SKILLS | DevOps and Cloud Computing

Cloud & Application Monitoring for DevOps

Objective

- Automate cloud monitoring for instances, databases, and network components.
- Understand the importance of monitoring in DevOps for system reliability and performance.
- Identify and track critical application metrics such as latenc error rates, and resource utilization.
- Explore infrastructure monitoring tools like Prometheus, Datadog, and Nagios.

PW SKILLS

Explaining the importance of automated cloud monitoring in DevOps.

Let’s see

Automated cloud monitoring is vital in DevOps because it enables continuous visibility into infrastructure and application performance.

- It helps detect issues early, supports rapid troubleshooting, and ensures system reliability and uptime.
- By automating this process, teams can proactively respond to incidents, optimize resources, and maintain consistent performance during frequent deployments, which aligns with the core DevOps goals of speed, efficiency, and reliability.

PW SKILLS

Discussing how cloud providers offer built-in monitoring tools (AWS CloudWatch, Azure Monitor, GCP Stackdriver).

Let’s discuss

Cloud providers offer built-in monitoring tools to help manage and optimize cloud resources:

- AWS CloudWatch collects and tracks metrics, monitors log files, sets alarms, and automatically reacts to changes in AWS resources.
- Azure Monitor gathers telemetry data from applications and infrastructure, offers advanced analytics, and integrates with automation tools for proactive management.
- GCP Stackdriver (now part of Google Cloud Operations Suite) provides monitoring, logging, and diagnostics for applications on Google Cloud and beyond, supporting multi-cloud and hybrid environments.

PW SKILLS

Demonstrating setting up CloudWatch Alarms to track EC2 performance (CPU, memory, disk usage).

Let’s do it

To set up CloudWatch Alarms for EC2 performance (CPU, memory, disk usage):

1. Open AWS CloudWatch Console.
2. Go to Alarms > Create Alarm.
3. Select a metric:
    - For CPU: EC2 > Per-Instance Metrics > CPUUtilization.
    - For Memory/Disk (requires CloudWatch Agent): CWAgent > Metrics.
4. Set a threshold (e.g., CPU > 80% for 5 minutes).
5. Configure actions: send notifications via SNS or trigger auto-scaling.

    Name and create the alarm.

Pop Quiz

Q. Which metric is natively available in CloudWatch for EC2 instances without installing extra agents?

A

CPU utilization

B

Memory usage

Pop Quiz

Q. Which metric is natively available in CloudWatch for EC2 instances without installing extra agents?

A

CPU utilization

B

Memory usage

Explaining the importance of regular backups and disaster recovery in cloud environments.

Let’s see

- Regular backups and disaster recovery are crucial in cloud environments to protect data from loss, corruption, or unexpected failures.
- They ensure business continuity, minimize downtime, and enable quick restoration of services during incidents like cyberattacks, hardware failures, or natural disasters.
- Backups protect against data loss from accidental deletion, corruption, cyberattacks, or system failures.
- Disaster recovery ensures that, in the event of major disruptions like outages or natural disasters, systems can be quickly restored with minimal downtime.

Discussing how to automate EBS snapshots, S3 backups, and RDS automated backups

Let’s discuss

To automate cloud backups:

- **EBS Snapshots:** Use Amazon Data Lifecycle Manager to automatically create and delete EBS snapshots based on schedules.
- **S3 Backups:** Enable S3 Versioning and Lifecycle Policies to back up and transition data to cheaper storage (like Glacier) automatically.
- **RDS Automated Backups:** Turn on automated backups in RDS settings, setting the backup window and retention period to regularly capture snapshots and transaction logs.

Pop Quiz

Q. Which AWS service is commonly used to automate EBS snapshot creation and deletion?

A

CloudFront

B

Data Lifecycle Manager (DLM)

Pop Quiz

Q. Which AWS service is commonly used to automate EBS snapshot creation and deletion?

A

CloudFront

B

Data Lifecycle Manager (DLM)

PW SKILLS

Explaining failover strategies such as multi-region replication and auto-recovery mechanisms.

Let’s see

Failover strategies like multi-region replication and auto-recovery mechanisms ensure high availability and resilience.

- Multi-region replication copies data and services across regions, so if one region fails, traffic can shift to another.
- Auto-recovery automatically restarts failed instances or services without manual intervention, minimizing downtime and maintaining business continuity.

Take A 5-Minute **Break!**

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

Explaining why monitoring is critical for uptime, performance, and troubleshooting.

Let’s see

Monitoring is crucial for several reasons:

- **Uptime:** It helps detect and alert teams about potential issues, allowing them to take corrective actions before they lead to downtime.
- **Performance:** Continuous tracking of system performance ensures resources are used efficiently and helps optimize workloads, preventing slowdowns or bottlenecks.
- **Troubleshooting:** Monitoring provides real-time insights, logs, and metrics, enabling rapid diagnosis of issues and faster resolution, minimizing the impact on users and business operations.

Pop Quiz

Q. How does monitoring improve system performance?.

A

By reducing application code.

B

By identifying resource bottlenecks and optimizing usage.

Pop Quiz

Q. How does monitoring improve system performance?.

A

By reducing application code.

B

By identifying resource bottlenecks and optimizing usage.

PW SKILLS

Discussing the role of monitoring in incident response and proactive problem detection.

Let’s discuss

Monitoring is essential for both incident response and proactive problem detection:

- **Incident Response:** Monitoring tools provide real-time alerts and detailed logs, enabling teams to quickly identify and respond to incidents, reducing downtime and minimizing business impact.
- **Proactive Problem Detection:** By continuously tracking system performance, monitoring can identify abnormal patterns or early signs of potential issues (like resource bottlenecks or security threats).

PW SKILLS

Providing real-world examples of system failures caused by a lack of monitoring.

Real-world examples

Real-world examples of system failures due to lack of monitoring include:

- **Amazon Web Services (AWS) Outage (2017):** Lack of proper monitoring and alerting led to a major S3 service disruption, causing outages for many websites.
- **Knight Capital Group (2012):** A lack of real-time monitoring in their trading system led to a software glitch that caused a $440 million loss in 45 minutes.
- **Target Data Breach (2013):** Inadequate monitoring of network traffic allowed hackers to access sensitive customer data, leading to a massive breach.

Defining and explaining key performance indicators (KPIs)

Let’s see

Key Performance Indicators (KPIs) are metrics used to measure system performance:

- **Latency:** The time it takes for a system to process and respond to a request. Lower latency indicates faster response times.
- **Error Rates:** The percentage of requests that fail or return errors. High error rates signal issues that need attention.
- **Throughput:** The number of requests the system can handle per second. Higher throughput indicates better capacity and performance.
- **Request Volume:** The total number of incoming requests over a set period. Monitoring request volume helps assess system load and potential bottlenecks.

PW SKILLS

Discussing how these metrics impact user experience and system stability.

Let’s discuss

These metrics directly impact user experience and system stability:

- **Latency:** High latency leads to slower response times, frustrating users and potentially causing them to abandon the service.
- **Error Rates:** A high error rate creates a poor user experience, as users encounter failures, which can damage trust and satisfaction.
- **Throughput:** Low throughput limits the system’s ability to handle traffic, leading to slow performance or outages during peak demand.
- **Request Volume:** Sudden spikes in request volume without adequate handling can overload the system, resulting in crashes or slowdowns.

PW SKILLS

Explaining how different applications require different monitoring strategies.

Let’s see

Different applications require tailored monitoring strategies based on their unique needs:

- **Web Applications:** Focus on response time, throughput, and error rates to ensure fast, reliable user interactions.
- **Databases:** Prioritize query performance, latency, and resource utilization to ensure efficient data retrieval and storage.
- **Microservices:** Monitor service-to-service communication, error rates, and latency to ensure smooth interactions across services.
- **Mobile Apps:** Track app crashes, network latency, and battery usage to optimize user experience across devices.

Discussing how to identify business-critical metrics for web apps, APIs, databases, and services.

Let’s discuss

To identify business-critical metrics for different components:

- **Web Apps:** Focus on response time, error rates, and conversion rates to ensure fast, reliable, and successful user interactions.
- **APIs:** Monitor latency, request success rate, and throughput to ensure smooth, reliable communication between services.
- **Databases:** Track query performance, read/write latency, and resource utilization to ensure fast data access and minimal downtime.
- **Services:** Measure uptime, service response time, and error rates to ensure availability, reliability, and smooth operations.

Providing examples of monitoring dashboards for different types of applications.

Examples

Here are examples of monitoring dashboards for different types of applications:

1. **Web Applications:**
   - Dashboard: A visual representation with line charts for response times, pie charts for user demographics, and bar graphs for traffic volume across different time periods.
2. **APIs:**
   - Dashboard: Real-time graphs displaying the number of requests per second, error counts, latency trends, and success/failure ratio.
3. **Databases:**
   - Dashboard: Tables and graphs showing query execution times, memory utilization trends, I/O operations, and active connections over time.
4. **Microservices/Services:**
   - Dashboard: A service health overview with status indicators (green/yellow/red), latency charts, and error tracking by service or endpoint.

PW SKILLS

Explaining why CPU, memory, disk, and network monitoring are essential for performance optimization.

Let’s see

- **CPU:** Monitors processing power to prevent overloads that slow down tasks.
- **Memory:** Tracks available memory to avoid crashes or slowdowns due to insufficient resources.
- **Disk:** Ensures fast data read/write speeds and prevents slowdowns from full or fragmented storage.
- **Network:** Monitors bandwidth and latency to avoid bottlenecks that can affect data transfer speeds.

Discussing common resource bottlenecks and troubleshooting techniques.

Let’s discuss

Common resource bottlenecks and troubleshooting techniques include:

- CPU Bottleneck
- Memory Bottleneck
- Disk Bottleneck
- Network Bottleneck

PW SKILLS

Demonstrating how to track system metrics using CloudWatch, Prometheus, or Datadog.

Let’s do it

CloudWatch:

- Navigate to CloudWatch Console.
- Select Metrics and choose your AWS service (e.g., EC2, S3).
- Create custom metrics or view predefined ones like CPU usage, memory, and disk I/O.
- Set up alarms for threshold-based notifications.

Let’s do it

Prometheus:

- Install Prometheus and configure targets (e.g., nodes, services).
- Define Prometheus queries to collect metrics like CPU usage, memory, etc.
- Visualize metrics with Grafana dashboards.

Let’s do it

Datadog:

- Install the Datadog Agent on your system.
- Configure integrations for cloud services, databases, or containers.
- View metrics like CPU, memory, and disk usage through Datadog’s dashboard.
- Set up alerts for thresholds or anomalies.

Discussing infrastructure monitoring best practices for servers, databases, and network resources.

Let’s discuss

Infrastructure monitoring best practices include:

- **Servers:** Track CPU, memory, disk usage, and uptime; set alerts for threshold breaches; regularly review logs for unusual activity.
- **Databases:** Monitor query performance, connection counts, replication status, and backup health to ensure data integrity and fast access.
- **Network Resources:** Measure bandwidth usage, latency, packet loss, and firewall health; use tools to detect and respond to traffic spikes or outages quickly.

PW SKILLS

Introducing monitoring tools and their use cases.

Let’s see

Here’s a quick intro to key monitoring tools:

- **Prometheus:** Ideal for metrics collection and alerting in dynamic environments like microservices and Kubernetes.
- **Datadog:** Best for cloud-native application monitoring, offering real-time metrics, logs, and tracing across distributed systems.
- **Nagios:** Great for traditional infrastructure monitoring, keeping tabs on servers, network devices, and on-premises resources.

PW SKILLS

Time for case study!

Important

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

Thanks
!