Configuration Manual

MSc Research Project
Programme Name

Swapnil Deshpande
Student ID: x22159401

School of Computing
National College of Ireland

Supervisor: Dr Ahmed Makki

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Swapnil Deshpande
Student ID:	x22159401
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Dr Ahmed Makki
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	539
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	13th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

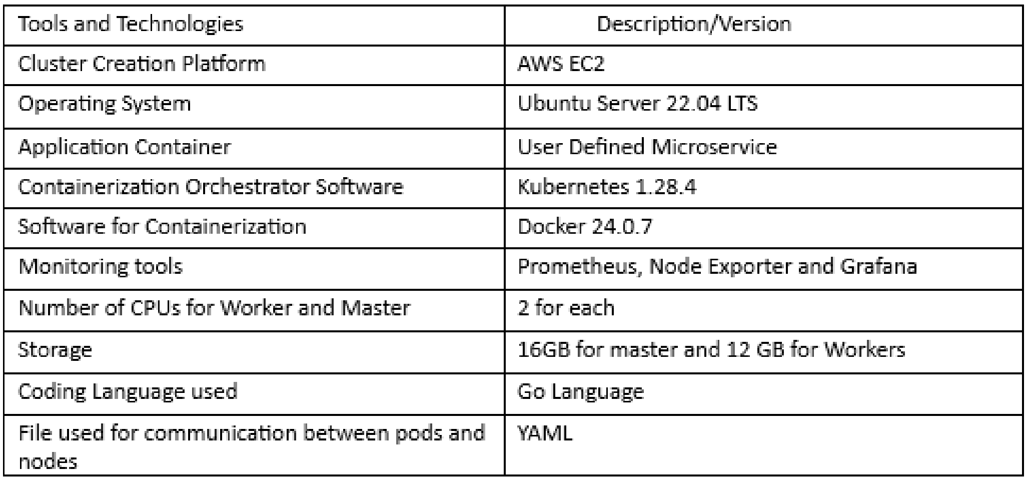
Configuration Manual

Swapnil Deshpande
x22159401

1 Introduction

The process of setting up a Kubernetes cluster on an Ubuntu server is explained simply in this report. The instructions and technology required to build the cluster will be covered in part 2. As we will discuss in the next section, we can use the concept of custom schedulers running concurrently to figure out how to check both the custom scheduler and the default scheduler. We will examine the code to have a deeper understanding of its functioning in accordance with the our custom scheduler. Finally, we will see how to configure the monitoring tools, prometheus and node exporter.

2 Tools and Technologies Required



Tools and Technologies	Description/Version
Cluster Creation Platform	AWS EC2
Operating System	Ubuntu Server 22.04 LTS
Application Container	User Defined Microservice
Containerization Orchestrator Software	Kubernetes 1.28.4
Software for Containerization	Docker 24.0.7
Monitoring tools	Prometheus, Node Exporter and Grafana
Number of CPUs for Worker and Master	2 for each
Storage	16GB for master and 12 GB for Workers
Coding Language used	Go Language
File used for communication between pods and nodes	YAML

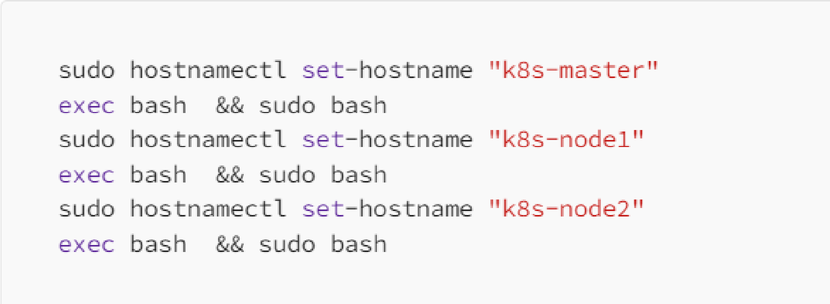
Figure 1: technology stack

3 Clustering using Kubernetes

I am utilizing AWS EC2 services for my research in order to benefit from cloud computing. I opted to use Ubuntu Server 22.04 because it provides improved compatibility with the latest Kubernetes features and upgrades, along with more recent software versions and updated kernel support.

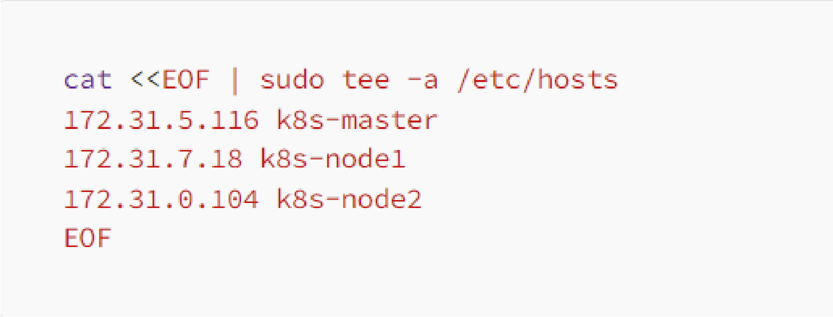
3.1 Node Creation

- Step1: Assign Unique Hostnames On the Master And Nodes Machine



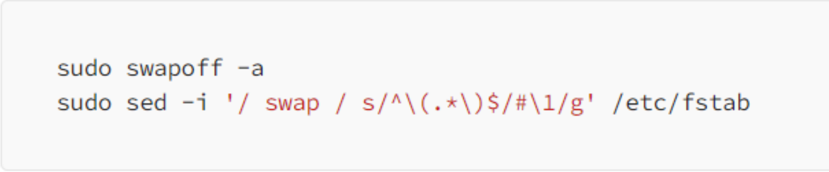
```
sudo hostnamectl set-hostname "k8s-master"
exec bash  && sudo bash
sudo hostnamectl set-hostname "k8s-node1"
exec bash  && sudo bash
sudo hostnamectl set-hostname "k8s-node2"
exec bash  && sudo bash
```

- Step2: Add IP's Address And Hostname To The Host file (ALL Nodes)



```
cat <<EOF | sudo tee -a /etc/hosts
172.31.5.116 k8s-master
172.31.7.18 k8s-node1
172.31.0.104 k8s-node2
EOF
```

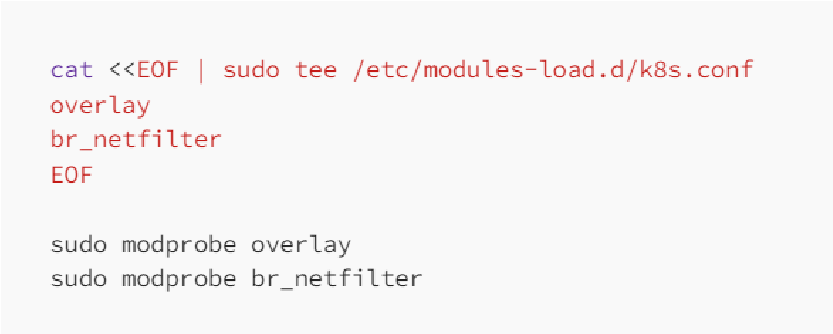
- Step3: Now Turn off the swap space. (ALL Nodes)



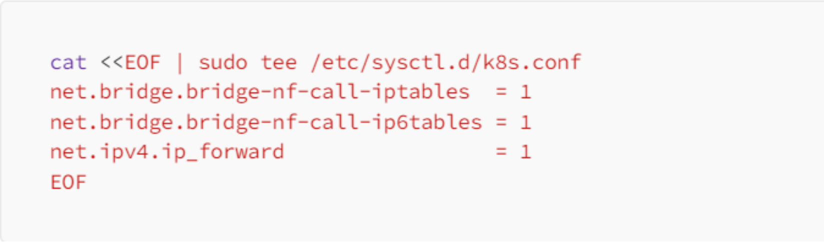
```
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

- Step4: Update the system package list and install the necessary packages for Container-D (ALL Nodes)

  Configure required modules



```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter
```



```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF
```

Then Apply sysctl parameters without rebooting to current running environment

```
sudo sysctl --system
```

- Step5: Now Install Docker In All Nodes

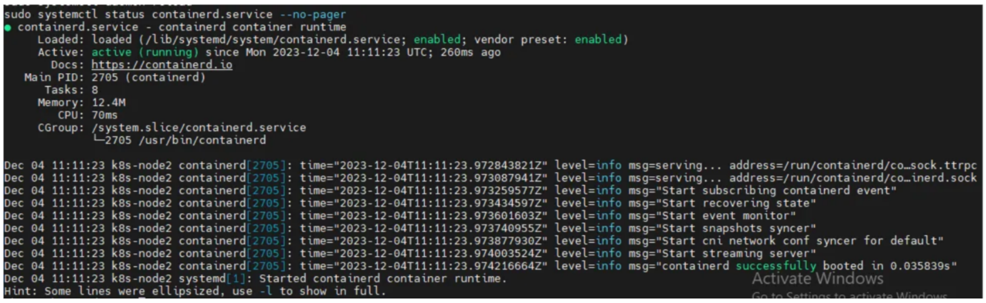
```
# Add Docker's official GPG key:
sudo apt-get update -y
sudo apt-get install ca-certificates curl gnupg -y
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
\
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update -y
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin -y
```

Configure Containerd To Start Using systemd as group

```
containerd config default | sudo tee /etc/containerd/config.toml
>/dev/null 2>&1
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g'
/etc/containerd/config.toml
```

- Step5: Start ContainerD Services and Check Status

```
sudo systemctl start containerd
sudo systemctl enable containerd
sudo systemctl restart containerd
sudo systemctl daemon-reload
sudo systemctl status containerd.service --no-pager
```



```
sudo systemctl status containerd.service --no-pager
● containerd.service - containerd container runtime
     Loaded: loaded (/lib/systemd/system/containerd.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2023-12-04 11:11:23 UTC; 260ms ago
       Docs: https://containerd.io
   Main PID: 2705 (containerd)
      Tasks: 8
     Memory: 12.4M
        CPU: 70ms
     CGroup: /system.slice/containerd.service
             └─2705 /usr/bin/containerd

Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.972843821Z" level=info msg=serving... address=/run/containerd/co…sock.ttrpc
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.973087941Z" level=info msg=serving... address=/run/containerd/co…inerd.sock
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.973259577Z" level=info msg="Start subscribing containerd event"
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.973434597Z" level=info msg="Start recovering state"
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.973601603Z" level=info msg="Start event monitor"
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.973740955Z" level=info msg="Start snapshots syncer"
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.973877930Z" level=info msg="Start cni network conf syncer for default"
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.974003524Z" level=info msg="Start streaming server"
Dec 04 11:11:23 k8s-node2 containerd[2705]: time="2023-12-04T11:11:23.974216664Z" level=info msg="containerd successfully booted in 0.035839s"
Dec 04 11:11:23 k8s-node2 systemd[1]: Started containerd container runtime.
Hint: Some lines were ellipsized, use -l to show in full.
```

- Step6: Now Install kubectl kubeadm and kubernetes cni

```
sudo apt-get update -y
sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update -y
sudo apt-get install -y kubelet kubeadm kubectl kubernetes-cni
sudo apt-mark hold kubelet kubeadm kubectl && sudo apt-mark hold docker
kubectl version --client && docker --version
```

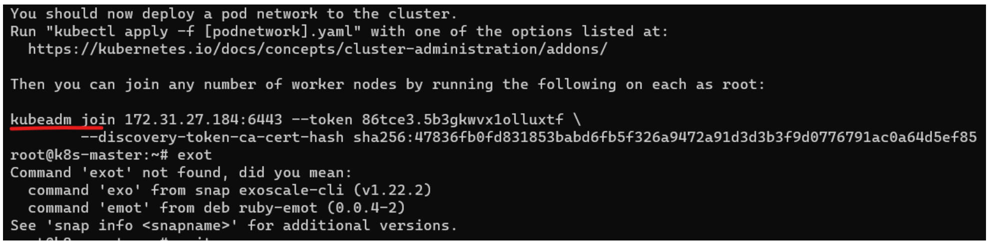
And check their status of installation

```
sudo systemctl daemon-reload
sudo systemctl start kubelet
sudo systemctl enable kubelet.service
sudo systemctl status kubelet.service --no-pager
```

- Step7: For Master Node, switch to root user, then initialize the kubeadm and set kubernetes directory path

```
sudo su -

kubeadm init

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

- Step8: For Node1 and Node 2, Run Token On Nodes As Root User

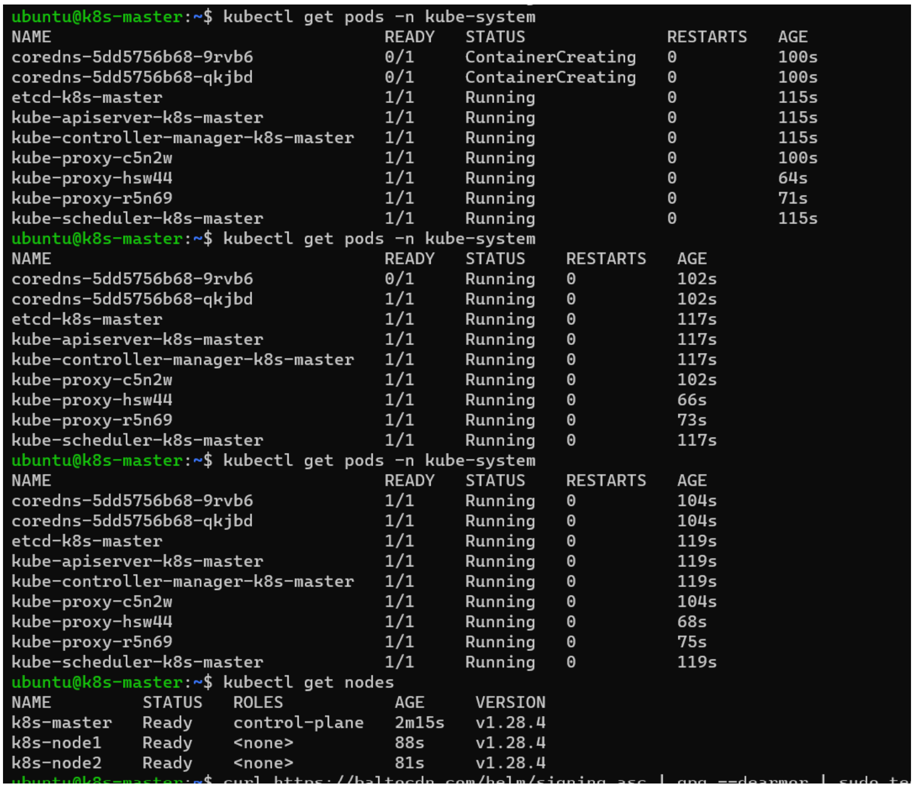


```
You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.27.184:6443 --token 86tce3.5b3gkwvx1olluxtf \
        --discovery-token-ca-cert-hash sha256:47836fb0fd831853babd6fb5f326a9472a91d3d3b3f9d0776791ac0a64d5ef85
root@k8s-master:~# exot
Command 'exot' not found, did you mean:
  command 'exo' from snap exoscale-cli (v1.22.2)
  command 'emot' from deb ruby-emot (0.0.4-2)
See 'snap info <snapname>' for additional versions.
```

- Step9: Install The Flannel pod network network

```
kubectl apply -f https://github.com/flannel-
io/flannel/releases/latest/download/kube-flannel.yml
```

- Step10: Now Let's Check: On Master As Normal User

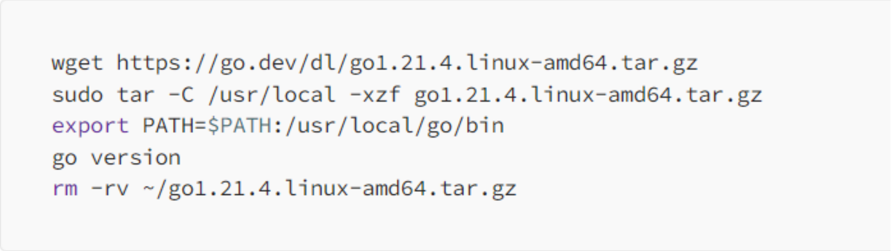


```
ubuntu@k8s-master:~$ kubectl get pods -n kube-system
NAME                               READY   STATUS              RESTARTS   AGE
coredns-5dd5756b68-9rvb6           0/1     ContainerCreating   0          100s
coredns-5dd5756b68-qkjbd           0/1     ContainerCreating   0          100s
etcd-k8s-master                    1/1     Running             0          115s
kube-apiserver-k8s-master          1/1     Running             0          115s
kube-controller-manager-k8s-master 1/1     Running             0          115s
kube-proxy-c5n2w                   1/1     Running             0          100s
kube-proxy-hsw44                   1/1     Running             0          64s
kube-proxy-r5n69                   1/1     Running             0          71s
kube-scheduler-k8s-master          1/1     Running             0          115s
ubuntu@k8s-master:~$ kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS   AGE
coredns-5dd5756b68-9rvb6           0/1     Running   0          102s
coredns-5dd5756b68-qkjbd           1/1     Running   0          102s
etcd-k8s-master                    1/1     Running   0          117s
kube-apiserver-k8s-master          1/1     Running   0          117s
kube-controller-manager-k8s-master 1/1     Running   0          117s
kube-proxy-c5n2w                   1/1     Running   0          102s
kube-proxy-hsw44                   1/1     Running   0          66s
kube-proxy-r5n69                   1/1     Running   0          73s
kube-scheduler-k8s-master          1/1     Running   0          117s
ubuntu@k8s-master:~$ kubectl get pods -n kube-system
NAME                               READY   STATUS    RESTARTS   AGE
coredns-5dd5756b68-9rvb6           1/1     Running   0          104s
coredns-5dd5756b68-qkjbd           1/1     Running   0          104s
etcd-k8s-master                    1/1     Running   0          119s
kube-apiserver-k8s-master          1/1     Running   0          119s
kube-controller-manager-k8s-master 1/1     Running   0          119s
kube-proxy-c5n2w                   1/1     Running   0          104s
kube-proxy-hsw44                   1/1     Running   0          68s
kube-proxy-r5n69                   1/1     Running   0          75s
kube-scheduler-k8s-master          1/1     Running   0          119s
ubuntu@k8s-master:~$ kubectl get nodes
NAME         STATUS   ROLES           AGE     VERSION
k8s-master   Ready    control-plane   2m15s   v1.28.4
k8s-node1    Ready    <none>          88s     v1.28.4
k8s-node2    Ready    <none>          81s     v1.28.4
ubuntu@k8s-master:~$ curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo te
```

4 Microservices Code Creation

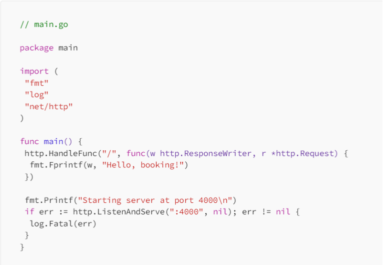
In this steps we are creating two microservices, for that initally we are installing go for coding the microservice functionality and then pushing code as docker container on docker hub,

For Go installation run following command,



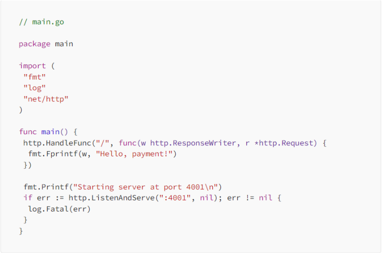
```
wget https://go.dev/dl/go1.21.4.linux-amd64.tar.gz
sudo tar -C /usr/local -xzf go1.21.4.linux-amd64.tar.gz
export PATH=$PATH:/usr/local/go/bin
go version
rm -rv ~/go1.21.4.linux-amd64.tar.gz
```

Microservices code:



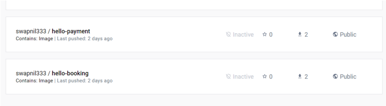
```
// main.go

package main

import (
 "fmt"
 "log"
 "net/http"
)

func main() {
 http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintf(w, "Hello, booking!")
 })

 fmt.Printf("Starting server at port 4000\n")
 if err := http.ListenAndServe(":4000", nil); err != nil {
  log.Fatal(err)
 }
}
```

Figure 2: Mircoservice 1



```
// main.go

package main

import (
 "fmt"
 "log"
 "net/http"
)

func main() {
 http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
  fmt.Fprintf(w, "Hello, payment!")
 })

 fmt.Printf("Starting server at port 4001\n")
 if err := http.ListenAndServe(":4001", nil); err != nil {
  log.Fatal(err)
 }
}
```

Figure 3: Mircoservice 2



swapnil333 / hello-payment
Contains: Image | Last pushed: 2 days ago
Inactive ☆ 0 2 Public

swapnil333 / hello-booking
Contains: Image | Last pushed: 2 days ago
Inactive ☆ 0 2 Public

Figure 4: Docker Image

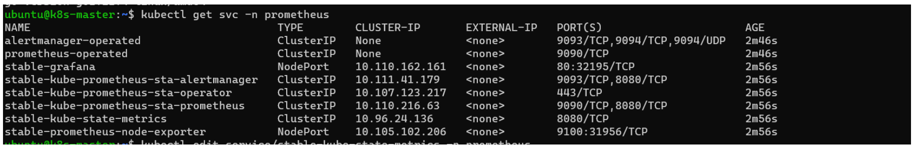
5 Prometheus, Grafana and Node Exporter Installation

For Installation of Prometheus, Grafana and Node Exporter we make use of Helm, Helm is a tool that combines your configuration files into a single reusable package, automating the development, packaging, configuration, and deployment of Kubernetes applications.

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update prometheus-community
helm search repo prometheus-community
kubectl create namespace prometheus
helm install stable prometheus-community/kube-prometheus-stack -n
prometheus
```

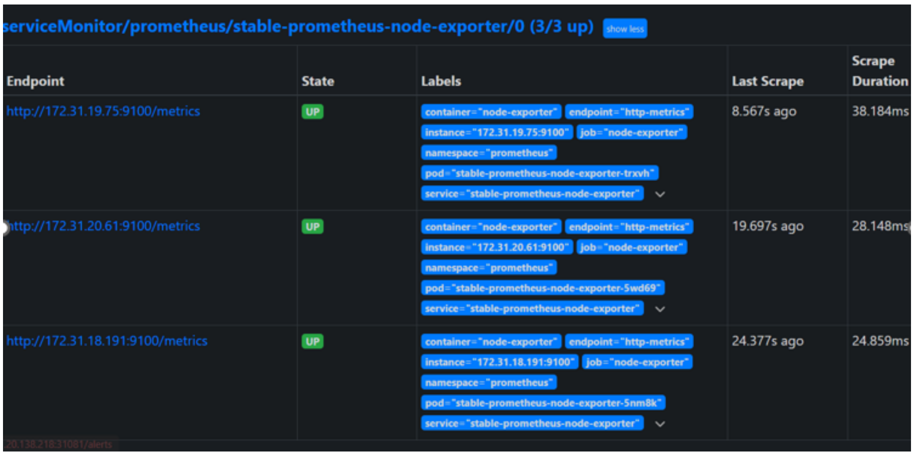
And then Expose the Prometheus and Grafana services to outside world by changing the ClusterIP to NodePort

```
kubectl get pods -n prometheus
kubectl get svc -n prometheus
kubectl edit service/stable-grafana -n prometheus
kubectl edit service/stable-kube-prometheus-sta-prometheus -n prometheus
kubectl edit service/stable-kube-prometheus-sta-alertmanager -n prometheus
kubectl edit service/stable-kube-state-metrics -n prometheus
kubectl edit service/stable-prometheus-node-exporter -n prometheus
kubectl edit service/stable-kube-prometheus-sta-operator -n prometheus
kubectl edit service/alertmanager-operated -n prometheus
kubectl edit service/prometheus-operated -n prometheus
kubectl describe secret stable-kube-prometheus-sta-prometheus -n prometheus
kubectl get svc -n prometheus
```



```
ubuntu@k8s-master:~$ kubectl get svc -n prometheus
NAME                                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
alertmanager-operated                     ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   2m46s
prometheus-operated                       ClusterIP   None             <none>        9090/TCP                     2m46s
stable-grafana                            NodePort    10.110.162.161   <none>        80:32195/TCP                 2m56s
stable-kube-prometheus-sta-alertmanager   ClusterIP   10.111.41.179    <none>        9093/TCP,8080/TCP            2m56s
stable-kube-prometheus-sta-operator       ClusterIP   10.107.123.217   <none>        443/TCP                      2m56s
stable-kube-prometheus-sta-prometheus     ClusterIP   10.110.216.63    <none>        9090/TCP,8080/TCP            2m56s
stable-kube-state-metrics                 ClusterIP   10.96.24.136     <none>        8080/TCP                     2m56s
stable-prometheus-node-exporter           NodePort    10.105.102.206   <none>        9100:31956/TCP               2m56s
```

Then Finally we will able to see the prometheus dashboard with all EC2 instance registered as target as shown below,



serviceMonitor/prometheus/stable-prometheus-node-exporter/0 (3/3 up)

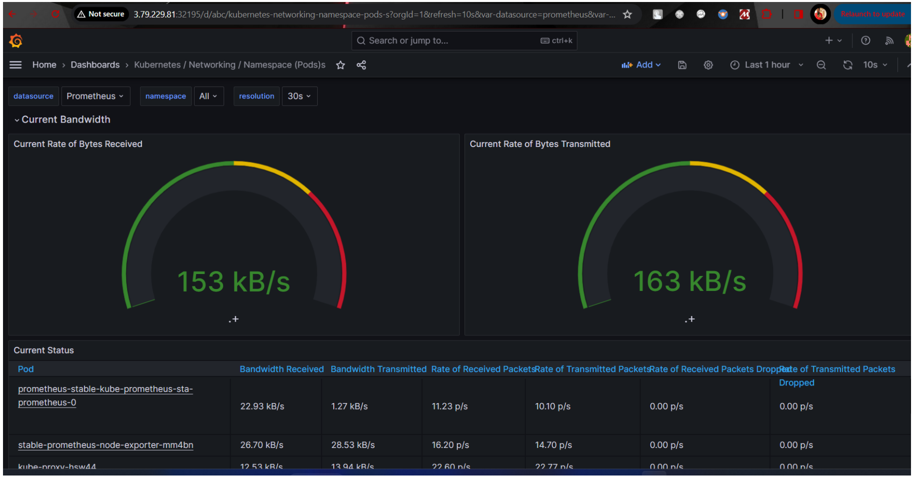
Endpoint	State	Labels	Last Scrape	Scrape Duration

http://172.31.19.75:9100/metrics	UP	container="node-exporter"
		endpoint="http-metrics"
		instance="172.31.19.75:9100"
		job="node-exporter"
		namespace="prometheus"
		pod="stable-prometheus-node-exporter-trxvh"
		service="stable-prometheus-node-exporter"
			8.567s ago	38.184ms

ttp://172.31.20.61:9100/metrics	UP	container="node-exporter"
		endpoint="http-metrics"
		instance="172.31.20.61:9100"
		job="node-exporter"
		namespace="prometheus"
		pod="stable-prometheus-node-exporter-5wd69"
		service="stable-prometheus-node-exporter"
			19.697s ago	28.148ms

http://172.31.18.191:9100/metrics	UP	container="node-exporter"
		endpoint="http-metrics"
		instance="172.31.18.191:9100"
		job="node-exporter"
		namespace="prometheus"
		pod="stable-prometheus-node-exporter-5nm8k"
		service="stable-prometheus-node-exporter"
			24.377s ago	24.859ms

For Grafana after accessing the website and login we need to import dashboard which is basically for kubernetes monitoring, as import number is 12125 , and then we can see dashboard which contains parameters related to scheduler using different namespace to assess the various parameters


Not secure
3.79.229.81:32195/d/abc/kubernetes-networking-namespace-pods-s?orgId=1&refresh=10s&var-datasource=prometheus&var-...
Relaunch to update
Search or jump to...
ctrl+k
Home › Dashboards › Kubernetes / Networking / Namespace (Pods)s
Add
Last 1 hour
10s
datasource
Prometheus
namespace
All
resolution
30s
Current Bandwidth
Current Rate of Bytes Received
153 kB/s
Current Rate of Bytes Transmitted
163 kB/s
Current Status

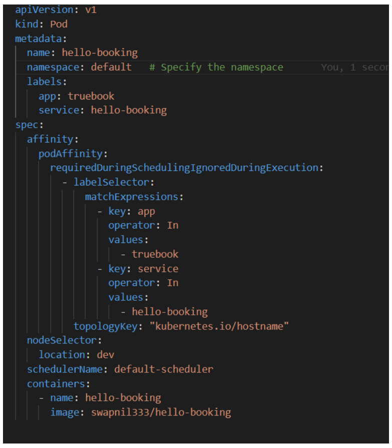
Pod	Bandwidth Received	Bandwidth Transmitted	Rate of Received Packets	Rate of Transmitted Packets	Rate of Received Packets Dropped	Rate of Transmitted Packets Dropped
prometheus-stable-kube-prometheus-sta-prometheus-0	22.93 kB/s	1.27 kB/s	11.23 p/s	10.10 p/s	0.00 p/s	0.00 p/s
stable-prometheus-node-exporter-mm4bn	26.70 kB/s	28.53 kB/s	16.20 p/s	14.70 p/s	0.00 p/s	0.00 p/s
kube-proxy-hsw44	12.53 kB/s	13.94 kB/s	22.60 p/s	22.77 p/s	0.00 p/s	0.00 p/s

6 Custom Scheduler Implementation

Following Commands needs to execute our custom scheduler, and our microservice will execute our scheduler using deployment file,

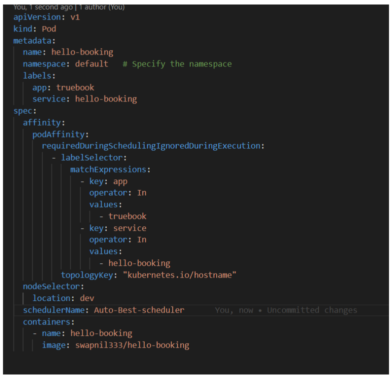
Firstly we need to import our custom scheduler,
https://github.com/swapnild333/mycustomscheduler

Followings are the deployment file and scheduler code,



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-booking
  namespace: default   # Specify the namespace
  labels:
    app: truebook
    service: hello-booking
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: app
                operator: In
                values:
                  - truebook
              - key: service
                operator: In
                values:
                  - hello-booking
          topologyKey: "kubernetes.io/hostname"
  nodeSelector:
    location: dev
  schedulerName: default-scheduler
  containers:
    - name: hello-booking
      image: swapnil333/hello-booking
```

Figure 5: Microservice 1 Deployment Script


You, 1 second ago | 1 author (You)
You, now • Uncommitted changes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-booking
  namespace: default   # Specify the namespace
  labels:
    app: truebook
    service: hello-booking
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchExpressions:
              - key: app
                operator: In
                values:
                  - truebook
              - key: service
                operator: In
                values:
                  - hello-booking
          topologyKey: "kubernetes.io/hostname"
  nodeSelector:
    location: dev
  schedulerName: Auto-Best-scheduler
  containers:
    - name: hello-booking
      image: swapnil333/hello-booking
```

Figure 6: Microservice 2 Deployment Script

References

Documentation — Grafana Labs — grafana.com (n.d.). `https://grafana.com/docs/`. [Accessed 08-12-2023].

Getting Started - Cloud Computing Tutorials for Building on AWS — aws.amazon.com (n.d.). `https://aws.amazon.com/getting-started/`. [Accessed 08-12-2023].

Installing Helm — helm.sh (n.d.). `https://helm.sh/docs/intro/install/`. [Accessed 08-12-2023].

Poniszewska-Maranda, A. and Czechowska, E. (2021). Kubernetes cluster for automating software production environment, *Sensors* **21**: 1910.

Prometheus (n.d.). Overview — Prometheus — prometheus.io, `https://prometheus.io/docs/introduction/overview/`. [Accessed 08-12-2023].

Wikipedia contributors (2023). Kubernetes — Wikipedia, the free encyclopedia. [Online; accessed 8-December-2023].
URL: *https://en.wikipedia.org/w/index.php?title=Kubernetesoldid=1188576677*