Event-driven scaling in Kubernetes using KEDA

Diploma Thesis

Faculty of Computer and Information Science

University of Ljubljana

Your Name

Supervisor: Supervisor's Name

June 14, 2025

Contents

Abstract iv

Povzetek v

Acknowledgements vi

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Goals and Objectives . . . 1
- 1.4 Thesis Outline . . . 1

2 Theoretical Background 2
- 2.1 Cloud-Native and Microservices . . . 3
- 2.2 Containers and Orchestration . . . 3
  - 2.2.1 Docker . . . 3
  - 2.2.2 Kubernetes . . . 3
- 2.3 Autoscaling in Kubernetes . . . 3
  - 2.3.1 Horizontal Pod Autoscaler (HPA) . . . 3
  - 2.3.2 Vertical Pod Autoscaler (VPA) . . . 3
  - 2.3.3 Cluster Autoscaler (CA) . . . 3
- 2.4 Event-Driven Architecture (EDA) . . . 3
  - 2.4.1 Principles of EDA . . . 3
  - 2.4.2 Common EDA Patterns . . . 3
  - 2.4.3 Message Brokers (e.g., RabbitMQ, Apache Kafka) . . . 3
- 2.5 KEDA: Kubernetes-based Event-driven Autoscaling . . . 3
  - 2.5.1 KEDA Architecture . . . 3
  - 2.5.2 How KEDA Works with HPA . . . 3
  - 2.5.3 Supported Scalers . . . 3
  - 2.5.4 The `ScaledObject` Custom Resource . . . 3

3 Analysis of the Problem **4**
3.1 Limitations of CPU/Memory-Based Scaling . . . 4
3.2 Need for Event-Driven Scaling . . . 4
3.3 Existing Solutions and Their Trade-offs . . . 4

4 Design and Implementation of the Solution **5**
4.1 System Architecture . . . 5
4.2 Tools and Technologies . . . 5
4.3 Test Application . . . 5
4.3.1 Application Logic . . . 5
4.3.2 Dockerfile . . . 5
4.4 Environment Setup . . . 5
4.4.1 Kubernetes Cluster Setup . . . 5
4.4.2 Deploying the Event Source . . . 5
4.4.3 Deploying the Application . . . 5
4.5 Configuring KEDA for Autoscaling . . . 5
4.5.1 The `ScaledObject` Configuration . . . 5
4.5.2 Verifying the Setup . . . 5

5 Evaluation and Results **6**
5.1 Test Environment . . . 6
5.2 Evaluation Metrics . . . 6
5.3 Test Scenarios . . . 6
5.3.1 Scenario 1: Sudden Burst of Traffic . . . 6
5.3.2 Scenario 2: Gradual Increase in Traffic . . . 6
5.3.3 Scenario 3: Scaling to Zero . . . 6
5.4 Presenting the Results . . . 6
5.5 Analysis of Results . . . 6

6 Conclusion **7**
6.1 Summary of Work . . . 7
6.2 Contribution . . . 7
6.3 Limitations . . . 7
6.4 Future Work . . . 7

A Appendices **9**
A.1 Source Code for the Test Application . . . 9
A.2 Kubernetes Deployment YAML . . . 9

List of Figures

List of Tables

Abstract

This thesis explores event-driven autoscaling in a Kubernetes environment using KEDA (Kubernetes-based Event-driven Autoscaler). ...

Keywords: Kubernetes, KEDA, event-driven architecture, autoscaling, cloud-native.

Povzetek (Slovenian Abstract)

To diplomsko delo raziskuje dogodkovno pogojeno skaliranje v okolju Kubernetes z uporabo KEDA (Kubernetes-based Event-driven Autoscaler). ...

Ključne besede: Kubernetes, KEDA, dogodkovno vodena arhitektura, samodejno skaliranje, v oblaku rojene aplikacije.

Acknowledgements

Chapter 1

Introduction

1.1 Motivation

Kubernetes has become the de facto standard for container orchestration in cloud-native environments [1]. As organizations increasingly adopt microservices architectures, the need for efficient scaling solutions becomes paramount.

1.2 Problem Statement

1.3 Goals and Objectives

1.4 Thesis Outline

In Chapter 2, we ...

Chapter 2

Theoretical Background

2.1 Cloud-Native and Microservices

2.2 Containers and Orchestration

2.2.1 Docker

2.2.2 Kubernetes

Architecture

Key Objects (Pods, Deployments, Services)

2.3 Autoscaling in Kubernetes

2.3.1 Horizontal Pod Autoscaler (HPA)

2.3.2 Vertical Pod Autoscaler (VPA)

2.3.3 Cluster Autoscaler (CA)

2.4 Event-Driven Architecture (EDA)

2.4.1 Principles of EDA

2.4.2 Common EDA Patterns

2.4.3 Message Brokers (e.g., RabbitMQ, Apache Kafka)

2.5 KEDA: Kubernetes-based Event-driven Autoscaling

2.5.1 KEDA Architecture

2.5.2 How KEDA Works with HPA

2.5.3 Supported Scalers

2.5.4 The `ScaledObject` Custom Resource

Chapter 3

Analysis of the Problem

3.1 Limitations of CPU/Memory-Based Scaling

3.2 Need for Event-Driven Scaling

3.3 Existing Solutions and Their Trade-offs

Chapter 4

Design and Implementation of the Solution

4.1 System Architecture

4.2 Tools and Technologies

4.3 Test Application

4.3.1 Application Logic

4.3.2 Dockerfile

4.4 Environment Setup

4.4.1 Kubernetes Cluster Setup

4.4.2 Deploying the Event Source

4.4.3 Deploying the Application

4.5 Configuring KEDA for Autoscaling

4.5.1 The `ScaledObject` Configuration

4.5.2 Verifying the Setup

Chapter 5

Evaluation and Results

5.1 Test Environment

5.2 Evaluation Metrics

5.3 Test Scenarios

5.3.1 Scenario 1: Sudden Burst of Traffic

5.3.2 Scenario 2: Gradual Increase in Traffic

5.3.3 Scenario 3: Scaling to Zero

5.4 Presenting the Results

5.5 Analysis of Results

Chapter 6

Conclusion

6.1 Summary of Work

6.2 Contribution

6.3 Limitations

6.4 Future Work

Bibliography

[1] Kelsey Hightower, Brendan Burns, and Joe Beda. *Kubernetes: Up and Running.* O'Reilly Media, 2nd edition, 2019.

Appendix A

Appendices

A.1 Source Code for the Test Application

```python
import time
import os

def main():
    print("Processing message...")
    time.sleep(1)
    print("Message processed.")

if __name__ == "__main__":
    main()
```

Listing A.1: Sample Python worker.

A.2 Kubernetes Deployment YAML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: python-worker
spec:
  replicas: 1
  selector:
    matchLabels:
      app: python-worker
  template:
    metadata:
      labels:
```

```
13          app: python-worker
14      spec:
15        containers:
16        - name: worker
17          image: your-repo/python-worker:latest
18          resources:
19            limits:
20              memory: "128Mi"
21              cpu: "500m"
```

Listing A.2: Deployment YAML for the test application.