National College of Ireland

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).	☐
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

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 Sanoma 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)

KubEdge: Information on KubEdge 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.

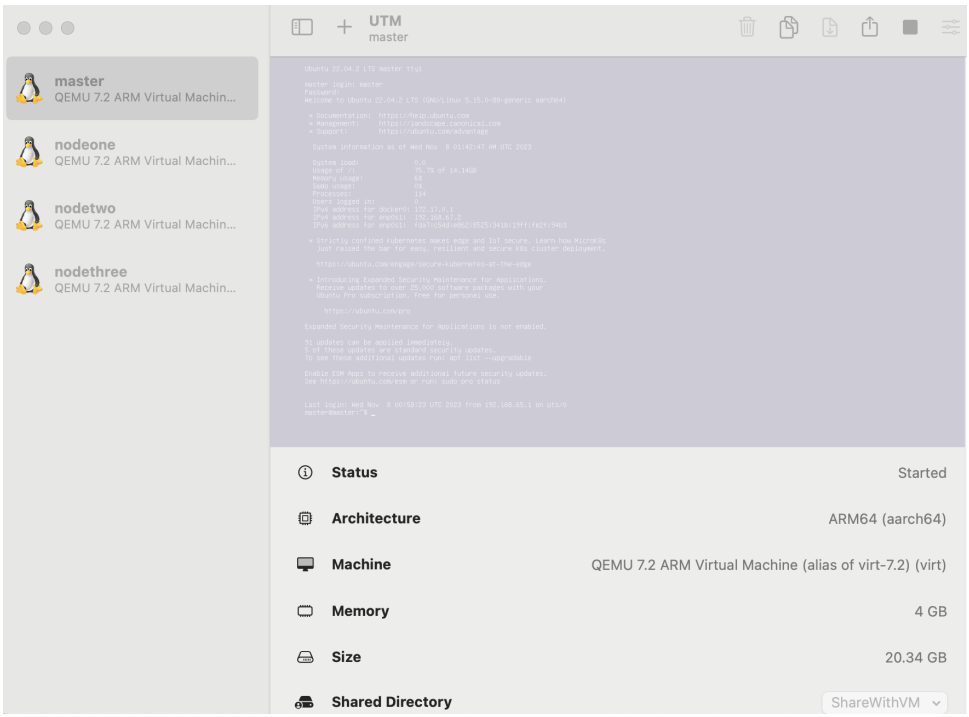

Figure 1: UTM Window

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:

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

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

```
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 KubdEdge can be downloaded and installed. For research purposes, the repo was downloaded, so GO is required to build.

```
    sudo apt install golang-go
export GOOS=linux
export GOARCH=arm64
source ~/.bashrc
export PATH=$PATH:/snap/bin:/usr/go/bin
export GOPATH=/usr/go
export GOBIN=$GOPATH/bin
export PATH=$PATH:$GOBIN:$GOROOT/bin
```

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

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

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

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

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

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

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

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

```
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:

```
kubectl label services kubernetes service.edgemesh.kubeedge.io/service-proxy-name=""
git clone https://github.com/kubeedge/edgemesh.git
cd edgemesh
kubectl apply -f build/crds/istio/
kubectl apply -f build/agent/resources/
kubectl get nodes --all-namespaces
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:

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

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

```
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-dfbrl                          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
kubeedge      cloudcore-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:

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

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

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

```
sudo apt install golang-go
export GOOS=linux
export GOARCH=arm64
source ~/.bashrc
export PATH=$PATH:/snap/bin:/usr/go/bin
export GOPATH=/usr/go
export GOBIN=$GOPATH/bin
export PATH=$PATH:$GOBIN:$GOROOT/bin
```

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

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

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

```
cd /root
$GOPATH/src/github.com/kubeedge/kubeedge
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:

```
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

.

Figure 6: EdgeMesh YAML change 3

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

```
systemctl restart edgecore.service
systemctl status edgecore.service
```

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

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

2.4 Gateway Setup at Master node

Run the below commands at Master node.

```
cd edgemesh
kubectl apply -f build/crds/istio/
kubectl apply -f build/agent/resources/
kubectl apply -f build/gateway/resources
kubectl get all -n kubeedge -o wide
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:

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

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

Figure 7: Sample App Host Modified

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

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

```
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 8: Deploy Sample App

3. **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:

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

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

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

Figure 9: Test Sample App URL

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

```
root@master:~/edgemesh# 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-bkxqx                   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-mpdmn                           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
kubeedge      cloudcore-5876c76687-4jwld                 1/1     Running   0          6h24m   192.168.67.2     master
kubeedge      edgemesh-agent-chp94                       1/1     Running   0          37h     192.168.67.4     nodetwo
kubeedge      edgemesh-agent-q5csz                       1/1     Running   0          37h     192.168.67.3     nodeone
kubeedge      edgemesh-agent-q7n99                       1/1     Running   0          37h     192.168.67.5     nodethree
kubeedge      edgemesh-agent-zdtxv                       1/1     Running   0          37h     192.168.67.2     master
kubeedge      edgemesh-gateway-6d477479f6-mvhhw          1/1     Running   0          37h     192.168.67.2     master
kubeedge      hostname-lb-edge-5cdf5c758c-5rwxg          1/1     Running   0          37h     172.17.0.4       nodethree
kubeedge      hostname-lb-edge-5cdf5c758c-6z8vp          1/1     Running   0          37h     172.17.0.3       nodeone
kubeedge      hostname-lb-edge-5cdf5c758c-9vvjn          1/1     Running   0          37h     172.17.0.4       nodetwo
kubeedge      hostname-lb-edge-5cdf5c758c-llxkp          1/1     Running   0          37h     172.17.0.4       nodeone
kubeedge      hostname-lb-edge-5cdf5c758c-tsxg4          1/1     Running   0          37h     172.17.0.3       nodethree
kubeedge      hostname-lb-edge-5cdf5c758c-x648b          1/1     Running   0          37h     172.17.0.3       nodetwo
root@master:~/edgemesh#
root@master:~/edgemesh# curl http://192.168.67.2:23333/
hostname-lb-edge-5cdf5c758c-5rwxg
root@master:~/edgemesh# curl http://192.168.67.2:23333/
hostname-lb-edge-5cdf5c758c-9vvjn
root@master:~/edgemesh#
```

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:

```
git clone https://github.com/kubeedge/edgemesh
```

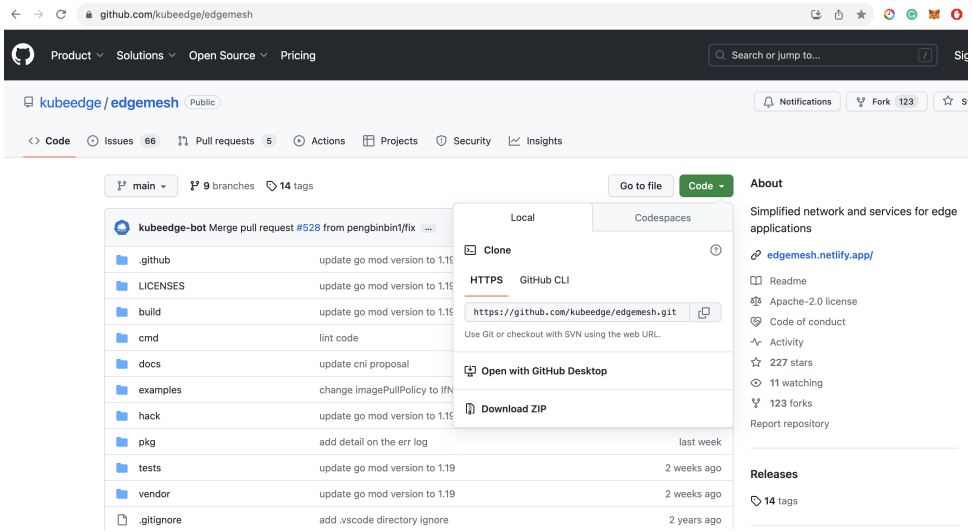

Figure 11: Git Repo

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

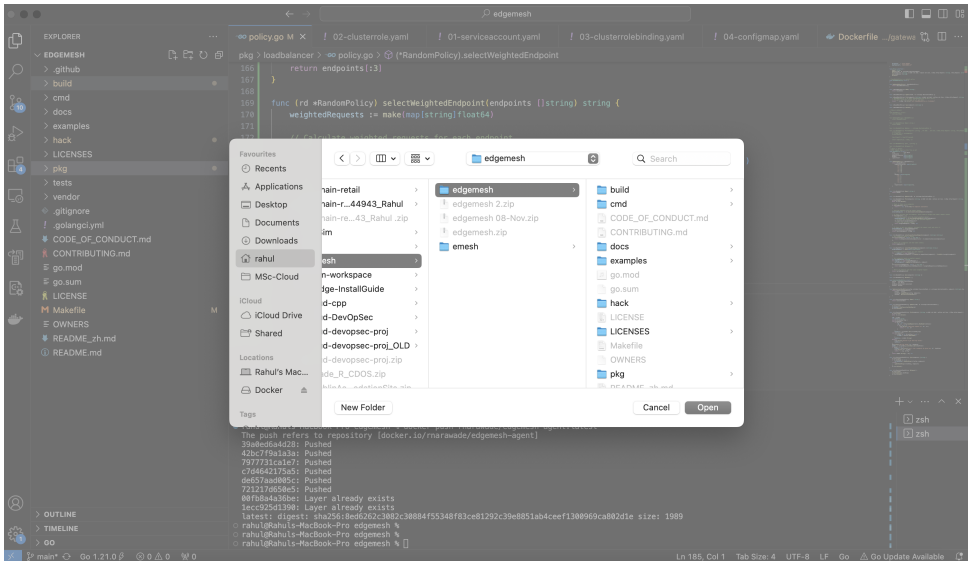

Figure 12: Open Repo

3.1 Load balancer policy Implementation

Open file `/github/EdgeMesh/edgemesh/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.

```go
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

```go
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.

```go
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` 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

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/edgemesh/build/agent/resources/05-daemonset.yaml` at both master and local as below:

Figure 22: YAML changes for Agent

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

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.

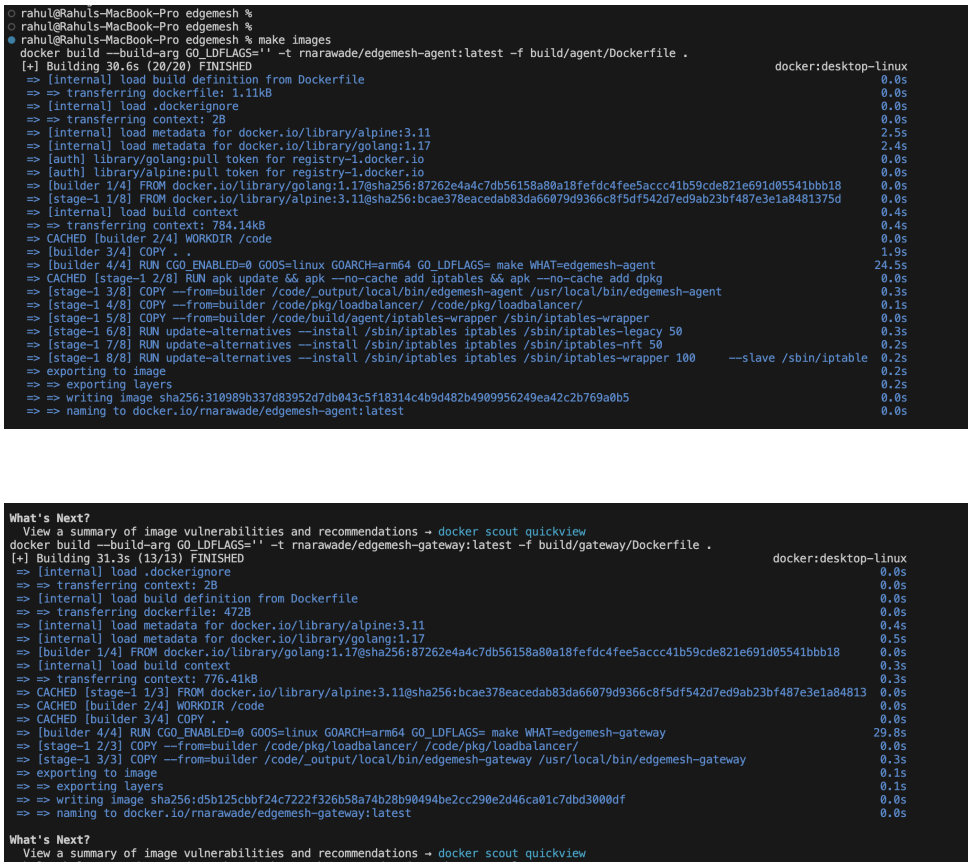

Figure 24: Make Images

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.

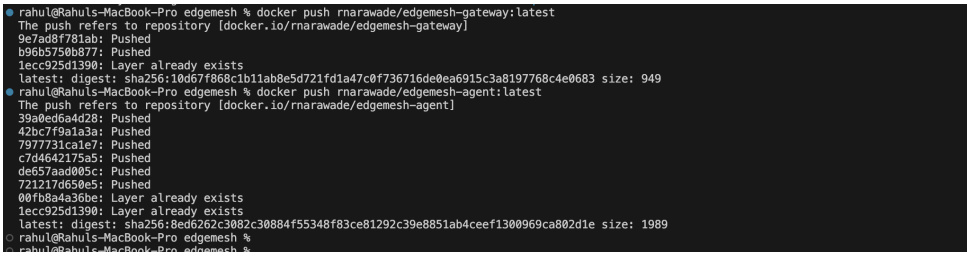

Figure 25: Docker Push Results

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

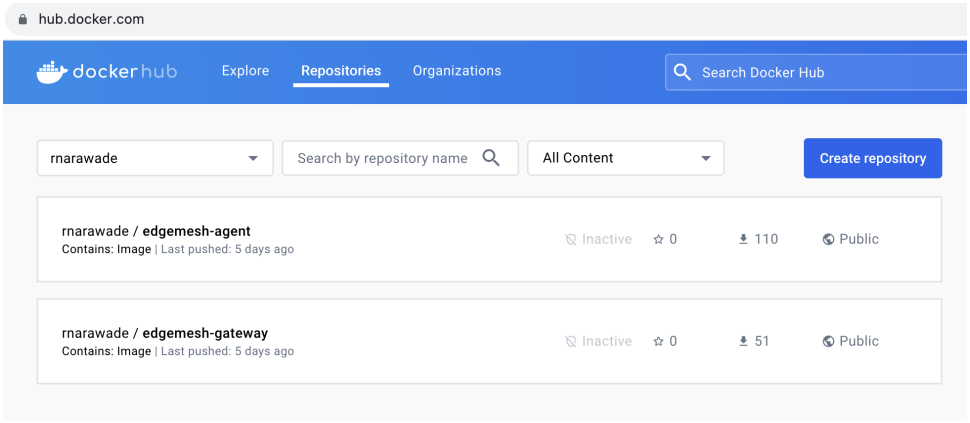

Figure 26: Docker Push Results

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

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

Figure 27: Delete Agent

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

```
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: deleet Gateway

3. **Step 3:** Delete Application:

```
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: Deleet test application

4. **Step 4:** Apply new build for Agent

```
kubectl apply -f build/agent/resources/
kubectl apply -f build/gateway/resources/
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:

```
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
kubeedge      cloudcore-5876c76687-mm6m9                 1/1     Running      0          58m     192.168.67.2     master
kubeedge      edgemesh-agent-7gmrq                       1/1     Running      0          7m19s   192.168.67.2     master
kubeedge      edgemesh-agent-sfwkx                       1/1     Running      0          7m19s   192.168.67.4     nodetwo
kubeedge      edgemesh-agent-ttgzm                       1/1     Running      0          7m19s   192.168.67.5     nodethree
kubeedge      edgemesh-agent-vspxg                       1/1     Running      0          7m19s   192.168.67.3     nodeone
kubeedge      edgemesh-gateway-6d477479f6-dxwjp          1/1     Running      0          6m32s   192.168.67.2     master
kubeedge      hostname-lb-edge-5cdf5c758c-26bd7          1/1     Running      0          27s     172.17.0.3       nodeone
kubeedge      hostname-lb-edge-5cdf5c758c-58j2w          1/1     Running      0          27s     172.17.0.4       nodetwo
kubeedge      hostname-lb-edge-5cdf5c758c-5rs9v          1/1     Running      0          27s     172.17.0.4       nodeone
kubeedge      hostname-lb-edge-5cdf5c758c-8lxvw          1/1     Running      0          27s     172.17.0.3       nodethree
kubeedge      hostname-lb-edge-5cdf5c758c-9n6ff          1/1     Running      0          27s     172.17.0.3       nodetwo
kubeedge      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-8lxvw
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-8lxvw
root@master:~/edgemesh# curl 192.168.67.2:23333
hostname-lb-edge-5cdf5c758c-8lxvw
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:

```
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.

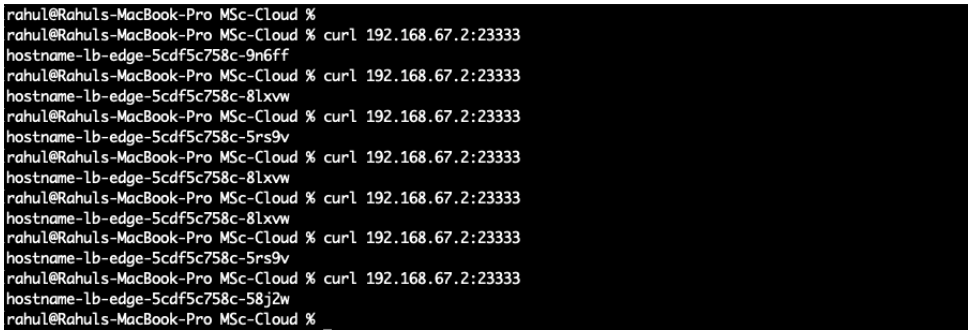

Figure 35: Verify test application works from Local system

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

```
brew install hey
```

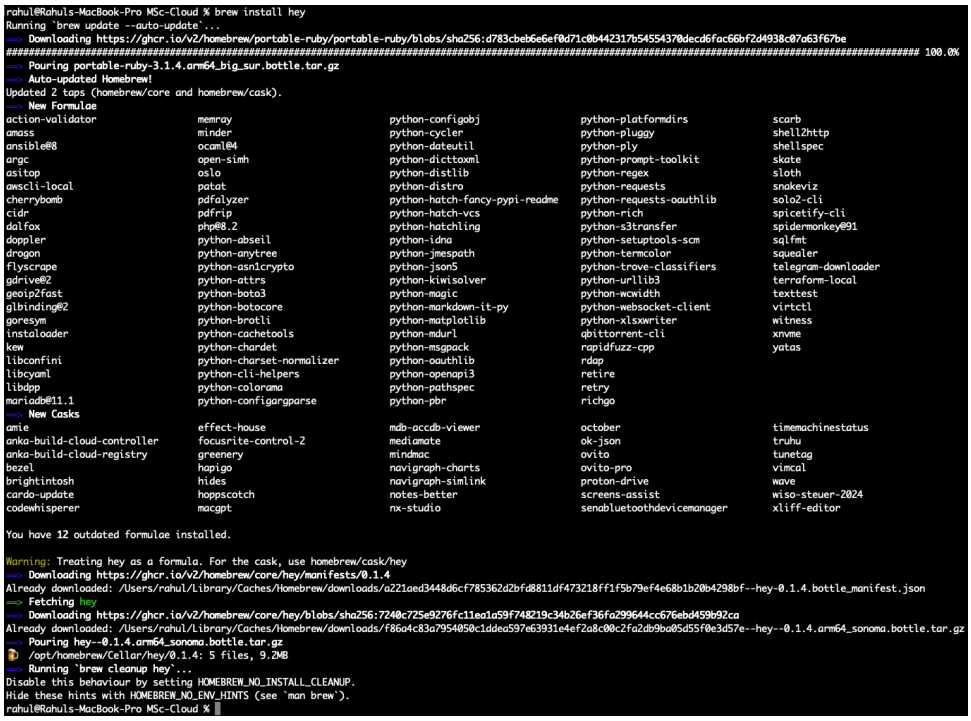

Figure 36: Install Hey tool on local system

.

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

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

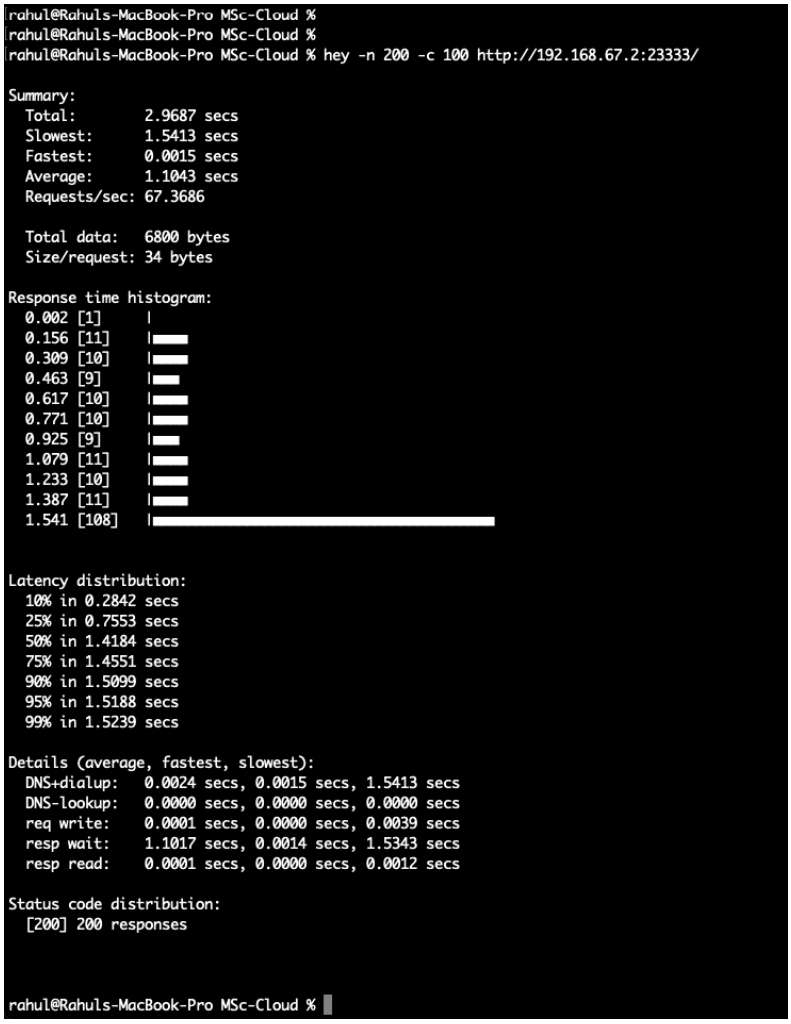

Figure 37: Sample Hey tool test

.

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). Edgemesh documentation: Edge gateway. Accessed: December 11, 2023.
URL: *https://edgemesh.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). Edgemesh: A high-performance and light-weight edge computing mesh, `https://github.com/kubeedge/edgemesh`. 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.