Assignment 5: Monitoring

Introduction

In this assignment, you will create a simple web application and deploy it in Kubernetes. You will then set up monitoring for the application using Prometheus, Grafana, and Alertmanager. Through this assignment, you will learn how to monitor a web application deployed in Kubernetes using Prometheus and Grafana. You will also learn how to set up alerts using Alertmanager and receive alerts in a telegram bot.

Part 1: Prepare the web application

1. Create a simple web application in Node.js or any other language of your choice. You can use examples from internet or create your own application with help of AI tools.
   - `/` - Home page that displays a welcome message and the values of the counter and switch.
   - `/inc` - Increment endpoint that increments a counter and returns the new value.
   - `/switch` - Switch endpoint that toggles a state between ON, OFF, and UNKNOWN.
   - `/metrics` - Metrics endpoint that returns the values of the counter and switch in Prometheus format.

Note: The metrics endpoint should return the data in Prometheus format. Refer to the Prometheus documentation for more details. The response content type should be `text/plain`.

For counter and switch, you can use in-memory variables or any other storage mechanism of your choice.

Note, Prometheus is a time series database and it stores the data in the form of key-value pairs. The key is the metric name and the value is the metric value. However, in this scenario, the switch is a state and it can have multiple state values. To handle this, you can use labels in Prometheus. For example, you can have a metric named `switch_state` with the label `state`. The `state` label can have values like `ON`, `OFF`, and `UNKNOWN`, and the value of the metric can be `1` or `0` based on the state label.

Example Prometheus format:

```
counter_value 10

switch_state{state="ON"} 1
switch_state{state="OFF"} 0
switch_state{state="UNKNOWN"} 0
```

2. Create a Dockerfile for the web application, build the Docker image, and

push it to the Docker Hub repository. Ensure that the repository is public to avoid any access issues.

Part 2: Deploy the web application in Kubernetes

1. Create a new deployment for the web application using the Docker image you have pushed to the Docker Hub repository.
2. Create a new service of NodePort type to expose the port to the outside. You can learn more about NodePort here. Note, if you are using minikube with docker desktop, then you need tunnel the service to access it from the host. See here.
3. Access the web application using the IP address of the host and the port you have specified in the service object. The application should be accessible in the browser.

Part 3: Set up monitoring using Prometheus and Grafana using Docker Compose

1. Create a `docker-compose.yml` file that includes the following services:
   - Prometheus: Use the official Prometheus image from Docker Hub. Configure Prometheus to scrape the metrics endpoint of the web application deployed in Kubernetes. You can use the `prometheus.yml` configuration file to define the scrape configuration.
   - Alertmanager: Use the official Alertmanager image from Docker Hub. Use a telegram bot for alerting. See this example configuration here. You have to create a telegram bot and get the token and chat ID to configure the alertmanager. Use `@BotFather` to create a new bot and get the token. Initiate a chat with the bot and get the chat ID. See this link.
   - Grafana: Use the official Grafana image from Docker Hub. Configure Grafana to connect to Prometheus as a data source and to Alertmanager for alerting.
2. Start the services using `docker-compose up` and access Grafana in the browser. You can access Grafana at `http://localhost:3000`. The default username and password for Grafana is `admin/admin`.
3. Configure Grafana to use Prometheus as a data source.
4. Configure Grafana to use Alertmanager for alerting.
5. Create a dashboard in Grafana to visualize the metrics from Prometheus. You can create graphs, tables, and other visualizations to monitor the web application metrics. You should make 4 visualizations for each of the following metrics:
   - Counter value over time.

- Switch state over time.
- Counter is increased by 10 in 1 minute.
- Switch state is in the `UNKNOWN` state for 1 minute.

6. Create alerts in Grafana to monitor the web application metrics. You should create alerts for the following scenarios:

    - If the counter is increased by 10 in 1 minute.
    - If the switch state is in the `UNKNOWN` state for 1 minute.

7. Test the alerts by triggering the conditions and verifying that the alerts are sent to Alertmanager and to the telegram bot.

Submission

You need to submit the following files as a zip archive:

1. The project folder contains the source code for the web application and the Dockerfile.
2. Kubernetes deployment and service configuration files.
3. Docker compose file for Prometheus, Alertmanager, and Grafana.
4. Screenshots of Minikube running on your local machine with `kubectl get pods --all-namespaces` output.
5. Screenshots of the web application running in Kubernetes and accessible in the browser. Screenshots should include the browser window showing the application and the terminal window showing the deployment and service status using `kubectl`.
6. Screenshots of the 4 visualizations in Grafana: counter value over time, switch state over time, counter is increased by 10 in 1 minute, and switch state is in the `UNKNOWN` state for 1 minute.
7. Screenshots of the alerts and their states in Grafana.
8. Screenshots of the alerts received in the telegram bot.