

Configuration Manual

MSc Research Project
Cloud Computing

Rahul Dhanapal Narawade
Student ID: 22144943

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing



Student Name:	Rahul Dhanapal Narawade
Student ID:	22144943
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	14/12/2023
Project Title:	Configuration Manual
Word Count:	3424
Page Count:	20

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:	Rahul Dhanapal Narawade
Date:	13th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Rahul Dhanapal Narawade
22144943

1 Introduction

This configuration manual provides guidance on the implementation of scheduling algorithm enhancements of KubeEdge with EdgeMesh for research to optimize cluster performance. This contains the prerequisites and specifications for building the experimental setup. The guide also includes a detailed, step-by-step implementation plan for the proposed solution that could be useful for others looking to replicate the work completed as part of the research project. This configuration manual provides technical information for other researchers regarding the methodologies used in the research.

1.1 Prerequisites

The project requires the following prerequisites

Computing system: A system that can support multiple virtual machines (VMs). Researcher has used MacBook M1 Pro (MacOS Sonoma Version 14.0) with 16 GB memory. This system provides sufficient resources for running multiple VMs and developing and testing the project code.

IDE: An integrated development environment (IDE) for implementing and building the code Microsoft Visual Studio Code (VS Code Version: 1.82.3) or a similar well-suited IDE for developing Golang applications. Visual Studio Code (2023)

GO Language: Familiarity or access to the Go language's syntax, data structures, and built-in functions. Some basic knowledge of the Go programming language recommended. (go version go1.18.1 linux/arm64) *The Go Programming Language* (2023)

Docker Hub: Access to and understanding of Docker Hub, a public registry for storing and sharing Docker images. Basic familiarity with Docker Hub concepts, such as creating repositories, pushing, and pulling images. (Docker Hub Desktop version 4.24.2) (Install Docker Desktop on Mac, 2023) Docker Documentation (2023)

Version controlling: Basic knowledge of version control systems, such as Git. Exposure to Github for versioning and basic git commands. *Downloading Package* (2023)

Kubernetes: Understanding of container orchestration platform Kubernetes. Exposure to basic workflow and Kubernetes concepts, such as pods, deployments, services, and clusters. (v 1.21.1) *Kubernetes Documentation* (2023)

KubeEdge: Information on KubeEdge architecture, its components, and its integration with Kubernetes. (v 1.12.1) *KubeEdge* (2023a)

2 Environment Setup

This section guides about the environment setup.

2.1 Creating VM on MacOS

- **Step 1:** Download Ubuntu Image from the official website: <https://ubuntu.com/download/server/arm> (Version 4.2.5) Ubuntu (2023)
- **Step 2:** Download UTM from the official website: <https://mac.getutm.app/UTM> (2023)
- **Step 3:** Run UTM and follow these sub-steps:
 1. Click on the "+" button in the top left corner of the UTM window.
 2. Select "Virtualize" from the menu.
 3. Select "Linux" from the list of available operating systems.
 4. Click on the "Browse" button and select the Ubuntu ISO image that was downloaded in Step 1.
 5. Choose the amount of Memory and CPU cores that you want to allocate to the virtual machine. For this reserach memory was 1 GB and CPU was 1.
 6. Click on the "Create" button.
- **Step 3:** Similarly create three more VM's by follwong steps 1-3 in total 4VMs.

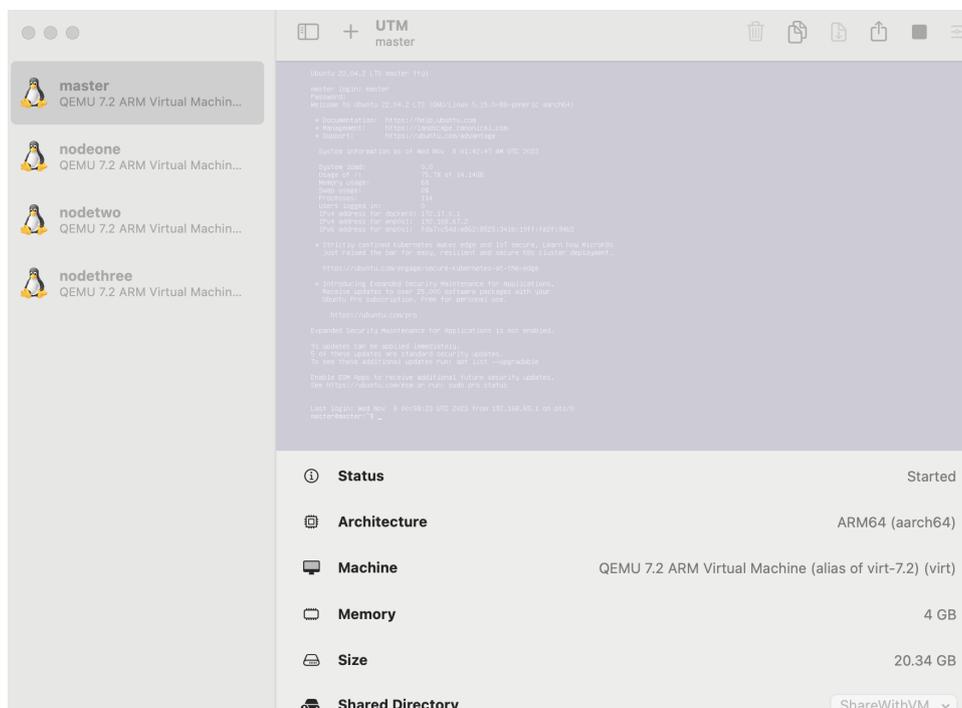


Figure 1: UTM Window

2.2 Setting up Master node – CloudCore

At the master node, we need to install Kubernetes and KubeEdge Cloud core. Follow the below steps to get these both installed (Nair; 2023; Gaponcic; 2023; KubeEdgeGit; 2023; KubeEdge; 2023b; EdgeMesh; 2023).

1. **Step 1:** Login to VM1. For illustration purposes, VM1 is considered the master node.

2. **Step 2:** Change user to root and disable swap with the following commands:

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

3. **Step 3:** Install Docker and containerd with the following command:

```
1 sudo apt-get install docker-ce docker-ce-cli containerd.io
  docker-buildx-plugin docker-compose-plugin
```

4. **Step 4:** Install GO language. Optionally, a specific version for KubeEdge can be downloaded and installed. For research purposes, the repo was downloaded, so GO is required to build.

```
1 sudo apt install golang-go
2 export GOOS=linux
3 export GOARCH=arm64
4 source ~/.bashrc
5 export PATH=$PATH:/snap/bin:/usr/go/bin
6 export GOPATH=/usr/go
7 export GOBIN=$GOPATH/bin
8 export PATH=$PATH:$GOBIN:$GOROOT/bin
```

5. **Step 5:** Clone KubeEdge and make a build:

```
1 git clone https://github.com/kubeedge/kubeedge $GOPATH/
  src/github.com/kubeedge/kubeedge
2 cd $GOPATH/src/github.com/kubeedge/kubeedge
3 git checkout release-1.11
4 apt install make
5 make all WHAT=keadm
```

6. **Step 6:** Install Kubernetes and CNI:

```
1 sudo apt-get install -y kubelet=1.21.1-00 kubeadm
  =1.21.1-00 kubectl=1.21.1-00
2 kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver
  -advertise-address=192.168.0.208
3 mkdir -p $HOME/.kube
4 sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
5 sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

```
6 kubectl taint nodes --all node-role.kubernetes.io/master-
7 kubectl apply -f https://docs.projectcalico.org/v3.11/
  manifests/calico.yaml
8 kubectl get nodes
```

7. **Step 7:** Copy KubeEdge build to the user path:

```
1 cd /root
2 $GOPATH/src/github.com/kubeedge/kubeedge
3 cp ./_output/local/bin/keadm /usr/bin/
```

8. **Step 8:** Initialize KubeEdge (change the IP to the IP of the Master node):

```
1 keadm init --advertise-address="192.168.67.2" --profile
  version=v1.12.1 --kube-config=/root/.kube/config --set
  cloudCore.modules.dynamicController.enable=true
```

9. **Step 9:** EdgeMesh for Master nodes:

```
1 kubectl label services kubernetes service.edgemesh.
  kubeedge.io/service-proxy-name=""
2 git clone https://github.com/kubeedge/edgemesh.git
3 cd edgemesh
4 kubectl apply -f build/crds/istio/
5 kubectl apply -f build/agent/resources/
6 kubectl get nodes --all-namespaces
7 kubectl get all -n kubeedge -o wide
```



```
root@master:~#
root@master:~# kubectl get nodes --all-namespaces
NAME      STATUS   ROLES                AGE    VERSION
master    Ready    control-plane,master 4d3h   v1.21.1
nodeone   Ready    agent,edge           4d3h   v1.22.6-kubeedge-v1.11.3
nodethree Ready    agent,edge           4d3h   v1.22.6-kubeedge-v1.11.3
nodetwo   Ready    agent,edge           4d3h   v1.22.6-kubeedge-v1.11.3
root@master:~#
root@master:~#
```

Figure 2: Get all nodes

10. **Step 10:** Install EdgeMesh Gateway:

```
1 kubectl apply -f build/gateway/resources
```

11. **Step 11:** Generate a token from Master nodes for edge nodes to connect and copy the token:

```
1 keadm gettoken
```

```

root@master:~#
root@master:~# kubectl get pods --all-namespaces -o wide
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE   IP              NODE
kube-system  calico-kube-controllers-5bcd7db644-d2pwn  1/1     Running   0           26m   192.168.219.66  master
kube-system  calico-node-dfbr1                         0/1     Init:Error 5         10m   192.168.67.4    nodetwo
kube-system  calico-node-l4f6j                         0/1     Init:Error 6         12m   192.168.67.3    nodeone
kube-system  calico-node-lkpjj                         1/1     Running   0           26m   192.168.67.2    master
kube-system  calico-node-s4v7b                         0/1     Init:Error 5         5m43s  192.168.67.5    nodethree
kube-system  coredns-558bd4d5db-2tfff                 1/1     Running   0           28m   192.168.219.67  master
kube-system  coredns-558bd4d5db-jxz4k                 1/1     Running   0           28m   192.168.219.65  master
kube-system  etcd-master                              1/1     Running   0           28m   192.168.67.2    master
kube-system  kube-apiserver-master                     1/1     Running   0           28m   192.168.67.2    master
kube-system  kube-controller-manager-master            1/1     Running   0           28m   192.168.67.2    master
kube-system  kube-proxy-75dwf                          1/1     Running   0           28m   192.168.67.2    master
kube-system  kube-proxy-gm768                          1/1     Running   0           10m   192.168.67.4    nodetwo
kube-system  kube-proxy-p9sqz                          1/1     Running   0           5m43s  192.168.67.5    nodethree
kube-system  kube-proxy-xpr74                          1/1     Running   0           12m   192.168.67.3    nodeone
kube-system  kube-scheduler-master                     1/1     Running   0           28m   192.168.67.2    master
kubedge     cloudfcore-5876c76687-mm6m9              1/1     Running   0           25m   192.168.67.2    master
root@master:~#
root@master:~#

```

Figure 3: Get all pods

2.3 Setting Up EdgeNode(s) - Edgecore

Unlike Master node, Kubernetes is not required on edge nodes. Follow the below steps to setup multiple edge nodes. For research purpose, Three edge nodes are connected to the master.(Nair; 2023; Gaponcic; 2023; KubeEdgeGit; 2023; KubeEdge; 2023b; EdgeMesh; 2023).

1. **Step 1:** Login to VM1. For illustration purposes, VM1 is considered the master node.

2. **Step 2:** Change user to root and disable swap with the following commands:

```

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

```

3. **Step 3:** Install Docker and containerd with the following commands:

```

1 sudo apt-get install docker-ce docker-ce-cli containerd.io
  docker-buildx-plugin docker-compose-plugin
2 sudo cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
3 net.bridge.bridge-nf-call-ip6tables = 1
4 net.bridge.bridge-nf-call-iptables = 1
5 EOF

```

4. **Step 4:** Install GO language. Optionally, a specific version for KubdEdge can be downloaded and installed. For research purposes, the repo was downloaded, so GO is required to build.

```

1 sudo apt install golang-go
2 export GOOS=linux
3 export GOARCH=arm64
4 source ~/.bashrc
5 export PATH=$PATH:/snap/bin:/usr/go/bin
6 export GOPATH=/usr/go
7 export GOBIN=$GOPATH/bin
8 export PATH=$PATH:$GOBIN:$GOROOT/bin

```

5. **Step 5:** Clone KubeEdge and make a build:

```
1 git clone https://github.com/kubeedge/kubeedge $GOPATH/  
  src/github.com/kubeedge/kubeedge  
2 cd $GOPATH/src/github.com/kubeedge/kubeedge  
3 git checkout release-1.11  
4 apt install make  
5 make all WHAT=keadm
```

6. **Step 7:** Copy KubeEdge build to the user path:

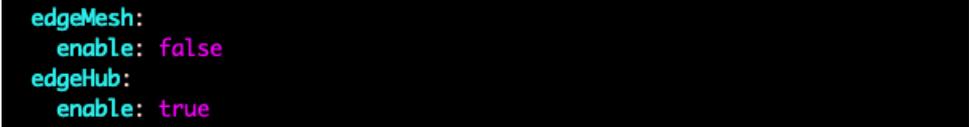
```
1 cd /root  
2 $GOPATH/src/github.com/kubeedge/kubeedge  
3 cp ./_output/local/bin/keadm /usr/bin/
```

7. **Step 8:** Connect to the cloud. Change the IP of the master and copy the token from the master:

```
1 keadm join --cloudcore-ipport=192.168.67.2:10000 --token  
  ="Token Generated from Master"
```

8. **Step 9:** Update the YAML file (vi /etc/kubeedge/config/cloudcore.yaml):

(a) EdgeMesh to false



```
edgeMesh:  
  enable: false  
edgeHub:  
  enable: true
```

Figure 4: EdgeMesh YAML change 1

(b) Enable Metamanager



```
metaManager:  
  contextSendGroup: hub  
  contextSendModule: websocket  
  enable: true  
metaServer:  
  enable: true  
  server: 127.0.0.1:10550  
  remoteQueryTimeout: 60
```

Figure 5: EdgeMesh YAML change 2

(c) Update cluster DNS

```
edged:
  cgroupDriver: cgroupfs
  cgroupRoot: ""
  cgroupsPerQOS: true
  clusterDNS: 169.254.96.16
  clusterDomain: cluster.local
  cniBinDir: /opt/cni/bin
  cniCacheDirs: /var/lib/cni/cache
```

Figure 6: EdgeMesh YAML change 3

9. **Step 10:** Restart edgecore services using the following commands:

```
1 systemctl restart edgecore.service
2 systemctl status edgecore.service
```

10. **Step 11:** Perform curl with the following command, expecting a response and not an error EdgeMesh (2023)

```
1 curl 127.0.0.1:10550/api/v1/services
```

2.4 Gateway Setup at Master node

Run the below commands at Master node.

```
1 cd edgmesh
2 kubectl apply -f build/crds/istio/
3 kubectl apply -f build/agent/resources/
4 kubectl apply -f build/gateway/resources
5 kubectl get all -n kubeedge -o wide
6 kubectl get pods --all-namespaces
```

2.5 Deploy Sample Application

1. **Step 1:** At the master node, deploy the sample application using the following commands:

```
1 cd edgmesh
2 vi examples/hostname-lb-random-gateway.yaml
```

Update the replicas to 6. Note: There are 3 nodes, so each will get two pods.

```
spec:
  replicas: 6
  selector:
    matchLabels:
      app: hostname-lb-edge
```

Figure 7: Sample App Host Modified

2. **Step 2:** Run the following command to deploy pods. EdgeMesh (2023)

```
1 kubectl apply -f examples/hostname-lb-random-gateway.yaml
```

```

root@master:~/edgemes#
root@master:~/edgemes# kubectl apply -f examples/hostname-lb-random-gateway.yaml
deployment.apps/hostname-lb-edge created
service/hostname-lb-svc created
gateway.networking.istio.io/edgemes-gateway created
destinationrule.networking.istio.io/hostname-lb-svc created
virtualservice.networking.istio.io/edgemes-gateway-svc created
root@master:~/edgemes#

```

Figure 8: Deploy Sample App

- Step 3:** Verify that the sample application works using the following commands. The IP address is the master node's exposed IP displayed in the output of the following command:

```

1 kubectl get pods --all-namespaces -o wide
2 curl 192.168.67.2:23333

```

- Step 4:** As this is a sample test application, it returns the pod name from where the request got served.

```

root@master:~/edgemes#
root@master:~/edgemes# curl http://192.168.67.2:23333/
hostname-lb-edge-5cdf5c758c-5rwxg
root@master:~/edgemes# curl http://192.168.67.2:23333/
hostname-lb-edge-5cdf5c758c-9vvjn
root@master:~/edgemes#

```

Figure 9: Test Sample App URL

- Step 5:** The image below shows the final cluster details.

```

root@master:~/edgemes# kubectl get pods --all-namespaces -o wide
NAMESPACE   NAME                                                    READY   STATUS    RESTARTS   AGE   IP              NODE
kube-system  calico-kube-controllers-5bcd7db644-5jqfz             1/1     Running   0           27d   192.168.219.67  master
kube-system  calico-node-g7957                                     1/1     Running   0           27d   192.168.67.2   master
kube-system  coredns-558bd4d5db-bkxq                             1/1     Running   0           27d   192.168.219.66  master
kube-system  coredns-558bd4d5db-ctwg8                            1/1     Running   0           27d   192.168.219.65  master
kube-system  etcd-master                                           1/1     Running   0           27d   192.168.67.2   master
kube-system  kube-apiserver-master                                1/1     Running   0           27d   192.168.67.2   master
kube-system  kube-controller-manager-master                       1/1     Running   2           27d   192.168.67.2   master
kube-system  kube-proxy-dg9lw                                      1/1     Running   1           27d   192.168.67.3   nodeone
kube-system  kube-proxy-mpdmm                                     1/1     Running   0           27d   192.168.67.2   master
kube-system  kube-proxy-xpsfr                                     1/1     Running   2           27d   192.168.67.4   nodetwo
kube-system  kube-proxy-zr8wq                                     1/1     Running   0           12d   192.168.67.5   nodethree
kube-system  kube-scheduler-master                               1/1     Running   2           27d   192.168.67.2   master
kubedge     claudcore-5876c76687-4jwld                          1/1     Running   0           6h24m  192.168.67.2   master
kubedge     edgemes-agent-chp94                                   1/1     Running   0           37h   192.168.67.4   nodetwo
kubedge     edgemes-agent-q5csz                                  1/1     Running   0           37h   192.168.67.3   nodeone
kubedge     edgemes-agent-q7n99                                  1/1     Running   0           37h   192.168.67.5   nodethree
kubedge     edgemes-agent-zdtvx                                  1/1     Running   0           37h   192.168.67.2   master
kubedge     edgemes-gateway-6d477479f6-mvhww                   1/1     Running   0           37h   192.168.67.2   master
kubedge     hostname-lb-edge-5cdf5c758c-5rwxg                   1/1     Running   0           37h   172.17.0.4     nodethree
kubedge     hostname-lb-edge-5cdf5c758c-6z8yp                   1/1     Running   0           37h   172.17.0.3     nodeone
kubedge     hostname-lb-edge-5cdf5c758c-9vvjn                   1/1     Running   0           37h   172.17.0.4     nodetwo
kubedge     hostname-lb-edge-5cdf5c758c-llxkp                   1/1     Running   0           37h   172.17.0.4     nodeone
kubedge     hostname-lb-edge-5cdf5c758c-tsxg4                   1/1     Running   0           37h   172.17.0.3     nodethree
kubedge     hostname-lb-edge-5cdf5c758c-x648b                   1/1     Running   0           37h   172.17.0.3     nodetwo
root@master:~/edgemes#
root@master:~/edgemes# curl http://192.168.67.2:23333/
hostname-lb-edge-5cdf5c758c-5rwxg
root@master:~/edgemes# curl http://192.168.67.2:23333/
hostname-lb-edge-5cdf5c758c-9vvjn
root@master:~/edgemes#

```

Figure 10: Cluster all pods

3 Implementation

1. **Step 1:** Download or clone the Git repository for EdgeMesh from GitHub. Use the following command in your terminal:

```
1 git clone https://github.com/kubeedge/edgemes
```

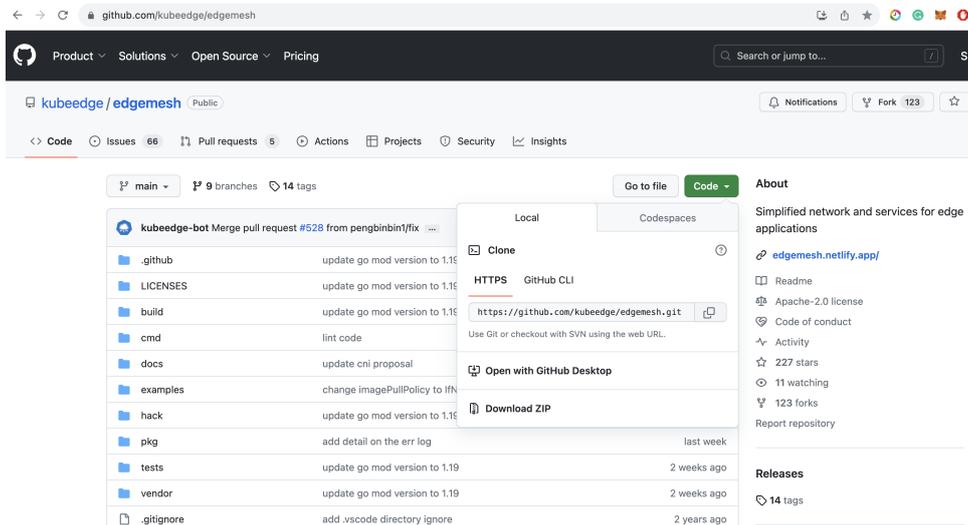


Figure 11: Git Repo

2. **Step 2:** Open the downloaded repository in your preferred Integrated Development Environment (IDE).

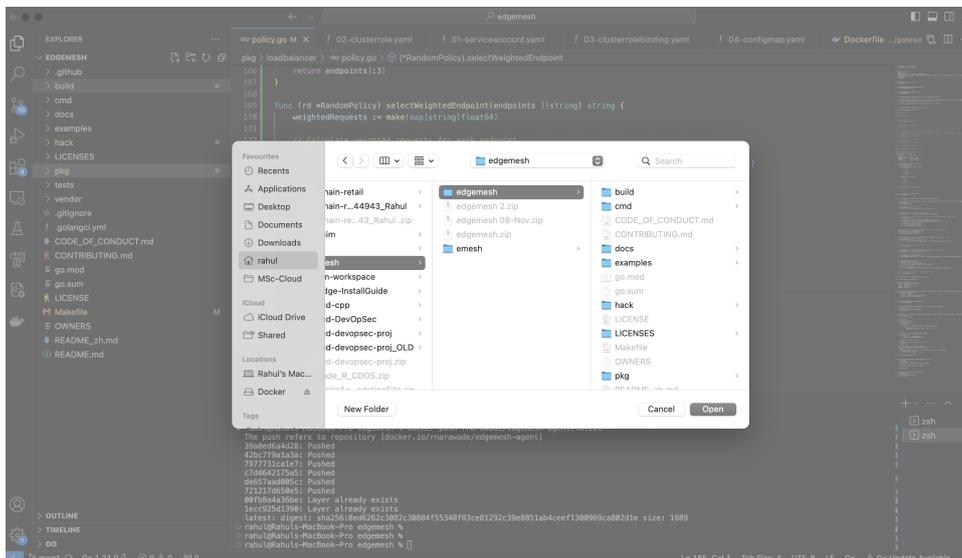


Figure 12: Open Repo

3.1 Load balancer policy Implementation

Open file `/github/EdgeMesh/edgemes/pkg/loadbalancer/policy.go` In this file the default load balancing policies are exists. Each of the policy has predefined code starting

with its 'struct', and methods such as 'policyname', 'update', 'pick', 'sync' and 'Release'. Out of these methods depending upon the research the struct and pick function are expected to be updated. The said research updates the existing methods only, as an example the struct updated as below.

1. **Step 1:** Update the existing random policy or write a new one. In this research updated existing.

```
type RandomPolicy struct {
    lock          sync.Mutex
    lockCounter   int
    endpointLatencies map[string]int
    weights       map[string]int
    requestCount  map[string]int
}
```

Figure 13: Struct of new policy

2. **Step 2:** The 'NewRandomPolicy' method creates a new instance of the 'RandomPolicy' struct. This method initializes the 'RandomPolicy' struct with predefined endpoint latencies, weights, and an empty request count map.

```
func NewRandomPolicy() *RandomPolicy {
    return &RandomPolicy{
        endpointLatencies: map[string]int{
            "0": 10,
            "1": 10,
            "2": 15,
            "3": 15,
            "4": 20,
            "5": 20,
        },
        weights: map[string]int{
            "0": 3,
            "1": 2,
            "2": 2,
            "3": 1,
            "4": 1,
            "5": 1,
        },
        requestCount: map[string]int{},
    }
}
```

Figure 14: Struct of new policy

3. **Step 3:** The 'Name' and 'Update' methods are empty declarations by default. The 'Name' method returns the name associated with the policy. The 'Update' method updates the 'RandomPolicy' based on old and new istioapi.DestinationRule instances and does not perform any specific update.
4. **Step 4:** The 'Pick' method selects an endpoint based on logic written for selection. The method selects endpoints with the lowest latency, and if multiple endpoints exist, the method selects one endpoint based on weights and the lowest request counter. The method increments the request count and returns the selected endpoint. Below is the code snippet. Initially, the small code can be changed like scheduling all requests locally Kim and Kim (2023).
5. **Step 5:** The 'selectLowestLatencyEndpoints' method sorts the endpoints based on latency and returns the first three with the lowest latency. Below is the code snippet.

```

func (rd *RandomPolicy) Pick(endpoints []string, srcAddr net.Addr, netConn net.Conn, cliReq *http.Request) (string, *http.Request,
rd.lock.Lock()
defer rd.lock.Unlock()

// Check if there are no endpoints
if len(endpoints) == 0 {
    return "", nil, errors.New("no endpoints available")
}

// Find the endpoints with the lowest latency
selectedEndpoints := rd.selectLowestLatencyEndpoints(endpoints)

// If there are more than one endpoint, select based on weights and lowest request counter
if len(endpoints) > 1 {
    // Find the endpoint with the lowest weighted request
    weightedEndpoint := rd.selectWeightedEndpoint(selectedEndpoints)
    // Increment the request count for the selected endpoint
    rd.requestCount[weightedEndpoint]++
    // Return the selected endpoint
    return weightedEndpoint, cliReq, nil
}

// If there is only one endpoint, increment the request count and return it
selectedEndpoint := selectedEndpoints[0]
rd.requestCount[selectedEndpoint]++
return selectedEndpoint, cliReq, nil
}

```

Figure 15: Pick method new policy

```

func (rd *RandomPolicy) selectLowestLatencyEndpoints(endpoints []string) []string {
    // Sort endpoints based on latency
    sort.Slice(endpoints, func(i, j int) bool {
        return rd.endpointLatencies[endpoints[i]] < rd.endpointLatencies[endpoints[j]]
    })

    // Return the n endpoints with the lowest latency
    return endpoints[:3]
}

```

Figure 16: Select Lowest Method

6. **Step 6:** The 'selectWeightedEndpoint' method calculates the weighted requests for each endpoint by dividing the request count by weights. Then the method sorts the endpoints based on weighted requests and only returns the one with the lowest weighted request. This method makes sure that the nodes are not overloaded and distributes the requests based on weights. Below is the code snippet.

```

func (rd *RandomPolicy) selectWeightedEndpoint(endpoints []string) string {
    weightedRequests := make(map[string]float64)

    // Calculate weighted requests for each endpoint
    for _, endpoint := range endpoints {
        weightedRequests[endpoint] = float64(rd.requestCount[endpoint]) / float64(rd.weights[endpoint])
    }

    // Sort endpoints based on weighted requests
    var sortedEndpoints []string
    for endpoint := range weightedRequests {
        sortedEndpoints = append(sortedEndpoints, endpoint)
    }
    sort.Slice(sortedEndpoints, func(i, j int) bool {
        return weightedRequests[sortedEndpoints[i]] < weightedRequests[sortedEndpoints[j]]
    })

    // Return the endpoint with the lowest weighted request
    return sortedEndpoints[0]
}

```

Figure 17: Select Highest weighed with lowest process counter

7. **Step 7:** The 'Sync' method synchronizes the policy, but here it is empty and does not perform any specific synchronization for the policy.
8. **Step 8:** The 'Release' method releases any resources associated with the 'Random-Policy,' but here it's empty and does not perform any specific release action for the

policy.

9. **Step 9:** Comment the contents of 'github/EdgeMesh/edgemesh/Makefile' and update 'github/EdgeMesh/edgemesh/Makefile' as below. This ensures that you can build your projects locally.

3.2 Docker files

1. **Step 1:** Update the EdgeMesh agent Dockerfile located at /github/EdgeMesh/edgemesh/build/a as shown below. This ensures that the changes performed are included while making the build.

```
COPY --from=builder /code/_output/local/bin/edgemesh-agent /usr/local/bin/edgemesh-agent

# Add support for auto-detection of iptables mode
COPY --from=builder /code/pkg/loadbalancer/ /code/pkg/loadbalancer/
COPY --from=builder /code/build/agent/iptables-wrapper /sbin/iptables-wrapper
RUN update-alternatives --install /sbin/iptables iptables /sbin/iptables-legacy 50
```

Figure 18: Docker file update for Agent

2. **Step 2:** Update the EdgeMesh gateway Dockerfile located at /github/EdgeMesh/edgemesh/build/a as below. This ensures that the changes performed are included while making the build.

```
COPY --from=builder /code/_output/local/bin/edgemesh-agent /usr/local/bin/edgemesh-agent

# Add support for auto-detection of iptables mode
COPY --from=builder /code/pkg/loadbalancer/ /code/pkg/loadbalancer/
COPY --from=builder /code/build/agent/iptables-wrapper /sbin/iptables-wrapper
RUN update-alternatives --install /sbin/iptables iptables /sbin/iptables-legacy 50
```

Figure 19: Docker file update for Gateway

3.3 Docker Hub

Sign up for the <https://hub.docker.com/signup> for the docker hub repo if not already. This repo username will be required for the next section. Docker Documentation (2023)

3.4 Build file changes (make file)

Open github/EdgeMesh/edgemesh/Makefile and update below. This file helps to create the build locally and contains the location of the remote docker hub repo where we will be updating the successful build. The 'rnarawade' is the docker hub repo name.

```
15 GOPATH?=$(shell go env GOPATH)
16 IMAGE_REPO ?= rnarawade
17 ARCH ?= amd64
18 IMAGE_TAG ?= latest
19 GO_LDFLAGS='${shell hack/make-rules/version.sh}'
```

Figure 20: Make file Change 1

```
.PHONY: images agentimage gatewayimage
images: agentimage gatewayimage
agentimage gatewayimage:
  docker build --build-arg GO_LDFLAGS=${GO_LDFLAGS} -t rnarawade/edgemes-${@:image=}:${IMAGE_TAG} -f build/${@:image=}/Dockerfile

.PHONY: push push-all push-multi-platform-images
push-all: push-multi-platform-images
```

Figure 21: Make file Change 2

3.5 YAML File at local system and Master node

Update the two YAML files below into local system where we are updating the code and update the same files onto Master node where we have cloned the EdgeMesh repo as a part of environment setup. These files helps us to get the latest build from docket hub.

1. **Step 1:** Update the YAML file located at `github/EdgeMesh/edgemes/build/agent/resources/05-daemonset.yaml` at both master and local as below:

```
privileged: true
image: rnarawade/edgemes-agent:latest
imagePullPolicy: Always
env:
```

Figure 22: YAML changes for Agent

2. **Step 2:** Update the YAML file located at `github/EdgeMesh/edgemes/build/gateway/resources/05-deployment.yaml` at both master and local as below:

```
- name: edgemes-gateway
  securityContext:
    privileged: true
  image: rnarawade/edgemes-gateway:latest
  imagePullPolicy: Always
  env:
```

Figure 23: YAML changes for Gateway

4 Build and Deployment

This section explains the build creation and deployment process for EdgeMesh.

4.1 Creating build – Make Images

Once all the changes mentioned in section 3 are completed, at the command of IDE execute the command ‘make images’ this will trigger the build for EdgeMesh Agent and Edgmesh Gateway both at the local system.

```
rahul@Rahuls-MacBook-Pro edgemesh %
rahul@Rahuls-MacBook-Pro edgemesh %
rahul@Rahuls-MacBook-Pro edgemesh % make images
docker build --build-arg GO_LDFLAGS='' -t rnarawade/edgemesh-agent:latest -f build/agent/Dockerfile .
[+] Building 30.6s (20/28) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.11kB
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:3.11
=> [internal] load metadata for docker.io/library/golang:1.17
=> [auth] library/golang:pull token for registry-1.docker.io
=> [auth] library/alpine:pull token for registry-1.docker.io
=> [builder 1/4] FROM docker.io/library/golang:1.17@sha256:87262e4a4c7db56158a80a18fefdc4fee5acc41b59cde821e691d05541bbb18
=> [stage-1 1/8] FROM docker.io/library/alpine:3.11@sha256:bcae378eacedab83da66079d9366c8f5df542d7ed9ab23bf487e3e1a8481375d
=> [internal] load build context
=> => transferring context: 764.14kB
=> CACHED [builder 2/4] WORKDIR /code
=> [builder 3/4] COPY . .
=> [builder 4/4] RUN CGO_ENABLED=0 GOOS=linux GOARCH=arm64 GO_LDFLAGS= make WHAT=edgemesh-agent
=> CACHED [stage-1 2/8] RUN apk update && apk --no-cache add iptables && apk --no-cache add dpkg
=> [stage-1 3/8] COPY --from=builder /code/_output/local/bin/edgemesh-agent /usr/local/bin/edgemesh-agent
=> [stage-1 4/8] COPY --from=builder /code/pkg/loadbalancer/ /code/pkg/loadbalancer/
=> [stage-1 5/8] COPY --from=builder /code/build/agent/iptables-wrapper /sbin/iptables-wrapper
=> [stage-1 6/8] RUN update-alternatives --install /sbin/iptables iptables /sbin/iptables-legacy 50
=> [stage-1 7/8] RUN update-alternatives --install /sbin/iptables iptables /sbin/iptables-nft 50
=> [stage-1 8/8] RUN update-alternatives --install /sbin/iptables iptables /sbin/iptables-wrapper 100 --slave /sbin/iptables
=> exporting to image
=> => exporting layers
=> => writing image sha256:310989b337d83952d7db043c5f18314c4b9d482b4909956249ea42c2b769a0b5
=> => naming to docker.io/rnarawade/edgemesh-agent:latest

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
docker build --build-arg GO_LDFLAGS='' -t rnarawade/edgemesh-gateway:latest -f build/gateway/Dockerfile .
[+] Building 31.3s (13/13) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 472B
=> [internal] load metadata for docker.io/library/alpine:3.11
=> [internal] load metadata for docker.io/library/golang:1.17
=> [builder 1/4] FROM docker.io/library/golang:1.17@sha256:87262e4a4c7db56158a80a18fefdc4fee5acc41b59cde821e691d05541bbb18
=> [internal] load build context
=> => transferring context: 776.41kB
=> CACHED [stage-1 1/3] FROM docker.io/library/alpine:3.11@sha256:bcae378eacedab83da66079d9366c8f5df542d7ed9ab23bf487e3e1a84813
=> CACHED [builder 2/4] WORKDIR /code
=> CACHED [builder 3/4] COPY . .
=> [builder 4/4] RUN CGO_ENABLED=0 GOOS=linux GOARCH=arm64 GO_LDFLAGS= make WHAT=edgemesh-gateway
=> [stage-1 2/3] COPY --from=builder /code/pkg/loadbalancer/ /code/pkg/loadbalancer/
=> [stage-1 3/3] COPY --from=builder /code/_output/local/bin/edgemesh-gateway /usr/local/bin/edgemesh-gateway
=> exporting to image
=> => exporting layers
=> => writing image sha256:d5b125cbbf24c7222f326b58a74b28b90494be2cc290e2d46ca01c7dbd3800df
=> => naming to docker.io/rnarawade/edgemesh-gateway:latest

What's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
```

Figure 24: Make Images

4.2 Pushing build to docker hub

Once the image creation is successful as shown in 4.1 Execute below commands to push the docker builds for Edgmesh Agent and Gateway individually.

-
- 1 `docker push rnarawade/edgemesh-agent:latest`
 - 2 `docker push rnarawade/edgemesh-gateway:latest`
-

The outcome of these commands is that the newest EdgeMesh builds for agent and gateway gets uploaded to the Docker hub. The successful operation is shown below. The prerequisite for this is that the docker hub application login exists on the local system.

```

rahul@Rahuls-MacBook-Pro edgemesh % docker push rnarawade/edgemesh-gateway:latest
The push refers to repository [docker.io/rnarawade/edgemesh-gateway]
9e7ad5f701ab: Pushed
b96b5750b877: Pushed
1ecc925d1390: Layer already exists
latest: digest: sha256:10d67f868c1b11ab9e5d721fd1a47c0f736716de0ea6915c3a8197768c4e0683 size: 949
rahul@Rahuls-MacBook-Pro edgemesh % docker push rnarawade/edgemesh-agent:latest
The push refers to repository [docker.io/rnarawade/edgemesh-agent]
390ed6a4d20: Pushed
42bc7f9a1a3a: Pushed
7977731ca1e7: Pushed
c7d4642175a5: Pushed
de657aad005c: Pushed
721217d650e5: Pushed
00f084a36e: Layer already exists
1ecc925d1390: Layer already exists
latest: digest: sha256:8ed6262c3082c30884f55348f83ce81292c39e8851ab4ceef1300969ca802d1e size: 1989
rahul@Rahuls-MacBook-Pro edgemesh %
rahul@Rahuls-MacBook-Pro edgemesh %

```

Figure 25: Docker Push Results

Uploaded and downloaded count of these images shown below on Docker Hub.

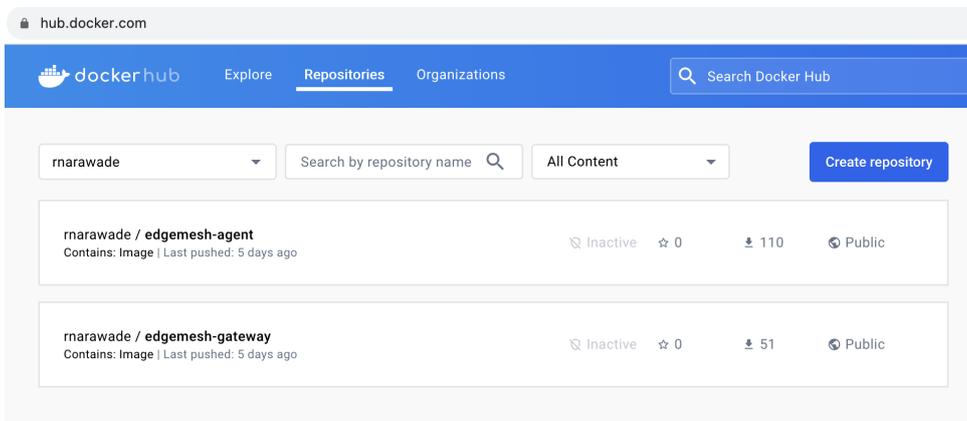


Figure 26: Docker Push Results

4.3 Deployment of EdgeMesh on Master

Section 3.5 3 has explains for the YAML files changes for deployment at the master node. As the start of the deployment process, we need to clean the previous deployments. To clean and apply new changes execute the following commands on the master node after navigating to the /edgemesh directory:

1. **Step 1:** Delete EdgeMesh Agnet

```
1 kubectl delete -f build/agent/resources/05-daemonset.yaml
```

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl delete -f build/agent/resources/05-daemonset.yaml
daemonset.apps "edgemesh-agent" deleted
root@master:~/edgemesh#

```

Figure 27: Delete Agent

2. **Step 2:** Delete EdgeMesh Gateway:

```
1 kubectl delete -f build/gateway/resources/05-deployment.yaml
```

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl delete -f build/gateway/resources/05-deployment.yaml
deployment.apps "edgemesh-gateway" deleted
root@master:~/edgemesh#

```

Figure 28: delete Gateway

3. Step 3: Delete Application:

```

1 kubectl delete -f examples/hostname-lb-random-gateway.yaml

```

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl delete -f examples/hostname-lb-random-gateway.yaml
deployment.apps "hostname-lb-edge" deleted
service "hostname-lb-svc" deleted
gateway.networking.istio.io "edgemesh-gateway" deleted
destinationrule.networking.istio.io "hostname-lb-svc" deleted
virtualservice.networking.istio.io "edgemesh-gateway-svc" deleted
root@master:~/edgemesh#

```

Figure 29: Delete test application

4. Step 4: Apply new build for Agent

```

1 kubectl apply -f build/agent/resources/
2 kubectl apply -f build/gateway/resources/
3 kubectl apply -f examples/hostname-lb-random-gateway.yaml

```

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl apply -f build/agent/resources/
serviceaccount/edgemesh-agent unchanged
clusterrole.rbac.authorization.k8s.io/edgemesh-agent unchanged
clusterrolebinding.rbac.authorization.k8s.io/edgemesh-agent unchanged
configmap/edgemesh-agent-cfg unchanged
configmap/edgemesh-agent-psk unchanged
daemonset.apps/edgemesh-agent created
root@master:~/edgemesh#

```

Figure 30: Deploy new agent

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl apply -f build/gateway/resources/
serviceaccount/edgemesh-gateway unchanged
clusterrole.rbac.authorization.k8s.io/edgemesh-gateway unchanged
clusterrolebinding.rbac.authorization.k8s.io/edgemesh-gateway unchanged
configmap/edgemesh-gateway-cfg unchanged
configmap/edgemesh-gateway-psk unchanged
deployment.apps/edgemesh-gateway created
root@master:~/edgemesh#

```

Figure 31: Deploy New Gateway

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl apply -f examples/hostname-lb-random-gateway.yaml
deployment.apps/hostname-lb-edge created
service/hostname-lb-svc created
gateway.networking.istio.io/edgemesh-gateway created
destinationrule.networking.istio.io/hostname-lb-svc created
virtualservice.networking.istio.io/edgemesh-gateway-svc created
root@master:~/edgemesh#

```

Figure 32: Deploy Test application again

5. Step 5: Get information about the pods in all namespaces:

```

1 kubectl get pods --all-namespaces -o wide

```

```

root@master:~/edgemesh#
root@master:~/edgemesh# kubectl get pods --all-namespaces -o wide
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE   IP              NODE
kube-system  calico-kube-controllers-5bcd7db644-d2pwn 1/1     Running   0           60m   192.168.219.66 master
kube-system  calico-node-dfbrl                          0/1     Init:Error 16          43m   192.168.67.4   nodetwo
kube-system  calico-node-l4f6j                          0/1     Init:Error 15          46m   192.168.67.3   nodeone
kube-system  calico-node-lkpjj                          1/1     Running   0           60m   192.168.67.2   master
kube-system  calico-node-s4v7b                          0/1     Init:Error 17          39m   192.168.67.5   nodethree
kube-system  coredns-558bd4d5db-2tfff                   1/1     Running   0           61m   192.168.219.67 master
kube-system  coredns-558bd4d5db-jxz4k                   1/1     Running   0           61m   192.168.219.65 master
kube-system  etcd-master                                1/1     Running   0           62m   192.168.67.2   master
kube-system  kube-apiserver-master                       1/1     Running   0           62m   192.168.67.2   master
kube-system  kube-controller-manager-master             1/1     Running   0           62m   192.168.67.2   master
kube-system  kube-proxy-75dwf                           1/1     Running   0           61m   192.168.67.2   master
kube-system  kube-proxy-gm768                           1/1     Running   0           43m   192.168.67.4   nodetwo
kube-system  kube-proxy-p9sqz                            1/1     Running   0           39m   192.168.67.5   nodethree
kube-system  kube-proxy-xpr74                            1/1     Running   0           46m   192.168.67.3   nodeone
kube-system  kube-scheduler-master                      1/1     Running   0           62m   192.168.67.2   master
kubedge     cloudcore-5876c76687-mm6m9                1/1     Running   0           58m   192.168.67.2   master
kubedge     edgemesh-agent-7gmrq                       1/1     Running   0           7m19s  192.168.67.2   master
kubedge     edgemesh-agent-sfwkx                       1/1     Running   0           7m19s  192.168.67.4   nodetwo
kubedge     edgemesh-agent-ttqzm                       1/1     Running   0           7m19s  192.168.67.5   nodethree
kubedge     edgemesh-agent-vspgx                       1/1     Running   0           7m19s  192.168.67.3   nodeone
kubedge     edgemesh-gateway-6d477479f6-dxwjp         1/1     Running   0           6m32s  192.168.67.2   master
kubedge     hostname-lb-edge-5cdf5c758c-26bd7         1/1     Running   0           27s   172.17.0.3     nodeone
kubedge     hostname-lb-edge-5cdf5c758c-58j2w         1/1     Running   0           27s   172.17.0.4     nodetwo
kubedge     hostname-lb-edge-5cdf5c758c-5rs9v         1/1     Running   0           27s   172.17.0.4     nodeone
kubedge     hostname-lb-edge-5cdf5c758c-8lxwv         1/1     Running   0           27s   172.17.0.3     nodethree
kubedge     hostname-lb-edge-5cdf5c758c-9n6ff         1/1     Running   0           27s   172.17.0.3     nodetwo
kubedge     hostname-lb-edge-5cdf5c758c-rqkmm         1/1     Running   0           27s   172.17.0.4     nodethree
root@master:~/edgemesh#
root@master:~/edgemesh#

```

Figure 33: All pods after deployment

```

root@master:~/edgemesh#
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-26bd7
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-8lxwv
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-26bd7
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-58j2w
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-58j2w
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-8lxwv
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-8lxwv
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-5rs9v
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-5rs9v
root@master:~/edgemesh#

```

Figure 34: Verify test application works

6. **Step 6:** Perform a curl command to test the deployment:

```
1 curl 192.168.67.2:23333
```

5 Evaluation Setup

This section describes the steps to test the evaluation.

1. **Step 1:** Test if the URL is accessible outside of the VMs.

```
rahul@Rahuls-MacBook-Pro MSc-Cloud %  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-9n6ff  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-8lxww  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-5rs9v  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-8lxww  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-8lxww  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-5rs9v  
rahul@Rahuls-MacBook-Pro MSc-Cloud % curl 192.168.67.2:23333  
hostname-lb-edge-5cdf5c758c-58j2w  
rahul@Rahuls-MacBook-Pro MSc-Cloud %
```

Figure 35: Verify test application works from Local system

2. **Step 2:** Install the Hey Tool using the following command. Dogan (2023)

```
1 brew install hey
```

```
rahul@Rahuls-MacBook-Pro MSc-Cloud % brew install hey  
Running 'brew update --auto-update' ...  
-> Downloading https://ghcr.io/v2/homebrew/portable-ruby/portable-ruby/blobs/sha256:d783cbe66e6f0d71c0b442317b54554370decdfac66b72d4938c07a63f67be  
##### 100.0%  
-> Pairing portable-ruby-3.1.4.arm64_big_sur.bottle.tar.gz  
-> Auto-updated Homebrew!  
Updated 2 taps (homebrew/core and homebrew/cask).  
-> New Formulae  
action-validator          memray                    python-configobj          python-platformdirs      scarb  
amass                     minder                   python-cycler             python-pluggy            shell2http  
anstyle@0                 ocean4                   python-detaiil            python-ply                 shellspec  
argc                       open-slmh                python-dicttoxml          python-prompt-toolkit     skate  
asitop                     oslo                      python-distlib            python-regex               sloth  
awscli-local              patat                    python-distro             python-requests           snakeviz  
cherrybomb                pdfalyzer                python-hatch-fancy-pypl-readme  python-requests-oauthlib  solo2-cli  
clidr                     pdfrip                   python-hatch-vcs          python-rich                 spicetify-cli  
dal-fax                    pyweb2                   python-hatchling          python-s3transfer          spidermonkey@91  
doppler                   python-abeel             python-ida                 python-setuptools-scm     sqflite  
dragon                     python-anytree            python-jmespath            python-termcolor           squealer  
flyscrape                  python-ansicrypto         python-json5               python-trove-classifiers  telegram-downloader  
gdrlv@2                    python-attrs              python-kiwisolver          python-urllib3             terraform-local  
geop2fast                  python-boto3               python-magic                python-wcwidth              texttest  
glbinding@2                python-botozone            python-markdown-it-py      python-wssocket-client    virtual  
goreplay                   python-brotli              python-metplotlib          python-xlsxwriter           wifness  
instaloader                 python-cachetools          python-mdurl                gbitorrent-cli            xname  
kew                         python-charset             python-msgpack              rapidfuzz-cpp              yotas  
libconfini                  python-charset-normalizer  python-oauthlib             rdap  
libcyaml                    python-clipboard             python-openapi3              retire  
libpnp                      python-colorama             python-pathspec              retry  
nortoad@11.1                python-configanparse         python-pbr                   richgo  
-> New Casks  
ame                         effect-house                mdb-accdb-viewer           october                    timemachinestatus  
anka-build-cloud-controller  focusrite-control-2         mediamate                  ok-json                     truhu  
anka-build-cloud-registry    greenery                    mindmac                     avito                       tunetag  
bezel                        hapigo                       navigraph-charts            ovito-pro                   vivalc  
brightintosh                  hides                        navigraph-stmlink           proton-drive                  wave  
cardo-update                   hoppscotch                  notes-better                 screens-assist               wiso-steuer-2024  
codewhisperer                 macgpt                       rx-studio                   senabluetoothdevicemanager  xlyff-editor  
  
You have 12 outdated formulae installed.  
  
Warning: Treating hey as a formula. For the cask, use homebrew/cask/hey  
-> Downloading https://ghcr.io/v2/homebrew/core/hey/manifests/0.1.4  
Already downloaded: /Users/rahul/Library/Caches/Homebrew/downloads/a221aed3448d6cf785362d2b0f0d811df473218ff1f5b79ef4e68b1b20b4298bf--hey-0.1.4.bottle.manifest.json  
-> Fetching hey  
-> Downloading https://ghcr.io/v2/homebrew/core/hey/blobs/sha256:7240c725e9276fc1ea1a59f748219c34b26f36fa29964cc676ebd459b92ca  
Already downloaded: /Users/rahul/Library/Caches/Homebrew/downloads/f8604c83a7954080c1dea597e63931e4ef2a800c2fa2d99a085d55f0e3d57e--hey-0.1.4.arm64_sonoma.bottle.tar.gz  
-> Pairing hey--0.1.4.arm64_sonoma.bottle.tar.gz  
D /opt/homebrew/Cellar/hey/0.1.4: 5 files, 9.2MB  
-> Running 'brew cleanup hey' ...  
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.  
Hide these hints with HOMEBREW_NO_ENV_HINTS (see 'man brew').  
rahul@Rahuls-MacBook-Pro MSc-Cloud %
```

Figure 36: Install Hey tool on local system

3. **Step 3:** Run a sample test using the Hey Tool with the following command:

```
1 hey -n 200 -c 100 http://192.168.67.2:23333/
```

```
rahul@Rahuls-MacBook-Pro MSc-Cloud %  
rahul@Rahuls-MacBook-Pro MSc-Cloud %  
rahul@Rahuls-MacBook-Pro MSc-Cloud % hey -n 200 -c 100 http://192.168.67.2:23333/  
  
Summary:  
Total:      2.9687 secs  
Slowest:    1.5413 secs  
Fastest:    0.0015 secs  
Average:    1.1043 secs  
Requests/sec: 67.3686  
  
Total data: 6800 bytes  
Size/request: 34 bytes  
  
Response time histogram:  
0.002 [1] |  
0.156 [11] |  
0.309 [10] |  
0.463 [9] |  
0.617 [10] |  
0.771 [10] |  
0.925 [9] |  
1.079 [11] |  
1.233 [10] |  
1.387 [11] |  
1.541 [108] |  
  
Latency distribution:  
10% in 0.2842 secs  
25% in 0.7553 secs  
50% in 1.4184 secs  
75% in 1.4551 secs  
90% in 1.5099 secs  
95% in 1.5188 secs  
99% in 1.5239 secs  
  
Details (average, fastest, slowest):  
DNS+diialup: 0.0024 secs, 0.0015 secs, 1.5413 secs  
DNS-lookup: 0.0000 secs, 0.0000 secs, 0.0000 secs  
req write: 0.0001 secs, 0.0000 secs, 0.0039 secs  
resp wait: 1.1017 secs, 0.0014 secs, 1.5343 secs  
resp read: 0.0001 secs, 0.0000 secs, 0.0012 secs  
  
Status code distribution:  
[200] 200 responses  
  
rahul@Rahuls-MacBook-Pro MSc-Cloud %
```

Figure 37: Sample Hey tool test

References

- Docker Documentation (2023). Install docker desktop on mac, <https://docs.docker.com/desktop/install/mac-install/>. Accessed: December 11, 2023.
- Dogan, J. B. (2023). Hey: Http load testing and benchmarking tool, <https://github.com/rakyll/hey>. Accessed: December 11, 2023.
- Downloading Package* (2023). <https://git-scm.com/download/mac>. Accessed: December 11, 2023.
- EdgeMesh, P. (2023). Edgemes documentation: Edge gateway. Accessed: December 11, 2023.
URL: <https://edgemes.netlify.app/guide/edge-gateway.html>
- Gaponcic, D. (2023). Getting started with kubeedge on virtual machines, *Medium* . Accessed: December 11, 2023.
- Kim, S. H. and Kim, T. (2023). Local scheduling in kubeedge-based edge computing environment, *Sensors* **23**(3): 1522.
- KubeEdge* (2023a). <https://kubeedge.io/>. Accessed: December 11, 2023.
- KubeEdge, P. (2023b). Edgemes: A high-performance and light-weight edge computing mesh, <https://github.com/kubeedge/edgemes>. Accessed: December 11, 2023.
- KubeEdgeGit, P. (2023). Kubeedge: Kubernetes native edge computing framework, <https://github.com/kubeedge/kubeedge>. Accessed: December 11, 2023.
- Kubernetes Documentation* (2023). <https://kubernetes.io/docs/home/>. Accessed: December 11, 2023.
- Nair, A. B. (2023). Edge computing in kubernetes using kubeedge, *Medium* . Accessed: December 11, 2023.
- The Go Programming Language* (2023). <https://go.dev/>. Accessed: December 11, 2023.
- Ubuntu (2023). Download ubuntu server for arm. Accessed: [Insert Access Date Here].
URL: <https://ubuntu.com/download/server/arm>
- UTM* (2023). <https://mac.getutm.app/>. Accessed: December 11, 2023.
- Visual Studio Code (2023). Download visual studio code - mac, linux, windows, <https://code.visualstudio.com/download>. Accessed: December 11, 2023.