Hello all, I have now passed my project 3, and am providing a **high level guide** (**"AS IS" WITHOUT WARRANTY OF ANY KIND**) to help fellow cohorts. This is ***not*** a ***get-out-of-jail-free card*** , you still need to perform your own work & troubleshooting to get your project done.

This project asks for:

1. Implement distributed tracing with Jaeger for the backend app. This is optional for the frontend app & trial app
2. Implement k8 resource monitoring with Prometheus

To do **Part 1** Jaeger tracing, complete these:

a. **prerequisite**: jaeger installed in ***observability*** ns (Part 4 Observability Ex 2.6) with ***cluster-wide permissions*** (this was omitted in class lessons, but you can read about it [here](https://github.com/jaegertracing/jaeger-operator#getting-started))

b. create a Jaeger instance using the script below. **SKIP**this step if it has been created in [Ex 4.4](https://knowledge.udacity.com/questions/699694) (using the instructions provided in the ***Accepted Answer***) & vice versa.

***This Jaeger instance should be in the same ns as the apps you are tracing (e.g. default ns) and ensure your Jaeger in observability is given cluster wide permissions***.

1

kubectl apply -n observability -f - <<EOF

2

apiVersion: jaegertracing.io/v1

3

kind: Jaeger

4

metadata:

5

name: simplest

6

EOF

c. Add tracing logic to [backend code](https://github.com/udacity/CNAND_nd064_C4_Observability_Starter_Files/blob/master/Project_Starter_Files-Building_a_Metrics_Dashboard/reference-app/backend/app.py) (similar to what [trial code](https://github.com/udacity/CNAND_nd064_C4_Observability_Starter_Files/blob/master/Project_Starter_Files-Building_a_Metrics_Dashboard/reference-app/trial/app.py) has).

d. Add any missing packages to requirements.txt

e. Add sidecar injection annotation to the Deployment section of [backend.yaml](https://github.com/udacity/CNAND_nd064_C4_Observability_Starter_Files/blob/master/Project_Starter_Files-Building_a_Metrics_Dashboard/manifests/app/backend.yaml)

1

apiVersion: apps/v1

2

kind: Deployment

3

metadata:

4

name: backend-app

5

annotations:

6

"sidecar.jaegertracing.io/inject": "true"

7

...

g. Read this [tutorial](https://www.digitalocean.com/community/tutorials/how-to-implement-distributed-tracing-with-jaeger-on-kubernetes) to better understand distributed tracing with Jaeger on K8 (**Part 4 Observability has a lot of gaps on this subject**).

h. For project submission, distributed tracing is only required for the backend app.

To do **Part 2** monitoring K8 resources with Prometheus, complete these:

a. **prerequisite:**prometheus/grafana installed in ***monitoring*** ns (see Part 4 Observability Ex 2.4)

b. create ***3*** service monitors, one each for backend, frontend & trial applications. Example yaml (replace app & port names to suit each app):

1

apiVersion: monitoring.coreos.com/v1

2

kind: ServiceMonitor

3

metadata:

4

name: backend

5

namespace: monitoring

6

labels:

7

app: backend

8

release: prometheus

9

spec:

10

selector:

11

matchLabels:

12

app: backend

13

endpoints:

14

- port: backend

15

path: /metrics

16

interval: 15s

17

namespaceSelector:

18

matchNames:

19

- default

c. add http request export logic to [backend](https://github.com/udacity/CNAND_nd064_C4_Observability_Starter_Files/blob/master/Project_Starter_Files-Building_a_Metrics_Dashboard/reference-app/backend/app.py), [frontend](https://github.com/udacity/CNAND_nd064_C4_Observability_Starter_Files/blob/master/Project_Starter_Files-Building_a_Metrics_Dashboard/reference-app/frontend/app.py) & [trial](https://github.com/udacity/CNAND_nd064_C4_Observability_Starter_Files/blob/master/Project_Starter_Files-Building_a_Metrics_Dashboard/reference-app/trial/app.py) using [Prometheus Flask Exporter](https://github.com/rycus86/prometheus_flask_exporter#prometheus-flask-exporter). **Read**[this](https://github.com/rycus86/prometheus_flask_exporter/tree/master/examples/gunicorn-internal) Github page (***Gunicorn example-internal metrics endpoint***) to understand the code logic you will need to add to the 3 apps you want prometheus to monitor.

d. add missing packages to each app's respective requirements.txt file

e. build new images for the 3 apps and push them to docker hub

f. update ***backend.yaml***, ***frontend.yaml*** & ***trial.yaml***to replace the image names with the ones you pushed to your Docker hub.

g. See [this](https://www.youtube.com/watch?v=mLPg49b33sA) to understand service monitor concept (***this is omitted in*** ***Part 4 Observability lessons***) & [this](https://github.com/rycus86/prometheus_flask_exporter/tree/master/examples/sample-signals) for examples of PromSQL (this is barely explained in the lessons).

Once the above 2 parts are done, deploy the 3 apps to your k3s cluster. then**generate some 20x, 40x & 50x status codes so you can create the required Grafana visuals for project submission. To do this:**

1. Read app.py for the 3 apps and figure out what requests to which endpoints will generate 20x, 40x & 50x status codes
2. use ***curl*** to send the requests to the endpoints, use a for loop to generate multiple requests in one go. See [Ex 5.7 video](https://youtu.be/D1NvpBGqs1I) for the examples.

**Follow the official project steps** to gather submission artifacts (i.e. dashboard visuals and all submission screenshots).

**Warning**: there are all sorts of defects in the starter code, which will render something not working. You will still need to ***scrutinize the code + yaml files*** to correct the ***hidden bombs***. Good luck & have fun ***treasure*** hunting.

**Disclaimer**: this is provided as a good will to help fellow cohorts navigate the pains induced by the errors and gaps in the official project instructions - ***until*** Udacity eventually corrects the official project instructions. This is ***not*** a replacement of Udacity official instructions.