Week - 11 Monitoring And Logging

- Installation Of grafana
- Database Installation MySQL
- Grafana Setup with My SQL
- Installation of Prometheus
- Setting Up Prometheus on the Kubernetes cluster
- Monitoring K8 Cluster with Prometheus
- Alerts in Grafana
- Grafana Plugins

Prometheus:

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. It is now a standalone open-source project and maintained independently by the community. Prometheus is designed to monitor highly dynamic service-oriented architectures.

Key Features:

- Multidimensional Data Model: Prometheus stores all its data as time series, i.e., streams of timestamped values belonging to the same metric and the same set of labeled dimensions.

- Flexible Query Language: PromQL (Prometheus Query Language) allows for slicing and dicing the collected time series data to generate ad-hoc graphs, tables, and alerts.

- No Dependency on Distributed Storage: Prometheus operates as a single node that can optionally use federation to scale out.

- Built-in Support for Alerting: It includes an alert manager that handles alerts, including deduplication, grouping, and routing to receivers such as email, PagerDuty, or OpsGenie.

- Service Discovery: Prometheus supports multiple modes for discovering target services dynamically.

Use Cases:

- Infrastructure Monitoring: Collecting metrics from servers, VMs, and containers.

- Application Performance Monitoring: Monitoring performance metrics such as request latencies, error rates, and throughput.

- Incident Management and Alerting: Generating alerts based on metric thresholds or anomalies and sending them to various notification channels.

Grafana:

Grafana is an open-source analytics and monitoring solution that integrates with various data sources. It provides a powerful and flexible dashboarding and visualization platform, making it a popular choice for monitoring and observability.

Key Features:

- Multi-Source Data Integration: Supports a wide range of data sources, including Prometheus, InfluxDB, Graphite, Elasticsearch, MySQL, and many more.

- Custom Dashboards: Create, explore, and share dashboards that visualize metrics data. Dashboards can be customized with various widgets such as graphs, tables, heatmaps, and alerts.

- Templating: Use variables and templates to create dynamic and reusable dashboards.

- Alerting: Set up alerts and notifications based on the data visualized in Grafana, using flexible alert rules.

Use Cases:

- Operational Dashboards: Visualize and monitor operational data across servers, applications, and services.

- Business Metrics: Track and display business metrics such as sales, user sign-ups, and engagement.

- Performance Analysis: Analyze and visualize application performance data to detect trends, anomalies, and bottlenecks.

Grafana Plugins and Alerts:

Grafana Plugins:
Grafana supports various plugins to extend its functionality:

- Data Source Plugins: Allow Grafana to fetch data from various sources (e.g., Prometheus, Elasticsearch, AWS CloudWatch).

- Panel Plugins: Add new visualizations to Grafana (e.g., Graphs, Tables, Pie Charts).

- App Plugins: Bundle data source and panel plugins to provide full application-like experiences (e.g., Kubernetes monitoring).

Examples of Popular Plugins:

- Loki: For logging and querying logs.
- Tempo: For tracing and performance analysis.
- Grafana Polystat: For multi-dimensional data visualization.

Grafana Alerts:

Grafana alerting allows you to configure alerts and notifications based on metric thresholds or complex expressions.

- Alert Rules: Define conditions under which an alert should be triggered.

- Notification Channels: Specify where alerts should be sent, such as email, Slack, PagerDuty, or custom webhooks.

- Alert Management: Monitor, acknowledge, and silence alerts from within Grafana.

Grafana Dashboards

Grafana dashboards are used to visualize data from various sources in an interactive and customizable way.

Key Features of Grafana Dashboards

- Visualization: Grafana supports a wide range of visualization types, including graphs, tables, heatmaps, and more.

- Data Sources: Grafana can connect to various data sources, such as Prometheus, InfluxDB, MySQL, PostgreSQL, Elasticsearch, and many others.

- Customizable: Dashboards are highly customizable, allowing usto create and arrange panels according to their needs.

- Alerting: Grafana allows users to set up alerts that can notify them via different channels (email, Slack, PagerDuty, etc.) when certain conditions are met.

- Templating: Grafana supports templating, which makes dashboards reusable and dynamic. You can create variables and use them to filter data in your dashboards.

- Annotations: Annotations allow you to mark points on your graphs with additional information, which can help in understanding events in context.

Exporters in Prometheus

Exporters are crucial components in the Prometheus ecosystem that help in collecting metrics from various sources. They are lightweight processes that extract data from different systems and expose it in a format that Prometheus can scrape.

Key Points about Exporters

- Functionality: Exporters gather metrics from third-party systems and expose them via an HTTP endpoint, typically in a format that Prometheus understands (usually plain text with a specific structure).

- Use Cases: Exporters are used to monitor various types of services, such as databases, hardware devices, and other software services that do not natively expose metrics in a format compatible with Prometheus.
- Custom Exporters: While Prometheus provides many official exporters, the community also contributes a wide range of exporters for various applications. Additionally, you can write custom exporters for specialized use cases.

Common Exporters

- Node Exporter: For hardware and OS metrics exposed by *NIX kernels.

- Blackbox Exporter: For probing endpoints over HTTP, HTTPS, DNS, TCP, ICMP, etc.

- SNMP Exporter: For collecting metrics from SNMP-enabled devices.

- JMX Exporter: For exposing Java Management Extensions (JMX) data.

Node Exporter in Prometheus

The Node Exporter is one of the most commonly used exporters in the Prometheus ecosystem. It is designed to collect hardware and operating system metrics from *NIX systems (Linux, UNIX, etc.). We can use Node Exporter to collect metrics from EC2 instances

K8s Helm Charts

Helm is a package manager for Kubernetes that simplifies the process of deploying and managing applications on Kubernetes clusters. It uses a packaging format called charts.

Key Concepts of Helm

- Helm: The CLI tool that interacts with the Helm library.
- Chart: A Helm package containing the resource definitions to deploy an application or service.
- Release: A running instance of a chart, deployed to a Kubernetes cluster.
- Repository: A collection of charts, similar to a package repository in other package managers.

How Helm Charts are Used

Helm charts are used to deploy applications to Kubernetes clusters. They provide a template-driven approach to manage Kubernetes manifests.

Monitoring Kubernetes (K8s) Cluster with Prometheus and Grafana:

Architecture:

- Prometheus Operator: Deploying the Prometheus Operator in a Kubernetes cluster simplifies the deployment and management of Prometheus instances.
- Node Exporter: Collects hardware and OS metrics from the host.
- kube-state-metrics: Exposes Kubernetes cluster-level metrics.
- Grafana: Visualizes metrics collected by Prometheus.

Follow the below steps:

- Install Helm - https://get.helm.sh/helm-v3.7.1-linux-amd64.tar.gz to */tmp/helm-v3.7.1-linux-amd64.tar.gz*

- Extrack helm - *#tar -xvzf /tmp/helm-v3.7.1-linux-amd64.tar.gz*

- Move the Helm to /usr/local/bin - *#mv /tmp/helm/linux-amd64/helm /usr/local/bin/helm*

- Add Prometheus Helm repository *#helm repo add prometheus-community https://prometheus-community.github.io/helm-charts*

- Update Helm repositories *#helm repo update*

- Install Prometheus using Helm *#helm install prometheus prometheus-community/prometheus*

- Expose Prometheus service using kubectl
  #kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext

- Add Help Repo for Grafana
  #helm repo add grafana https://grafana.github.io/helm-charts

- Install Grafana using Helm
  #helm install grafana grafana/grafana

- Expose Grafana service using kubectl
  #kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext

Prometheus And Grafana

- Create a `prometheus.yml`(Prometheus Configuration file with the content given in this repo)
  https://github.com/sudhanshuvlog/GFG-Devops18/blob/main/Monitoring/prometheus.yml

- Create Prometheus Server - `*#docker run -d --name prometheus-container -e TZ=IST -v ./prometheus.yml:/etc/prometheus/prometheus.yml -p 9090:9090 ubuntu/prometheus:2.46.0-22.04_stable*`

- Download the latest version of Node-exporter

#wget
https://github.com/prometheus/node_exporter/releases/download/v1.7.0/
node_exporter-1.7.0.linux-amd64.tar.gz

- Untar the downloaded file
  #tar -xvzf node_exporter-1.7.0.linux-amd64.tar.gz

- Move the binary to a directory in your PATH
  #cp node_exporter-1.7.0.linux-amd64/node_exporter /usr/bin

- Run the program on your EC2 Instance
  #node_exporter

- Expose Docker Metrics - `*#vi /etc/docker/daemon.json*` Add The content given in this repo
  {
  "metrics-addr": "0.0.0.0:9323"
  }

- Restart Docker Engine - `*#systemctl restart docker*`. Now Docker will be exposing it metrics at IP:9323/metrics

- Restart Prometheus Server, When you made any configuration changes - `*#docker restart prometheus-container*`

- Start Grafana Server - `*docker run -d --name=grafana -p 3000:3000 grafana/grafana*`

- Download the latest version of Prometheus on EC2
  #wget
  https://github.com/prometheus/prometheus/releases/download/v2.49.0-rc.1/prometheus-2.49.0-rc.1.linux-amd64.tar.gz

- Untar the downloaded file
  #tar -xvzf prometheus-2.49.0-rc.1.linux-amd64.tar.gz

- Move the binary to a directory in your PATH
  #cp prometheus-2.49.0-rc.1.linux-amd64/prometheus /usr/bin

- Run the program
  #prometheus

Database Installation MySQL & Grafana Setup with My SQL

Follow the below k8s configuration file - https://github.com/sudhanshuvlog/GFG-Devops18/tree/main/K8s/MySQL-Mysqld-Exporter

This Pod setup allows you to run a MySQL database and simultaneously collect its metrics using the MySQL exporter. The metrics are exposed on port 9104, which can then be scraped by Prometheus for monitoring purposes.

We are launching 2 containers in a pod(We will be launching one sidecar container named MySQL exporter for exposing the metrics)

- MySQL Container: Runs a MySQL database server.
- MySQL Exporter Container: Collects metrics from the MySQL server and exposes them on port 9104 for Prometheus to scrape.