Cloud Native Monitoring

In this section we will cover

- Sock Shop: A Microservice Demo Application.

Run the Sock Shop

Read the documentation located at:
https://microservices-demo.github.io/microservices-demo

```
$ git clone https://github.com/microservices-demo/microservices-demo.

$ kubectl create ns sock-shop
$ kubectl apply -f microservices-demo/deploy/kubernetes/complete-demo
```

Expose the Front-end

- Modify the front-end service of the Sock Shop to be of type LoadBalancer.
- Then access the application via the public DNS name.

Installing Prometheus

- `cd` to the `/deploy/kubernetes/manifests-monitoring` directory.
- Create the monitoring namespace with `kubectl apply -f monitoring-ns.yaml`
- Create an alertrules volume. Although we're not going to use alerts, it is required by the deployment: `kubectl apply -f prometheus-alertrules.yaml`

- Create a `ConfigMap` that inform prometheus to scrape kubernetes services `kubectl apply -f prometheus-configmap.yaml`
- Deploy and expose prometheus: `kubectl apply -f prometheus-dep.yaml` and `kubectl apply -f prometheus-svc.yaml`

Connecting to Prometheus

First, connect to Prometheus. The service exposed a `NodePort` on port `31090`.

As before we will need the external IP of one of the cluster nodes

```
$export EXTERNAL_IP=$(kubectl get nodes \
  -o jsonpath='{.items[1].status.addresses[?(@.type=="ExternalIP")].a
```

- Using your local browser, browse to `http://<EXTERNAL_IP>:31090`. You should see the prometheus UI.

Using Prometheus

Now take a look around. Try:

- Read the documentation at: https://prometheus.io/docs/querying/basics/
- Plot the rate of requests (the rate of `request_duration_seconds_count`) - your results may vary, this depends on the requests to the sock shop.

- Deploy the `load-test` manifest. This starts a container that loads the sock-shop. `kubectl apply -f /deploy/kubernetes/manifests/loadtest-dep.yaml`
- Watch the rate of requests again.
- Use the `request_duration_seconds_bucket` parameter to plot the median and 95% percentile duration of the requests made by the load-test. (see the docs)

Installing Grafana

Now we're going to install Grafana. Grafana is an open source dashboarding application that has good integration with Prometheus.

Rather than applying individual Grafana deployments and services, it is easier to just deploy the entire `kubernetes-monitoring` directory.

```
cd ~ ; kubectl apply -f microservices-demo/deploy/kubernetes/manifests-monitoring/
```

- Get the pods for the monitoring namespace and make sure the Grafana pod starts.

```
$ kubectl get pods -n monitoring
```

Using Grafana

Our service manifest declares a `NodePort` of 31300.

- Open up the GKE firewall to allow port 31300
- Login with the default username and password of `admin` and `admin`
- Browse to the `sock-shop-performance` dashboard.

Tasks

Grafana documentation can be found here: http://docs.grafana.org/guides/basic_concepts/

- Create your own dashboard.
- Create a 95th percentile latency plot
- Create a "metric" showing the number of sock orders.

Bonus Exercise

If you've finished all this, and you have some extra time, browse to the alertmanager directory:

```
cd /deploy/kubernetes/manifests-alerting/
```

Investigate the Prometheus alertmanager which provides an alert mechanism for Prometheus.