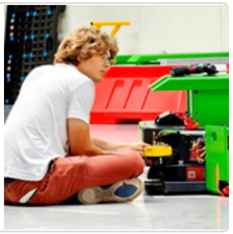
Kubernetes Autoscaling: KEDA with prometheus on EKS with traefik provisioned by Terraform

Check GitHub for helpful DevOps tools:

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics



Ask Personal AI Document assistant to learn interactively (FASTER)!

1 Download PDF

2 Go to website

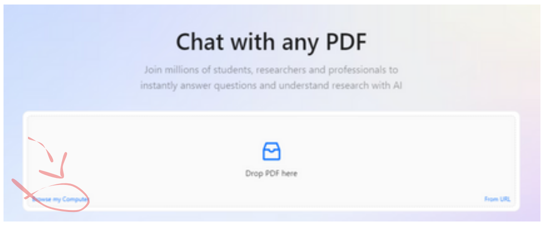
3 Browse file

4 Chat with Document

1 https://github.com/MichaelRobotics/DevOpsTools/blob/main/KubernetesAutoscaleP3.pdf

Click there to go to ChatPdf website 2

3


Chat with any PDF
Join millions of students, researchers and professionals to instantly answer questions and understand research with AI
Drop PDF here
Browse my Computer
From URL

Ask questions about document! 4

Complety new to Linux and Networking?

Essential for this PDF is a thorough knowledge of networking. I highly recommend the HTB platform's networking module, which offers extensive information to help build a comprehensive understanding.

HTB - Your Cyber Performance Center

We provide a human-first platform creating and maintaining high performing cybersecurity individuals and organizations.

https://www.hackthebox.com/

What is Kubernetes?

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications. It helps manage clusters of nodes running containers, ensuring efficient and reliable operation.

How Kubernetes clusters are made?

Kubernetes clusters consist of a control plane and multiple worker nodes. The control plane manages cluster operations, while worker nodes run the actual container workloads.

Why and When use Kubernetes

Kubernetes is ideal for deploying scalable, resilient, and automated containerized applications. It is used when managing multiple containers across different environments is necessary.

Example: Running a microservices-based e-commerce platform that scales up during peak hours.

System Requirements

- RAM: 2 GB per node (1 GB can work for testing but may lead to limited performance)
- 10 GB free storage
- Ubuntu

Kubernetes: Main components & packages

- **kube-apiserver:** Central management component that exposes the Kubernetes API; acts as the front-end for the cluster.
- **etcd:** Distributed key-value store for storing all cluster data, ensuring data consistency across nodes.
- **kube-scheduler:** Assigns pods to available nodes based on resource requirements and policies.
- **kube-controller-manager:** Manages core controllers that handle various functions like node status, replication, and endpoints.
- **kubelet:** Agent that runs on each node, responsible for managing pods and their containers.
- **kube-proxy:** Manages networking on each node, ensuring communication between pods and services within the cluster.

Kubernetes Autoscaling: KEDA

1) What is KEDA?

KEDA (Kubernetes-based Event-Driven Autoscaling) is a Kubernetes operator that enables event-driven, dynamic scaling of workloads based on external event sources. It extends Kubernetes' native Horizontal Pod Autoscaler (HPA) by introducing event triggers that allow scaling decisions to be made on metrics beyond CPU and memory, such as message queue length or database activity.

2) Key Benefits of KEDA

- **Custom Metrics Scaling**: Enables scaling based on custom metrics like Kafka consumer lag, ensuring efficient workload handling.
- **Wide Support**: Works seamlessly with diverse triggers like queue size and lag from tools like Kafka, ActiveMQ, and more.

3) How Does KEDA Work?

KEDA integrates with Kubernetes' HPA API to support event-driven scaling by using scalers. Scalers define triggers for scaling based on external metrics such as Kafka consumer lag or queue size. When triggered, KEDA dynamically scales workloads like Deployments, StatefulSets, and Jobs. In environments like GKE, EKS.

Kubernetes Autoscaling: KEDA with prometheus & k6s & robusta on EKS deployed with Terraform

1) Create EKS cluster

Follow instructions from my PDF about EKS deployment through IaC tools: choose eksctl, Terraform or CloudFormation

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

or use Hashicorp Template:

Provision an EKS cluster (AWS)

AWS's Elastic Kubernetes Service (EKS) is a managed service that lets you deploy, manage, and scale containerized applications on Kubernetes.

https://developer.hashicorp.com/terraform

Or create cluster manualy:

Create an Amazon EKS cluster

This topic provides an overview of the available options and describes what to consider when you create an Amazon EKS

https://docs.aws.amazon.com/eks/latest/userguide/create-cluster.html

2) Add traefik

```
helm repo add traefik \
https://helm.traefik.io/traefik

helm repo update

helm upgrade --install traefik traefik/traefik \
--namespace traefik --create-namespace --wait
```

and get ip to reach your app:

```
export INGRESS_HOSTNAME=$(kubectl --namespace traefik \
 get svc traefik \
 --output jsonpath="{.status.loadBalancer.ingress[0].hostname}")

export INGRESS_HOST=$(dig +short $INGRESS_HOSTNAME)
```

Wait a minute then:

```
echo $INGRESS_HOST
```

Pass one of ip’s to variable:

```
export INGRESS_HOST=<IP>
```

3) Deploy app

Clone directory and deploy app

```
git clone https://github.com/MichaelRobotics/Kubernetes.git
cd Kubernetes/KEDA

kubectl create namespace production
kubectl --namespace production \
apply --filename k8s/
```

4) Install Prometheus and Keda on cluster

```
helm repo add kedacore \
https://kedacore.github.io/charts

helm repo add prometheus-community \
https://prometheus-community.github.io/helm-charts

helm repo update

helm install keda kedacore/keda \
--namespace keda \
--create-namespace \
--wait

helm upgrade --install \
prometheus prometheus-community/prometheus \
--namespace monitoring \
--create-namespace \
--wait
```

5) Install Robusta

```
helm repo add robusta \
https://robusta-charts.storage.googleapis.com
```

Execute only if you do not already have Robusta CLI

```
pip install -U robusta-cli --no-cache
```

Follow the instructions from the Wizard dont add prometheus. Create slack account etc.

```
robusta gen-config
```

```
helm upgrade --install robusta robusta/robusta \
--namespace monitoring --create-namespace \
--values generated_values.yaml \
--values robusta-values.yaml \
--set clusterName=dot --wait
```

6) Install k6s

```
sudo apt update
sudo apt install -y k6
```

7) Deploy KEDA and configure k6s

```
helm install keda kedacore/keda \
--namespace keda \
--create-namespace \
--wait
```

```
kubectl --namespace production apply \
--filename keda-prom.yaml
```

Install yq

```
sudo apt update
sudo apt install yq
```

```
yq --inplace \
".spec.rules[0].host = \"dot.$INGRESS_HOST.nip.io\"" \
k8s/ing.yaml
```

```
cat k6.js \
| sed -e "s@http\.get.*@http\.get('http://dot.$INGRESS_HOST.nip.io');@g" \
| tee k6.js
```

```
cat k6-100.js \
| sed -e "s@http\.get.*@http\.get('http://dot.$INGRESS_HOST.nip.io');@g" \
| tee k6-100.js
```

7) Stress test with k6s

```
k6 run k6.js
```

Check if pod scaled:

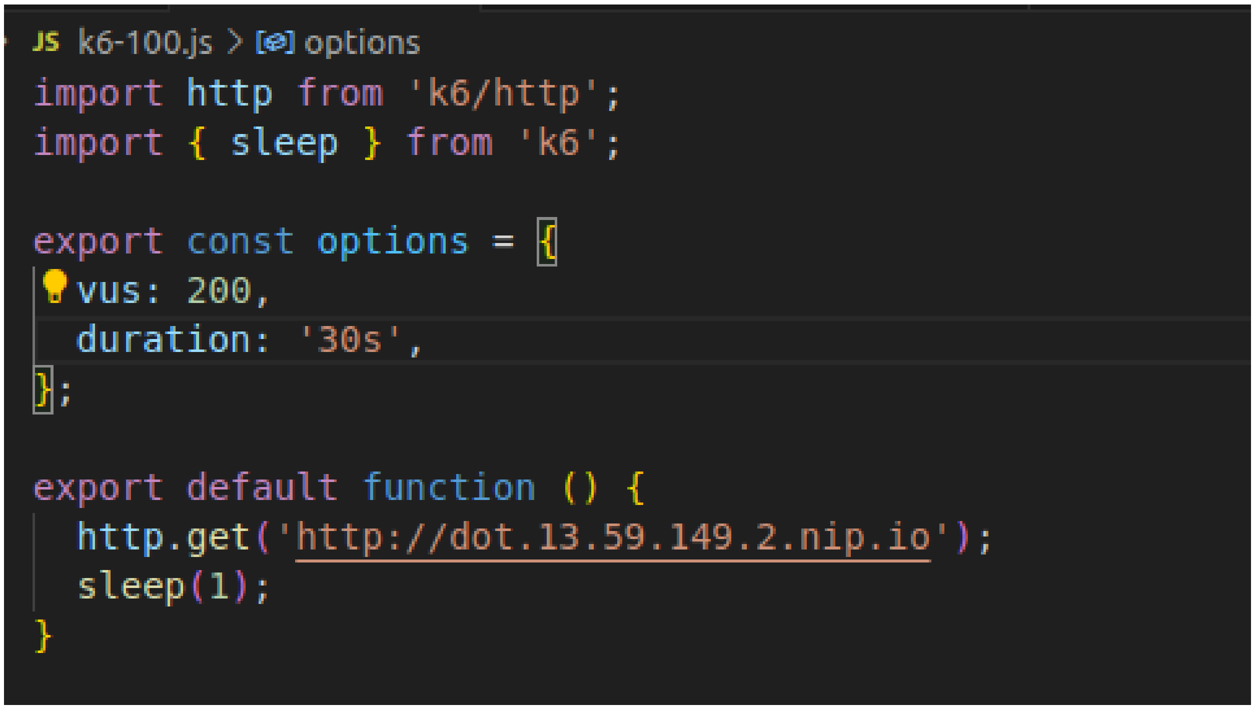
```
kubectl --namespace production \
get pods,hpa,scaledobjects
```

This generate more stress, check if pod scaled:

```
k6 run k6-100.js
```

```
kubectl --namespace production \
get pods,hpa,scaledobjects
```

Modify for acordingly for your needs. vus: 200 means the test will simulate 200 concurrent virtual users performing the specified tasks. Duration: '30s' means the test will run for 30 seconds.



```
import http from 'k6/http';
import { sleep } from 'k6';

export const options = {
  vus: 200,
  duration: '30s',
};

export default function () {
  http.get('http://dot.13.59.149.2.nip.io');
  sleep(1);
}
```

common troubleshooting

1) KEDA Not Scaling Pods

Cause: Incorrect Prometheus configuration or metric queries.
Solution: Verify that the Prometheus server endpoint is reachable from the KEDA deployment. Check the ScaledObject's metric configuration using **kubectl describe scaledobject <scaledobject-name>.** Validate the PromQL query by testing it directly in Prometheus. Ensure the Prometheus server and Traefik configuration expose metrics correctly.

2) Prometheus Metrics Not Available for KEDA

Cause: Misconfigured Traefik or Prometheus service discovery.
Solution: Confirm Traefik is exposing the /metrics endpoint by checking its configuration and logs. Ensure Prometheus has the correct scrape configuration for Traefik using **kubectl get configmap -n <prometheus-namespace>.** Test the scrape targets in the Prometheus UI under Status > Targets to verify connectivity.

3) ScaledObject Not Triggering Scale Events

Cause: Permissions or resource constraints in EKS.
Solution: Ensure the Kubernetes service account used by KEDA has the necessary RBAC permissions. Use **kubectl describe clusterrole <role-name>** and check for required actions like get, list, and watch on custom metrics. Validate the resource limits of the KEDA operator to ensure it has sufficient CPU and memory.

4) Check my Kubernetes Troubleshooting series:

Michael Robotics
Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

Learn more about Kubernetes

Check Kubernetes and piyushsachdeva - great docs!

Setup a Multi Node Kubernetes Cluster

kubeadm is a tool to bootstrap the Kubernetes cluster

https://github.com/piyushsachdeva/CKA-2024/tree/main/Resources/Day27

Kubernetes Documentation

This section lists the different ways to set up and run Kubernetes

https://kubernetes.io/docs/setup/

Share, comment, DM and check GitHub for scripts & playbooks created to automate process.

Check my GitHub

Michael Robotics

Hi, I'm Michal. I'm a Robotics Engineer and DevOps enthusiast. My mission is to create skill-learning platform that combats skill information overload by adhering to the set of principles: simplify, prioritize, and execute.

https://github.com/MichaelRobotics

PS.

If you need a playbook or bash script to manage KVM on a specific Linux distribution, feel free to ask me in the comments or send a direct message!