Task 0:

Follow the readme, setup a cluster and deploy the sample microservice application. Now that you have the application running, let's improve this stack.

Activity 1: Monitoring

When there are many servers and even more applications running inside each, monitoring resources and applications becomes important. Systems break, applications crash. To build a robust product, monitoring becomes key. With kubernetes and other cloud native solutions, a Time Series Database called Prometheus is often used. Prometheus collectes metrics from agents and applications which are running in the cluster and provides a query language to create meaningful visualizations.

Another open source dashboard called Grafana is used to create graphs and metrics with the use of prometheus.

Let's start by installing Prometheus into the cluster and collect metrics like Node and Pod cpu, memory, etc.

Guides:

1. How to monitor your Kubernetes cluster with Prometheus and Grafana

2. prometheus-community/helm-charts: Prometheus community Helm charts

3. grafana/helm-charts

Note: The guides follow the older version, checkout links 2 and 3 for installing prometheus and grafana respectively.

Use the above guides as a starting point to set up prometheus and grafana. Most of these would be using predefined templates for creating all the components. The popular way is to use Helm. Prometheus and Grafana helm charts can be found in the links above.

If you want to customize your graphs, take a look at this introduction to prometheus query language or PromQL: PromQL tutorial for beginners and humans | by Aliaksandr Valialkin | Medium

Look through the collections of publicly available dashboards here: Grafana Dashboards - discover and share dashboards for Grafana.

Task 1.1:

Setup Prometheus and Grafana and have some dashboards working, monitor the resource utilization for the application we created.

- Prometheus Node Exporter Full dashboard for Grafana
- Kubernetes / Compute Resources / Cluster dashboard for Grafana
- Kubernetes / Compute Resources / Node (Pods) dashboard for Grafana
- Kubernetes / Compute Resources / Pod dashboard for Grafana

Guides: If you want to start collecting more metrics for a particular pod, it has to expose more metrics, often using a prometheus client which is integrated with the application. These integrations can be done in the applications using supported clients or using exporters.

Annotations can be used to tell prometheus to also scrape the pods when clients or exporters are configured. Annotations are key-value pairs that can be added to pod/deployment specs. For scraping using prometheus these annotations can be used:

- prometheus.io/scrape: The default configuration will scrape all pods and, if set to false, this annotation will exclude the pod from the scraping process.
- prometheus.io/path: If the metrics path is not /metrics, define it with this annotation.
- prometheus.io/port: Scrape the pod on the indicated port instead of the pod's declared ports.

More information: Kubernetes & Prometheus Scraping Configuration

Task 1.2: Sending more traffic to the UI and see if this reflects in the metrics. More traffic should use more pod resources. You can use something like Hey to generate load.