**Tutorial #1:** <https://github.com/jeremyjordan/ml-monitoring>

**Google Kubernetes Engine**

* Create Cluster
* Connect to Cluster -> 3 dots at the GKE Clusters -> Connect
* Copy paste command: gcloud container clusters …. (into VM terminal)

**Kubectl**

* Get Namespace: kubectl get ns
* Get All resources (to verify deployment): kubectl get all -n my-namespace
* Delete all in namespace: kubectl delete all --all -n my-namespace

**Helm**

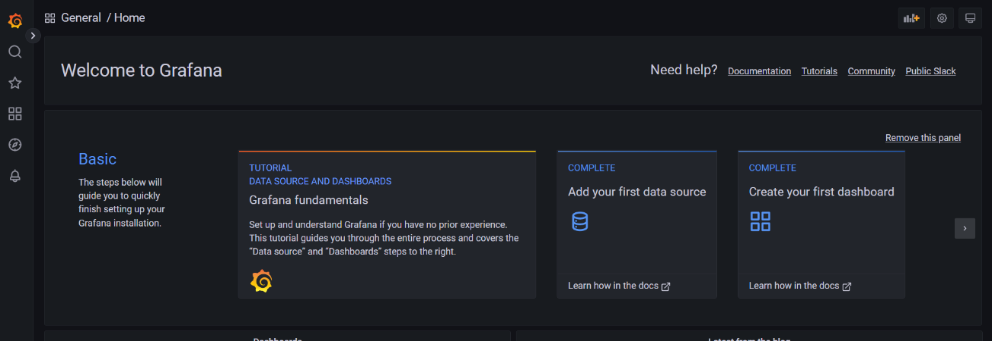
* Install Helm: <https://helm.sh/docs/intro/install/>
* Check if helm is installed: helm version
* List all helm repos: helm repo list
* Install helm charts (on Kube/Prometheus Stack): <https://artifacthub.io/packages/helm/prometheus-community/kube-prometheus-stack>

**Instructions to Deploy Kube-Prometheus-Stack to Kubernetes Cluster**

* kubectl create namespace monitoring
* helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>
* helm repo update
* helm install prometheus-stack prometheus-community/kube-prometheus-stack -n monitoring
* Verify all resource deployed successfully: kubectl get all -n monitoring

**Accessing Grafana Dashboard (NEED TO DO IN LOCAL MACHINE -> CANNOT BE DONE IN CLOUD SHELL)**

* Install gcloud sdk, kubectl, helm on local machine
* Port Forward (from TCP 80 -> Localhost 8000): kubectl port-forward svc/prometheus-stack-grafana 8000:80 -n monitoring
  + Username: admin
  + Password: prom-operator
* Go to <http://localhost:8000/> to view Grafana Dashboard
* A screenshot of a computer

  Description automatically generated with low confidence****Search -> General -> Pick any dashboard to view default dashboard
* Import Custom Dashboard from Repo
  + Dashboard -> Import -> Upload JSON file (can use below for example or custom built dashboard)
  + <https://github.com/jeremyjordan/ml-monitoring/blob/main/dashboards/model.json>

**Flask**

* Define API methods
* When testing locally:
  + Run python flask\_app.py
  + Test endpoint in http://localhost:8080
  + WARNING: “/metrics” endpoint only works if ENV Variable PROMETHEUS\_MULTIPROC\_DIR is set -> (Endpoint will work inside docker container)
* Prometheus-Flask-Exporter (<https://github.com/rycus86/prometheus_flask_exporter>) for simple metrics
* Need to use Prometheus python client for custom metrics (ex: Tracking Prediction value outputs)

**Docker**

* Create a new folder called docker to keep everything clean
* Cd into that directory
* Build Images (name is wine-quality-model:v1): docker build -t wine-quality-model:v1 .
* Build Container (map to port 8080): docker run -it -p 8080:8080 wine-quality-model:v1
* Stop container: docker stop containerID
* View all container: docker ps -a
* View all images: docker images

**Push Local Docker image to GCR**

* Tag local image: docker tag LOCAL\_IMAGE\_ID gcr.io/PROJECT-ID/IMAGE\_NAME:TAG
* List all images: docker images
* Need Service Account to publish to GCR
  + Service Account Permission: Storage Admin
  + Login: gcloud auth login
  + Login as service acct: gcloud auth activate-service-account [pushtogcrservacct@PROJECTNAME.iam.gserviceaccount.com](mailto:pushtogcrservacct@PROJECTNAME.iam.gserviceaccount.com) --key-file=keyfile.json
  + gcloud auth configure-docker
* Push to GCR: docker push HOSTNAME/PROJECT-ID/IMAGE:TAG

**ML Model Deploy on Kubernetes**

* Use YAML file to create deployment
* Apply Command to Create Deployment (file is inside Kubernetes folder): kubectl apply -f wine-quality-model.yaml -n monitoring
* Verify service is deployed: kubectl get all -n moni
* toring
* Test out Flask API (from TCP 8080 -> Localhost 8080): kubectl port-forward svc/wine-quality-model-service 8080:8080 -n monitoring
* Verify Deployment: kubectl get prometheuses.monitoring.coreos.com prometheus-stack-kube-prom-prometheus -n monitoring -o yaml
* Test out Flask API (in Postman): <http://localhost:8080>
* “/metrics” -> should display metrics scaped from Prometheus

**Prometheus UI**

* Prometheus UI: kubectl port-forward svc/prometheus-stack-kube-prom-prometheus 9090:9090 -n monitoring
* Look at Service scraped: Status -> Targets -> wine-quality-model-service
* Graph Tab -> Explore PromQL in Prometheus UI

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidence

**Custom Grafana Dashboard**

**Graphical user interface

Description automatically generated**

**Prometheus Query used in Dashboard**

* Model Score: sum(increase(regression\_model\_output\_sum[30m])) / sum(increase(regression\_model\_output\_count[30m]))
* Model Prediction: sum(increase(regression\_model\_output\_bucket[30m])) by (le)
* HTTP success rate: sum(increase(request\_count\_total{http\_status="200", exported\_endpoint='/predict'}[30m])) / sum(increase(request\_count\_total{exported\_endpoint='/predict'}[30m]))
* Requests: sum by (exported\_endpoint) (rate(request\_latency\_seconds\_bucket[1m]))
* Latency: histogram\_quantile(0.99, sum(rate(request\_latency\_seconds\_bucket[1m])) by (le))

**Load Test using Locust**

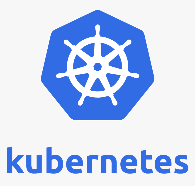
* Build Images: docker build -t load-test-locust:v1 .
* Build Container (map to port 8089): docker run -it -p 8089:8089 load-test-locust:v1
* Upload Container to GCR just like the model container
* Use YAML file to create deployment
* Create Deployment (file is inside Kubernetes folder): kubectl apply -f load-test.yaml -n monitoring
* Verify service is deployed: kubectl get all -n monitoring
* Open Locust UI (from TCP 8089 -> Localhost 8089): kubectl port-forward svc/locust-master-service 8089:8089 -n monitoring
* Go to Locust UI: [http://localhost:808](http://localhost:8080)9
* Run Locust Test -> Host = <http://wine-quality-model-service-ip:port>
* How to find wine-quality-model-service-ip?
  + A screenshot of a computer

    Description automatically generated with medium confidenceGet all resources in kubernetes: kubectl get all -n monitoring
* **A screenshot of a computer

  Description automatically generated**Locust UI to get summary of Locust Test

**Teardown Instructions**

* Kubectl delete -f load-test.yaml -n monitoring
* Kubectl delete -f wine-quality-model.yaml -n monitoring
* Helm uninstall Prometheus-stack -n monitoring
* Kubectl delete namespace monitoring

**Architecture Diagram**

Dashboard

Visualize

Pull Metrics

Load Test

Deployed on

**Graphical user interface

Description automatically generated****Text

Description automatically generated**

Logo, icon

Description automatically generated

Model wrapped in Flask API

**Chart

Description automatically generated**

**Reference:**

* <https://github.com/jeremyjordan/ml-monitoring>
* <https://github.com/Anishmourya/flask-prometheus-gunicorn-docker-real-app> (To Create Flask Prometheus App)