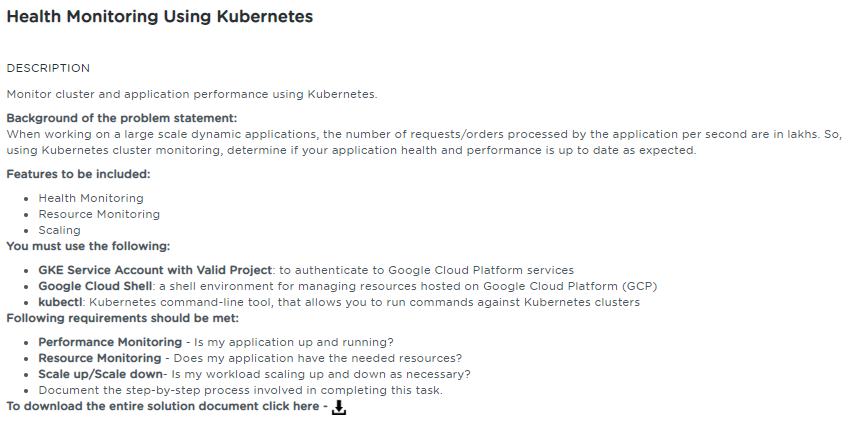
**Health Monitoring Using Kubernetes - Assessment**



**Table of Contents**

[**1.** **Project Abstract** 1](#_Toc44454771)

[**2.** **Work Environment** 2](#_Toc44454772)

[**2.1.** **Prerequisites** 2](#_Toc44454773)

[**2.2.** **Setting up Kubernetes Cluster** 2](#_Toc44454774)

[**3.** **Health/Resource Monitoring** 5](#_Toc44454775)

[**3.1.** **Kubernetes Metrics Server** 5](#_Toc44454776)

[**3.1.1.** **Deploy Metrics-Server** 5](#_Toc44454777)

[**3.1.2.** **Metrics-Server Service Details** 6](#_Toc44454778)

[**3.1.3.** **View the Metrics of Pods/Nodes** 7](#_Toc44454779)

[**4.** **Scale up/Scale down application cluster size/number of nodes** 8](#_Toc44454780)

[**4.1.** **Create Horizontal Pod Auto scale (HPA) definition file** 8](#_Toc44454781)

[**4.2.** **Auto scale (Scale up & Scale down event)** 9](#_Toc44454782)

[**5.** **References** 10](#_Toc44454783)

# **Project Abstract**

The intent of this project is to perform the following activities:

* Monitor the performance of your application and identify if it is up and running as expected.
* Monitor resources in terms of CPU/Memory/disk so as to ensure that your application has enough resources required for its proper functioning.
* Scale up/scale down the application cluster size/no. of nodes.

# **Work Environment**

## **Prerequisites**

Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster.

## **Setting up Kubernetes Cluster**

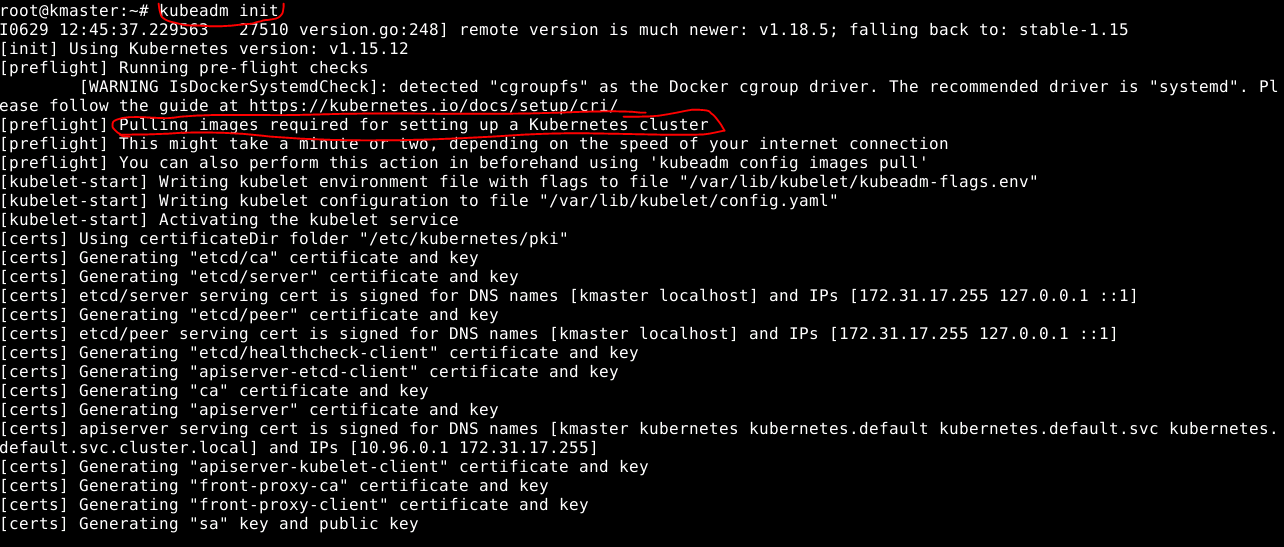
Using the SimpliLearn Practice Lab session, Kubernetes Cluster is being setup with one master and two worker nodes

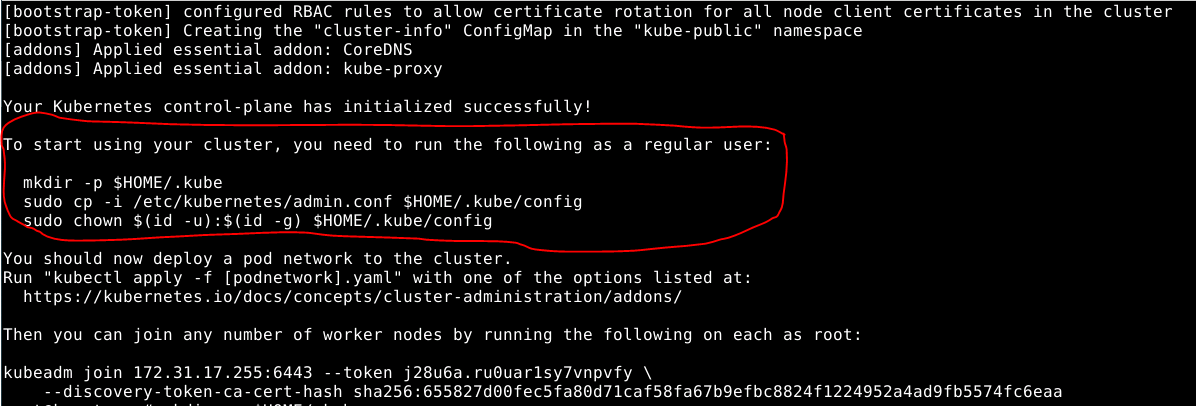
Run the below command on the master

|  |
| --- |
| $ sudo –i  $ kubeadm reset |

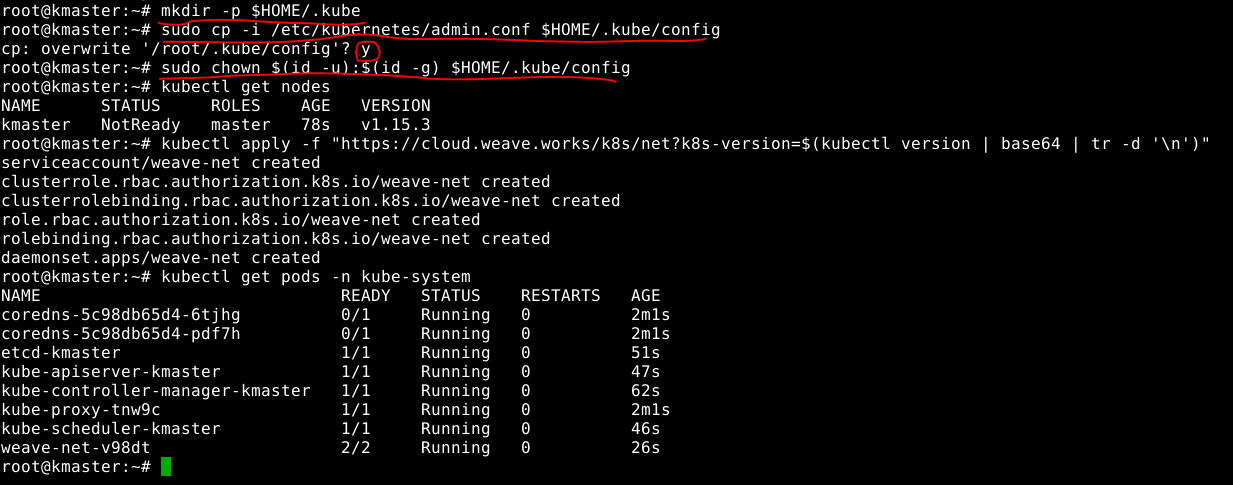
**kubeadm reset** is responsible for cleaning up a node local file system from files that were created using the kubeadm init or kubeadm join commands

Run the below command in order to set up the Kubernetes control plane

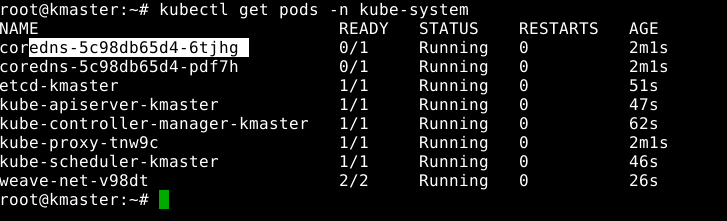




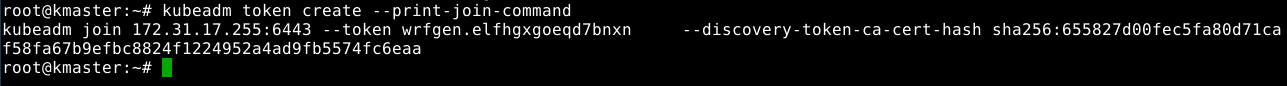
After executing the init command, below screenshot shows various commands used to initialize the configuration and set the correct permissions



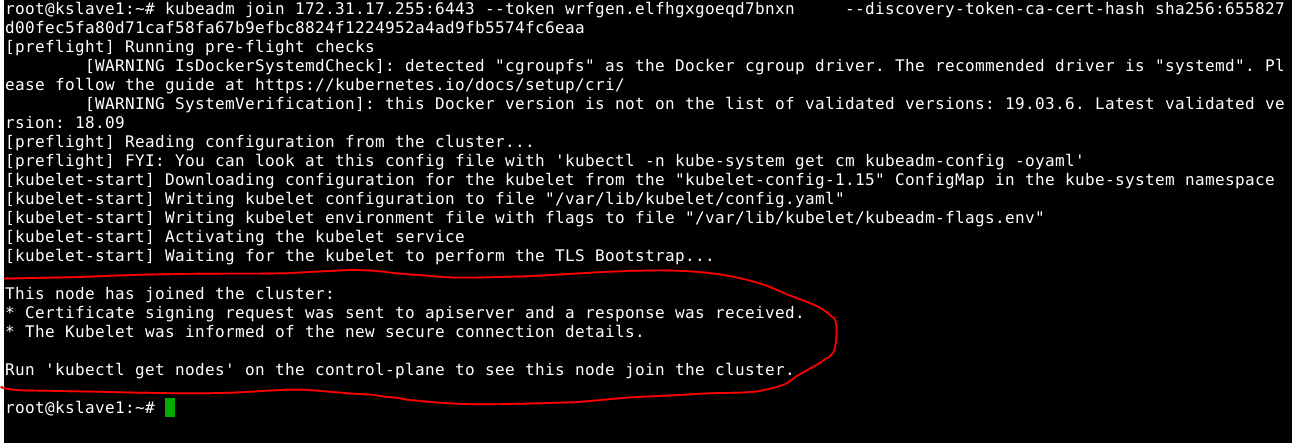
Below screenshot shows the command to verify the master has all the required components running successfully



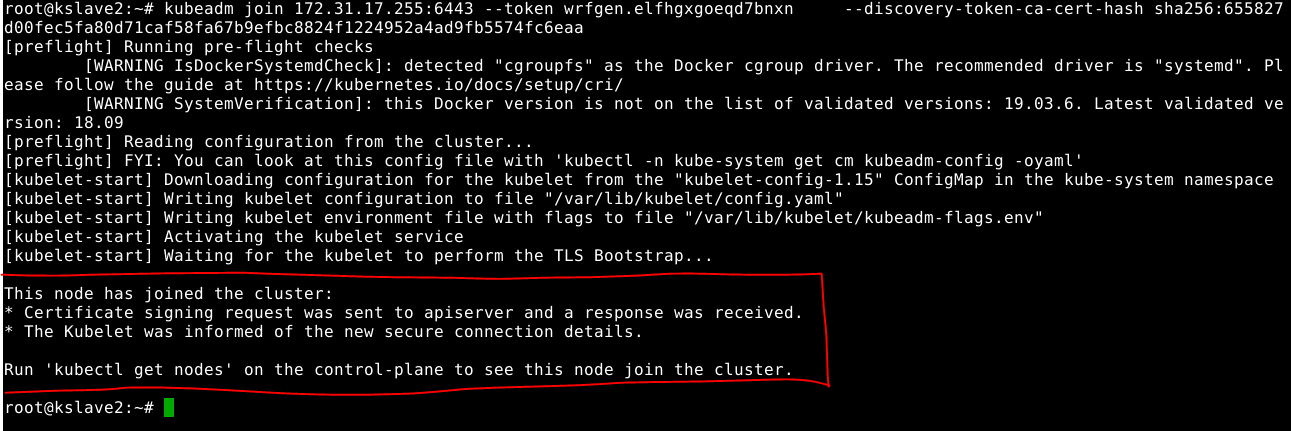
Below screenshot shows the command usage on generating the token for joining the slave nodes in the cluster



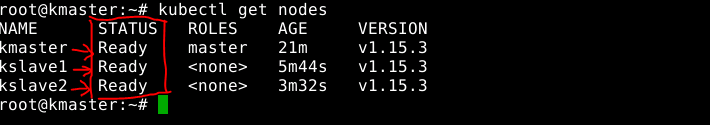
Below screenshot shows the command usage on to join the slave node (kslave1) to the cluster



Below screenshot shows the command usage on to join the slave node (kslave2) to the cluster



Below screenshot shows the all the nodes status in the cluster



# **Health/Resource Monitoring**

## **Kubernetes Metrics Server**

Metrics Server is a scalable, efficient source of container resource metrics for Kubernetes built-in auto scaling pipelines.

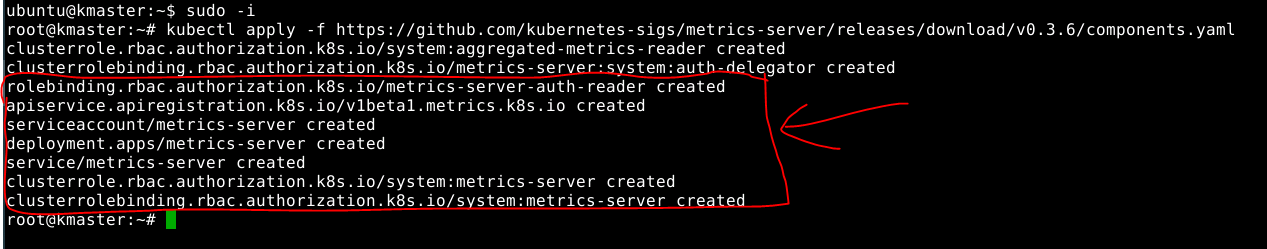
Metrics Server collects resource metrics from Kubelets and exposes them in Kubernetes apiserver through [Metrics API](https://github.com/kubernetes/metrics) for use by [Horizontal Pod Autoscaler](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/) and [Vertical Pod Autoscaler](https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler). Metrics API can also be accessed by **kubectl top**, making it easier to debug autoscaling pipelines.

You can use Metrics Server for:

* CPU/Memory based horizontal autoscaling (learn more about [Horizontal Pod Autoscaler](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/))
* Automatically adjusting/suggesting resources needed by containers (learn more about [Vertical Pod Autoscaler](https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler))

## **Deploy Metrics-Server**

The below screenshot shows on the deployment of kubernetes metrics server



Deploy the Metrics server patch for the correct execution

|  |
| --- |
| $ kubectl patch deploy metrics-server -p "$(cat k8s-metrics-server.patch.yaml)" -n kube-system |

**Note**:

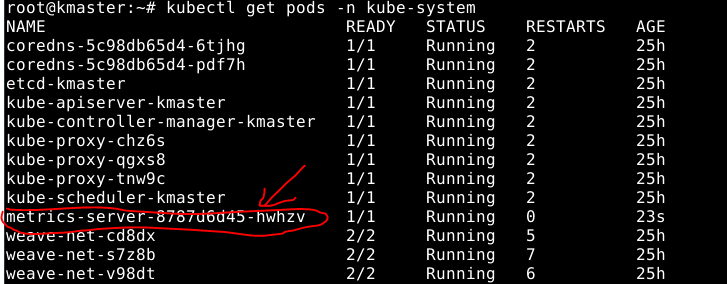
Get the **k8s-metrics-server.patch.yaml** file from the link given below

wget-c https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml

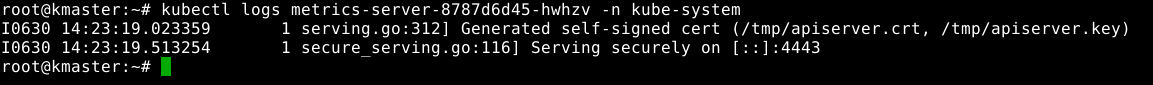
Run the below command to view the status of the Metrics server

|  |
| --- |
| $ kubectl get pods -n kube-system |

The below screenshot shown is the result of executing the above command

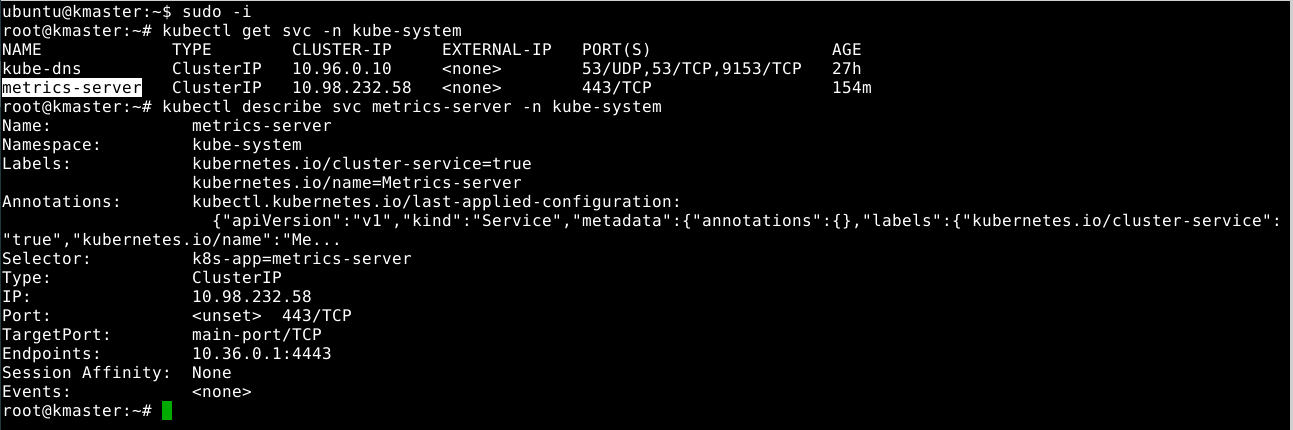


The below screenshot shown is the logs of the metrics-server



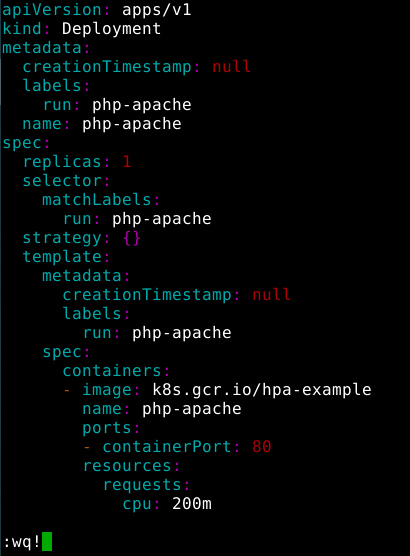
## **Metrics-Server Service Details**

Below screenshot shows the details of the metrics server

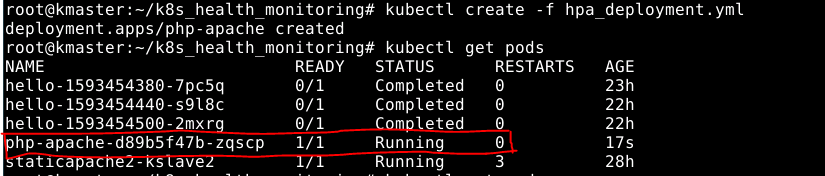


## **View the Metrics of Pods/Nodes**

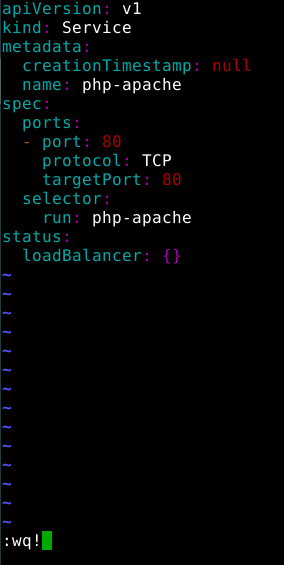
Below screenshot shows the Deployment yml file (image - php-apache)



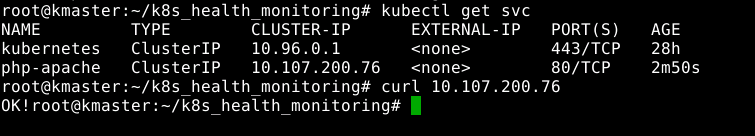
Below screenshot shows the deployment object creation and it’s running status



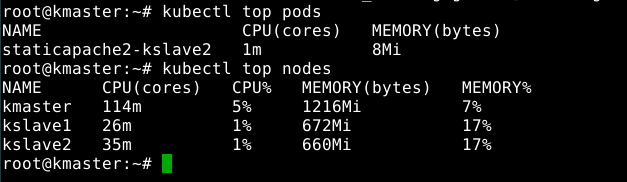
Below screen shown is the content of the Service definition file to enable the communication for the Deployment object(php-apache)



Below screenshot shows the pod communication through the Service using the curl command



Below screenshot shows the command which is useful for fetching the metrics of all the running pods and the available nodes

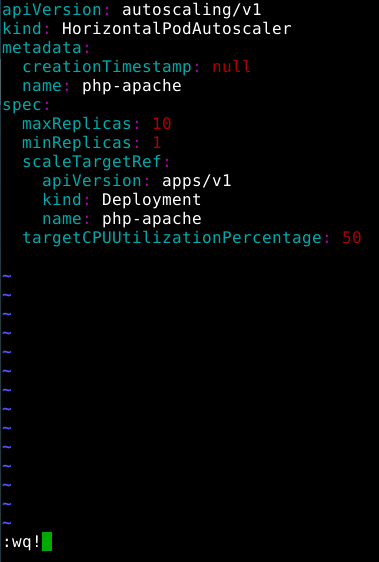


# **Scale up/Scale down application cluster size/number of nodes**

This will allow you to auto scale your application at runtime based upon the number of nodes a cluster requires to execute a particular number of requests/orders to handle large scale data applications.

## **Create Horizontal Pod Auto scale (HPA) definition file**

Below screenshot shows the Horizontal Pod Auto scale definition file which has the minimum and maximum replicas definition based on the targetCPUUtilizationPercentage of the php-apache Deployment object

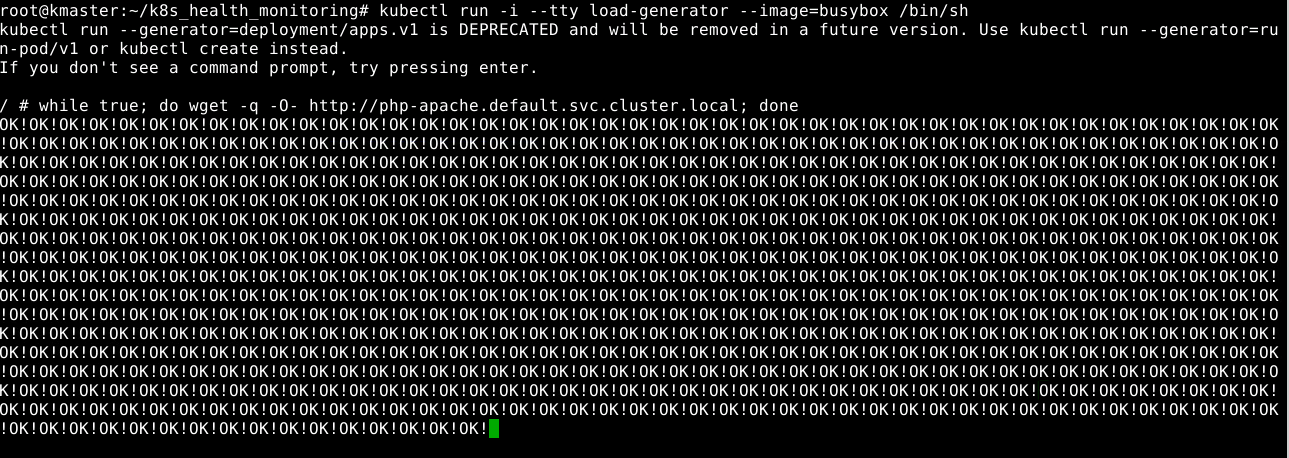


Below screenshot shows the creation of HorizontalPodAutoScale object

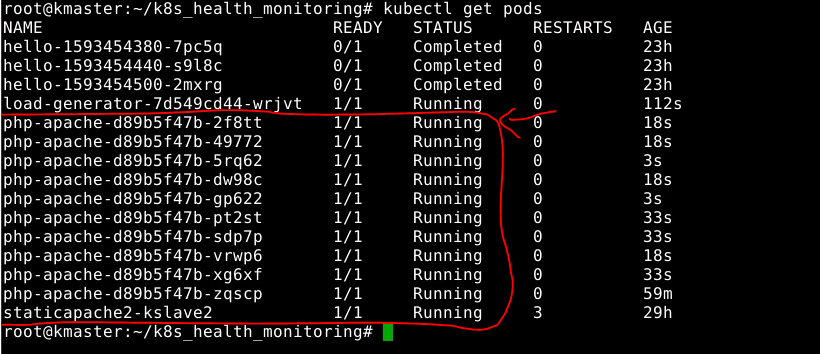


## **Auto scale (Scale up & Scale down event)**

Below screenshot shows the deployment command generates the message to simulate the message generation to increase the CPU load so that we can prove the Scale up capability

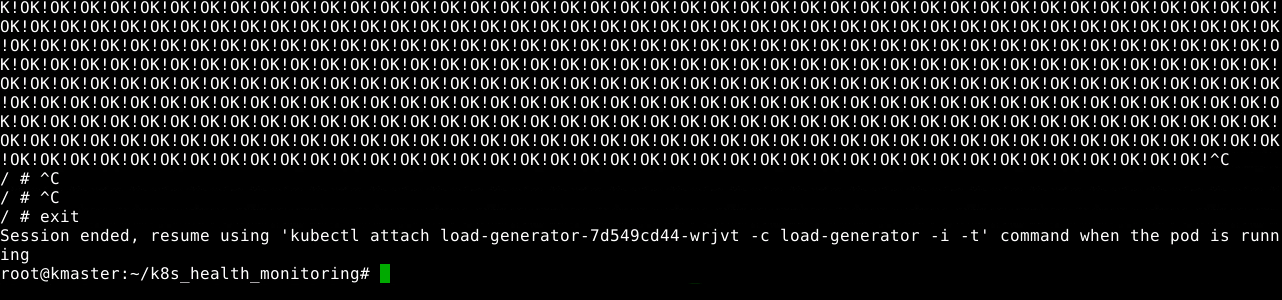


Below screenshot shows the Pod gets auto scaled according to the definition of the HorizontalPodAutoScale object i.e when the targetCPUUtilizationPercentage reaches 50%, the pod gets auto scaled.



Once it reaches the maximum replicas (defined as 10), we can stop generating the message just to prove the scale down capability i.e when the load decrease, the pod gets automatically scaled down to 1.

Below screenshot shows the message generation get stopped. This stops generating the message which in turn decreases the CPU load of the node



Below screenshot shows that only one pod (php-apache) is running as the pods get scaled down automatically when the CPU load gets decreased.



# **References**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Components** | **Reference** |
| 1 | Write-up Document | 3.Health\_Monitoring\_using\_k8s\_writeup.docx |
| 2 | Sources | 3.Health\_Monitoring\_using\_k8s\_sources.docx |