**Overview**

Create a solution (in either golang or python) designed to run on a Kubernetes Cluster to monitor internet urls and provide Prometheus metrics, once completed please upload your solution to your [github.com](https://nam04.safelinks.protection.outlook.com/?url=http%3A%2F%2Fgithub.com%2F&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=fxLVfW0flNWP%2BeLgYiNel91K2rcsFKB1UR298BxN%2F%2FA%3D&reserved=0) account.  Please make it public.

**Requirements**

* A service written in python or golang that queries 2 urls ([https://httpstat.us/503](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F503&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=S2EnHJCLgI3jkRIT155IdYylnwTjNngchRec2LUGvyk%3D&reserved=0) & [https://httpstat.us/200](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F200&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=noWQTAkRQUGzmoh7kJlnS5uhfAmucfaNppMy0%2FN8bpQ%3D&reserved=0))
* The service will check the external urls ([https://httpstat.us/503](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F503&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=S2EnHJCLgI3jkRIT155IdYylnwTjNngchRec2LUGvyk%3D&reserved=0) & [https://httpstat.us/200](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F200&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=noWQTAkRQUGzmoh7kJlnS5uhfAmucfaNppMy0%2FN8bpQ%3D&reserved=0) ) are up (based on http status code 200) and response time in milliseconds
* The service will run a simple http service that produces metrics using appropriate Prometheus libraries and outputs on /metrics
* Expected response format:
  + sample\_external\_url\_up{url="[https://httpstat.us/503](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F503&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=S2EnHJCLgI3jkRIT155IdYylnwTjNngchRec2LUGvyk%3D&reserved=0) "}  = 0
  + sample\_external\_url\_response\_ms{url="[https://httpstat.us/503](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F503&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=S2EnHJCLgI3jkRIT155IdYylnwTjNngchRec2LUGvyk%3D&reserved=0) "}  = [value]
  + sample\_external\_url\_up{url="[https://httpstat.us/200](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F200&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=noWQTAkRQUGzmoh7kJlnS5uhfAmucfaNppMy0%2FN8bpQ%3D&reserved=0) "}  = 1
  + sample\_external\_url\_response\_ms{url="[https://httpstat.us/200](https://nam04.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhttpstat.us%2F200&data=04%7C01%7Cjaporter%40vmware.com%7C8a433f61b9044245c1cc08d9fbad5518%7Cb39138ca3cee4b4aa4d6cd83d9dd62f0%7C0%7C1%7C637817544156607810%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C2000&sdata=noWQTAkRQUGzmoh7kJlnS5uhfAmucfaNppMy0%2FN8bpQ%3D&reserved=0) "}  = [value]

In order to prepare for the above problem statement, I started with the python code for the image.

This repository collects Kubernetes manifests, [Grafana](http://grafana.com/) dashboards, and [Prometheus rules](https://prometheus.io/docs/prometheus/latest/configuration/recording_rules/) combined with documentation and scripts to provide easy to operate end-to-end Kubernetes cluster monitoring with [Prometheus](https://prometheus.io/).

So, here I would be using 2 types of metrics to fetch the required data from the Kubernetes application.

**Counter**

A counter is a cumulative metric that represents a single [monotonically increasing counter](https://en.wikipedia.org/wiki/Monotonic_function) whose value can only increase or be reset to zero on restart. For example, you can use a counter to represent the number of requests served, tasks completed, or errors.

Do not use a counter to expose a value that can decrease. For example, do not use a counter for the number of currently running processes; instead use a gauge.

**Histogram**

A histogram samples observations (usually things like request durations or response sizes) and counts them in configurable buckets. It also provides a sum of all observed values.

A histogram with a base metric name of <basename> exposes multiple time series during a scrape:

cumulative counters for the observation buckets, exposed as <basename>\_bucket{le="<upper inclusive bound>"}

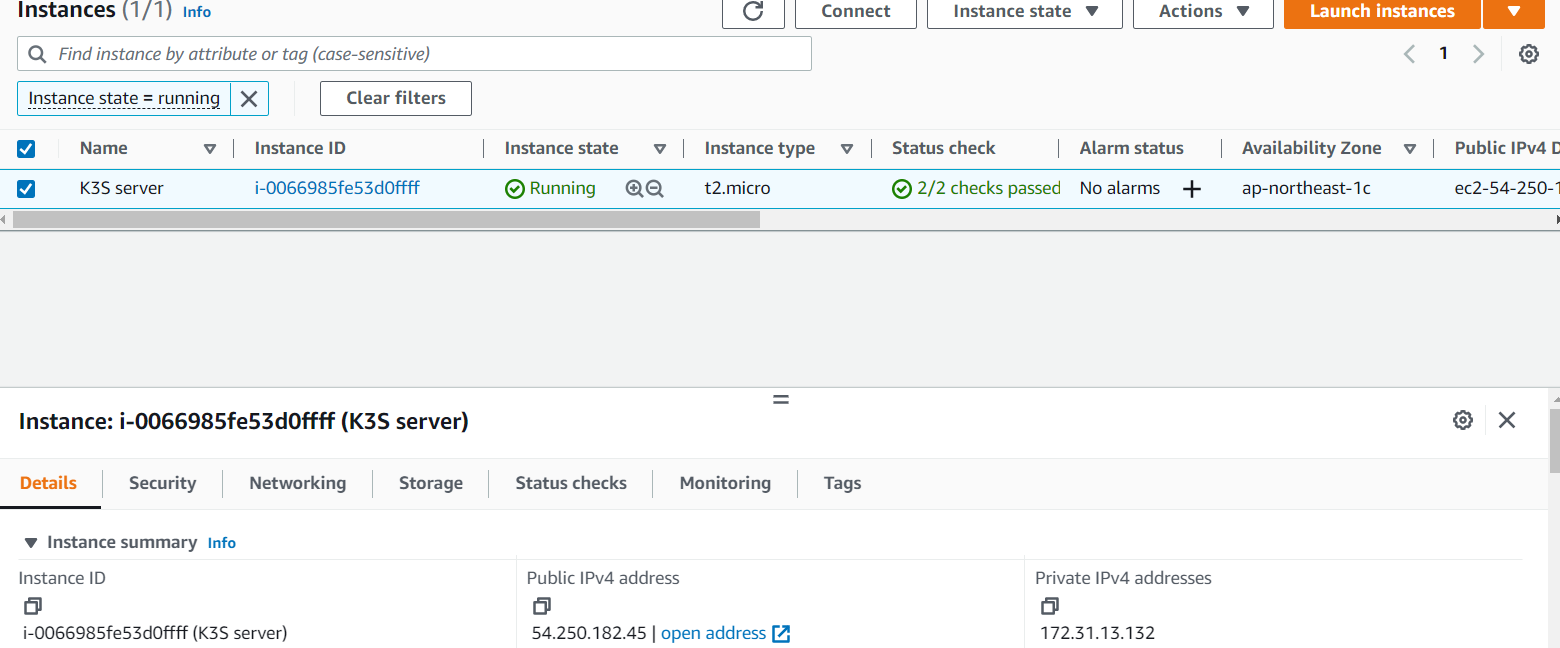
the total sum of all observed values, exposed as <basename>\_sum

the count of events that have been observed, exposed as <basename>\_count (identical to <basename>\_bucket{le="+Inf"} above)

Use the [histogram\_quantile() function](https://prometheus.io/docs/prometheus/latest/querying/functions/" \l "histogram_quantile) to calculate quantiles from histograms or even aggregations of histograms. A histogram is also suitable to calculate an [Apdex score](https://en.wikipedia.org/wiki/Apdex). When operating on buckets, remember that the histogram is [cumulative](https://en.wikipedia.org/wiki/Histogram#Cumulative_histogram). See [histograms and summaries](https://prometheus.io/docs/practices/histograms) for details of histogram usage and differences to [summaries](https://prometheus.io/docs/concepts/metric_types/#summary).

Post creation of the python code (***server.py***), we need to create a docker image for the same.

Here, I used AWS EC2 instances to build my job, so first I tried to install docker post launching the EC2 instance:



Update the packages on your instance

*sudo yum update -y*

Install Docker

*sudo yum install docker -y*

Start the Docker Service

*sudo service docker start*

Add the ec2-user to the docker group so you can execute Docker commands without using sudo.

*sudo usermod -a -G docker ec2-user*

**DockerFile for the python code**:

FROM python:3.10.5-alpine3.16 AS build

RUN mkdir /app/

WORKDIR /app/

COPY ./src/requirements.txt /app/requirements.txt

RUN pip install -r requirements.txt

COPY ./src/ /app/

ENV FLASK\_APP=server.py

EXPOSE 8000

EXPOSE 5000

CMD flask run -h 0.0.0.0 -p 5000

**Commands:**

*# docker build -t server/python:1.0 -f Dockerfile* .

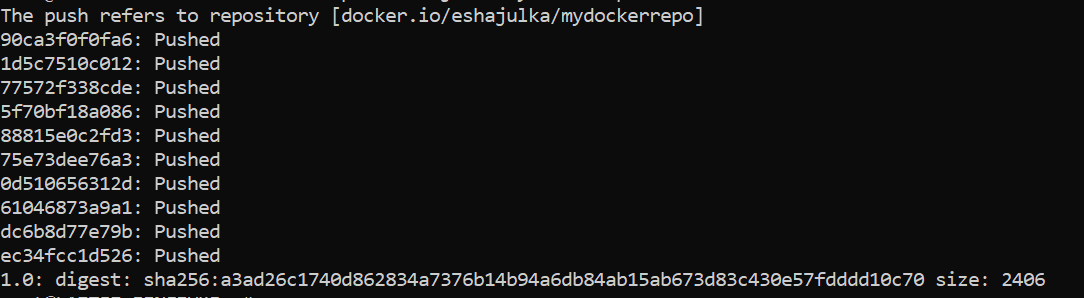
Once the command is executed, we can check under : docker images

**Pushing Image to DockerHub:**

*# docker login (Enter the credentials for the dokerhub)*

*# docker tag server/python:1.0 eshajulka/mydockerrepo:1.0*

*# docker push eshajulka/mydockerrepo:1.0 (push to docker hub)*



Now we, need to install the kubernetes cluster on the EC2 instance. So for this here I have used k3s (lightweight Kubernetes cluster) installation.

#### **K3S Installation: (The certified Kubernetes distribution built for IoT & Edge computing)**

This can be installed on the same EC2 instance with the help of the below command from ([K3s](https://k3s.io/)).

*curl -sfL https://get.k3s.io | sh -*

# Check for Ready node, takes ~30 seconds

*k3s kubectl get node*

Post the installation is successful, we can execute the kubectl commands to verify.

**Creating deployments in the Kubernetes cluster**:

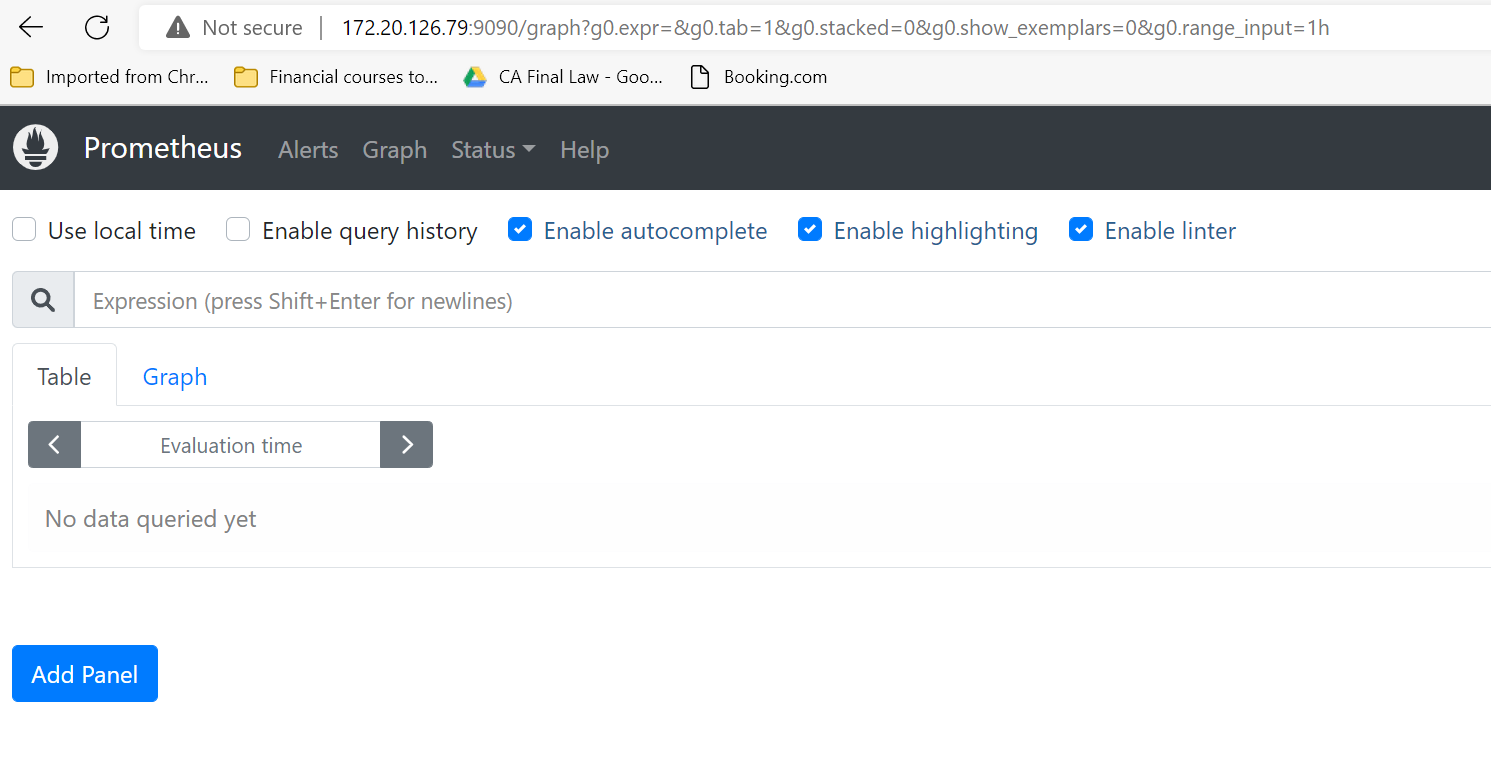
The deployments and the respective services can be created with the help of Yaml manifest files.



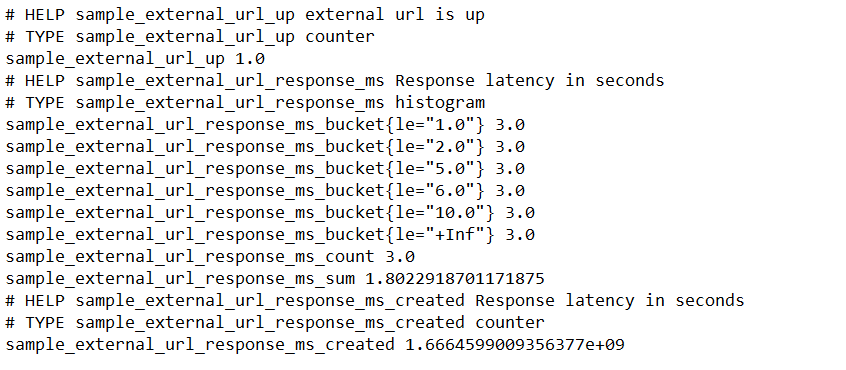
For application, I have given the below command:

Kubectl create deployment server-python --image=eshajulka/mydockerrepo --replicas=2

Post this I tried accessing the links described in the overview sections:



<https://httpstat.us/200>



<https://httpstat.us/503>

