Ghulam Ishaq Khan Institute (GIKI)

DevOps Assignment # 2

Subject: DevOps	**Course Code:** CS - 328 – Spring - 2025
Class: BS CY **Batch:** Fall – 2022	**Submission Deadline:** 20 Feb 2025 Thursday (11:00 AM)
Course Instructor: Muhammad Ahmad Nawaz	**Total Marks:** 10
Course TA: Muhammad Hamza	

Note (Read notes & instructions first)

- First, read the instructions and statements of each exercise/question carefully then write the solution.
- *For Assignment:*
  - The name of your zip file should contain your assignment number and your roll number as shown in following example, For Example if your roll number is 2022532 and you have done assignment number 1 then the name of file should be as ---> 2022532_1.zip
    - monitor_and_push.sh
    - config.cfg
    - README.md
    - Any additional files used for testing.
  - Then upload that pdf file at Microsoft teams. Remember the sequence of pages should be right.

CHEATING/COPY CASE or **LATE SUBMISSION** will be graded as **STRAIGHT ZERO MARKS**.

Automating Collaborative Development Workflow Using Bash Scripting

Objective:
In a collaborative development environment, keeping track of changes and ensuring seamless synchronization among team members is crucial. Manually committing code, pushing updates, and notifying collaborators can lead to delays and inconsistencies. To solve this, you are tasked with designing an automated system using a Bash script that continuously monitors a specific file or directory for modifications and performs the following tasks:

Requirements:

1. **Change Detection:**
   The script should monitor a specified file or directory for any modifications. Use a checksum-based approach (e.g., SHA-256) to detect changes.
2. **Version Control Automation:**
   Upon detecting a change, the script should:
   - Automatically stage the modified file(s).
   - Commit the changes with a descriptive message (e.g., "Auto-commit: Changes detected in <file_name>").
   - Push the changes to the remote GitHub repository.
3. **Real-Time Notification:**

   After pushing the changes, the script should send an email notification to all collaborators using the **SendGrid Email API**. The email should include:
   - A subject line (e.g., "Repository Update Notification").
   - A body message indicating that changes have been detected and pushed to the repository.
4. **Configuration:**
   The script should be configurable to allow users to specify:
   - The path to the local repository.
   - The file or directory to monitor.
   - The Git remote and branch name.
   - The list of collaborators' email addresses.

- SendGrid API credentials (API key, sender email, etc.).

5. **Error Handling:**
   The script should include basic error handling to manage:
   - Git push failures.
   - SendGrid API errors.
   - File monitoring issues.

Deliverables:

1. **Bash Script:**
   Write a Bash script (monitor_and_push.sh) that implements the above functionality. Ensure the script is well-documented with comments explaining each step.
2. **Configuration File:**
   Create a configuration file (config.cfg) to store user-specific settings such as:
   - Repository path.
   - File/directory to monitor.
   - Git remote and branch.
   - Collaborators' email addresses.
   - SendGrid API key and sender email.
3. **SendGrid Integration:**
   Use the SendGrid Email API to send email notifications. Include the necessary curl command in your script to interact with the API.
4. **Testing:**
   Test the script in a controlled environment to ensure it:
   - Detects file changes.
   - Commits and pushes changes to GitHub.
   - Sends email notifications to collaborators.
5. **Documentation:**
   Provide a README.md file that includes:
   - Instructions for setting up and running the script.
   - Prerequisites (e.g., Git, SendGrid API key, etc.).
   - Example usage.

Grading Criteria:

1. **Functionality (50%):**
   - The script correctly detects changes, commits, pushes, and sends notifications.
   - Proper error handling is implemented.
2. **Code Quality (20%):**
   - The script is well-structured, readable, and documented.
3. **Configuration (10%):**
   - The configuration file is correctly implemented and used.
4. **Testing (10%):**
   - Evidence of testing (e.g., screenshots, logs) is provided.
5. **Documentation (10%):**
   - The README.md file is clear and provides sufficient instructions.

This assignment will help you gain hands-on experience with Bash scripting, Git automation, and API integration, which are essential skills for collaborative software development. Good luck!

Here are some **resources** to help you complete the assignment:

1. Bash Scripting

- **Bash Scripting Tutorial**:
  https://ryanstutorials.net/bash-scripting-tutorial/
  A beginner-friendly guide to writing Bash scripts.
- **Advanced Bash-Scripting Guide**:
  https://tldp.org/LDP/abs/html/
  A comprehensive guide for advanced Bash scripting concepts.

- **Checksum in Bash**:
  Use `sha256sum` to generate file checksums. Learn more here:
  https://linux.die.net/man/1/sha256sum

2. Git Automation
- **Git Documentation**:
  https://git-scm.com/doc
  Official Git documentation for commands like `git add`, `git commit`, and `git push`.
- **Git Hooks**:
  Learn about Git hooks for automating tasks:
  https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks

3. SendGrid Email API
- **SendGrid API Documentation**:
  https://docs.sendgrid.com/for-developers/sending-email/api-getting-started
  Learn how to use the SendGrid API to send emails.
- **SendGrid API Key Setup**:
  https://docs.sendgrid.com/ui/account-and-settings/api-keys
  Steps to create and use an API key.
- **cURL for API Requests**:
  https://curl.se/docs/manual.html
  Learn how to use `curl` to make HTTP requests to the SendGrid API.

4. File Monitoring
- **Inotify (Advanced File Monitoring)**:
  https://linux.die.net/man/1/inotifywait
  A tool to monitor file system events in real-time (optional for advanced users).
- **Checksum-Based Monitoring**:
  Use `sha256sum` or `md5sum` to detect file changes:
  https://linux.die.net/man/1/md5sum

5. Error Handling in Bash
- **Bash Error Handling**:
  https://www.tutorialspoint.com/how-to-handle-errors-in-bash
  Learn how to handle errors in Bash scripts.
- **Exit Codes**:
  Understand Bash exit codes for error handling:
  https://tldp.org/LDP/abs/html/exit-status.html

---

6. Configuration Files
- **Using Configuration Files in Bash**:
  https://stackoverflow.com/questions/10929453/read-a-file-line-by-line-assigning-the-value-to-a-variable
  Learn how to read and use configuration files in Bash.

7. Testing and Debugging
- **Bash Debugging**:
  Use `set -x` to debug Bash scripts:
  https://www.gnu.org/software/bash/manual/bash.html#The-Set-Builtin
- **Testing Scripts**:
  Test your script in a controlled environment (e.g., a local Git repository).

8. Example Tools and Libraries

- **jq (for JSON Parsing)**:
  If you need to parse JSON responses from the SendGrid API, use `jq`:
  https://stedolan.github.io/jq/
- **SSMTP (for Sending Emails)**:
  If you prefer not to use SendGrid, you can use `ssmtp` for sending emails:
  https://wiki.debian.org/sSMTP

9. Example Repositories

- **GitHub Repo for Bash Scripts**:
  Explore examples of Bash scripts for inspiration:
  https://github.com/awesome-lists/awesome-bash
- **SendGrid API Examples**:
  https://github.com/sendgrid/sendgrid-python
  (Python examples, but the API logic is similar for `curl`).

10. Additional Reading

- **Cron Jobs for Automation**:
  Learn how to schedule your script to run automatically using `cron`:
  https://opensource.com/article/17/11/how-use-cron-linux
- **Bash Best Practices**:
  https://kvz.io/bash-best-practices.html
  Follow best practices for writing clean and efficient Bash scripts.

Prerequisites Checklist

Before starting, ensure you have:

1. A **GitHub repository** for testing.
2. **Git** installed on your system.
3. A **SendGrid account** and API key.
4. Basic knowledge of **Bash scripting** and **command-line tools**.

These resources will help you build the script step-by-step and ensure you understand each component of the assignment. Let me know if you need further clarification or assistance!