DevOps Intern Assignment – Theory Explanation

Step 1: Environment Setup

We started by creating a cloud server using AWS EC2 with Ubuntu as the operating system. After launching the instance, we connected to it securely using SSH. A new user named devops_intern was created to simulate a real DevOps working environment. This user was given passwordless sudo access to allow smooth execution of administrative tasks. The server hostname was also changed to a custom name to make system identification easier and more professional.

Step 2: Web Server Configuration

Next, we installed the Nginx web server on the EC2 instance. A simple HTML page was created and configured to display important system information such as the user's name, EC2 instance ID, and server uptime. This confirmed that the server was accessible publicly and that the web service was running correctly.

Step 3: Monitoring Script Creation

We then created a monitoring script named system_report.sh. This script collects system performance data including date and time, uptime, CPU usage, memory usage, disk usage, and the top three CPU-consuming processes. The output of this script was continuously stored in a log file for future reference and system tracking.

Step 4: Automation of Monitoring

To automate execution, the script was scheduled to run every five minutes. Initially this was done using cron, but later a more professional approach was implemented using systemd service and timer. This ensured that the script runs reliably even after system reboots and follows industry-standard automation practices.

Step 5: AWS CloudWatch Integration

To enable centralized monitoring, the system logs were connected to AWS CloudWatch. An IAM user was created for secure AWS CLI access. The log data generated by the monitoring script was formatted and pushed to a CloudWatch Log Group and Log Stream, allowing system performance to be viewed directly from the AWS dashboard.

Step 6: Bonus – Proactive Alert System

As an advanced feature, an email alert system was added. The monitoring script was enhanced to check disk usage and send an automatic email whenever disk usage exceeded 80%. This helps prevent system failures and demonstrates real-world proactive monitoring techniques.