PW SKILLS | DevOps and Cloud Computing

Monitoring Fundamentals

Objective

- Differentiate between infrastructure monitoring and application monitoring and identify key metrics to track.
- Set up basic monitoring configurations in AWS CloudWatch Prometheus, or Datadog.
- Implement instrumentation techniques to collect and visualize application metrics.
- Build real-time dashboards and alerts to track system health and performance.
- Understand active vs. historical monitoring and how they support operational decisions.

PW SKILLS

Explaining infrastructure monitoring and its focus on hardware and system performance.

Let’s see

Infrastructure monitoring is the practice of observing and managing the physical and virtual components of IT systems, such as servers, storage devices, networks, and operating systems.

- It focuses on hardware and system performance by tracking metrics like CPU usage, memory consumption, disk I/O, and network activity.
- The main goal is to detect problems early, prevent downtime, optimize resource use, and ensure systems stay reliable and efficient.

PW SKILLS

Discussing key infrastructure metrics

Let’s discuss

- **CPU usage:** Monitors processing load to spot overutilization or underutilization.
- **Memory usage:** Detects memory leaks and ensures efficient memory management.
- **Disk I/O:** Measures read/write speeds to find storage bottlenecks.
- **Network performance:** Tracks bandwidth and latency to identify network issues.

PW SKILLS

Explaining application monitoring and how it tracks service-level performance.

Let’s see

Application monitoring is the process of continuously observing and analyzing software applications to ensure they perform correctly and efficiently.

- It tracks service-level performance by measuring metrics such as response times, error rates, transaction volumes, and uptime.
- This helps detect issues early, maintain user satisfaction, and meet service-level agreements (SLAs).

Pop Quiz

Q. Datadog can monitor which of the following?

A

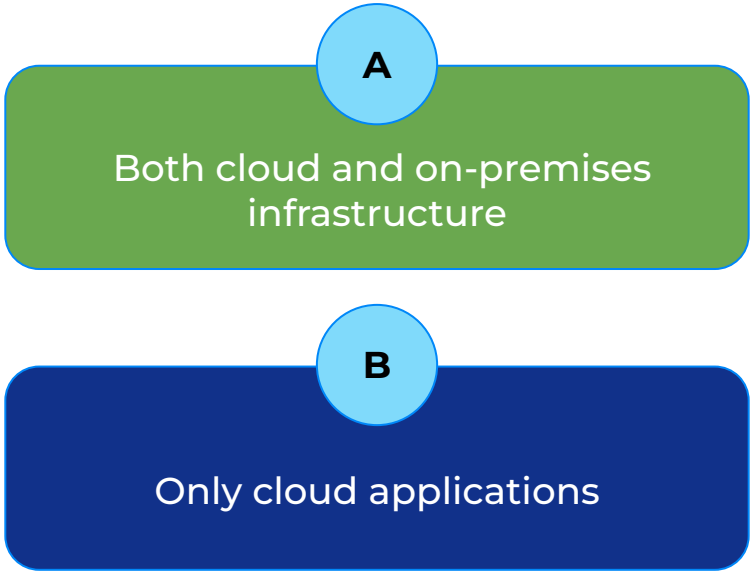
Both cloud and on-premises infrastructure

B

Only cloud applications

Pop Quiz

Q. Datadog can monitor which of the following?



PW SKILLS

Discuss key application metrics

Let’s discuss

- **Error rates:** Show the percentage of failed requests, indicating reliability issues.
- **Request latency:** Measures how long it takes to respond to a request, reflecting performance speed.
- **Throughput:** Tracks the number of requests handled per second, showing application capacity.

PW SKILLS

Explaining how monitoring tools collect and analyze metrics.

Let’s see

- Monitoring tools collect metrics by using agents, APIs, or log scraping to gather data from systems and applications.
- They then analyze this data in real time to detect patterns, identify issues, and generate alerts or reports for better decision-making.

PW SKILLS

Demonstrating basic configuration in AWS CloudWatch

Let’s do it

In AWS CloudWatch:

- **Create a dashboard:** Go to CloudWatch, choose Dashboards, click Create dashboard, add a widget, and select your EC2 instance metrics (like CPU usage).
- **Set up an alarm:** In Alarms, click Create alarm, choose the EC2 instance’s CPU utilization metric, set a threshold (e.g., above 80%), and configure notifications (like sending an email).

PW SKILLS

Introducing Datadog and Prometheus as alternative monitoring solutions.

Datadog & Prometheus

Datadog and Prometheus are popular alternatives for monitoring:

- **Datadog:** A cloud-based platform that offers real-time monitoring, dashboards, and alerting for applications, infrastructure, and services.
- **Prometheus:** An open-source tool that collects and stores metrics in a time-series database, ideal for highly customizable and scalable monitoring setups.

Take A 5-Minute Break!

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

Explaining instrumentation and why developers need to expose custom application metrics.

Let’s see

Instrumentation is the process of embedding code or using libraries within an application to collect and expose metrics, logs, and traces that provide insights into its performance and behavior.

- Developers expose custom application metrics to monitor specific business logic, track application-specific events, and optimize performance.
- This helps detect issues early, improve troubleshooting, and ensure the application meets desired performance standards.

PW SKILLS

Demonstrating how Prometheus client libraries can collect real-time application metrics.

Let’s do it

1. Install the Prometheus client:

```
pip install prometheus_client
```

2. Create metrics and start HTTP server (in Python):

```python
from prometheus_client import start_http_server, Counter
import random
import time

REQUESTS = Counter('app_requests_total', 'Total number of requests')
start_http_server(8000)

while True:
    REQUESTS.inc(random.randint(1, 10))  # Simulate random requests
    time.sleep(5)  # Update every 5 seconds
```

3. Expose metrics: Metrics are available at **http://localhost:8000/metrics**, and Prometheus can scrape them.

PW SKILLS

Discussing Grafana for visualizing Prometheus data in dashboards.

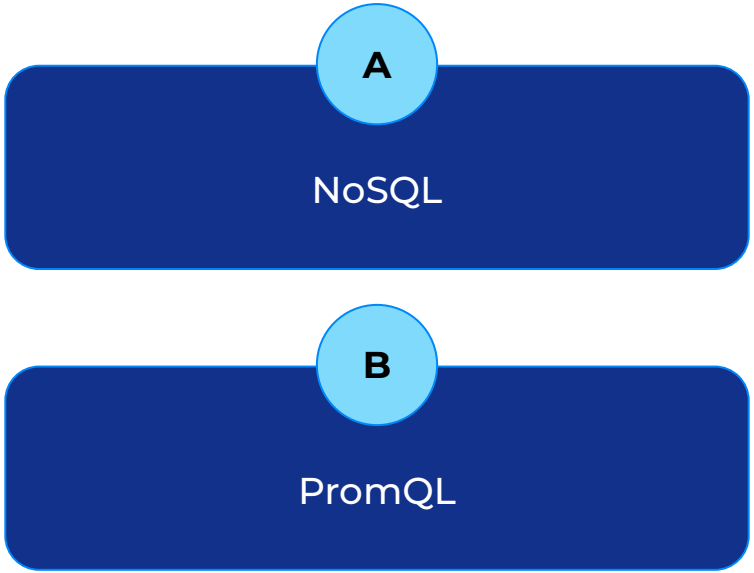
Let’s discuss

Grafana is a visualization tool that integrates with Prometheus to create interactive dashboards.

- It allows users to visualize Prometheus metrics using customizable panels like graphs and tables. With PromQL queries, Grafana fetches data from Prometheus and displays real-time metrics, such as CPU usage and request latency.
- It also supports alerting, enabling proactive monitoring based on metric thresholds.

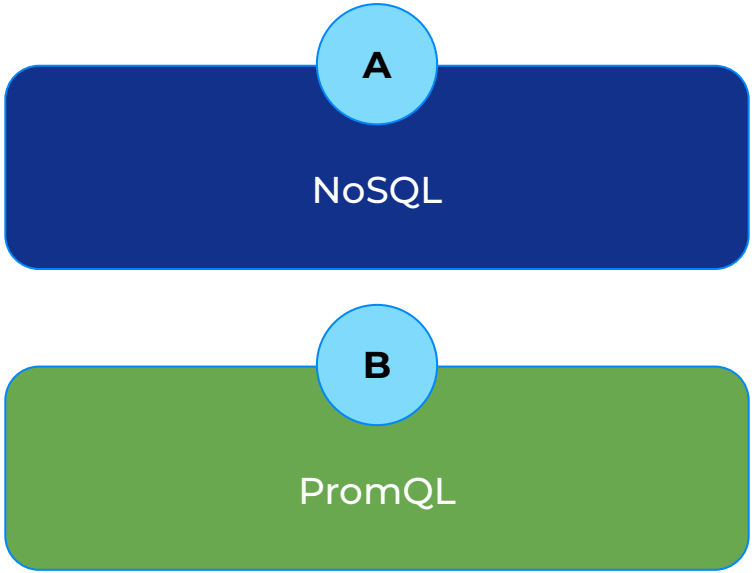
Pop Quiz

Q. Which query language does Grafana use to fetch data from Prometheus?



Pop Quiz

Q. Which query language does Grafana use to fetch data from Prometheus?



PW SKILLS

Explaining why dashboards are essential for observability.

Let’s see

Dashboards are essential for observability because they provide a centralized, visual representation of key metrics and performance data.

- They help quickly identify issues, track system health, and monitor trends in real time, enabling faster decision-making and proactive problem resolution.

PW SKILLS

Demonstrating how to create real-time dashboards in Grafana or CloudWatch.

Let’s do it

In Grafana:

- Add Prometheus as a data source.
- Create a new dashboard, then add a panel.
- Use PromQL queries to pull metrics (e.g., avg(rate(cpu usage[5m]))).
- Customize the visualization (e.g., graph, gauge).
- Save the dashboard and set up alerts if needed.

Let’s do it

In CloudWatch:

- Go to the CloudWatch Console.
- Create a new Dashboard.
- Add widgets (e.g., graphs, numbers) to visualize EC2 or custom metrics.
- Choose metrics from CloudWatch and adjust the time range.
- Save and share the dashboard for real-time monitoring.

PW SKILLS

Discuss setting up alert thresholds

Let’s discuss

- High CPU usage: Set an alert if CPU usage exceeds 80% for a certain period.
- Error rate spikes: Configure alerts for significant increases in error rates, indicating potential system failures.
- Memory leaks: Set alerts for unusually high or increasing memory usage, which could point to memory leaks.
- Slow response times: Define an alert for response times that exceed a threshold, indicating performance degradation.

PW SKILLS

Demonstrating configuring alerts via email, Slack, and webhook integrations.

Let’s do it

In Grafana:

1. Go to Alerting > Notification channels.
2. Add a new channel: choose Email, Slack, or Webhook.
    - For Email, enter the SMTP server settings.
    - For Slack, provide the Slack Webhook URL.
    - For Webhook, specify the URL to send alerts to.
3. Create an alert rule on a panel and configure it to trigger the chosen notification channel.

Let’s do it

In CloudWatch:

1. Create an Alarm with the desired metric and threshold.
2. Under Actions, select Send a notification.
3. Choose an existing SNS topic or create a new one, and subscribe to email, Slack, or webhook integrations via SNS.

PW SKILLS

Explaining the difference between active (real-time) monitoring and historical monitoring.

Let’s see

- **Active (Real-time) Monitoring** focuses on continuously tracking and analyzing system performance as it happens. It alerts teams instantly when metrics exceed predefined thresholds, allowing for quick issue resolution and minimizing downtime.
- **Historical Monitoring** involves reviewing past data to identify trends, patterns, or recurring issues over time. It helps with performance analysis, capacity planning, and understanding long-term system behavior, but it doesn't provide immediate alerts.

PW SKILLS

Discussing use cases for each

Let’s discuss

- **Real-time monitoring:** Used for incident detection, where it helps identify issues like system failures, high CPU usage, or performance degradation as they occur, enabling immediate response.
- **Historical monitoring:** Used for trend analysis and capacity planning, helping track long-term performance patterns, forecast future resource needs, and plan infrastructure growth.

PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this sessio
- Review your lecture notes
- Note down questions and queries regarding this session ar
  consult the teaching assistants

Thanks !