How to setup GKE Cluster directly using Docker Desktop for Mac

Kubernetes - Beginners | Intermediate | Advanced

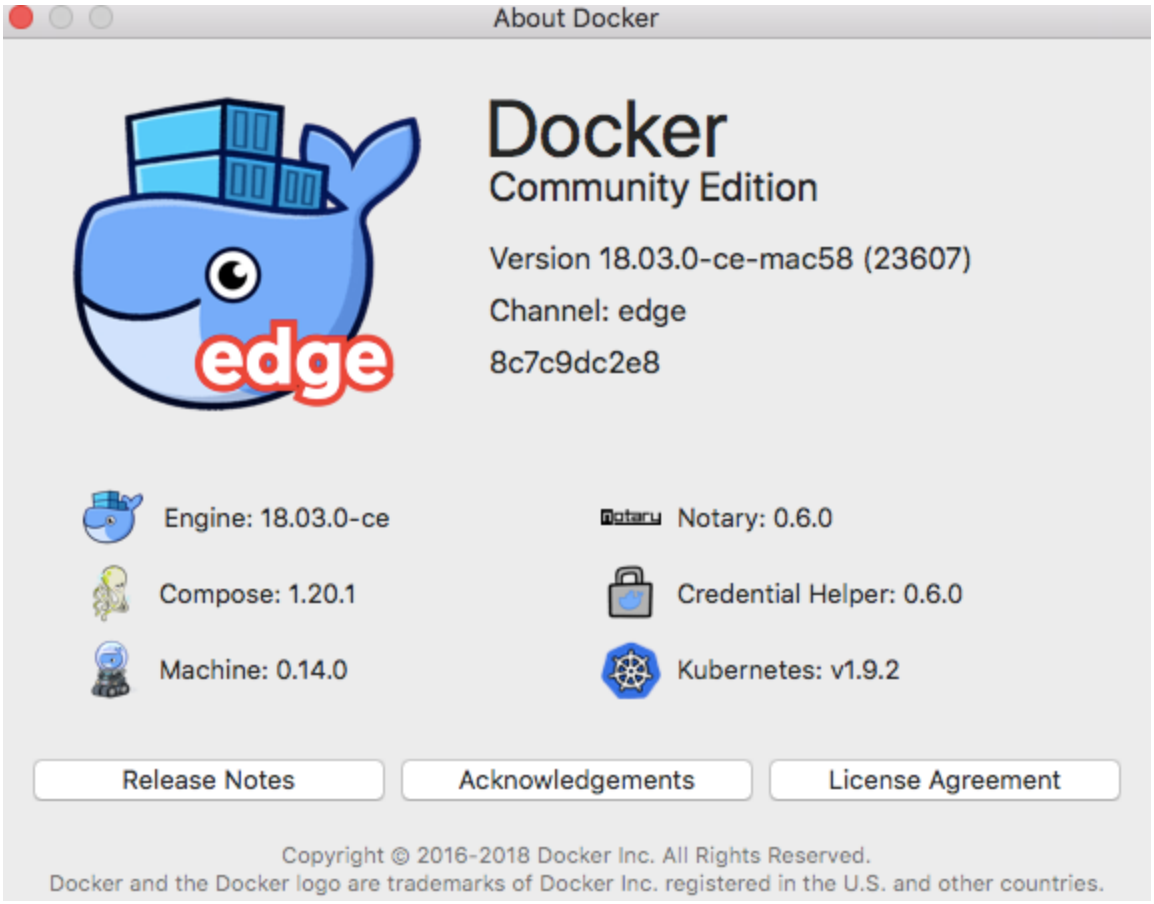
View on GitHub Join Slack

How to setup GKE Cluster directly using Docker Desktop for Mac

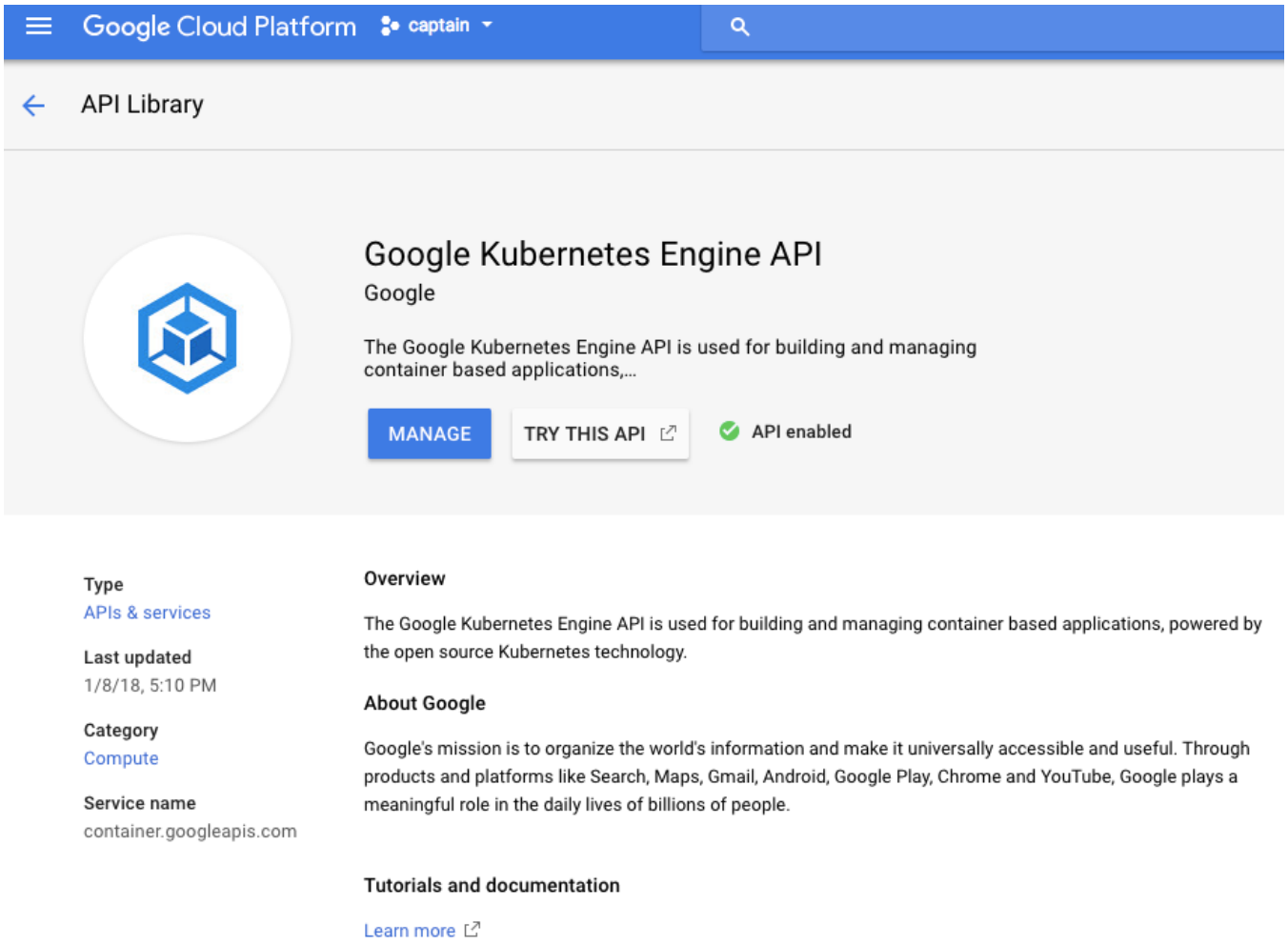
Refer this link

Pre-requisite

- Install/Upgrade Docker for Mac 18.03 CE Edition



- Install google-cloud-sdk
- Enable Google Cloud Engine API



- Authenticate Your Google Cloud using `gcloud auth`

Step-1

Installing Google Cloud SDK on your macOS

Make sure that Python 2.7 is installed on your system:

```
Ajeets-MacBook-Air:~ ajeetraina$ python -V
Python 2.7.10
```

Download the corresponding version of Google Cloud SDK. In this case the Mac OS version for 64-bits systems is downloaded.

```
wget https://dl.google.com/dl/cloudsdk/channels/rapid/downloads/google-cloud
```

Untar the downloaded file, as follows:

```
tar xfz google-cloud-sdk-195.0.0-darwin-x86_64.tar.gz
```

and execute the following command to install Google Cloud SDK in your system:

```
./google-cloud-sdk/install.sh
```

Initializing the SDK

```
gcloud init
```

In your browser, log in to your Google user account when prompted and click `Allow to grant permission to access Google Cloud Platform resources.`

Enabling Kubernetes Engine API

Authenticate Your Google Cloud

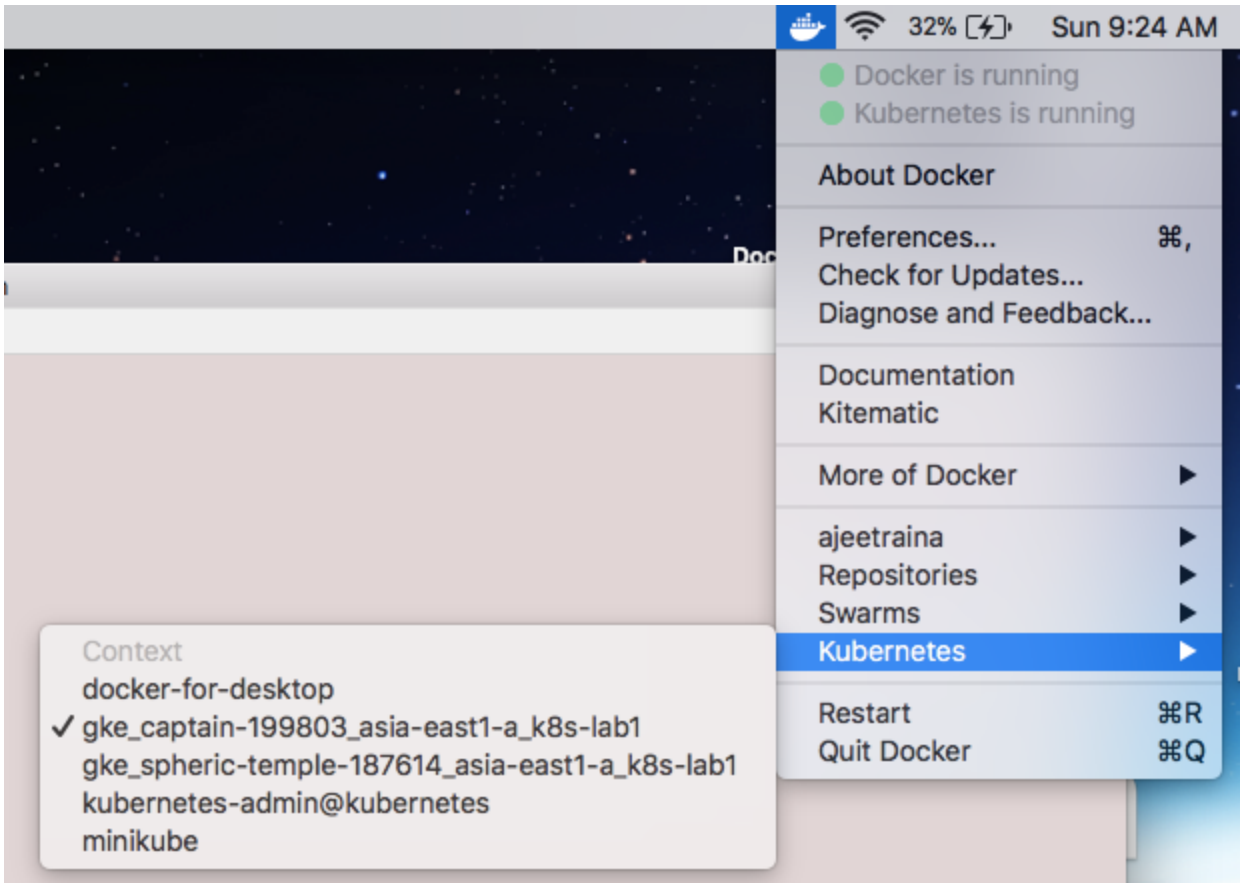
```
gcloud auth login
```

Creating GKE Cluster

```
gcloud container clusters create k8s-lab1 --disk-size 10 --zone asia-east1-a
```

Viewing it on Docker for Mac UI

You should be able to view GKE cluster under Preference UI by now.



Be aware that your Kubernetes context can be named differently and it depends on the project's name under which the Kubernetes cluster is being deployed.

Listing the Nodes

```
kubectl get nodes
```

You can connect to your cluster via command-line or using a dashboard. **Remember** your project's name can be different.

```
gcloud container clusters get-credentials k8s-lab1 --zone asia-east1-a --pro
```

Deploy Nginx on GKE Cluster

```
$ kubectl run nginx --image=nginx --replicas=3


deployment "nginx" created
```

Verify that the pods are running

```
kubectl get pods -owide
NAME                     READY     STATUS    RESTARTS   AGE       IP
nginx-7c87f569d-glczj    1/1       Running   0          8s        10.12.2.6
nginx-7c87f569d-pll76    1/1       Running   0          8s        10.12.0.8
nginx-7c87f569d-sf8z9    1/1       Running   0          8s        10.12.1.8
```

You can see that each `nginx` pod is now running in a different node (virtual machine).

Expose the nginx cluster as an external service

```
$ kubectl expose deployment nginx --port=80 --target-port=80 \
--type=LoadBalancer

service "nginx" exposed
```

Find the network load balancer address:

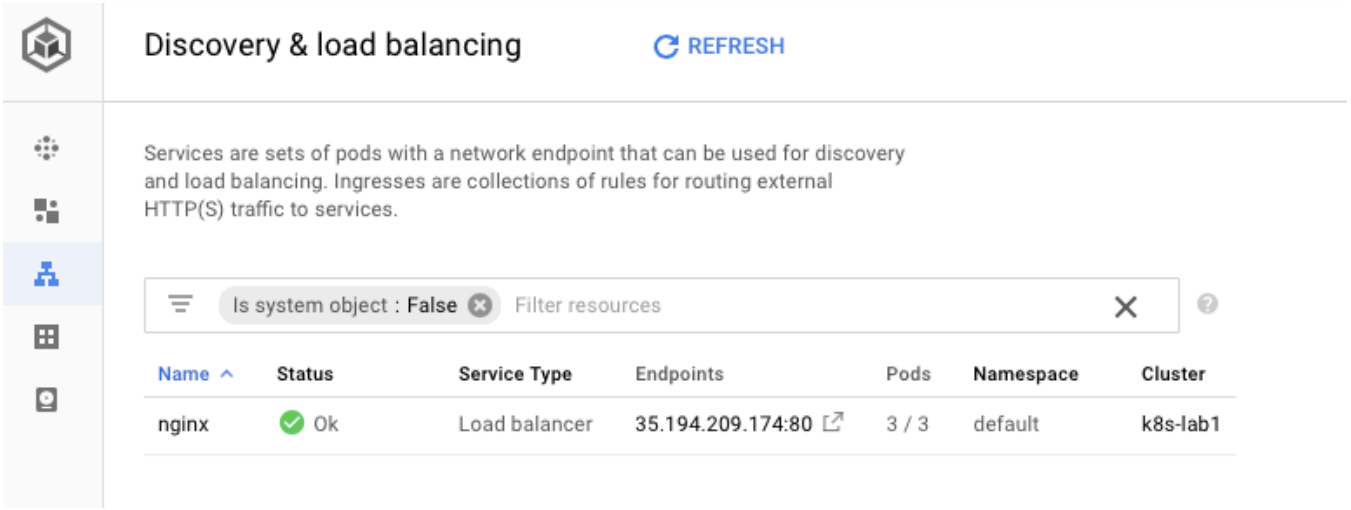
```
kubectl get service nginx
NAME      TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
nginx     LoadBalancer   10.15.247.8   <pending>     80:30253/TCP   12s
```

It may take several minutes to see the value of `EXTERNAL_IP` . If you don't see it the first time with the above command, retry every minute or so until the value of `EXTERNAL_IP` is displayed.

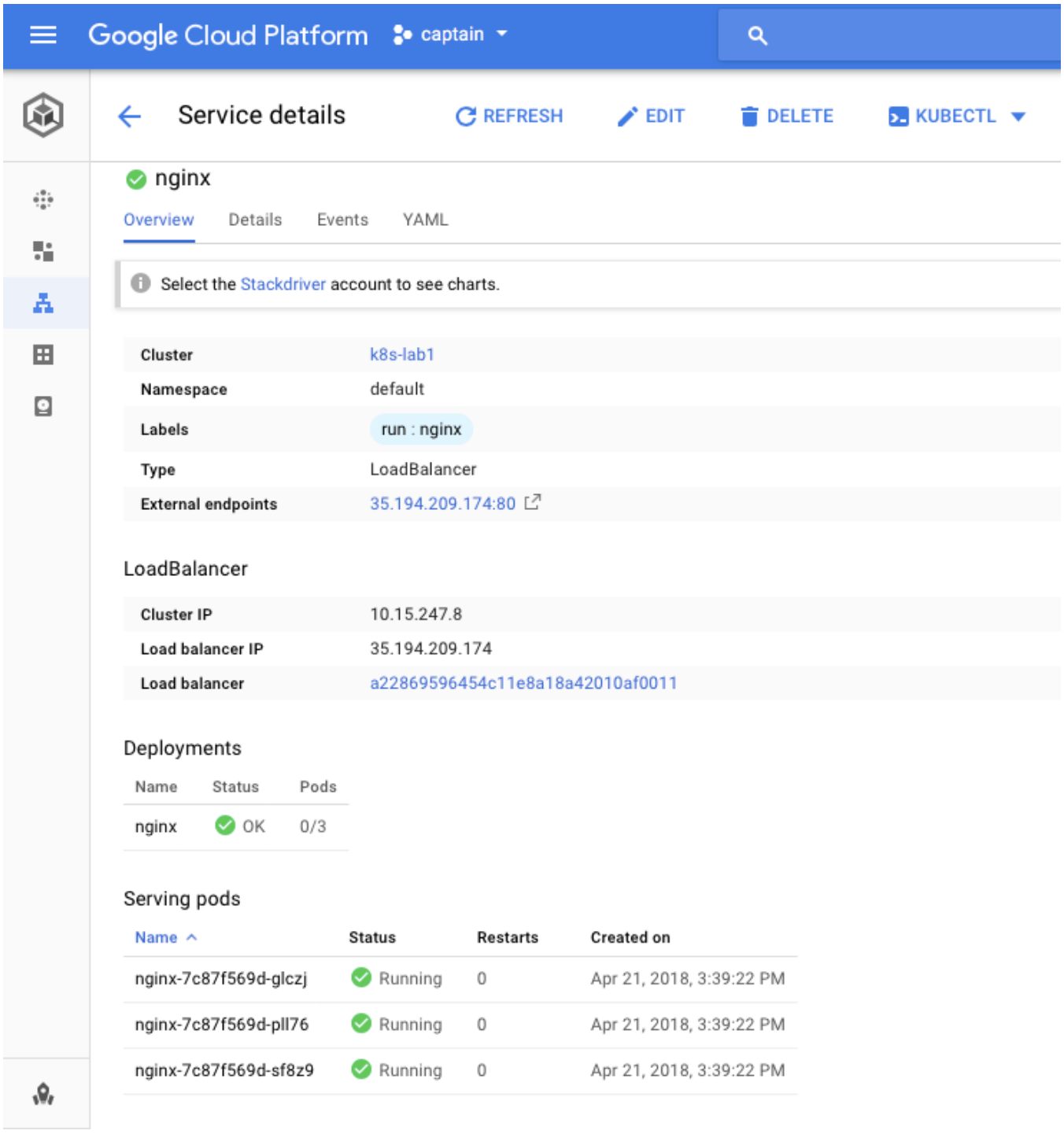
You can then visit `http://EXTERNAL_IP/` to see the server being served through network

load balancing.

GKE provides amazing platform to view `Workloads & Load-balancer` as shown below:



GKE also provides UI for displaying `Loadbalancer` :



Cleaning Up

```
gcloud container clusters delete k8s-lab1 --zone asia-east1-a
```

Next »

Join KubeDaily

10 Members Online

Support

MEMBERS ONLINE

- h3ll_boy
- Heisenberg
- Klu
- MEE6
- Nitinkashyap — Apex Legends
- ojaswa
- Parmeshwar
- prasad
- trimankaur
- vikas027

Free voice chat from Discord

Connect

Tweets by collabnix

kubelabs is maintained by collabnix.

This page was generated by GitHub Pages.