Helm Setup for Local Minikube

Prerequisite:

- Minikube
- Kubectl
- Helm

Step 1: Install Minikube.

Link: https://kubernetes.io/docs/tasks/tools/install-minikube/

Step 2: Check Minikube is properly installed or not by using:

$ minikube status

Output:

host: Running
kubelet: Running
api server: Running
kubectl: correctly configured: Pointing to minikube VM at 192.168.10.100

Step 3: Now Install and Set Up Kubectl.

link: https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-linux/

Step 4: Install helm in your system and config it's tiller.

link: https://helm.sh/docs/using_helm/#installing-helm

For configuring your tiller:

$ kubectl create serviceaccount --namespace kube-system tiller

$ kubectl create clusterrolebinding tiller-cluster-rule --clusterrole=cluster-admin --

serviceaccount=kube-system:tiller

$ helm init --service-account tiller --upgrade

Step 4.1: You can download helm version according to your need for this click on below links

link: https://github.com/helm/helm/releases

Step 5: Now follow the steps README file for the deployment of helm chart.

Step 6: You can check the status using:

$ helm status name_of_the_release This command show status of Helm chart installed.

Output:

```
LAST DEPLOYED: Tue Jun 11 17:06:58 2019
NAMESPACE: nexaops
STATUS: DEPLOYED

RESOURCES:
==> v1/ConfigMap
NAME        DATA  AGE
prometheus  1     3h31m

==> v1/Pod(related)
NAME                                                READY  STATUS   RESTARTS  AGE
testing-stuart-microservice-stack-database-cd9d46947-w4t4n       1/1    Running  0         3h31m
testing-stuart-microservice-stack-grafana-7644fd944f-7cfwf       1/1    Running  0         3h31m
testing-stuart-microservice-stack-microservice-api-689d795lphlz  1/1    Running  0         3h31m
testing-stuart-microservice-stack-prometheus-75f965cff9-slgbl    1/1    Running  0         3h31m

==> v1/Service
NAME                                        TYPE      CLUSTER-IP      EXTERNAL-IP  PORT(S)         AGE
testing-stuart-microservice-stack-database          NodePort  10.102.17.132   <none>       5432:32459/TCP  3h31m
testing-stuart-microservice-stack-grafana           NodePort  10.104.227.61   <none>       3000:31188/TCP  3h31m
testing-stuart-microservice-stack-microservice-api  NodePort  10.98.20.40     <none>       5000:32583/TCP  3h31m
testing-stuart-microservice-stack-prometheus        NodePort  10.111.189.141  <none>       9090:30718/TCP  3h31m

==> v1beta2/Deployment
NAME                                        READY  UP-TO-DATE  AVAILABLE  AGE
testing-stuart-microservice-stack-database          1/1    1           1          3h31m
testing-stuart-microservice-stack-grafana           1/1    1           1          3h31m
testing-stuart-microservice-stack-microservice-api  1/1    1           1          3h31m
testing-stuart-microservice-stack-prometheus        1/1    1           1          3h31m
```

Step 7: Now check all pods are running or not by using:

$ kubectl get pods This command will show you all the pods.

Output:

```
testing-stuart-microservice-stack-database-cd9d46947-w4t4n       1/1    Running  0         108m
testing-stuart-microservice-stack-grafana-7644fd944f-7cfwf       1/1    Running  0         108m
testing-stuart-microservice-stack-microservice-api-689d795lphlz  1/1    Running  0         108m
testing-stuart-microservice-stack-prometheus-75f965cff9-slgbl    1/1    Running  0         108m
```

Step 8: If all the pods are in the Running state then check your all services by using:

$ kubectl get svc This command will show you all the services.

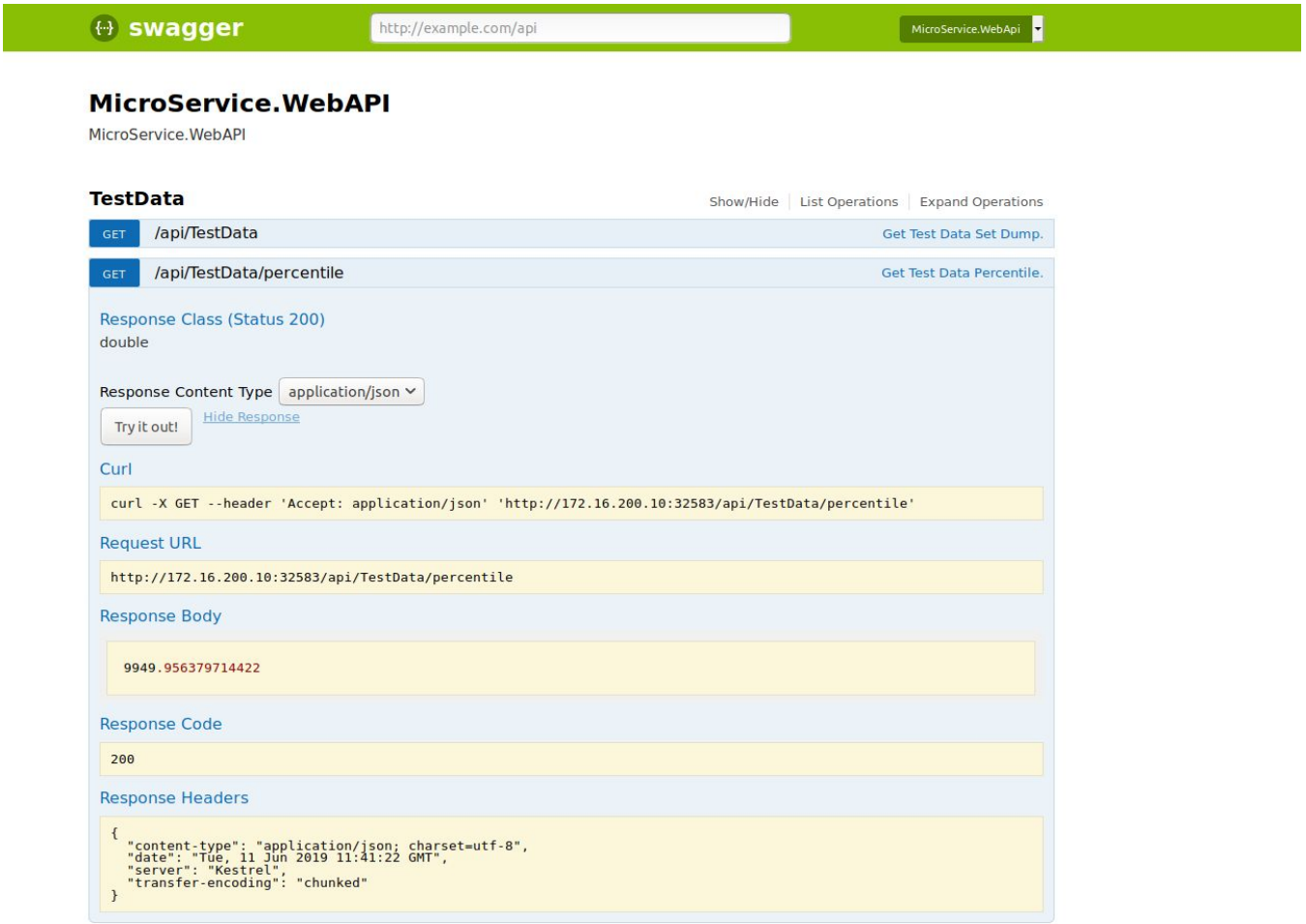
Output:

```
testing-stuart-microservice-stack-database          NodePort       10.102.17.132    <none>        5432:32459/TCP
                                                                                                                 137m
testing-stuart-microservice-stack-grafana           NodePort       10.104.227.61    <none>        3000:31188/TCP
                                                                                                                 137m
testing-stuart-microservice-stack-microservice-api  NodePort       10.98.20.40      <none>        5000:32583/TCP
                                                                                                                 137m
testing-stuart-microservice-stack-prometheus        NodePort       10.111.189.141   <none>        9090:30718/TCP
                                                                                                                 137m
```

Step 9: When all the pods are in the running state then you can check the output in your web browser.

Screenshots:

Microservices-api: URL: <NodeIP:Portno/swagger>



Prometheus: URL: <NodeIP:Portno**>**

Grafana: URL: <NodeIP:Portno>