Loop AI Devops Engineer Assignment

Log Monitoring and Visualization Setup Documentation

This documentation outlines the process of setting up a complete log monitoring solution using **Elasticsearch**, **Kibana**, **Metricbeat**, **Filebeat**, and a **Python script** to generate logs. The system setup provides real-time log ingestion, analysis, and visualization.

---

1. Elasticsearch Setup

1.1. Set Proxy Environment Variables (if required)

If your network requires the use of a proxy, set the environment variables as follows:

```
export http_proxy=http://jpe3proxy.jaws.jio.com:8080
export https_proxy=http://jpe3proxy.jaws.jio.com:8080
```

1.2. Install Dependencies

If `wget` is not installed, use the following command to install it:

```
yum install wget -y
```

1.3. Download and Install Elasticsearch RPM

Download the Elasticsearch RPM package and install it:

```
wget
https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-
7.17.2-x86_64.rpm
yum install -y elasticsearch-7.17.2-x86_64.rpm
```

1.4. Check Elasticsearch Service Status

Verify that Elasticsearch is installed and the service is active:

```
systemctl status elasticsearch.service
```

1.5. Create Directories for Logs and Data

Create necessary directories for Elasticsearch logs and data storage:

```
mkdir /eslogs /esdata /kibanalogs
```

1.6. Set Correct Permissions for Elasticsearch Directories

Ensure the directories have the correct ownership:

```
chown elasticsearch:elasticsearch /eslogs /esdata /kibanalogs
```

1.7. Modify Elasticsearch Configuration

Edit the `elasticsearch.yml` configuration file:

```
vi /etc/elasticsearch/elasticsearch.yml
```

Add the following configuration:

```
yaml
# Elasticsearch Cluster Configuration
cluster.name: EKScluster
node.name: 10.159.177.138
path.data: /esdata
path.logs: /eslogs
bootstrap.memory_lock: true
# Network settings
network.host: 10.159.177.138
http.port: 9200
# Node roles
node.roles: [ master, data ]
# Discovery settings
discovery.seed_hosts: [ "10.159.177.138" ]
cluster.initial_master_nodes: ["10.159.177.138"]
# Transport settings
transport.port: 9300
# Enable security features
xpack.security.enabled: true
xpack.security.transport.ssl.enabled: true
xpack.security.transport.ssl.verification_mode: certificate
xpack.security.transport.ssl.keystore.path: elastic-certificates.p12
xpack.security.transport.ssl.truststore.path: elastic-
certificates.p12
```

1.8. Start Elasticsearch

Start Elasticsearch:

```
systemctl start elasticsearch
```

1.9. Check Elasticsearch Service Status Again

Verify Elasticsearch is running:

```
systemctl status elasticsearch.service
```

1.10. Elasticsearch Security Setup

Enable SSL and set up keystore for secure communication:

```
# Generate CA Certificate
/usr/share/elasticsearch/bin/elasticsearch-certutil ca
# Generate the certificate using the CA
/usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca
/etc/elasticsearch/elastic-stack-ca.p12
# Add Secure Passwords to the Keystore
```

```
/usr/share/elasticsearch/bin/elasticsearch-keystore add
xpack.security.transport.ssl.keystore.secure_password
/usr/share/elasticsearch/bin/elasticsearch-keystore add
xpack.security.transport.ssl.truststore.secure_password
```

1.11. Reset Password for `elastic` User (if required)

If necessary, reset the password for the `elastic` user:

```
/usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic
-interactive
```

1.12. Setup Elasticsearch Passwords for Users

Use the following command to set up passwords for users:

```
/usr/share/elasticsearch/bin/elasticsearch-setup-passwords
interactive
```

1.13. Add a New User (Admin)

Add a new user (`admin`) with superuser role:

```
/usr/share/elasticsearch/bin/elasticsearch-users useradd admin -p
<password> -r superuser
```

1.14. Check Authentication

Verify if the admin user can authenticate successfully:

```
curl -X GET
"http://10.159.177.138:9200/_security/_authenticate?pretty" -u
admin:<password>
```

1.15. Restart Elasticsearch After Changes

After updating configurations or adding security settings, restart Elasticsearch:

```
systemctl restart elasticsearch
```

1.16. Check Elasticsearch Cluster Health

Check the health of your Elasticsearch cluster:

```
curl -X GET "http://10.159.177.138:9200/_cluster/health?pretty" -u
admin:<password>
```

---

2. Kibana Setup

2.1. Install Kibana

Download and install the Kibana RPM package:

```
sudo yum install kibana-7.17.2-x86_64.rpm
```

2.2. Configure Kibana

Edit the `kibana.yml` configuration file:

```
vi /etc/kibana/kibana.yml
```

Add the following configuration:

```
yaml
server.host: "10.159.177.138"
server.name: "10.159.177.138"
server.port: 5601
elasticsearch.hosts: ["http://10.159.177.138:9200"]
elasticsearch.username: "admin"
elasticsearch.password: "<password>"
logging.dest: /kibanalogs/kibana.log
server.publicBaseUrl: http://10.159.177.138:5601
```

2.3. Set Directory Permissions for Kibana Logs

Set the correct permissions for the Kibana logs directory:

```
sudo chown -R kibana:kibana /kibanalogs/*
```

2.4. Start Kibana

Start the Kibana service:

```
sudo systemctl start kibana
```

2.5. Check Kibana Status

Check if Kibana is running properly:

```
sudo systemctl status kibana
```

2.6. Access Kibana

Access Kibana in a browser at:

```
http://10.159.177.138:5601
```

---

3. Metricbeat Setup

3.1. Download and Install Metricbeat

Download and install Metricbeat RPM:

```
wget https://artifacts.elastic.co/downloads/beats/metricbeat/metricbeat-
7.17.13-x86_64.rpm
sudo rpm -vi metricbeat-7.17.13-x86_64.rpm
```

3.2. Configure Metricbeat

Edit the `metricbeat.yml` configuration file:

```
vi /etc/metricbeat/metricbeat.yml
```

Add the following configuration:

```
yaml
setup.kibana:
  host: "10.159.177.138:5601"
output.elasticsearch:
  hosts: ["http://10.159.177.138:9200"]
  username: "admin"
  password: "<password>"
```

3.3. Enable Elasticsearch Module for Metricbeat

Enable the Elasticsearch module:

```
metricbeat modules enable elasticsearch-xpack
```

3.4. Setup Metricbeat

Initialize Metricbeat's dashboards and index templates:

```
metricbeat setup -e
```

3.5. Start Metricbeat

Start the Metricbeat service:

```
sudo systemctl start metricbeat
```

3.6. Verify Metricbeat Status

Check if Metricbeat is running correctly:

```
sudo systemctl status metricbeat
```

3.7. Verify Data in Kibana

Once Metricbeat is running, go to the **Metrics** dashboard in Kibana to view the data.

---

4. Filebeat Setup

4.1. Download and Install Filebeat

Download and install Filebeat RPM:

```
wget https://artifacts.elastic.co/downloads/beats/filebeat/filebeat-
7.17.14-x86_64.rpm
```

```
rpm -vi filebeat-7.17.14-x86_64.rpm
```

4.2. Configure Filebeat

Backup Default Configuration:

```
cp /etc/filebeat/filebeat.yml /tmp
```

Edit the Configuration File:

```
vi /etc/filebeat/filebeat.yml
```

Add the following configuration:

```
yaml
filebeat.inputs:
  - type: log
    paths:
      - /root/application.log  # Path to your application log file
    fields:
      service: application
    fields_under_root: true
    multiline.pattern: '^[[:space:]]'
    multiline.negate: false
    multiline.match: after
processors:
  - dissect:
      tokenizer: "%{timestamp} %{+timestamp} %{severity}"
      field: "message"
      target_prefix: "parsed"
      ignore_missing: true
  - rename:
      fields:
        - from: "parsed.timestamp"
          to: "timestamp"
        - from: "parsed.severity"
          to: "severity"
      ignore_missing: true
  - add_fields:
      target: ''
      fields:
        service: "application"
output.elasticsearch:
  hosts: ["http://10.159.177.138:9200"]
  index: "application-logs-%{+yyyy.MM.dd}"
  username: "admin"
  password: "<password>"
  ssl.verification_mode: "none"
setup.kibana:
  host: "http://10.159.177.138:5601"
setup.template.name: "application-logs"
setup.template.pattern: "application-logs-*"
setup.template.enabled: true
setup.ilm.enabled: false
```

4.3. Start and Enable Filebeat Service

Start the Filebeat service:

```
systemctl start filebeat
```

4.4. Check Filebeat Status

Verify if Filebeat is running correctly:

```
systemctl status filebeat
```

---

5. Log Generation Script

5.1. Python Log Generation Script

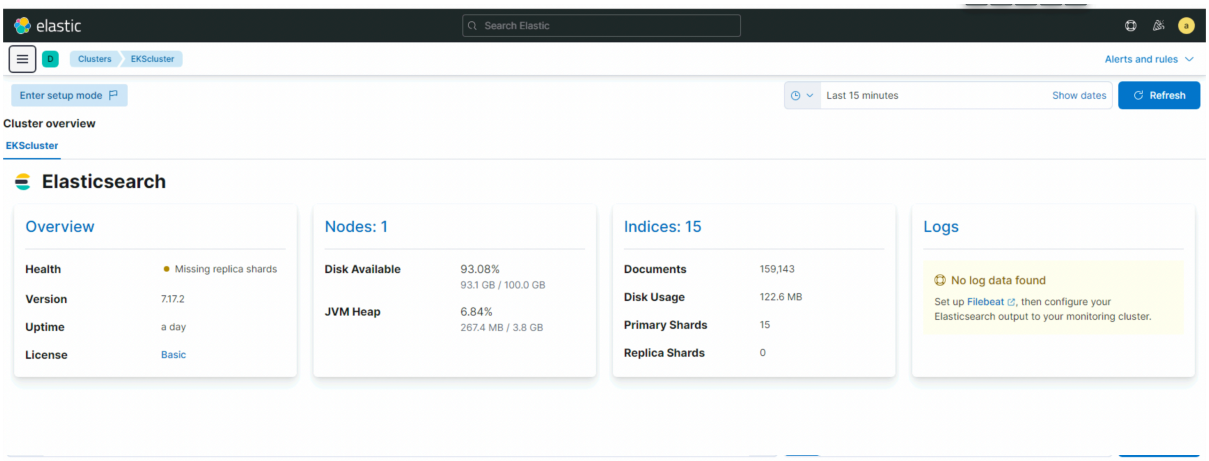
Here's the Python script to generate logs:

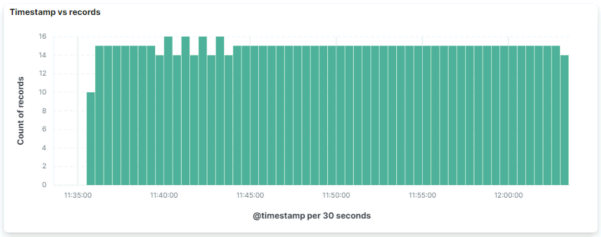
```
python
import logging
import random
import time
# Set up logging configuration
logging.basicConfig(
    filename='application.log',  # Specify the log file to write to
    level=logging.DEBUG,  # Capture all levels from DEBUG and above
    format='%(asctime)s %(levelname)s',  # Log format with timestamp
and severity level
    datefmt='%Y-%m-%d %H:%M:%S'  # Custom date format without
milliseconds
)
# Function to generate random log severity
def generate_random_log():
    severity = random.choice(['INFO', 'WARNING', 'ERROR'])
    if severity == 'INFO':
        logging.info('')
    elif severity == 'WARNING':
        logging.warning('')
    elif severity == 'ERROR':
        logging.error('')
# Generate logs at random intervals
def generate_logs(interval=2, duration=60):
    start_time = time.time()
    while time.time() - start_time < duration:
        generate_random_log()
        time.sleep(interval)
# Generate logs for 60 seconds with an interval of 2 seconds
if __name__ == "__main__":
    print("Generating logs...")
    generate_logs(interval=2, duration=6000)
    print("Log generation complete.")
```

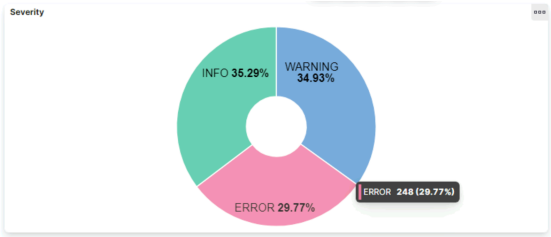
6. Sample Logs

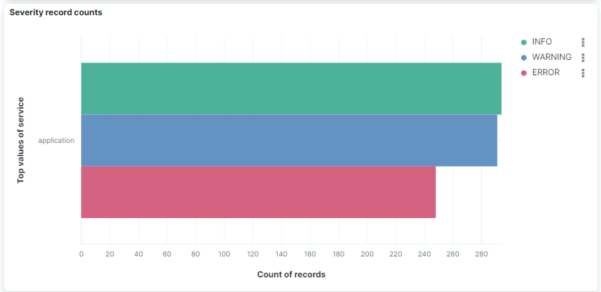
2024-12-19 04:01:45 ERROR
2024-12-19 04:01:47 WARNING
2024-12-19 04:01:49 INFO
2024-12-19 04:01:51 INFO
2024-12-19 04:01:53 ERROR
2024-12-19 04:01:55 WARNING
2024-12-19 04:01:57 ERROR
2024-12-19 04:01:59 ERROR
2024-12-19 04:02:01 WARNING
2024-12-19 04:02:03 WARNING

7. Kibana Snippets

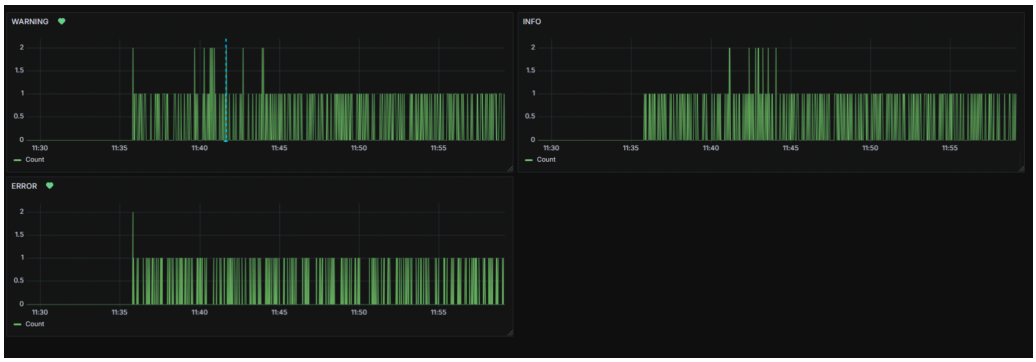




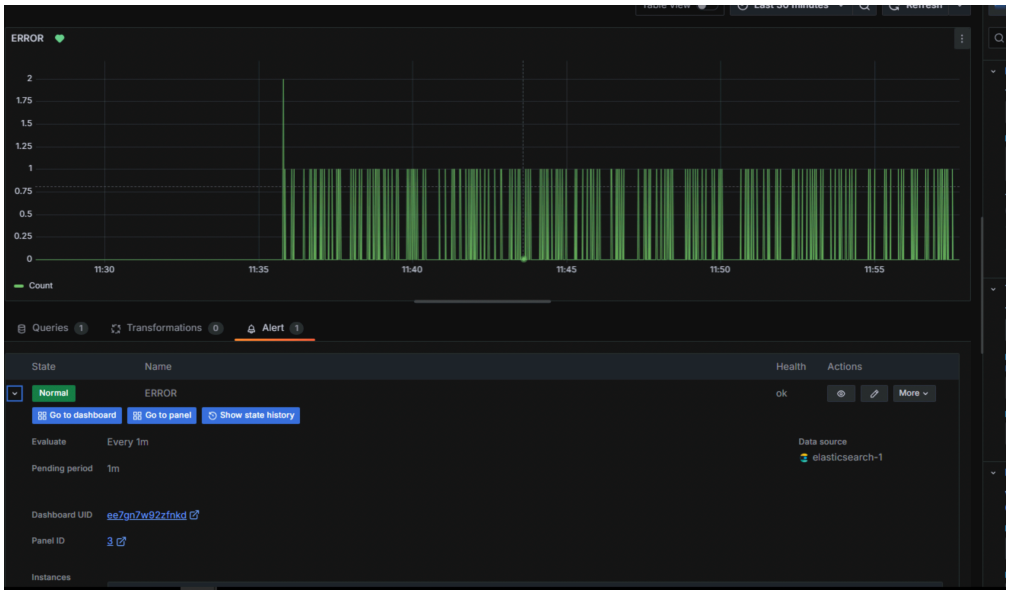




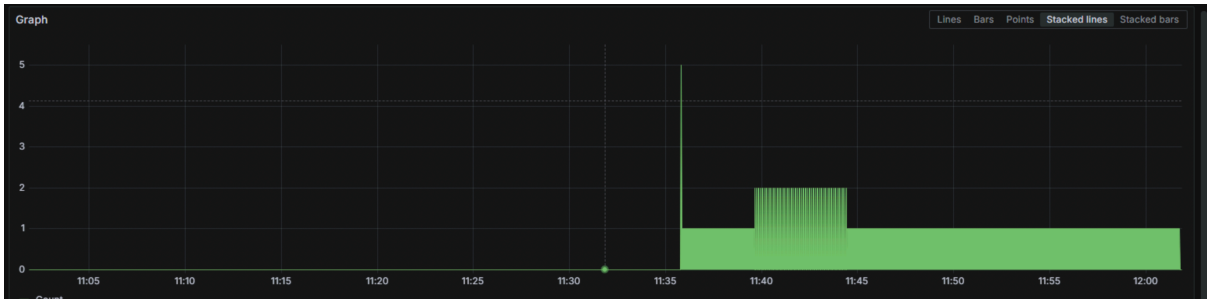
8. Grafana Snippets



Alert Configuration



Total Record Count



Markdown Linting Pipeline for Pull Requests

This repository contains an Azure Pipelines configuration that automatically detects any Markdown files (.md) in pull requests (PRs). If such files are found, it runs the `markdownlint` package to check for linting violations, ensuring that your Markdown files meet predefined standards.

Table of Contents

Overview

This CI pipeline is designed to:

1. Detect Markdown files (.md) in a pull request.
2. If Markdown files are found, it will run the `markdownlint` tool to check for common linting violations.
3. If violations are detected, the pipeline will fail and provide feedback in the CI output.
4. If no Markdown files are found, the pipeline will exit successfully without running linting.

Setup Instructions

Prerequisites

1. **Azure DevOps account**: You must have access to an Azure DevOps organization and project.
2. **Repository with Markdown files**: This pipeline works in repositories containing `.md` files.

Steps to Set Up the Pipeline

1. **Create a new repository** or use an existing one in Azure Repos.
2. **Add the following YAML configuration to your repository**:
    - Create a `.azure-pipelines.yml` file in the root of your repository.

```
yaml

trigger:
  branches:
    include:
      - main
      - release/*
```

```
pr:
  branches:
    include:
      - main
      - release/*

jobs:
  - job: LintMarkdownFiles
    displayName: 'Lint Markdown Files'
    pool:
      vmImage: 'ubuntu-latest'

    steps:
      - task: Checkout@2
        displayName: 'Checkout Code'

      - script: |
          # Check if there are any markdown files in the PR
          MARKDOWN_FILES=$(git diff --name-only ${{ parameters.baseBranch
}} ${{ parameters.headBranch }} | grep -E '\.md$')
          if [[ -z "$MARKDOWN_FILES" ]]; then
            echo "No markdown files changed in the PR. Skipping
markdownlint."
            exit 0
          fi
        displayName: 'Detect Markdown Files'

      - task: UseNode@2
        inputs:
          versionSpec: '16.x'
        displayName: 'Set up Node.js'

      - script: |
          # Install markdownlint-cli package
          npm install -g markdownlint-cli
        displayName: 'Install markdownlint'

      - script: |
          # Run markdownlint on the changed markdown files
          markdownlint $MARKDOWN_FILES
        displayName: 'Run markdownlint'

      - script: |
          # Ensure markdownlint exits with non-zero code if violations are
found
          if [[ $? -ne 0 ]]; then
            echo "Markdown linting failed. Please fix the issues and try
again."
            exit 1
          else
            echo "Markdown files passed linting successfully."
          fi
        displayName: 'Check for Linting Violations'
```

Explanation of the YAML Pipeline

1. **Trigger**:
    - This pipeline is triggered on pushes to the `main` branch and any `release/*` branches. It also runs on pull requests targeting these branches.

2. **Job**:
    - A single job (`LintMarkdownFiles`) runs in an Ubuntu-based environment (`ubuntu-latest`).
3. **Steps**:
    - **Checkout**: Checks out the latest code from the repository.
    - **Detect Markdown Files**: This step checks if any `.md` files are included in the PR changes.
    - **Set up Node.js**: This installs Node.js, as `markdownlint-cli` is a Node.js-based tool.
    - **Install markdownlint-cli**: Installs the `markdownlint-cli` package globally on the CI environment.
    - **Run markdownlint**: Runs `markdownlint` on the detected Markdown files to check for violations.
    - **Check for Violations**: If `markdownlint` reports violations, the step will fail, providing feedback on the issues.

Pipeline Flow

- The pipeline will check the PR to see if there are any `.md` files modified.
- If no `.md` files are present, the pipeline will exit successfully.
- If `.md` files are found, it installs `markdownlint` and runs it on those files.
- If any linting violations are detected, the pipeline will fail, and the output will list the issues.

How the Pipeline Works

Detect Markdown Files

In the pipeline configuration, the step that checks for Markdown files uses the `git diff` command to determine which files have been changed between the base and head branches in the PR. If any `.md` files are detected, the pipeline proceeds to the linting step.

Run markdown-lint

If Markdown files are found, the pipeline installs the `markdownlint-cli` package and runs it on the modified `.md` files. The tool checks for common Markdown style issues, such as:

- Trailing spaces
- Headers not followed by a space
- Incorrect list formatting
- And other style-related issues.

If `markdownlint` reports any issues, the pipeline will fail and output the details of the violations.

Reporting Violations

Any violations identified by `markdownlint` will be displayed in the Azure Pipelines log output. The pipeline will fail with a non-zero exit code, indicating the need for corrections in the affected Markdown files.

How to Trigger the Linting Process

To trigger the linting process, simply open a pull request that modifies or adds Markdown files (`*.md`).

- When the PR is created or updated, the pipeline will run automatically.
- If the PR contains `.md` files, the linting process will be triggered.
- If no violations are found, the PR will pass the pipeline checks.
- If violations are found, the PR will fail, and feedback will be provided in the form of detailed linting errors.

Common Linting Violations and Fixes

Here are some common violations you might encounter and how to fix them:

- **Violation**: `MD009 - Trailing spaces`
  - **Fix**: Remove any extra spaces at the end of lines.
- **Violation**: `MD010 - Hard tabs`
  - **Fix**: Replace any tabs with spaces (usually 2 or 4 spaces per tab).
- **Violation**: `MD013 - Line length`
  - **Fix**: Break lines that are too long (usually over 80 characters) into shorter lines.
- **Violation**: `MD001 - Header levels should only increment by one`
  - **Fix**: Ensure that headers follow a logical hierarchy (i.e., no skipping header levels like `#` to `###`).

File Structure

Here is an example of what your repository's file structure might look like:

```
bash
```

```
.
├── .azure-pipelines.yml        # Pipeline configuration file
├── README.md                   # Example Markdown file
├── CONTRIBUTING.md             # Example Markdown file
└── docs/
    └── guide.md                # Additional Markdown files
```