**AKS Ingress Controller Auto-Scaling using Custom Http Request Count Metrics with taking advantage of Prometheus Adapter Plugin.**

**What we achieve**: Kubernetes default HPA is only scaling pods based CPU and Memory metrics and most of time scaling pods based on memory it is not working well. Sometimes it is not convenient because CPU and memory based scaling works late and this case we can lost our business requests.In Additional We want to scale our apps based on different metric value and in this case default HPA memory nd cpu metric won't be relevant and also If we emphasize our problem to scaling Nginx Ingress controller using custom http request count in which ingress accept this traffic the specific period of time

About [More Theory](https://learnk8s.io/autoscaling-apps-kubernetes)

**Used Tech Stacks:**

- Azure AKS

- Nginx Ingress

- Prometheus

- Promethues Adapter

- Sample Apache Web App with HPA

- Hey [Http Load Generator](https://github.com/rakyll/hey)

**Tools:**

- kubectl: k8s client cli

- helm: k8s manifest templating tool

- kustomize: k8s manifest templating tool

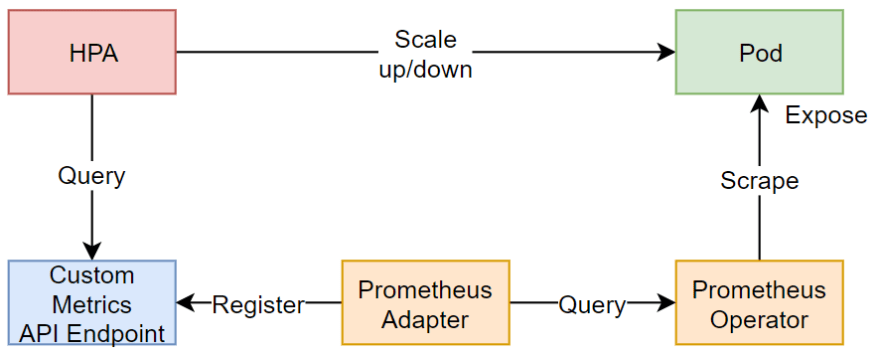
- stern: multipod logging

- watch: actively track command result tool

- curl: terminal based http tool

- jq: json parser cli

**Architecture**



I suppose everything clearly is understandable when we look at architecture. Pod can be anything microservice, nginx api gateway, statefull app ad .etc

**Hands-On Steps.**

**1. Spin UP AKS cluster in Azure**

If you have already cluster skip this step.

[Setup\_link](https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough-portal)

Set enough worker node pool count for testing pod based scaling. Minimum 3 three nodes with enough resources.

Check AKS cluster status

$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

aks-agentpool-21280117-vmss00000b Ready agent 160m v1.21.9

aks-agentpool-21280117-vmss00000c Ready agent 160m v1.21.9

aks-agentpool-21280117-vmss00000d Ready agent 160m v1.21.9

**2. Apply Nginx Ingress Controller with exposed metrics version.**

Create Namespace in AKS

$ kubectl create ns nginx-ingress

Add helm chart repo

$ helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

Install Helm

$ helm install nginx-ingress ingress-nginx/ingress-nginx \

--version 4.0.13 \

--namespace ingress-nginx --create-namespace \

--set controller.replicaCount=1 \

--set controller.metrics.enabled=true \

--set-string controller.podAnnotations."prometheus\.io/scrape"="true" \

--set-string controller.podAnnotations."prometheus\.io/port"="10254"

Check deployed values.

$ helm get values ingress-controller --namespace ingress-nginx

If you have already installed ingress-controller with helm then [you can upgrade using helm](https://kubernetes.github.io/ingress-nginx/deploy/) but must use above metric parameters.

Check pod status

$ kubectl get pods -n ingress-nginx

NAME READY STATUS RESTARTS AGE

nginx-ingress-ingress-nginx-controller-d86885b4b-6l9xp 1/1 Running 0 3h5m

Check metrics of ingress pod

$ kubectl get svc -n ingress-nginx

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

nginx-ingress-ingress-nginx-controller LoadBalancer 10.0.77.19 20.85.163.231 80:31882/TCP,443:32733/TCP 19h

nginx-ingress-ingress-nginx-controller-admission ClusterIP 10.0.243.207 <none> 443/TCP 19h

nginx-ingress-ingress-nginx-controller-metrics ClusterIP 10.0.89.112 <none> 10254/TCP 19h

prometheus-server NodePort 10.0.181.120 <none> 9090:32724/TCP 19h

It is just for getting metrics of ingress as following image

$ kubectl -n ingress-nginx port-forward svc/nginx-ingress-ingress-nginx-controller-metrics 10254:10254

Forwarding from 127.0.0.1:10254 -> 10254

Forwarding from [::1]:10254 -> 10254

Text

Description automatically generated

**3. Setup Prometheus**

For simplicity I installed as below. You can also use [promethues operator](https://github.com/prometheus-community/helm-charts/tree/main/charts/kube-prometheus-stack), for our case it doesn’t matter. For production We must enable volume mount in Prometheus for accessing old metrics

$ kubectl -n nginx-ingress apply --kustomize github.com/kubernetes/ingress-nginx/deploy/prometheus/

$ kubectl get pods -n ingress-nginx

NAME READY STATUS RESTARTS AGE

nginx-ingress-ingress-nginx-controller-d86885b4b-6l9xp 1/1 Running 0 3h13m

prometheus-server-6444d9478c-5nn9b 1/1 Running 0 3h13m

$ kubectl get svc -n ingress-nginx

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

nginx-ingress-ingress-nginx-controller LoadBalancer 10.0.77.19 20.85.163.231 80:31882/TCP,443:32733/TCP 19h

nginx-ingress-ingress-nginx-controller-admission ClusterIP 10.0.243.207 <none> 443/TCP 19h

nginx-ingress-ingress-nginx-controller-metrics ClusterIP 10.0.89.112 <none> 10254/TCP 19h

prometheus-server NodePort 10.0.181.120 <none> 9090:32724/TCP 19h

It is just for getting metrics of ingress from the Prometheus as following image

$ kubectl -n ingress-nginx port-forward svc/prometheus-server 9090:9090

Forwarding from 127.0.0.1:9090 -> 9090

Forwarding from [::1]:9090 -> 9090

Graphical user interface, text, application

Description automatically generated

**4. Install Prometheus Adapter**

$ kubectl create ns monitoring

$ cd ./scale-ingress-custom-metrics/prometheus-adapter

Firstly change below configurations ./2-deployment.yaml

- --prometheus-url=http://prometheus-server.ingress-nginx.svc:9090 # Prometheus endpoint in which get metrics

- --metrics-relist-interval=10s # Get Metrics interval from promethues

- --v=6 # Activate Debug Logs in Prometheus Adapter

$ kubectl apply -f 0-adapter/

$ kubectl apply -f 1-custom-metrics/

$ kubectl apply -f 2-resource-metrics/

$ kubectl get pods -n monitoring

$ kubectl get pods,svc,cm -n monitoring

NAME READY STATUS RESTARTS AGE

pod/custom-metrics-prometheus-adapter-5d7b66f7d-6n8hc 1/1 Running 0 3h34m

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/custom-metrics-prometheus-adapter ClusterIP 10.0.136.43 <none> 443/TCP 19h

NAME DATA AGE

configmap/custom-metrics-prometheus-adapter 1 19h

Let’s check Prometheus adapter logs

$ kubectl -n monitoring logs -f custom-metrics-prometheus-adapter-5d7b66f7d-6n8hc

**I0421 06:40:37.560790 1 adapter.go:101] successfully using in-cluster auth**

I0421 06:40:37.595081 1 round\_trippers.go:445] GET https://10.0.0.1:443/api?timeout=32s 200 OK in 33 milliseconds

I0421 06:40:37.662326 1 round\_trippers.go:445] GET https://10.0.0.1:443/apis/apiregistration.k8s.io/v1?timeout=32s 200 OK in 4 milliseconds

I0421 06:40:37.662378 1 round\_trippers.go:445] GET https://10.0.0.1:443/apis/events.k8s.io/v1beta1?timeout=32s 200 OK in 4 milliseconds

I0421 06:40:37.662404 1 round\_trippers.go:445] GET https://10.0.0.1:443/apis/flowcontrol.apiserver.k8s.io/v1beta1?timeout=32s 200 OK in 2

I0421 06:40:37.957020 1 api.go:74] GET **http://prometheus-server.ingress-nginx.svc:9090/api/v1/series?match%5B%5D=nginx\_ingress\_controller\_nginx\_process\_requests\_total%7Bcontroller\_namespace%21%3D%22%22%2Ccontroller\_pod%21%3D%22%22%7D&start=1650523177.862 200 OK**

Let’s check our custom rule is applied, exec adapter pod and check it. **It is most important part in which convert Prometheus ingress request count metric to k8s custom api variant and then k8 hpa will use it following step**

$ kubectl -n monitoring exec -it custom-metrics-prometheus-adapter-5d7b66f7d-6n8hc sh

/ $ cat /etc/adapter/config.yaml

rules:

- seriesQuery: 'nginx\_ingress\_controller\_nginx\_process\_requests\_total{controller\_namespace!="",controller\_pod!=""}'

resources:

overrides:

controller\_namespace: {resource: "namespace"}

controller\_pod: {resource: "pod"}

name:

matches: ""

as: "nginx\_ingress\_controller\_nginx\_process\_requests\_total

metricsQuery: round(sum(rate(<<.Series>>{<<.LabelMatchers>>}[1m])) by (<<.GroupBy>>), 0.001)

You should be more careful in the **seriesQuery and overrides** section in which **controller\_namespace** an **controller\_pod variables** must be same variable result of nginx\_ingress\_controller\_nginx\_process\_requests\_total metric in Prometheus

We will see k8s custom api metric endpoint successfully created

$ kubectl get APIService -A | grep custom.metrics.k8s.io

v1beta1.custom.metrics.k8s.io monitoring/custom-metrics-prometheus-adapter True 19h

**Check metric – Important Part**

$ kubectl get --raw "/apis/custom.metrics.k8s.io/v1beta1/namespaces/ingress-nginx/pods/\*/nginx\_ingress\_controller\_nginx\_process\_requests\_total" | jq .

{

"kind": "MetricValueList",

"apiVersion": "custom.metrics.k8s.io/v1beta1",

"metadata": {

"selfLink": "/apis/custom.metrics.k8s.io/v1beta1/namespaces/ingress-nginx/pods/%2A/nginx\_ingress\_controller\_nginx\_process\_requests\_total"

},

"items": [

{

"describedObject": {

"kind": "Pod",

"namespace": "ingress-nginx",

"name": "nginx-ingress-ingress-nginx-controller-d86885b4b-8cxpm",

"apiVersion": "/v1"

},

"metricName": "nginx\_ingress\_controller\_nginx\_process\_requests\_total",

"timestamp": "2022-04-20T21:47:19Z",

"value": "1300m",

"selector": null}]}

**Files attached …**

**5. Sample Apache Web App with HPA**

$ cd ./scale-ingress-custom-metrics/demo-app

$ kustomize build . | kubectl apply -f –

Check status and open it in browser

$ kubectl get pod,svc,ing -n autoscale-all-the-things

NAME READY STATUS RESTARTS AGE

pod/apache2-7548fdc69f-5856n 1/1 Running 0 3h59m

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

service/apache2 ClusterIP 10.0.187.197 <none> 80/TCP 20h

NAME CLASS HOSTS ADDRESS PORTS AGE

ingress.networking.k8s.io/apache2 <none> **apache2.example.com** 20.85.163.231 80, 443 20h

Add below record your OS local dns resolver

20.85.163.231 **apache2.example.com**

Graphical user interface, application, Word

Description automatically generated

We will see that we can get our app over ingress controller.

**Files attached …**

**6. Hey** [**Http Load Generator**](https://github.com/rakyll/hey)

Let’s open four separate linux terminal tap and type below commands

Interactively watch hpa target increased

$ watch -n 1 kubectl get hpa -n ingress-nginx

Watch interactively pod scaling count

$ watch -n 1 kubectl get pods -n ingress-nginx

Simple look scaling ingress pod logs at the same time

$ stern -n ingress-nginx nginx-ingress-ingress-nginx-controller

Simple load our app over nginx

$ hey -n 10000 -z 120s -c 10 https://apache2.example.com/

**Good Luck!**

**Resources**:

https://stacksimplify.com/azure-aks/azure-kubernetes-service-cluster-autoscaler/

https://www.thorsten-hans.com/aks-cluster-auto-scaler-inside-out/

https://stacksimplify.com/azure-aks/azure-kubernetes-service-pod-autoscaler/ (commands to use)

The Horizontal Pod Autoscaler checks by default the metrics every 15 seconds. You can configure the interval with the -horizontal-pod-autoscaler-sync-period flag. Take Consider App Start

https://www.programmingwithwolfgang.com/auto-scale-kubernetes-hpa

Another Solution examples Keda

https://blog.bart.je/adding-keda-aks-cluster/

https://joachim8675309.medium.com/aks-with-aad-pod-identity-7c2cbf906eb9

https://jonathan18186.medium.com/azure-kubernetes-service-aks-with-kubernetes-event-driven-autoscaling-keda-aa2f886e7cd0

https://www.youtube.com/watch?v=Bq2CpEcRtPw&list=PLOPXEdTIaFO8D73ufHmoGCauRRhCy5uBh

https://www.youtube.com/watch?v=TftaxqNFsZY

https://promitor.io/

https://keda.sh/docs/2.6/operate/metrics-server/

https://docs.microsoft.com/en-us/azure/aks/monitor-aks

https://docs.microsoft.com/en-us/samples/azure-samples/virtual-node-autoscale/virtual-node-autoscale-demo/

https://itnext.io/using-prometheus-in-azure-kubernetes-service-aks-ae22cada8dd9

https://dev.to/anirudhgarg\_99/scale-up-and-down-a-http-triggered-function-app-in-kubernetes-using-keda-4m42

https://blog.sighup.io/scale-workloads-with-ingress-traffic/

https://github.com/sighupio/blog-posts-example/tree/main/scale-workloads-with-ingress-traffic

https://docs.microsoft.com/en-us/answers/questions/438444/pod-scaling-based-on-the-http-requests.html

Prometheus and Grafana installation

https://kubernetes.github.io/ingress-nginx/user-guide/monitoring/

Prometheus Adapter

https://github.com/kubernetes-sigs/prometheus-adapter

https://blog.sighup.io/scale-workloads-with-ingress-traffic/

https://icicimov.github.io/blog/kubernetes/Kubernetes\_HPA\_Autoscaling\_with\_Custom\_Metrics/