After our application runs, we now want to operate this application. This tutorial cares about montoring and logging. All relevant resources are located in https://gitlab.fhnw.ch/cloud/devops/templates/monitoring-logging-tracing .

Task 1: Install Logging

For centralized logging handling, we install Loki quite similar to prometheus.

1. Go to the folder "loki " and follow the installation. I recommend to use the namespace "logging" for accessing the logs.
2. Verify, that the pods are running in the namespace.

Task 2: Adapt Monitoring

Our Monitoring is running with the help of the kube-prometheus-stack. However, as for now, only metrics related to kubernetes are collected. Since we would like to get custom quarkus/flask-metrics as well as business metrics, we need to adapt the kube-prometheus-stack deployment.

1. Follow the path under the Readme in the folder "kube-prometheus-stack". Please note that the values.yml assumes, that loki is installed in the namespace "logging".
   You should have the config for scraping the applications applied. Furthermore, Loki should be registered as datasource.
2. Forward the pod "prometheus" with its service listening to port 9090. Verify that the config is scraping quarkus and flask-apps.
3. Login into grafana, and check of the Loki-datasource is registered. Maybe you need to kill the grafana pod to force it to reload dashboards.

Task 3: Provision additional dashboards

For Loki as well as the chatbots-pods, we need to deploy additional dashboards.

1. Go to the folder "grafana" and follow the path in the Readme.
2. Check of the dashboards appear. Maybe you need to kill the grafana pod to force it to reload dashboards. There should be a dashboard for quarkus, one for flask und one for the logs.

Task 4: Adapt your business metrics and provision an own dashboard

As for now, you created in the last tutorial a business metric. Now we would like to see it in the monitoring.

1. Reflect your implemented business metric and refactor them. Think about what metric would make sense and make sure, roberta as well as eliza are generating the same metric with similar labels.
2. Create a new dashboard for the metric. The dashboard should summarize data from both chatbots. Export the dashboard and create an new configmap for you. You might want to fork the monitoring-logging-tracing-repo for this purpose.

Debug Tips:

- Verify your implemented metric by starting the pods locally and chat with them locally.
- Forward the chatbots to the local machine to verify that the metrics are exposed over time.
- If the dashboard does not appear, check the tips in the grafana-folder in the repository.