Jenkins Pipeline for Shell Script Execution and Task Automation

This documentation outlines the steps to create a Jenkins pipeline for executing shell scripts stored on GitHub, monitoring disk utilization, sending email alerts, and managing processes.

Objective

1. Execute a shell script using a Jenkins pipeline.
2. Automate tasks for:
    - Monitoring disk utilization and sending email alerts when usage exceeds 80%.
    - Managing system processes (e.g., stopping/starting services).

Task Overview

Shell Scripts

- **Disk Utilization Monitoring Script**:
  Checks disk usage and sends an alert if it exceeds 80%.
- **Process Management Script**:
  Manages system processes by stopping, starting, or checking their status.

Steps to Create the Pipeline

1. Write Shell Scripts

Disk Utilization Monitoring Script

Create a script (`Disk_Utilisation.sh`) to monitor disk usage:

```
#!/bin/bash

THRESHOLD=2
#As per the question this should be 80% but for now for testing purposes i
have done this on 2% in order to recieve email

echo "This Email Is Regards To The DiskUsage"
echo " "
echo " "

#Use grep to filter the '/' line and awk to extract the usage percentage

USAGE=$(df -h | grep ' /$' | awk '{print $5}' | awk '{print substr($0, 1,
length($0)-1)}') #Removes '%'

if [ "$USAGE" -gt "$THRESHOLD" ]; then
    echo "Disk usage critical: ${USAGE}% (Threshold: ${THRESHOLD}%)"
    exit 1 # Non-zero exit code for critical disk usage
else
    echo "Disk usage is under control: ${USAGE}% (Threshold: ${THRESHOLD}%)"
    exit 0 # Zero exit code for normal usage
fi
```

Process Management Script

Create a script (`processmanagement.sh`) to manage processes:

```
bash
Copy code
#!/bin/bash

# Process management script
ACTION=$1
SERVICE=$2

if [ "$ACTION" == "start" ]; then
  sudo systemctl start "$SERVICE"
  echo "Started $SERVICE"
elif [ "$ACTION" == "stop" ]; then
  sudo systemctl stop "$SERVICE"
  echo "Stopped $SERVICE"
elif [ "$ACTION" == "status" ]; then
  sudo systemctl status "$SERVICE"
else
  echo "Usage: $0 {start|stop|status} service_name"
fi
```

---

2. Push Scripts to GitHub

1. Create a GitHub repository and push the scripts:
   - `Disk_Utilisation.sh`
   - `processmanagement.sh`
2. Ensure the repository is public or accessible to Jenkins.

---

3. Configure Mail Service on the Jenkins Node

Install and Set Up Mail Utilities

1. Install mail utilities:

   ```
   sudo apt update
   sudo apt install mailutils
   ```

2. During installation, choose **Internet Site** when prompted.
3. Set the system mail name (e.g., `your-server-name`).

Configure Postfix for Email Notifications

1. Create a file for SMTP credentials:

   ```
   sudo nano /etc/postfix/sasl_passwd
   ```

2. Add the SMTP configuration (for Gmail as an example):

   ```
   [smtp.gmail.com]:587    youremail@gmail.com:app_password

   #password here is the google app password associated with your email.
   ```

3. Secure the credentials file:

   ```
   sudo chmod 600 /etc/postfix/sasl_passwd
   sudo postmap /etc/postfix/sasl_passwd
   ```

4. Update the Postfix configuration:

   ```
   sudo nano /etc/postfix/main.cf
   ```

   Add or update the following lines:

   ```
   relayhost = [smtp.gmail.com]:587
   smtp_sasl_auth_enable = yes
   smtp_sasl_security_options = noanonymous
   smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
   ```

5. Reload Postfix:

   ```
   sudo systemctl reload postfix
   ```

6. Test the mail setup:

   ```
   echo "Test email from Jenkins node" | mail -s "Test Email"
   recipient@example.com
   ```

---

4. Create the Jenkins Pipeline

Pipeline Configuration

1. **Clone Repository**:
    - Fetch the scripts from GitHub.
2. **Monitor Disk Utilization**:
    - Execute `Disk_Utilisation.sh`.
3. **Manage Processes**:
    - Execute `processmanagement.sh`.

Sample Jenkinsfile

Create a `Jenkinsfile` in your GitHub repository:

```
groovy
Copy code
pipeline {
    agent any
    environment {
        REPO_URL = 'https://github.com/your-repo-url.git'
        BRANCH = 'main'
    }
    stages {
        stage('Clone Repository') {
            steps {
                git branch: "${BRANCH}", url: "${REPO_URL}"
            }
        }
        stage('Monitor Disk Utilization') {
            steps {
                sh 'bash Disk_Utilisation.sh '
            }
        }
        stage('Process Management') {
            steps {
                sh 'bash processmanagement.sh status apache2'
            }
        }
    }
    post {
        always {
            echo 'Pipeline execution completed.'
        }
        success {
            mail to: 'recipient@example.com',
                 subject: "Pipeline Success",
                 body: "The Jenkins pipeline executed successfully."
        }
        failure {
            mail to: 'recipient@example.com',
                 subject: "Pipeline Failure",
                 body: "The Jenkins pipeline encountered an error."
        }
    }
}
```

5. Test and Verify

1. Run the pipeline and verify:
   - Jobs clone the repository.
   - Disk utilization is checked, and emails are sent if needed.
   - Processes are managed correctly.
2. Check email alerts and logs for results.

Conclusion

This Jenkins pipeline integrates disk monitoring and process management tasks using shell scripts hosted on GitHub. Email notifications ensure that disk utilization issues are promptly addressed, while process management scripts enable seamless system administration.