Building Observability into Cloud-Native Applications

Muhammad Ahmad Saeed

What is Observability?

- Ability to understand and gain insights into the internal state of a system based on the data it produces.

- Observability vs Monitoring
  - Monitoring: something is wrong
  - Observability: why it's wrong

Pillars of Observability

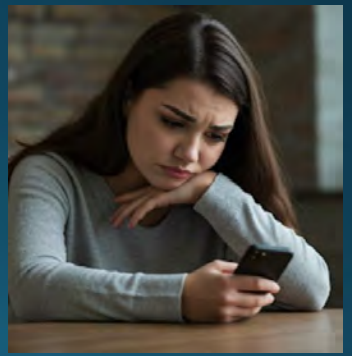
- Logs
  - Timestamped records of events that occurred in the system.
- Metrics
  - Quantitative data about system performance.
- Traces
  - End-to-end tracking of requests as they flow through distributed systems.

Why is Observability Crucial for Cloud-Native?

- Distributed: Microservices communicate over networks, introducing latency and failure points.
- Dynamic: Containers and orchestration platforms like Kubernetes constantly spin up and tear down resources.
- Ephemeral: Instances are short-lived, making it harder to track issues.

Lack of Observability

- Increased Mean Time to Resolution (MTTR)
  - Resolving issues takes significantly longer
- Limited Performance Optimization
  - Optimizing performance is a guessing game
- Difficulty in Ensuring Reliability
  - Reliability of the entire application becomes challenging

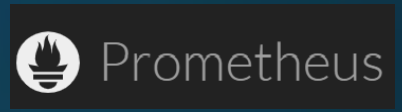


Implementing Observability in Cloud-Native Applications

- **Instrumentation**
  - Instrument code to collect metrics, logs, and traces.
- **Centralized Collection and Storage**
  - Collect data from all the services and store it in a centralized location.
- **Visualization and Analysis**
  - Use dashboards and visualization tools explore your data.
- **Contextualization**
  - Connect your metrics, logs, and traces together to provide a holistic view of your system.
- **Automation**
  - Automate the process of collecting, analyzing, and visualizing the observability data.

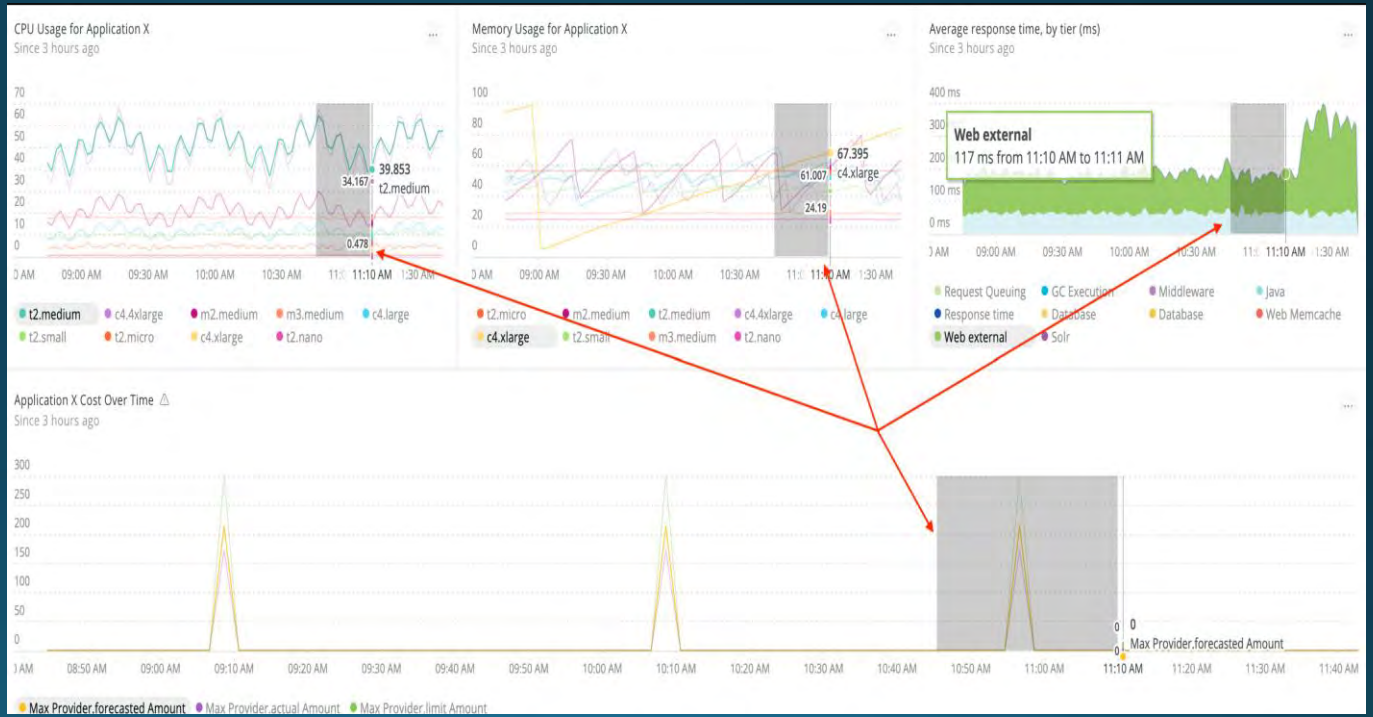
Tools and Technologies:

- Metrics: Prometheus, StatsD, Telegraf
- Logs: Elasticsearch, Loki, Fluentd, Filebeat
- Traces: Jaeger, Zipkin, OpenTelemetry
- Visualization: Grafana, Kibana



Platforms for Cloud-Native Observability

- Datadog
- New Relic
- Honeycomb



Best Practices

- Instrument early and often
- Use meaningful metrics
- Structure logs
- Correlate data
- Automate observability pipeline
- Cost Management

Case Study: Observability in Action

Scenario:

A cloud-native e-commerce application experiences periodic slowdowns.

Solution:

- **Logs** identify increased error rates in a microservice.
- **Metrics** reveal high CPU utilization on a node.
- **Traces** pinpoint a slow database query affecting response times.
- **Resolution**: Optimize the query, autoscale the service, and reduce latency.

Thank you