‹ Return to Classroom

DISCUSS ON STUDENT HUB ›

Building a Metrics Dashboard

REVIEW

CODE REVIEW

HISTORY

Requires Changes

4 specifications require changes

Hi there!

This was a good start point for this project 👍 You did a great job with performing Jaeger traces on the Python services 👏
Please make the below modifications to meet all rubric items

1. Change the period time of the **Frontend 40X-50X errors** dashboard to the **last 24 hours** rather than the last 6 hours.
2. Add at least one more screenshot of a dashboard that shows the **uptime of Backend and Frontend services during the last 24 hours**.
3. Add at least **one more SLI/SLO** for guaranteeing that our application has a 99.95% uptime per month.
4. List **2-3 KPIs per each SLI as well as a short description** of why those KPIs were chosen by you for each SLI.
5. Write a **text description of what graphs are represented in your final dashboard.**

EXTERNAL RESOURCES

- Monitoring Flask microservices with Prometheus
- Monitoring Kubernetes with Prometheus Operator
- Prometheus operator - enable monitoring for everything in all namespaces
- How to create a ServiceMonitor for prometheus-operator?
- How to visualize average response time of all requests
- If you are curious to learn about **How To Implement Distributed Tracing with Jaeger on Kubernetes** please refer to here

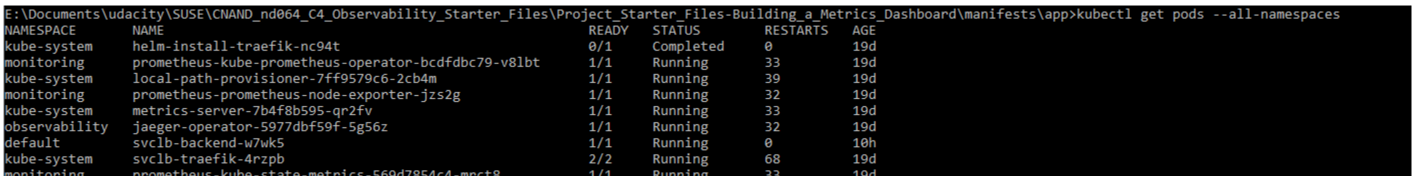
Good luck

Project and Cluster Staging

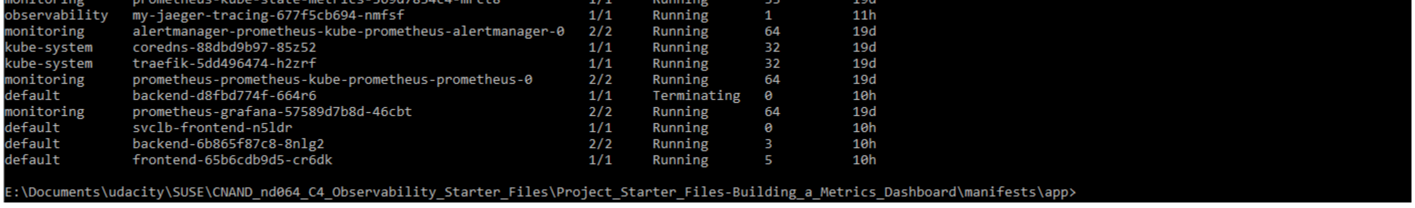
✓

The README includes a screenshot of the `kubectl get pods` output, showing successful installation of the components.

Thanks for listing all pods of all namespaces. All PODs are in the running state. Well done ☑



```
E:\Documents\udacity\SUSE\CNAND_nd064_C4_Observability_Starter_Files\Project_Starter_Files-Building_a_Metrics_Dashboard\manifests\app>kubectl get pods --all-namespaces
NAMESPACE       NAME                                                READY   STATUS      RESTARTS   AGE
kube-system     helm-install-traefik-nc94t                          0/1     Completed   0          19d
monitoring      prometheus-kube-prometheus-operator-bcdfdbc79-v8lbt 1/1     Running     33         19d
kube-system     local-path-provisioner-7ff9579c6-2cb4m              1/1     Running     39         19d
monitoring      prometheus-prometheus-node-exporter-jzs2g           1/1     Running     32         19d
kube-system     metrics-server-7b4f8b595-qr2fv                      1/1     Running     33         19d
observability   jaeger-operator-5977dbf59f-5g56z                    1/1     Running     32         19d
default         svclb-backend-w7wk5                                 1/1     Running     0          10h
kube-system     svclb-traefik-4rzpb                                 2/2     Running     68         19d
monitoring      prometheus-kube-state-metrics-569d7854c4-mpct8      1/1     Running     33         19d
```



```
monitoring      prometheus-kube-state-metrics-5b9d7854c4-mrct8     1/1     Running       35         19d
observability   my-jaeger-tracing-677f5cb694-nmfsf                 1/1     Running       1          11h
monitoring      alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running       64         19d
kube-system     coredns-88dbd9b97-85z52                            1/1     Running       32         19d
kube-system     traefik-5dd496474-h2zrf                            1/1     Running       32         19d
monitoring      prometheus-prometheus-kube-prometheus-prometheus-0 2/2     Running       64         19d
default         backend-d8fbd774f-664r6                            1/1     Terminating   0          10h
monitoring      prometheus-grafana-57589d7b8d-46cbt                2/2     Running       64         19d
default         svclb-frontend-n5ldr                               1/1     Running       0          10h
default         backend-6b865f87c8-8nlg2                           2/2     Running       3          10h
default         frontend-65b6cdb9d5-cr6dk                          1/1     Running       5          10h

E:\Documents\udacity\SUSE\CNAND_nd064_C4_Observability_Starter_Files\Project_Starter_Files-Building_a_Metrics_Dashboard\manifests\app>
```

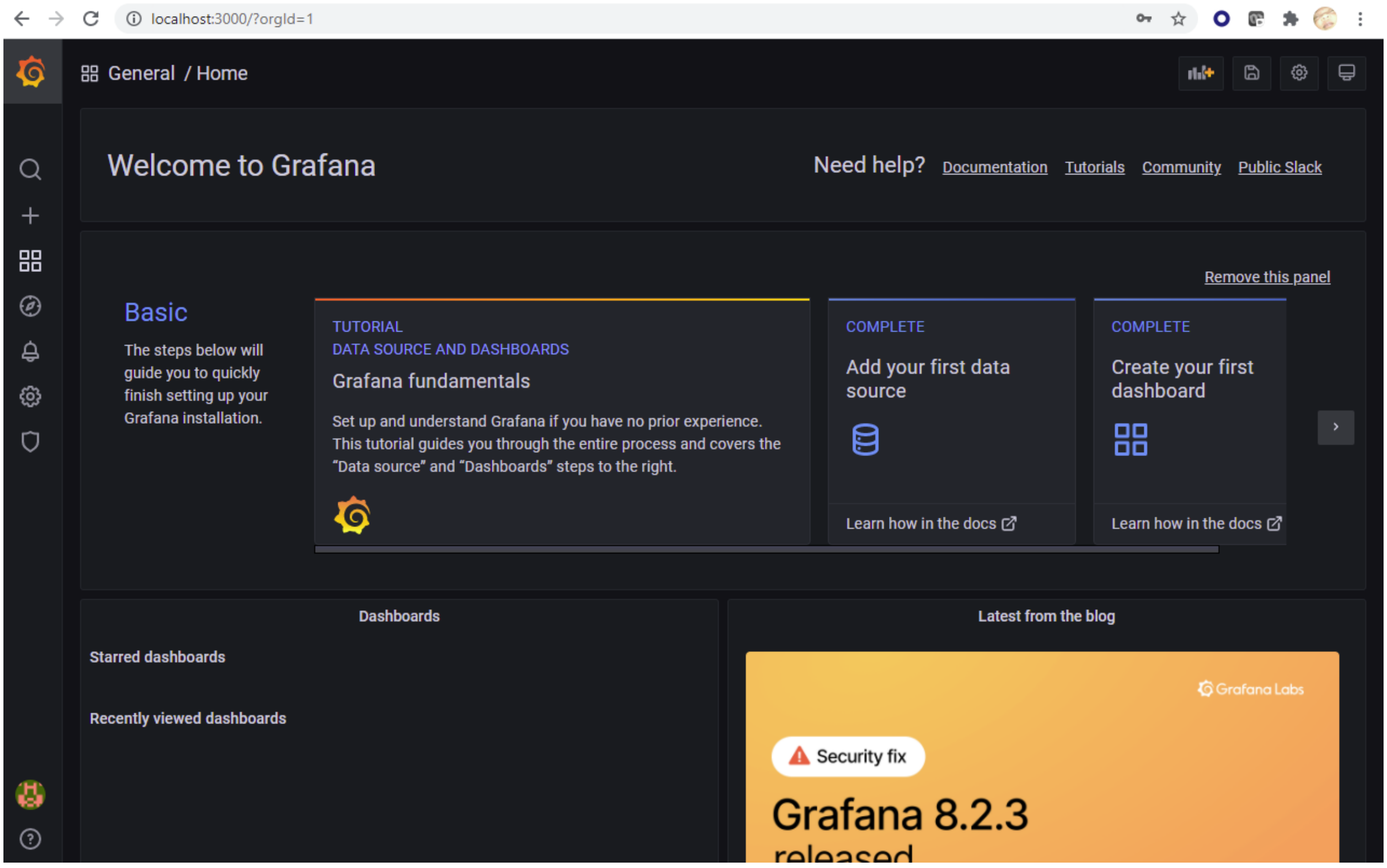
EXTERNAL RESOURCE

🗄 Let’s take a look at how Kubernetes Namespaces can make managing your Kubernetes resources easier.

✓

The README has a screenshot showing the home page after login.

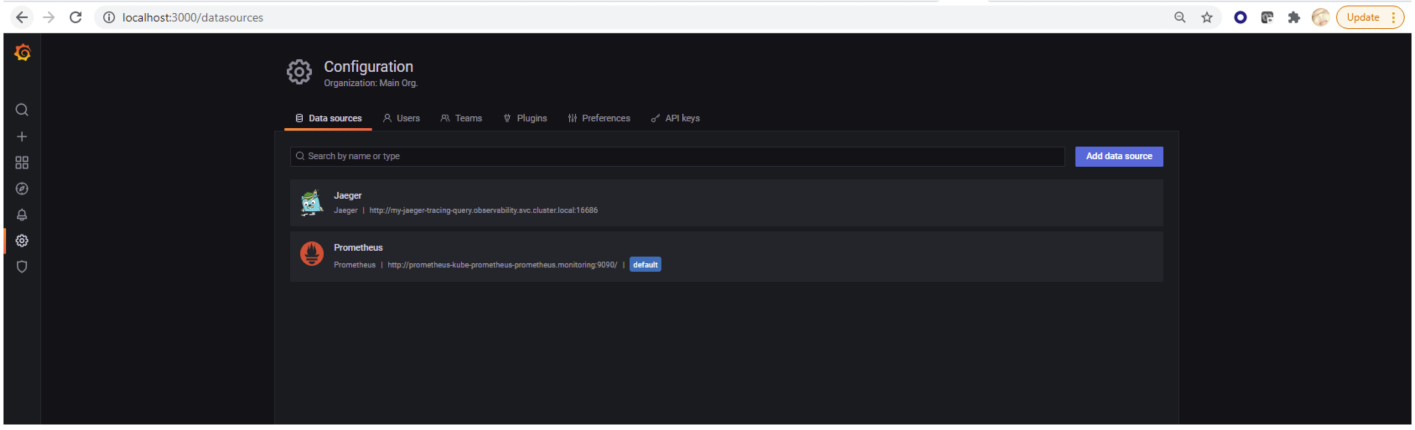
The `README.md` file contains a screenshot that shows the Grafana homepage.


localhost:3000/?orgId=1
General / Home
Welcome to Grafana
Need help?
Documentation
Tutorials
Community
Public Slack
Remove this panel
Basic
The steps below will guide you to quickly finish setting up your Grafana installation.
TUTORIAL
DATA SOURCE AND DASHBOARDS
Grafana fundamentals
Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the “Data source” and “Dashboards” steps to the right.
COMPLETE
Add your first data source
Learn how in the docs
COMPLETE
Create your first dashboard
Learn how in the docs
Dashboards
Starred dashboards
Recently viewed dashboards
Latest from the blog
Grafana Labs
Security fix
Grafana 8.2.3 released

✓

The README includes a screenshot of the dashboard in Grafana that shows Prometheus as a source.

There is a screenshot that confirms the **Prometheus** has been defined correctly as a data source for **Grafana**.


localhost:3000/datasources
Update
Configuration
Organization: Main Org.
Data sources
Users
Teams
Plugins
Preferences
API keys
Search by name or type
Add data source
Jaeger
Jaeger | http://my-jaeger-tracing-query.observability.svc.cluster.local:16686
Prometheus
Prometheus | http://prometheus-kube-prometheus-prometheus.monitoring:9090/ | default

Create Dashboards to Measure SLIs

✓

The README contains a definition of the SLIs, based on an SLO of monthly uptime and request response time.

Good job you have elaborated the SLI definition based on an SLO of monthly uptime and request-response time with an example.

EXTERNAL RESOURCE

After discussing the motivation behind SLOs and error budgets in the CloudNative ND program, this chapter provides a step-by-step recipe to get you started thinking about SLOs, and also some advice about how to iterate from there. We'll then

cover how to use SLOs to make effective business decisions, and explore some advanced topics.

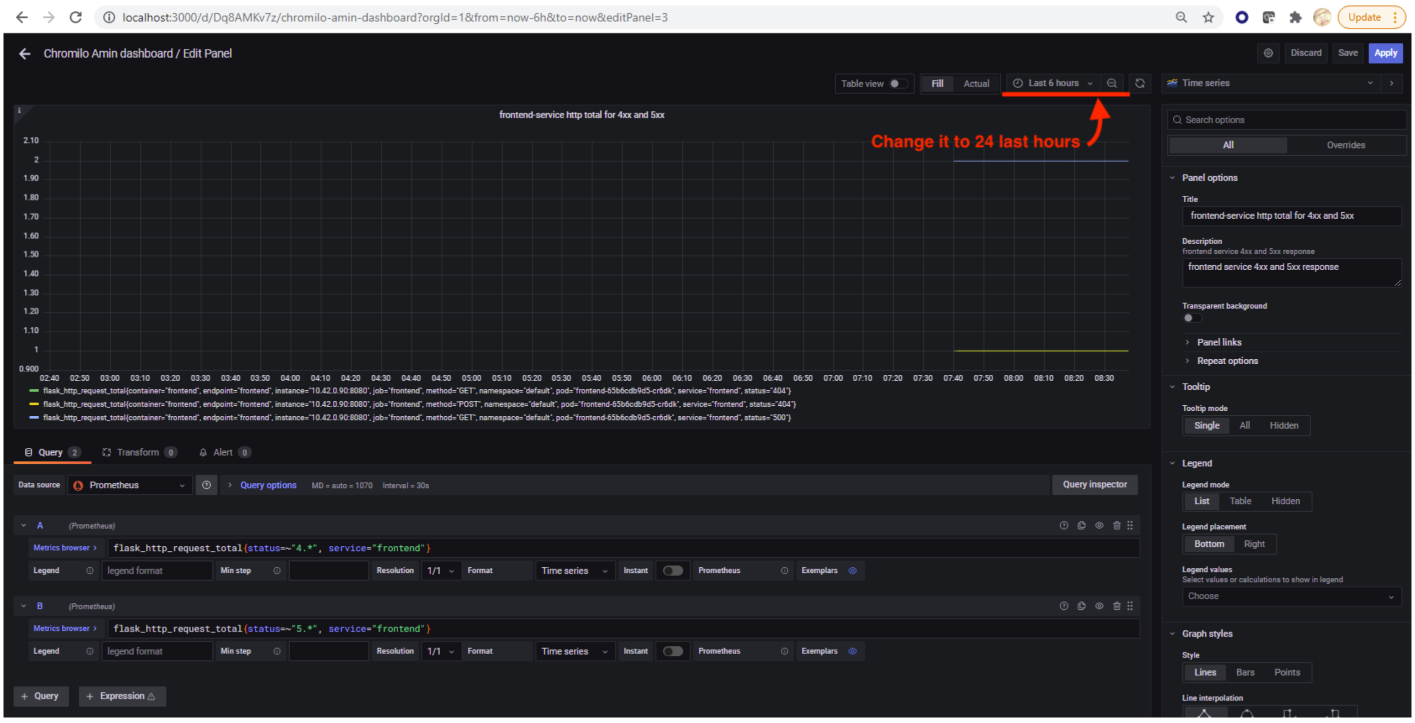
✓

The README contains KPI metrics based on the SLIs given in the project instructions.

Acceptable discussion in five KPI metrics to measure the pre-defined SLIs.

The README will include a screenshot of the finished dashboard containing panels with graphs that measure the required metrics provided by the course.

- According to the project instructions you should provide a screenshot of a dashboard that measures the 40X-50X errors of Backend and Frontend services during the **last 24 hours**. However, as you can see in the screenshot below, the Frontend 40X-50X dashboard shows that information only for the last 6 hours. Please change the period time of that dashboard to the **last 24 hours**.



localhost:3000/d/Dq8AMKv7z/chromilo-amin-dashboard?orgId=1&from=now-6h&to=now&editPanel=3

Chromilo Amin dashboard / Edit Panel

Change it to 24 last hours

frontend-service http total for 4xx and 5xx

```
flask_http_request_total{container="frontend", endpoint="frontend", instance="10.42.0.90:8080", job="frontend", method="GET", namespace="default", pod="frontend-65b6cdb9d5-cr6dk", service="frontend", status="404"}
flask_http_request_total{container="frontend", endpoint="frontend", instance="10.42.0.90:8080", job="frontend", method="POST", namespace="default", pod="frontend-65b6cdb9d5-cr6dk", service="frontend", status="404"}
flask_http_request_total{container="frontend", endpoint="frontend", instance="10.42.0.90:8080", job="frontend", method="GET", namespace="default", pod="frontend-65b6cdb9d5-cr6dk", service="frontend", status="500"}
```

Data source: Prometheus — Query options MD = auto = 1070 Interval = 30s

A (Prometheus)

```
flask_http_request_total{status=~"4.*", service="frontend"}
```

B (Prometheus)

```
flask_http_request_total{status=~"5.*", service="frontend"}
```

Legend: legend format — Min step — Resolution: 1/1 — Format: Time series — Instant — Prometheus — Exemplars

Panel options — Title: frontend-service http total for 4xx and 5xx — Description: frontend service 4xx and 5xx response

- Please add at least one more screenshot of a dashboard that shows the **uptime of Backend and Frontend services during the last 24 hours**.

Tracing

✓

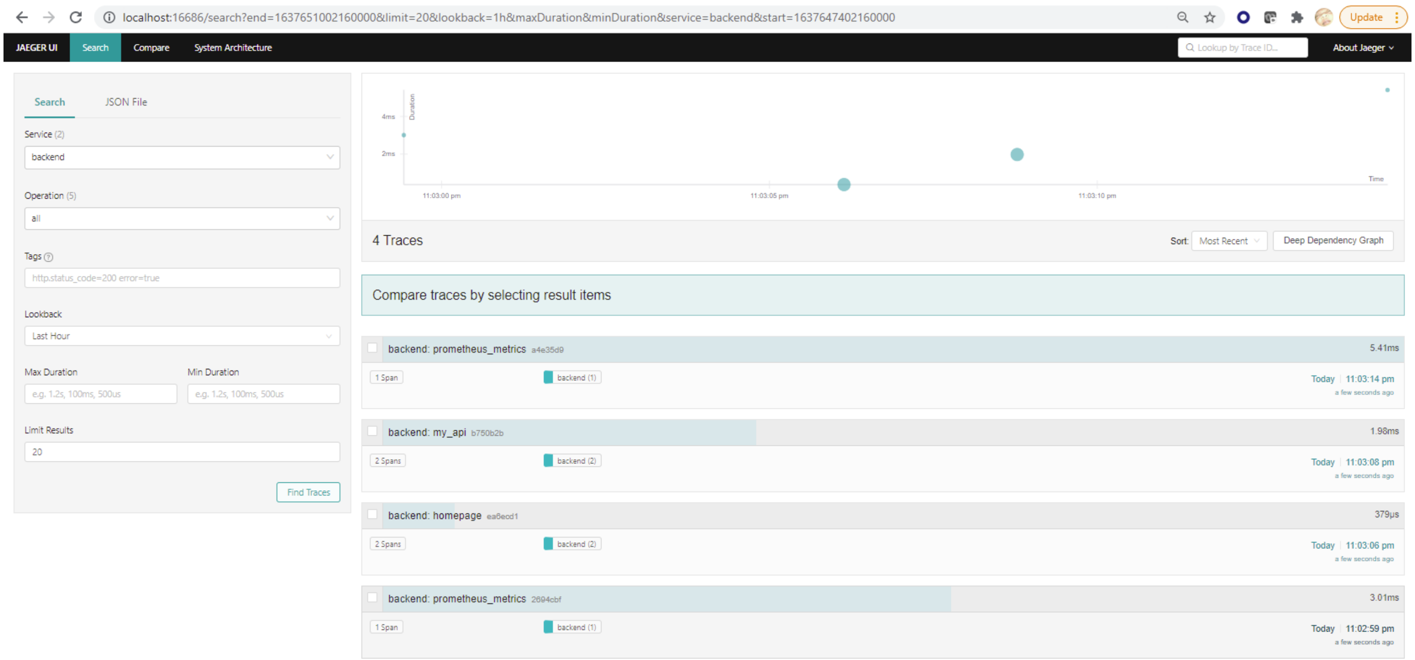
The sample Python file contains a trace and span code to perform Jaeger traces on the Python services.

The Backend reference app has been used in this project as a Python app that contains a trace and span code to perform Jaeger traces on the Python services.

✓

The README contains a screenshot displaying the trace history of the application.

Well done 👍 The `README.md` file contains a screenshot displaying the trace history of the Backend application.



localhost:16686/search?end=1637651002160000&limit=20&lookback=1h&maxDuration&minDuration&service=backend&start=1637647402160000

Service (2): backend

Operation (5): all

Tags: http.status_code=200 error=true

Lookback: Last Hour

Max Duration: e.g. 1.2s, 100ms, 500us

Min Duration: e.g. 1.2s, 100ms, 500us

Limit Results: 20

4 Traces

Sort: Most Recent

Compare traces by selecting result items

backend: prometheus_metrics a4e35d9 — 5.41ms
1 Span — backend (1) — Today | 11:03:14 pm — a few seconds ago

backend: my_api b750b2b — 1.98ms
2 Spans — backend (2) — Today | 11:03:08 pm — a few seconds ago

backend: homepage ea6ecd1 — 379μs
2 Spans — backend (2) — Today | 11:03:06 pm — a few seconds ago

backend: prometheus_metrics 2694cbf — 3.01ms
1 Span — backend (1) — Today | 11:02:59 pm — a few seconds ago

✓

The README contains a comprehensive trouble ticket written for developers.

Good job! The purpose of this exercise is to include the tracer span in the ticket to demonstrate how we can use a tracer to locate the problem easily.

Suggestion

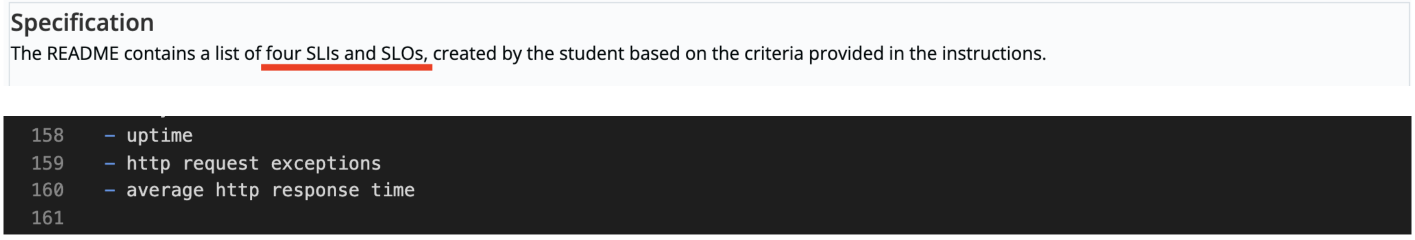
Please include the tracer span in your trouble ticket. Because the purpose of this exercise is to include the tracer span in the ticket to demonstrate how we can use a tracer to locate the problem easily.

Creating Your own Dashboard

🔄

The README contains a list of four SLIs and SLOs, created by the student based on the criteria provided in the instructions.

At the moment, your `README.md` file contains a list of *three* SLIs and SLOs. According to this rubric item we are looking for **four** SLIs and SLOs. Please add at least one more SLI/SLO to meet this rubric item.



Specification

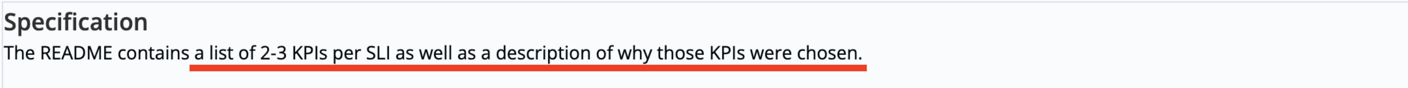
The README contains a list of four SLIs and SLOs, created by the student based on the criteria provided in the instructions.

```
- uptime
- http request exceptions
- average http response time
```

🔄

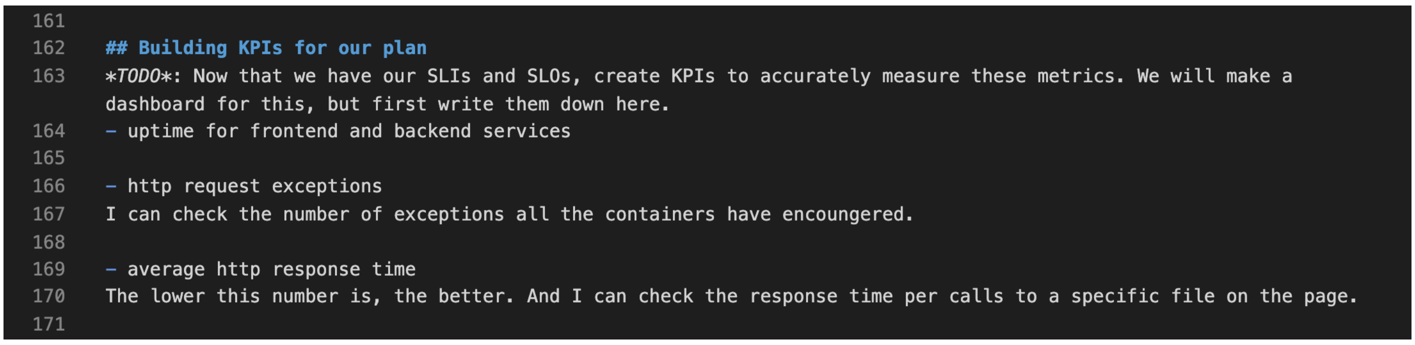
The README contains a list of 2-3 KPIs per SLI as well as a description of why those KPIs were chosen.

Thanks for listing KPIs based on your SLIs. However, according to this rubric item, you should **list 2-3 KPIs per each SLI as well as a short description of why those KPIs were chosen** by you for each SLI.



Specification

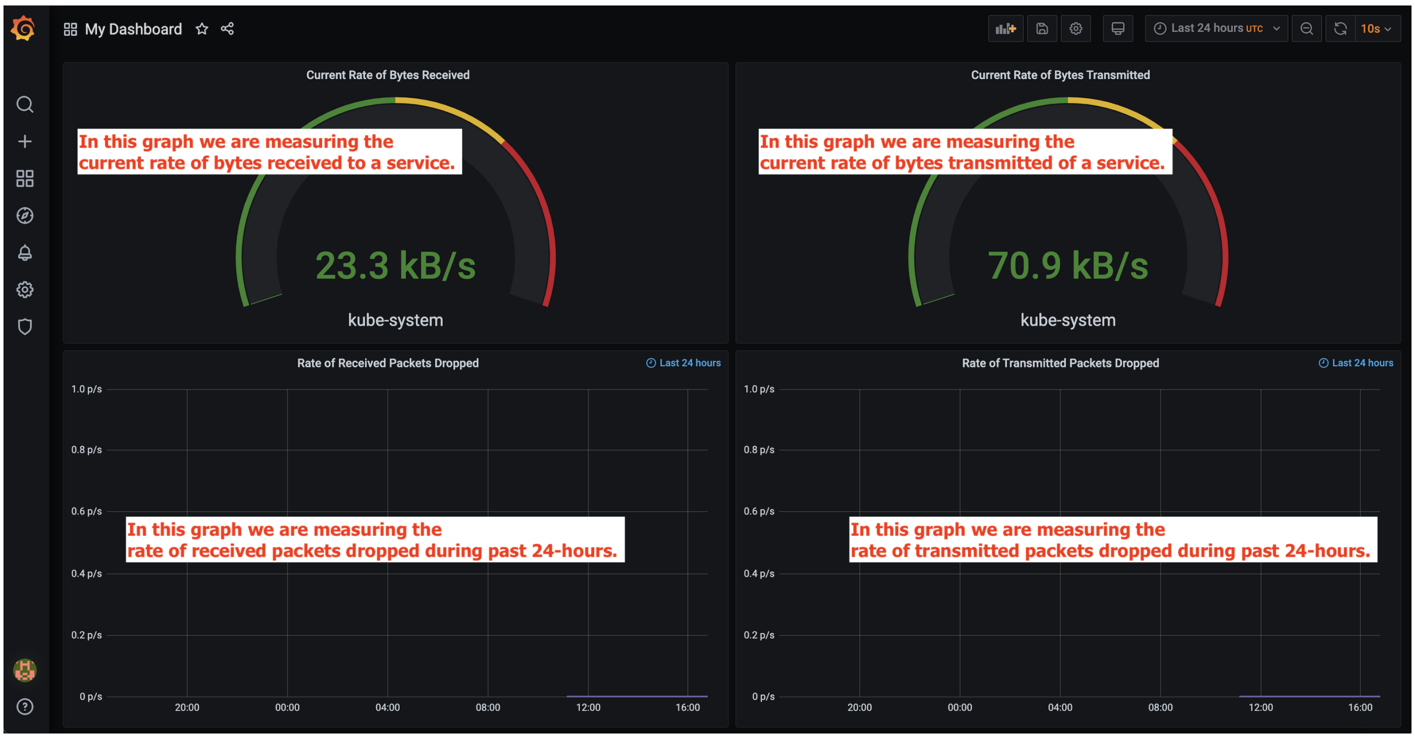
The README contains a list of 2-3 KPIs per SLI as well as a description of why those KPIs were chosen.



```
## Building KPIs for our plan
*TODO*: Now that we have our SLIs and SLOs, create KPIs to accurately measure these metrics. We will make a dashboard for this, but first write them down here.
- uptime for frontend and backend services

- http request exceptions
I can check the number of exceptions all the containers have encoungered.

- average http response time
The lower this number is, the better. And I can check the response time per calls to a specific file on the page.

```

The README contains a screenshot of the completed dashboard as well as a brief description of each graph.

Thanks for including two screenshots as your final dashboard. However, based on this rubric item, the completed dashboard should include **a brief description of each graph.** Please see the screenshot below as a reference. You can also enter a description of each chart in the `README.md` file.



In this graph we are measuring the current rate of bytes received to a service.

In this graph we are measuring the current rate of bytes transmitted of a service.

In this graph we are measuring the rate of received packets dropped during past 24-hours.

In this graph we are measuring the rate of transmitted packets dropped during past 24-hours.

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH