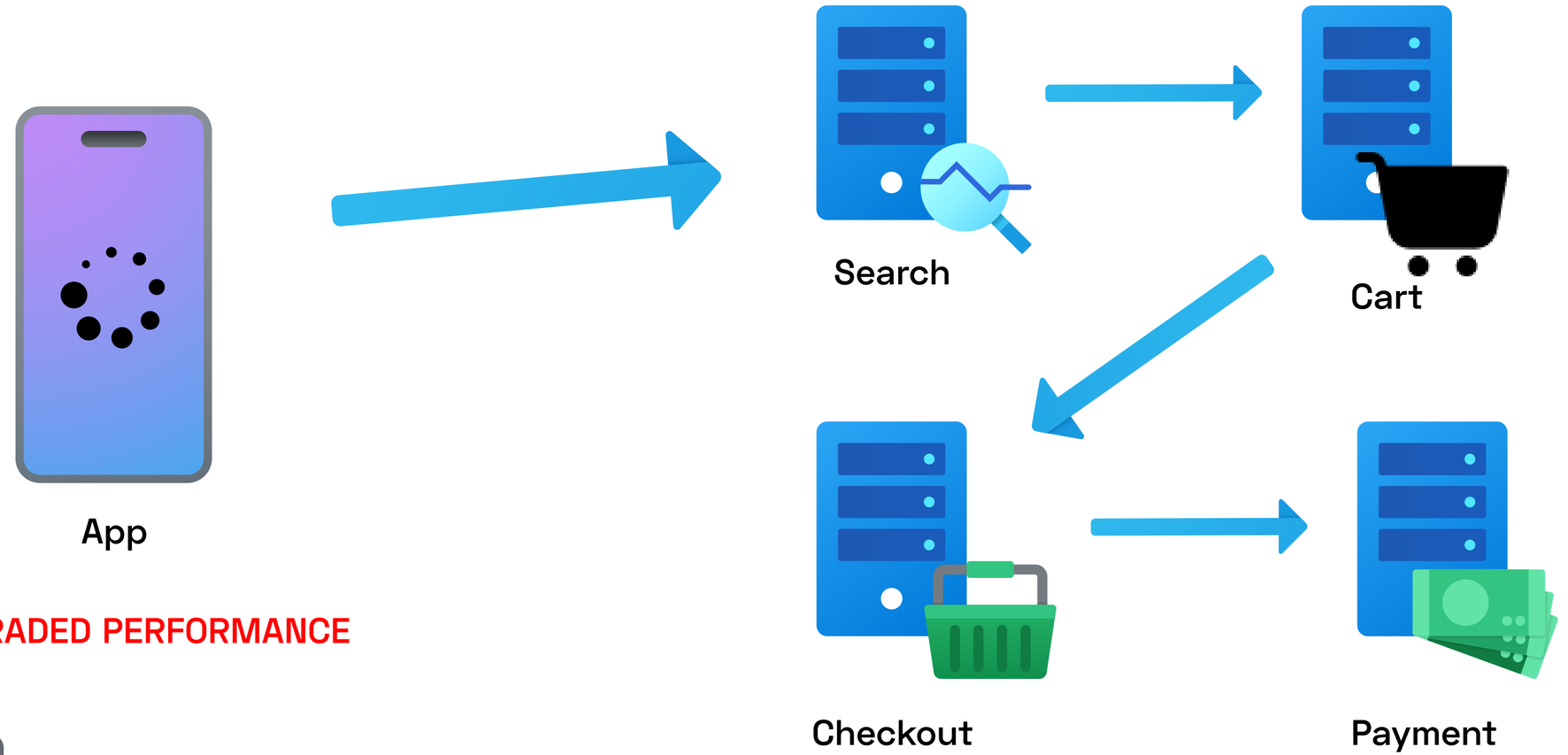
Try Pitch

SITUATION:
AN ECOMMERCE APP



Try Pitch

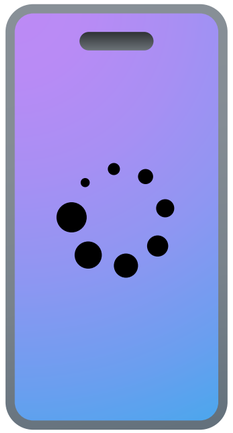
SITUATION:
AN ECOMMERCE APP



DEGRADED PERFORMANCE

Try Pitch

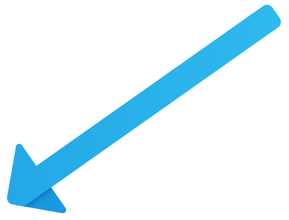
SITUATION:
AN ECOMMERCE APP



App

Search

Cart



Checkout

Payment

DEGRADED PERFORMANCE

Very slow app. I don't recommend using this app!

What if we got notifications for this?

What if we could track all this?

What if we could know whats causing it?

Try Pitch

WHAT IS OBSERVABILITY

For a software application to have observability, you must be able to:

- Understand the inner workings of your application
-
- Understand any system state your application many have gotten itself into
-
- Understand the things above, solely by observing that with external tools
-
- Understand that state, no matter how extreme or unusual

Try Pitch

WHAT IS OBSERVABILITY

A good litmus test:

- Can you understand what any particular user of your software may be experiencing?
- Can you determine anomalies even if you have never seen or debugged this particular state or failure before?
- Can you determine the things above even if this anomaly has never happened before?

Try Pitch

To achieve observability, an application must generate specific data that helps in detecting issues and resolving problems.

and 3 types of data are needed:

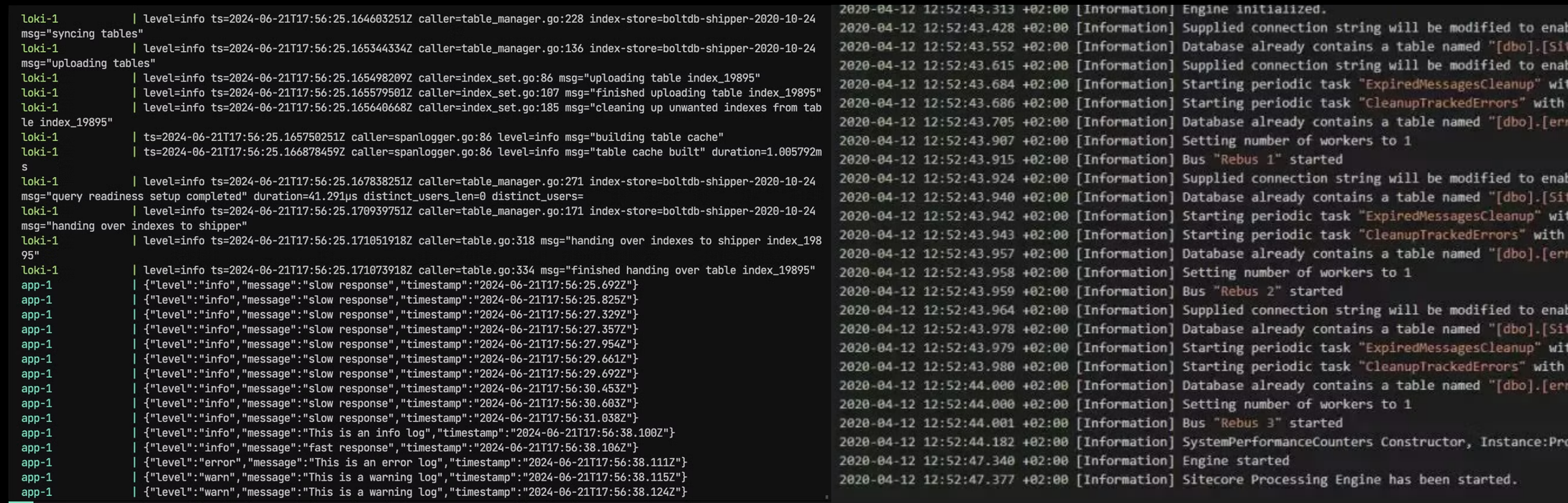
LOGS
TRACES
METRICS

Try Pitch

LOGS

Logs provide a chronological record of events or transactions within a system. They provide insights into when something happened and what exactly happened.

Tools like Elasticsearch, Logstash, or Loki are often used to deal with logs.

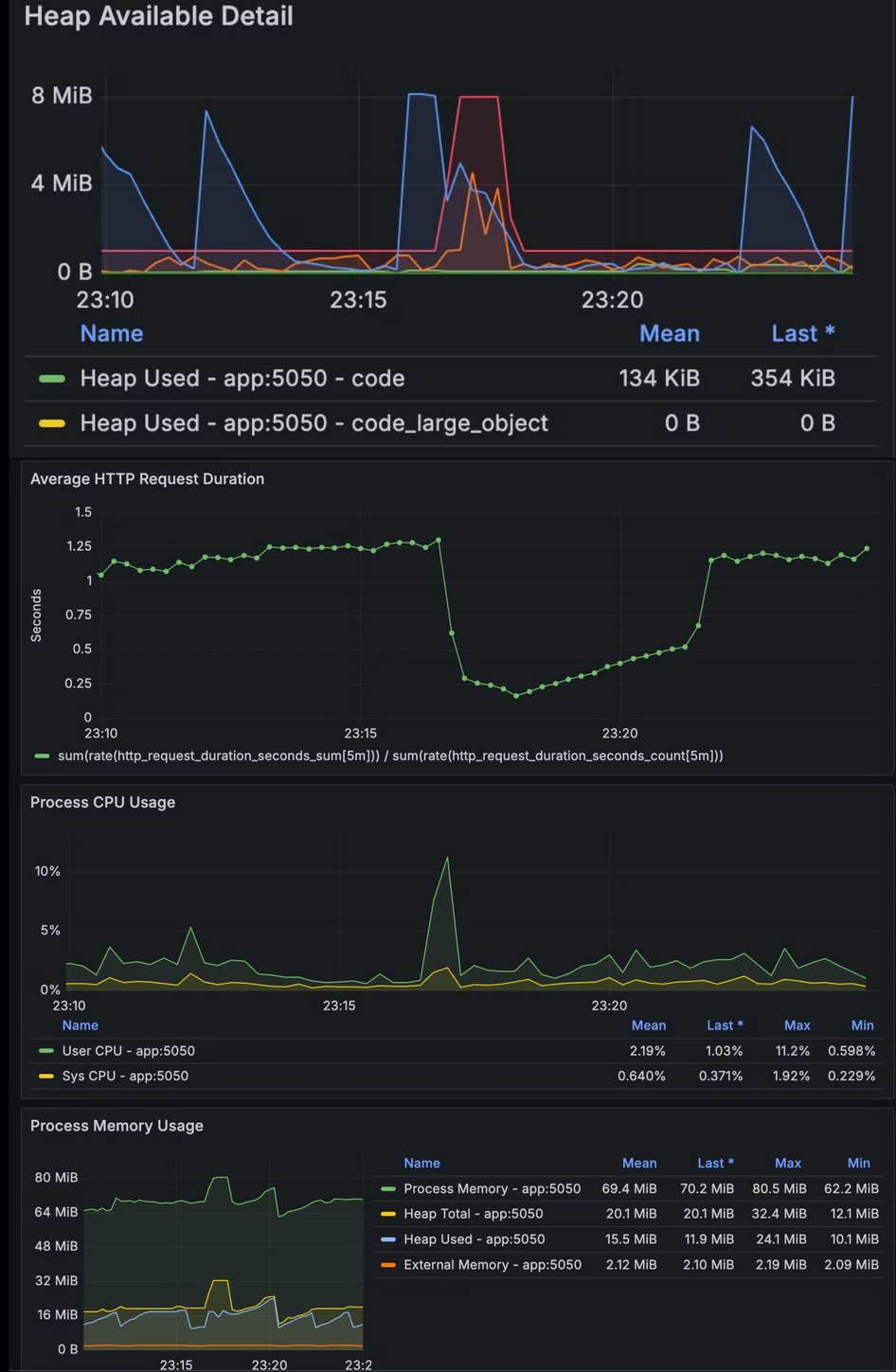


Try Pitch

METRICS

Metrics are quantitative measurements that offer a snapshot of a system's performance over time. These include numerical values such as response time, error rate, or resource utilization.

Tools like Prometheus collect metrics from various sources such as nodes, Kubernetes services, and pods, and provide querying and alerting mechanisms.



Try Pitch

TRACES

Tracing helps us to track the flow of requests through various services and components of a system

Tools like Tempo, Jaeger, and Zipkin help us track requests as they travel through various services in a microservices architecture.

Logs and metrics, when combined with traces, bring out better insights that help us to troubleshoot issues faster.

Try Pitch

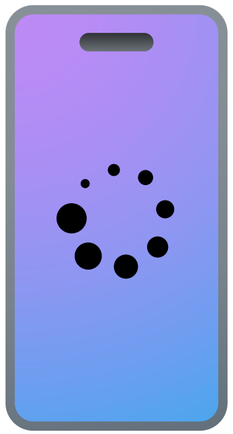
OBSERVABILITY BEST PRACTICES

- ✓ Decide what to log
- ✓ Cleanup logs
- ✓ The end goal of observability is that whenever there is an issue, we should have enough context to ensure that troubleshooting is effective enough to resolve any issue.

Try Pitch

SITUATION: AN ECOMMERCE APP

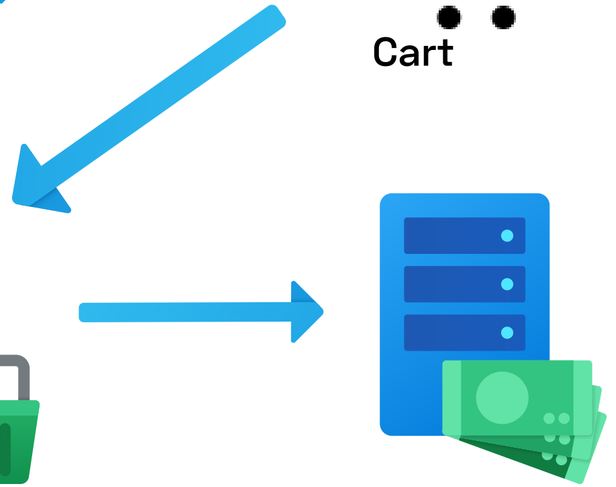
Enter Monitoring



App

Search

Cart



Checkout

Payment

What if we got notifications for this?

What if we could track all this?

What if we could know whats causing it?

DEGRADED PERFORMANCE

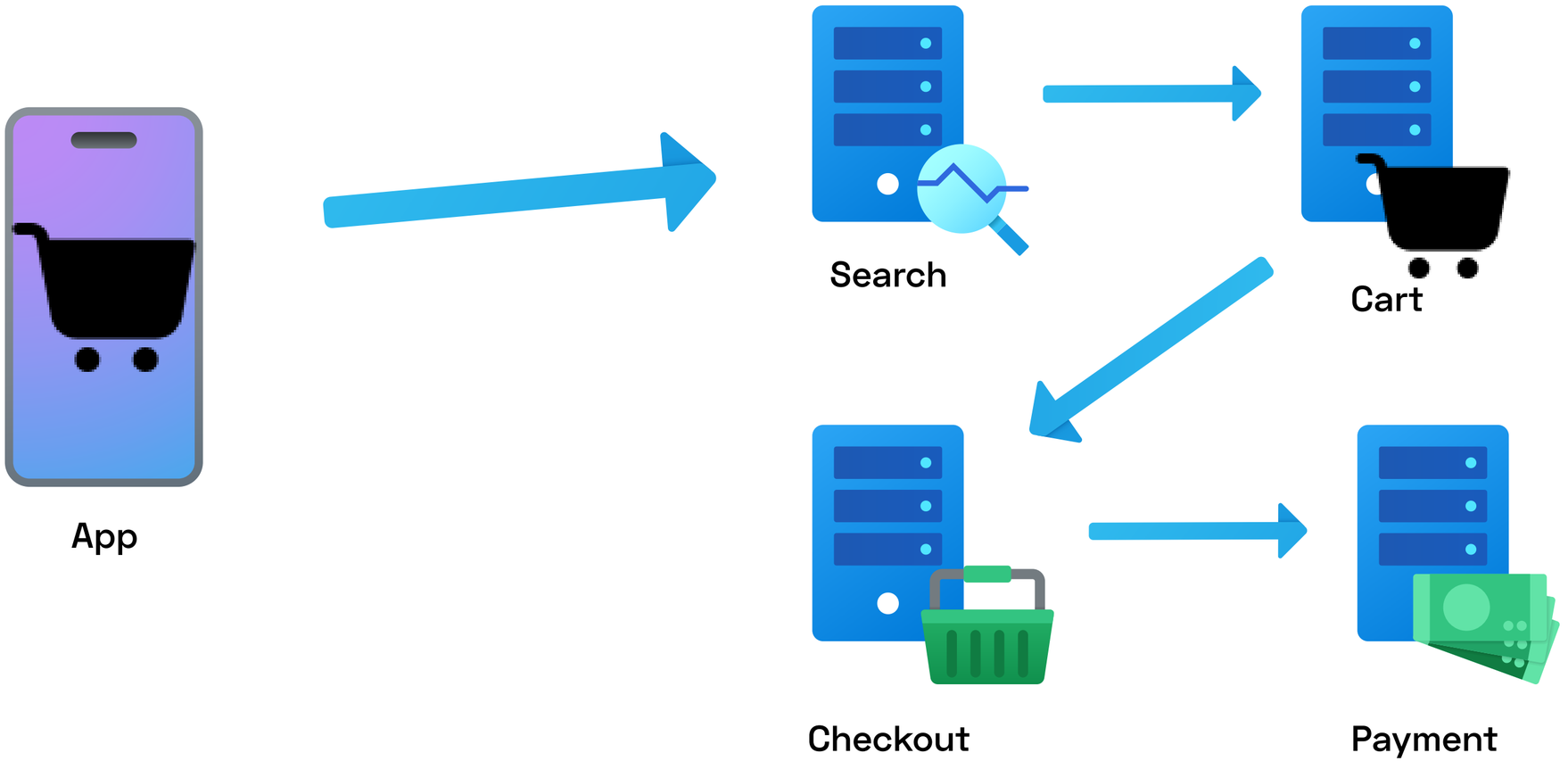
Very slow app. I don't recommend using this app!

Try Pitch

SITUATION: AN ECOMMERCE APP

Enter Monitoring

- spot issues before they escalate
- monitor the average response time
- receive notifications when for a specified threshold.



Try Pitch

MONITORING:

REDUCE DOWNTIME

INCREASE PERFORMANCE

PROVIDE BETTER USER EXPERIENCE

Try Pitch

MONITORING:

REDUCE DOWNTIME

INCREASE PERFORMANCE

PROVIDE BETTER USER EXPERIENCE

Monitoring tools like Prometheus and Grafana help us to track the health of individual components including servers, databases, network devices, and other infrastructure.

Try Pitch

MONITORING:

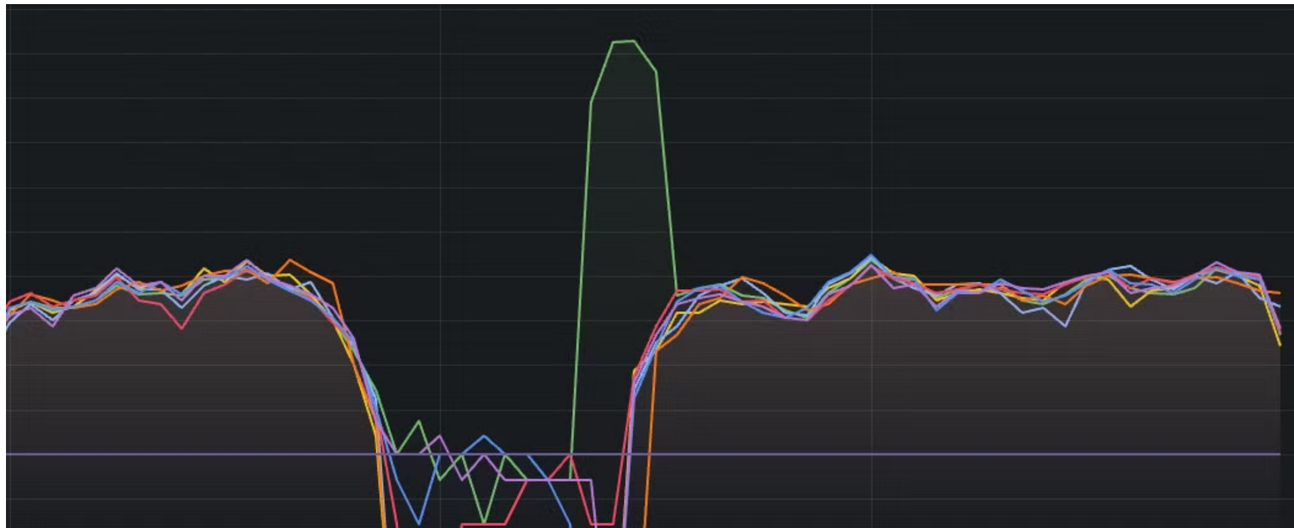
REDUCE DOWNTIME

INCREASE PERFORMANCE

PROVIDE BETTER USER EXPERIENCE

Monitoring tools like Prometheus and Grafana help us to track the health of individual components including servers, databases, network devices, and other infrastructure.

With monitoring, we can measure and consolidate key metrics of an application that are relevant to the business and actionable. These essential metrics can then be displayed on a dashboard and used to trigger alerts.



Try Pitch

MONITORING:

WHAT KIND OF METRICS SHOULD WE MONITOR?

Monitoring everything is not a best practice, so to implement monitoring effectively, we need to know what to track.

Try Pitch

MONITORING:

WHAT KIND OF METRICS SHOULD WE MONITOR?

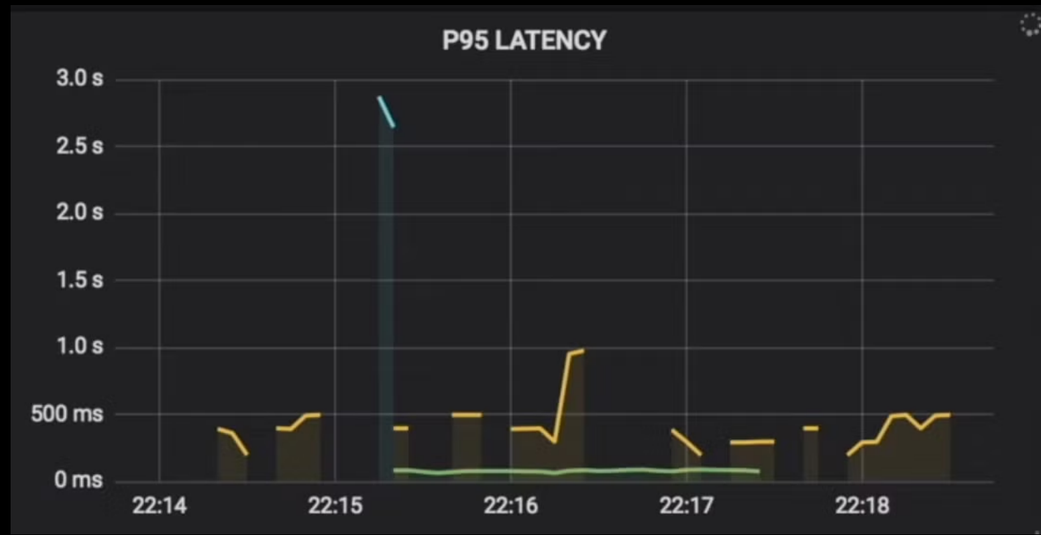
Monitoring everything is not a best practice, so to implement monitoring effectively, we need to know what to track.

LATENCY
TRAFFIC
ERROR RATE
SATURATION

Try Pitch

LATENCY

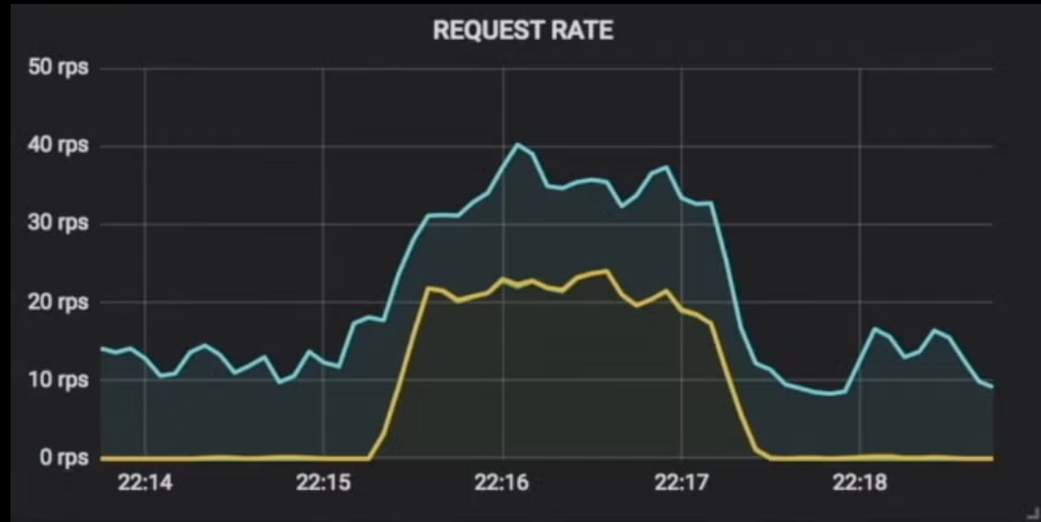
- Latency measures the time it takes for a request to travel from the client to the server and back.
- It measures how fast a response we are getting. Lower latency generally leads to a better user experience.
- In our case, when users search for a product, fast loading of search results is important. If search results take many seconds to appear, users start shifting to faster alternatives.



Try Pitch

TRAFFIC

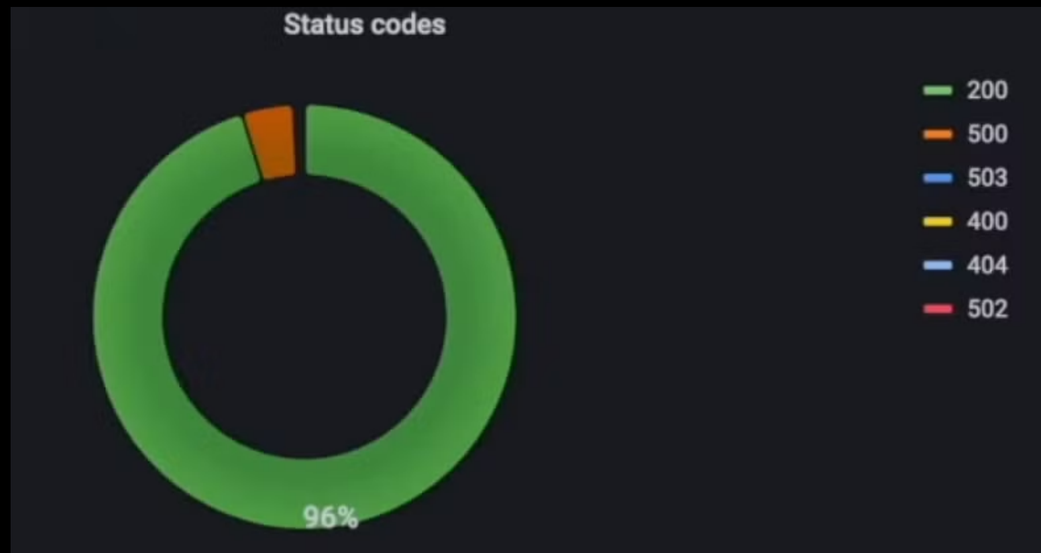
- Traffic denotes the number of requests a system receives over a specific period. Monitoring traffic ensures the system can manage load and scale appropriately as required.
- Again, in our case, during a flash sale event on our e-commerce web application, a significant increase in traffic can occur as users rush to make purchases.
- Monitoring this traffic helps ensure that the system can handle the load and scale appropriately whenever needed.



Try Pitch

ERROR RATE

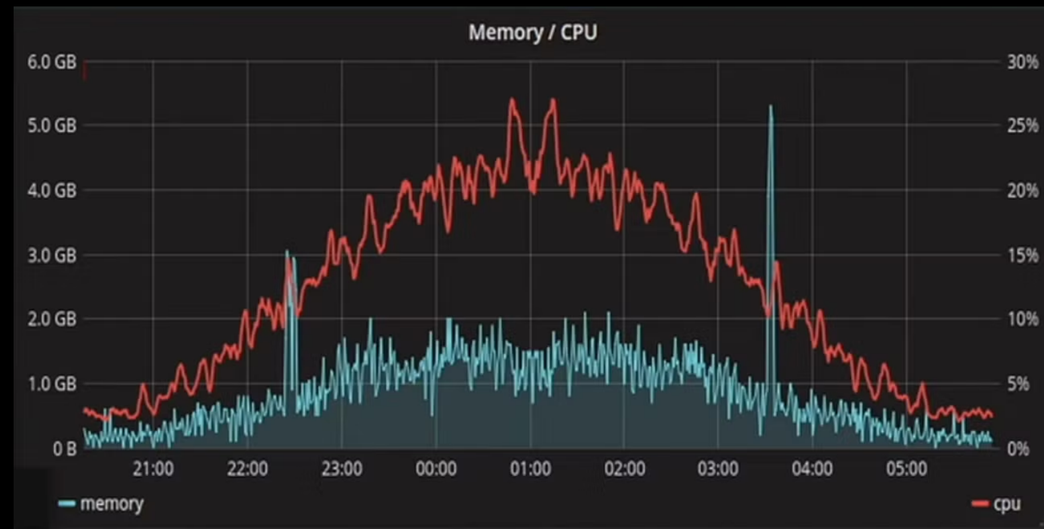
- The error rate metric represents the percentage of requests resulting in errors such as 404 page not found errors or 500 internal server errors.
- A high error rate indicates system instability or issues. Saturation measures resource utilization, including CPU, memory, and disk space.
- Heavy resource utilization can lead to performance degradation.



Try Pitch

SATURATION

- Saturation measures resource utilization, including CPU, memory, and disk space. Heavy resource utilization can lead to performance degradation.
- For example, if a server's CPU usage remains near 90% due to extensive processing, it means it is saturated.
- Such high CPU usage might delay the processing of user requests, affecting the application's responsiveness.
- Monitoring saturation always helps us to prevent performance degradation due to resource outages.



Try Pitch

By tracking these four golden metrics, teams can easily evaluate overall performance and application health, enabling informed decisions about optimization, scaling, and enhancements.

Without monitoring, we’re operating blindly as there is no way to determine if the service is even functioning.

Try Pitch

MONITORING BEST PRACTICES

- ✓ Monitoring should be adopted as soon as possible.
- ✓ We must take time to figure out what things we should monitor. Pay attention to the four golden signals.
- ✓ Dashboard and alert messages must easy to understand and gets right to the point.
  If we send too many alerts, people might start ignoring them because there are just too many. So always limit the alerts based on priority.

Try Pitch

So far, we learned that with monitoring, we can identify issues such as a specific service running slow or running low on CPU, and we get notified of such issues.

But getting these notifications alone isn't always enough.

Whenever any such issue arises, developers need sufficient data to fully detect the root cause and apply a permanent fix instead of applying some random fixes which waste time and resources. That is where observability comes into the picture.

Try Pitch

OBSERVABILITY + MONITORING

- When an alert is received from monitoring, SREs or developers investigate using observability data.
- Both serve the goal of identifying problems and delivering a high-quality user experience.
- Monitoring and observability are different concepts that complement each other.

Try Pitch

OBSERVABILITY + MONITORING

Key Differences:

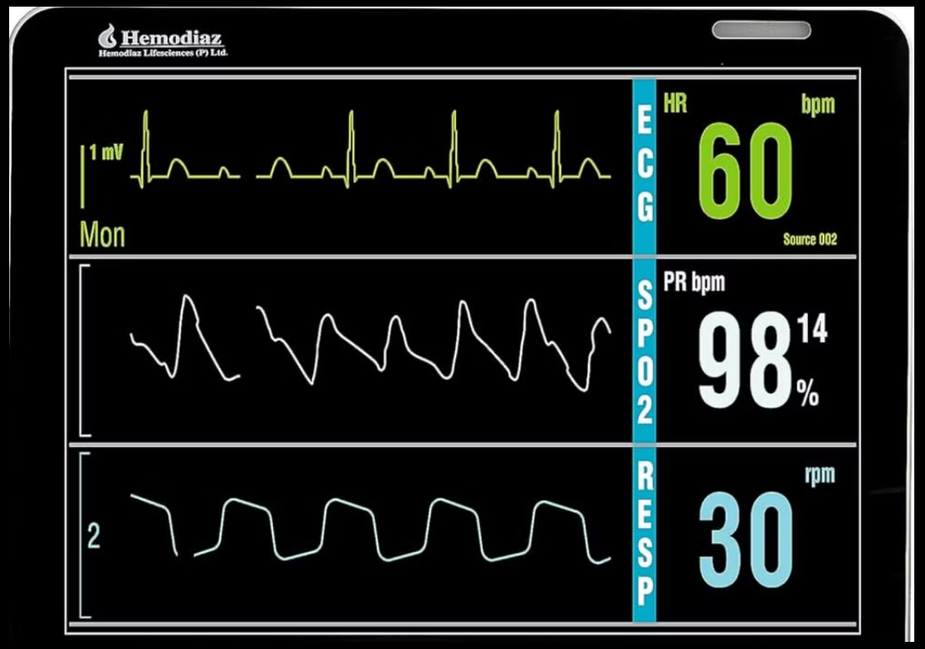
- **Monitoring:**
  - Alerts us when something goes wrong.
  - Example: Alerts when the application response rate is slow.
- **Observability:**
  - Helps us understand why there is an error and how to fix it.
  - Example: Identifies which specific microservice is causing the problem, allowing for a permanent fix.

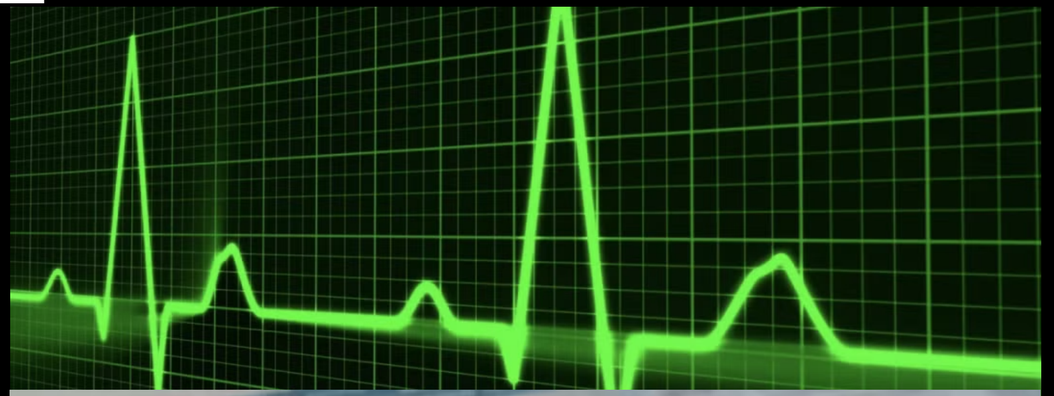
Practical Example:

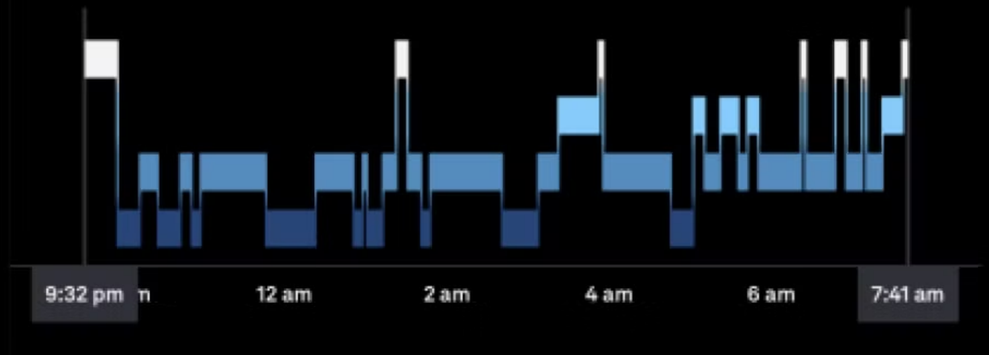
- **Monitoring:**
  - Detects that application response rate is no longer fast.
- **Observability:**
  - Pinpoints the specific microservice causing the slowdown.
  - Provides the necessary data to apply a permanent fix.

Try Pitch

ONE FINAL EXAMPLE







Try Pitch

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)