DevOps Internship Assignment

Log Analysis and Monitoring Script

Objective:

Create a script that automates the analysis and monitoring of log files

Requirements:

- Write the script using bash shell / Python scripting.
- Implement basic log analysis and monitoring functionalities.
- Include error handling and logging to provide feedback on script execution.

Tasks:

Log File Monitoring:

- Write a script that continuously monitors a specified log file for new entries.
- Use tail or similar commands to track and display new log entries in real time.
- Implement a mechanism to stop the monitoring loop (e.g., using a signal like Ctrl+C).

Log Analysis:

- Enhance the script to perform basic analysis on log entries:
- Count occurrences of specific keywords or patterns (e.g., error messages, HTTP status codes).
- Generate summary reports (e.g., top error message)

Deliverables:

- The completed shell script (log-monitor.sh) / python script (log-monitor.py) with necessary functionalities implemented.
- A brief README.md file explaining how to use and test the script, including any prerequisites or dependencies.
- Host the solution on GitHub and share the repository link

Evaluation Criteria:

- Correctness and completeness of the script's functionality.

- Clarity of code structure, comments, and variable naming.
- Error handling and robustness of the script.
- Creativity and additional features implemented beyond basic requirements.

Hint

To create a Python application that alternately logs INFO, DEBUG, and ERROR messages, you can use the logging module in Python. This script will continuously loop and log messages at different levels (INFO, DEBUG, ERROR) to demonstrate logging behavior for monitoring purposes. Below is an example of such a script:

```
import logging
import time
import random

# Configure logging
logging.basicConfig(level=logging.DEBUG)  # Set the root logger level to
DEBUG

# Create a logger
logger = logging.getLogger(__name__)

# Define log message formats
formats = {
    logging.INFO: "INFO message",
    logging.DEBUG: "DEBUG message",
    logging.ERROR: "ERROR message"
}

# Define log levels to cycle through
log_levels = [logging.INFO, logging.DEBUG, logging.ERROR]

# Main loop to log messages
while True:
    try:
        # Randomly select a log level
        log_level = random.choice(log_levels)

        # Get the log message format for the selected log level
        log_message = formats[log_level]

        # Log the message
        logger.log(log_level, log_message)
```

```
        # Sleep for a short interval
        time.sleep(1)

    except KeyboardInterrupt:
        # Handle keyboard interrupt (Ctrl+C)
        print("\nLogging interrupted. Exiting.")
        break
```