

**CLUSTER AUTOSCALER FOR UNMANAGED
KUBERNETES CLUSTER DEPLOYMENT ON CLOUD**

S. A. Chathura Madhushanka Siriwardhana

199366F

Degree of Master of Science

Department of Computer Science and Engineering

**University of Moratuwa
Sri Lanka**

August 2020

CLUSTER AUTOSCALER FOR UNMANAGED KUBERNETES CLUSTER DEPLOYMENT ON CLOUD

S. A. Chathura Madhushanka Siriwardhana

199366F

**Project report submitted in partial fulfillment of the requirements for the
degree Master of Science in Computer Science Specialising in Cloud Computing**

Department of Computer Science and Engineering

**University of Moratuwa
Sri Lanka**

August 2020

DECLARATION

I declare that this is my own work and this project report does not incorporate without acknowledgment any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text. Also, I hereby grant to the University of Moratuwa the non-exclusive right to reproduce and distribute my thesis, in whole or in part in print, electronic, or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

Date:

The above candidate has carried out research for the Masters Project report under my supervision.

Name of the supervisor: Prof. Gihan Dias

Signature of the supervisor:

Date:

ABSTRACT

This project report comprises details of the research "Cluster Autoscaler for Unmanaged Kubernetes Cluster Deployment on Cloud". Underutilization of server resources is a huge issue in enterprise data centers. When it comes to Kubernetes, underutilization and overutilization issue exists as is. In Kubernetes main course of having this issue is the use of fixed number of Kubernetes worker nodes. The Kubernetes community provides a cluster autoscaler solution to reduce underutilization and overutilization on Kubernetes clusters. This solution is only supported by a few major cloud providers like Google, AWS, DigitalOcean, and few others. Also, this solution is tightly bound to the auto scale group concept in those clouds. Hence this solution provided by the Kubernetes community cannot be used elsewhere. Therefore, there is a necessity for a general auto scaling approach that can be used on a wide range of cloud platforms and hardware virtualization platform. This research is to design and develop a Kubernetes Cluster Autoscaler which can be used on any cloud platform. This is achieved by removing the tightly bound auto scale group in the solution proposed by this research. Proposed solution use API and SDK provided by cloud provider and using libvirt which is a general purpose API library to manage KVM, Xen, VMWare ESXi and QEMU.

ACKNOWLEDGMENT

Firstly, I would like to express my sincere gratitude to Professor Gihan Dias at the University of Moratuwa for his valuable support and contribution in my research and the dedicated guidance provided to me throughout the research in terms of motivation, and immense knowledge.

My sincere thanks goes to the Cloud Native Computing Foundation, Google Inc. and Docker for developing Kubernetes and container standards, and introducing it to the opensource community.

Last but not the least, I would like to thank my family and fellow colleagues for the precious support given to me throughout this period in order to successfully complete the research work.

Table of Contents

CLUSTER AUTOSCALER FOR UNMANAGED KUBERNETES CLUSTER DEPLOYMENT ON CLOUD	i
CLUSTER AUTOSCALER FOR UNMANAGED KUBERNETES CLUSTER DEPLOYMENT ON CLOUD	ii
DECLARATION	iii
ABSTRACT	iv
ACKNOWLEDGMENT	v
Table of Contents	vi
List of Figures	viii
List of Tables	x
List of Abbreviations	xi
1. INTRODUCTION	1
1.1 Background	1
1.2 What is Autoscaling?	2
1.2.1 Schedule-based Autoscaling	2
1.2.2 Predictive Autoscaling	2
1.3 Kubernetes Cluster Autoscaler	3
1.4 Limitations in managed Kubernetes clusters	3
1.5 Problem Definition	3
1.6 Motivation	4
1.7 Objective	4
2. LITERATURE REVIEW	5
2.1 Container and containerization	5
2.2 Hardware Virtualization vs. containerization	6
2.3 Container orchestration	7
2.4 Kubernetes Container Orchestration Platform	7
2.5 SaltStack Container Orchestration	9
2.6 Helios Docker orchestration platform	9
2.7 Docker Swarm	9
2.8 Why Kubernetes over other container orchestration platforms	11
2.9 Functionality of Virtual Machine Auto Scaling	12
2.10 How does Kubernetes Scheduler work?	13

2.11 Why VM auto scaler can't use to scaler the Kubernetes Clusters directly	14
2.12 Managed vs. Unmanaged Kubernetes cluster on the cloud	14
2.13 Predictive Autoscaling vs. Rule-based Autoscaling	15
2.14 Kubernetes cluster autoscaler for OpenStack with Magnum	15
2.15 Kubernetes Cluster Architecture	16
2.16 How official Kubernetes Cluster Autoscaler Works	20
2.17 Summary	21
3. ARCHITECTURAL DESIGN	22
3.2 Kubernetes worker node removing module workflow	24
3.3 Kubernetes cluster balance module	25
3.3.1 hostPath and local volume types	26
3.3.2 Node Affinity and Node Selectors	28
3.4 Architectural Design Summary	28
4. IMPLEMENTATION	30
4.1 How to run the Kubernetes Cluster Autoscaler?	35
4.2 Add support to other cloud providers and virtualization platforms	37
4.3 Results	37
4.3.1 Testing Environment	38
4.3.2 Saturation in terms of Kubernetes	38
4.3.3 Manual Test Scenarios	40
4.3.4 Manual Testing Result	41
4.3.6 Automated Testing Results	42
5. CONCLUSION	44
6. REFERENCES	45

List of Figures

		Pages
Figure 1	The architecture of virtualization and containerization	6
Figure 2	Illustration of Kubernetes Architecture	8
Figure 3	Predictive autoscaling	15
Figure 4	Kubernetes Cluster Autoscaler workflow	17
Figure 5	Kubernetes Cluster Autoscaler node removal workflow	18
Figure 6	Autoscaler implementation on the Google Cloud Platform	19
Figure 7	High Level Architecture Diagram	22
Figure 8	Proposed Kubernetes worker node adding workflow	23
Figure 9	Proposed Kubernetes worker node removal workflow	24
Figure 10	Kubernetes cluster before and after running this module	26
Figure 11	hostPath pod definition	27
Figure 12	Persistent volume definition with the local volume type	27
Figure 13	Source Code Directory Structure	31
Figure 14	Main Program Code Snippet	32
Figure 15	Kube config searching and loading	33
Figure 16	Modified and Deleted events filter	34
Figure 17	Sample Configmap Resources	35
Figure 18	Minimum permissions requirement	36
Figure 19	Kubernetes Deployment Resource Configuration	36

Figure 20	Log Output of worker node adding process	37
Figure 21	Worker Node List	38
Figure 22	Log Output of Worker Node Deleting Process	38
Figure 23	Kubernetes Deployment resource YAML	39
Figure 24	Cluster autoscaler scale down graph node count vs. time	41
Figure 25	Pending pod against time in fix worker node cluster	42
Figure 26	Pending pod against time with cluster autoscaler	43

List of Tables

		Pages
Table 1	The architecture of virtualization and containerization	10
Table 2	Test Scenario result (Iteration vs. Time)	41

List of Abbreviations

LXC	Linux containers
OCI	Open Container Initiative
CNCF	Cloud Native Computing Foundation
HPA	Horizontal Pod Autoscaler
CI/CD	Continuous Integration / Continuous Delivery
GCE	Google Compute Engine
GKE	Google Container Engine
ASG	AutoScaleGroups
AWS	Amazon Web Services
EKS	Elastic Kubernetes Service
API	Application Program Interface
AKS	Azure Kubernetes Service