EMQX Setup and Configurations

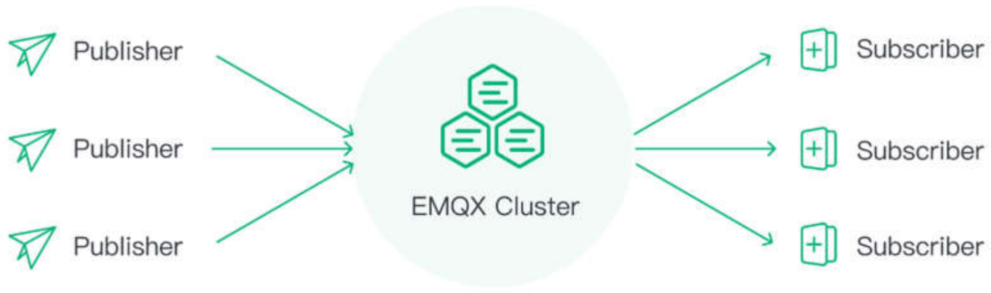


Figure: EMQX Basic Architecture

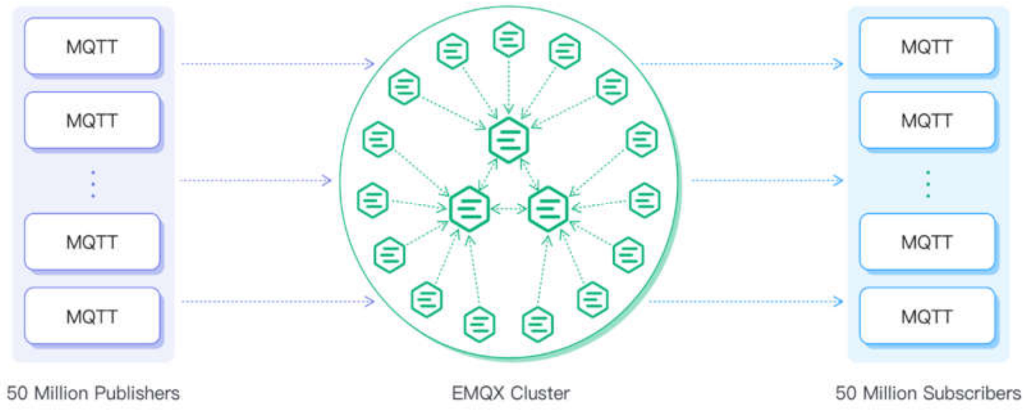


Figure: EMQX Cluster Architecture

BY: SAGAR MALLA
MARCH 21, 2024
V1.1

Table of Contents

EMQX Setup and Configurations....................i

A. Introductions....................1

1. Basic Setup....................2

2. Node Setup....................2

    2.1 Install with YUM Source. *(Same for all Nodes)*....................2

    2.3 Check EMQX Web Dashboard....................2

3. Cluster Setup....................4

    3.1 Cluster Creating/Joining....................5

3. Load Balancing by HAProxy....................6

    3.1 Installation of HAProxy....................6

    3.2 Configure HAProxy as LB....................6

4. MQTTX Load testing....................8

5. Load Testing via golan (Download File: LINK)....................10

    5.1 Creating loadtest.go file....................10

    (replace with your HaProxy /Node1 IP:....................10

    5.2 Go Inatallation:....................11

    5.3 Load testing....................11

6. Uninstallation of EMQX....................13

7. Using Docker Container....................14

8. Using Docker Compose....................15

9. Performence Tunning....................17

A. Introductions

MQTT is the most commonly used messaging protocol for the Internet of Things (IoT). MQTT stands for MQ Telemetry Transport. The protocol is a set of rules that defines how IoT devices can publish and subscribe to data over the Internet. MQTT is used for messaging and data exchange between IoT and industrial IoT (IIoT) devices, such as embedded devices, sensors, industrial PLCs, etc. The protocol is event driven and connects devices using the publish /subscribe (Pub/Sub) pattern. The sender (Publisher) and the receiver (Subscriber) communicate via Topics and are decoupled from each other. The connection between them is handled by the MQTT broker. The MQTT broker filters all incoming messages and distributes them correctly to the Subscribers.

- It requires minimal resources since it is **lightweight and efficient**
- Supports **bi-directional** messaging between device and cloud
- Can **scale to millions** of connected devices
- Supports **reliable message** delivery through 3 QoS levels
- Works well over **unreliable networks**
- **Security enabled**, so it works with TLS and common authentication protocols

Read More About MQTT: VideoLink | LINK2

EMQX is a cloud-native, MQTT-based, IoT messaging platform designed for high reliability and massive scale. EMQX is a tool in the Message Queue category of a tech stack.

EMQX is currently the most scalable MQTT broker for IoT applications. It processes millions of MQTT messages in a second with sub-millisecond latency and allows messaging among more than 100 million clients within a single cluster. EMQX is compliant with MQTT 5.0 and 3.x. It's ideal for distributed IoT networks and can run on the cloud, Microsoft Azure, Amazon Web Services, and Google Cloud. The broker can implement MQTT over TLS/SSL and supports several authentication mechanisms like PSK, JWT, and X.509. Unlike Mosquitto, EMQX supports clustering via CLI, HTTP API, and a Dashboard.

Read More About EMQX: LINK

1. Basic Setup

```
systemctl stop firewalld
systemctl disable firewalld
vi /etc/selinux/config
      SELINUX=disabled

# For RHEL Subscriptions
sudo subscription-manager register --username='dummyUserName' --password='dummyPassword'
subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms

dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
dnf update -y
yum install -y telnet curl wget net-tools
sudo yum install -y epel-release
yum clean all -y
reboot
```

2. Node Setup

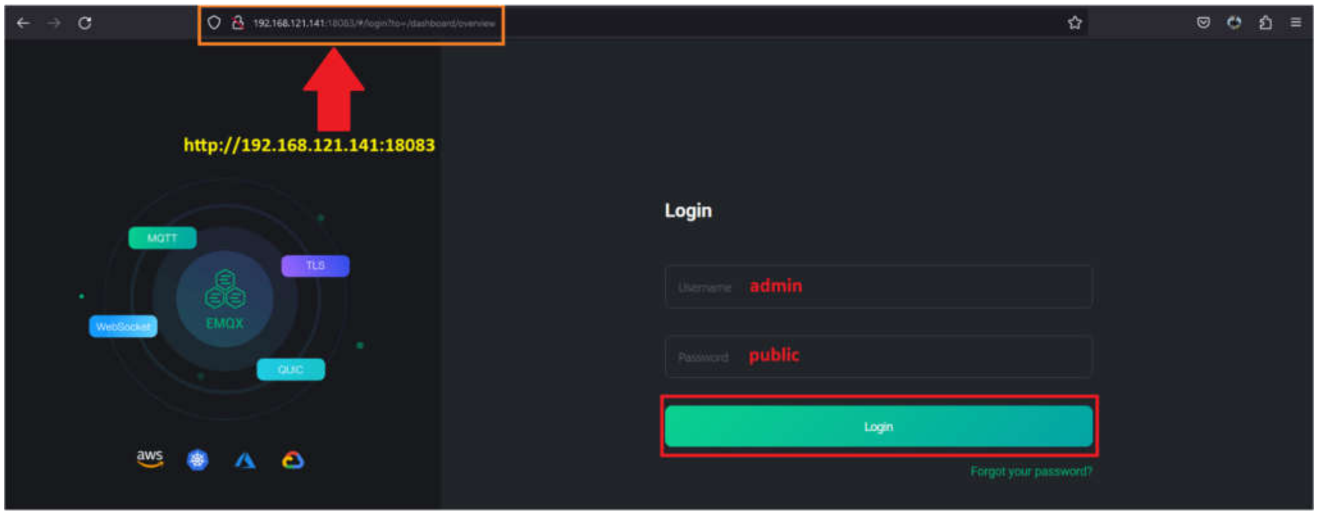
2.1 Install with YUM Source. *(Same for all Nodes)*

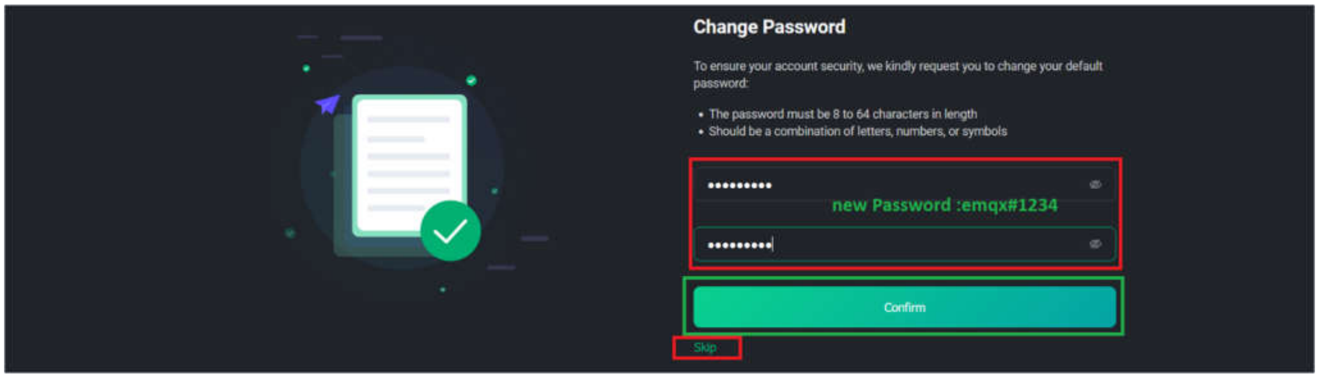
```
curl -s https://assets.emqx.com/scripts/install-emqx-rpm.sh | sudo bash
yum install epel-release -y ;
yum install -y openssl11 openssl11-devel ;
sudo yum install emqx -y;

# Start and Status Check
sudo systemctl start emqx
systemctl status emqx

#Port Check
ss -ltn
netstat -tupln | grep emqx

# Reload Daemon and restart emqx after config change
systemctl daemon-reload
systemctl restart emqx

# Admin/Users password reset (not for first-time setup)
emqx ctl admins passwd <Username> <NewPassword>
e.g.: emqx ctl admins passwd admin NewPassword#1234
```

Read More : LINK | LINK2

2.3 Check EMQX Web Dashboard

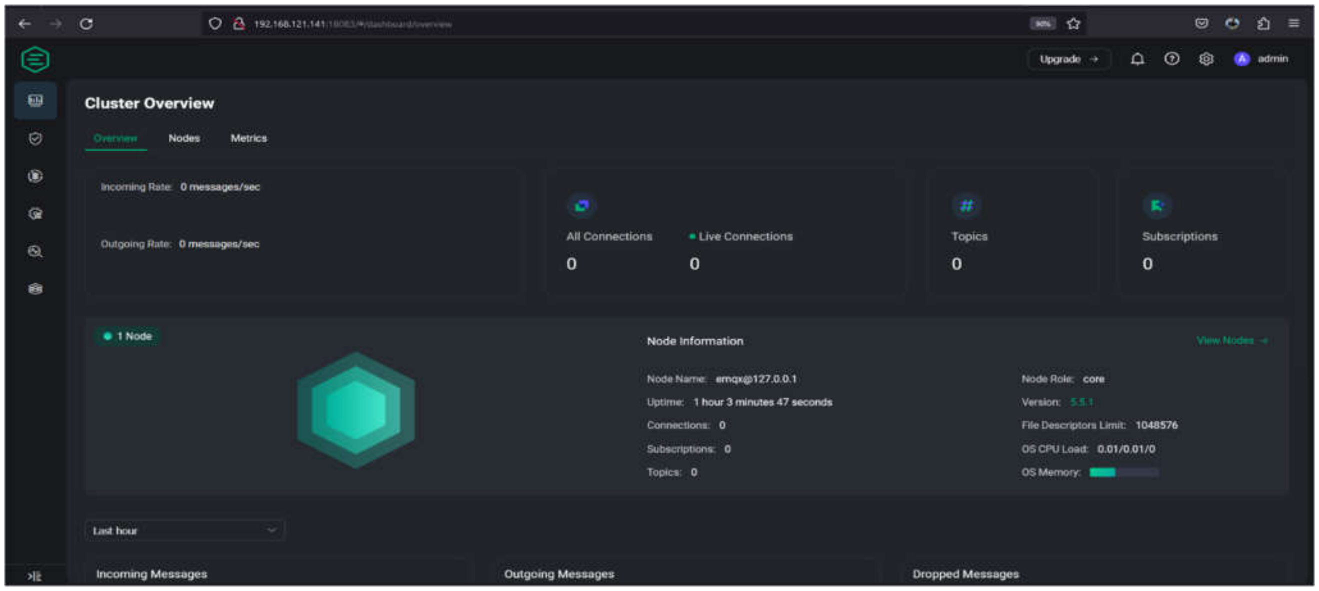
Web Dashboard: http://IP:18083/
Default-login:
  Default-username: admin
  Default-password: public
  new password: emqx#1234





To ensure your account security, we kindly request you to change your default password:

- The password must be 8 to 64 characters in length
- Should be a combination of letters, numbers, or symbols



Incoming Rate: 0 messages/sec

Outgoing Rate: 0 messages/sec

All Connections 0

Live Connections 0

Topics 0

Subscriptions 0

1 Node

Node Information

View Nodes →

Node Name: emqx@127.0.0.1

Uptime: 1 hour 3 minutes 47 seconds

Connections: 0

Subscriptions: 0

Topics: 0

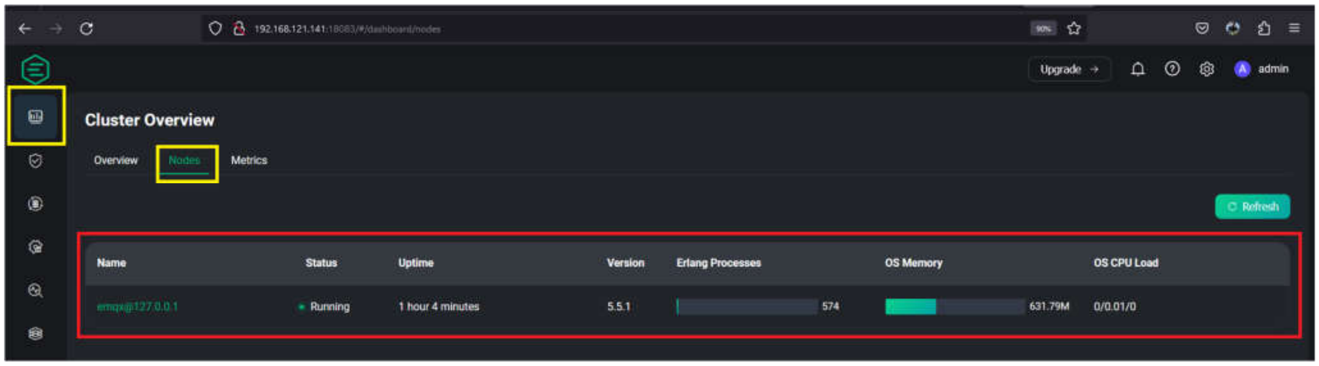
Node Role: core

Version: 5.5.1

File Descriptors Limit: 1048576

OS CPU Load: 0.01/0.01/0

OS Memory:



Name	Status	Uptime	Version	Erlang Processes	OS Memory	OS CPU Load
emqx@127.0.0.1	Running	1 hour 4 minutes	5.5.1	574	631.79M	0/0.01/0

3. Cluster Setup

```
Node1 – Specs
OS: RHEL/RockeyLinux 9.3
RAM: minimum 8GB
CPU Core: minimum 4 Cores
Storage: minimum 30GB
IP : 192.168.121.141
Hostname: node1.emqx.com
```

```
Node1 – Specs
OS: RHEL/RockeyLinux 9.3
RAM: minimum 8GB
CPU Core: minimum 4 Cores
Storage: minimum 30GB
IP : 192.168.121.142
Hostname: node2.emqx.com
```

```
Node1 – Specs
OS: RHEL/RockeyLinux 9.3
RAM: minimum 8GB
CPU Core: minimum 4 Cores
Storage: minimum 30GB
IP : 192.168.121.143
Hostname: node3.emqx.com
```

EMQX installation same for all three nodes, the configuration for the all theree nodes are as follows,
Config file path: `/etc/emqx/emqx.conf`

Note: just **change** the **node name** of each and **cookie** must be **same** for **all node**, because this cookie will be used later cluster formation (Joining).
Node name: *emqx@IP_Address* or *emqx@validDomainName* for e.g. *emqx@192.168.1.10* or *emqx@this.is.mytestdomain.com*

```
# Node1 configurations
# vi /etc/emqx/emqx.conf
node {
  name = "emqx@192.168.121.141"
  cookie = "emqxsecretcookie"
  data_dir = "/var/lib/emqx"
}
cluster {
  name = emqxcl
  discovery_strategy = manual
}
dashboard {
    listeners.http {
        bind = 18083
    }
}
```

```
# Node2 configurations
# vi /etc/emqx/emqx.conf
node {
  name = "emqx@192.168.121.142"
  cookie = "emqxsecretcookie"
  data_dir = "/var/lib/emqx"
}
cluster {
  name = emqxcl
  discovery_strategy = manual
}
dashboard {
    listeners.http {
        bind = 18083
    }
}
```

```
# Node3 configurations
# vi /etc/emqx/emqx.conf
node {
  name = "emqx@192.168.121.143"
  cookie = "emqxsecretcookie"
  data_dir = "/var/lib/emqx"
}
cluster {
  name = emqxcl
  discovery_strategy = manual
}
dashboard {
    listeners.http {
        bind = 18083
    }
}
```

Here you can also use ***Auto clustering*** by usnig ***discovery_strategy = static*** ratherthan discovery_strategy = manual, ***Read More*** : LINK

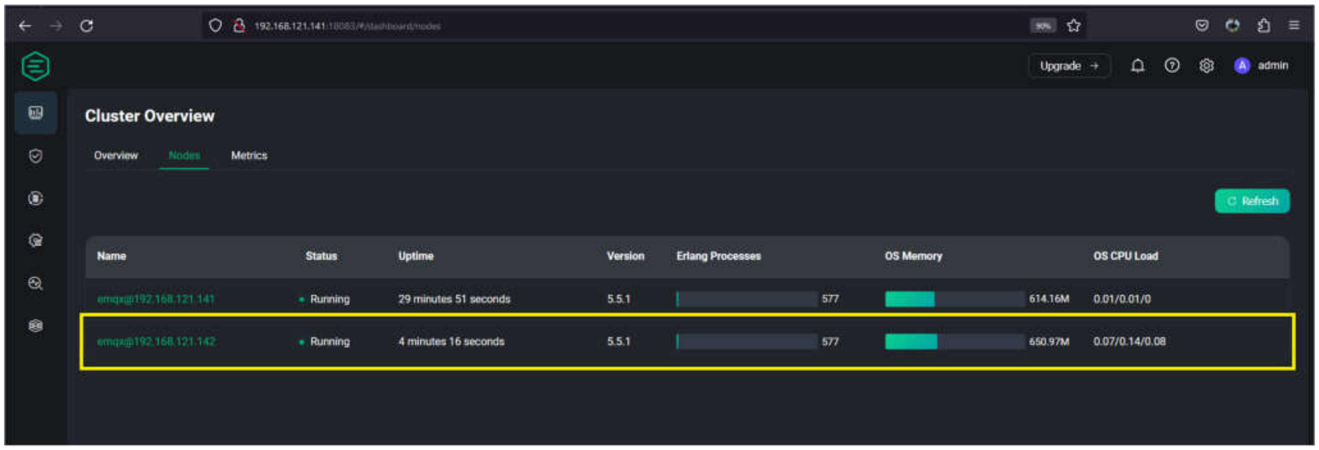
After Configurations Save run the following command to reload the daemon and apply the emqx configuration.

```
# Reload Daemon and restart emqx after config change
systemctl daemon-reload
systemctl restart emqx
```

3.1 Cluster Creating/Joining

```
# joining a master/Node1 by Node2
emqx ctl cluster join emqx@192.168.121.141        # Joining a master/Node1 using IP
emqx ctl status                                   # To Check node status
emqx ctl cluster status                           To Check node status

# joining a master/Node1 by Node3
emqx ctl cluster join emqx@192.168.121.141        # Joining a master/Node1 using IP
emqx ctl status                                   # To Check node status
emqx ctl cluster status                           # To Check node status

# After joining nodes each-other don't forget to run this cmd (in every node).
systemctl daemon-reload
systemctl restart emqx
systemctl status emqx

# Leaving a master/Node1 by other Node
emqx ctl cluster leave emqx@192.168.121.141
emqx ctl cluster force-leave emqx@192.168.121.141
```

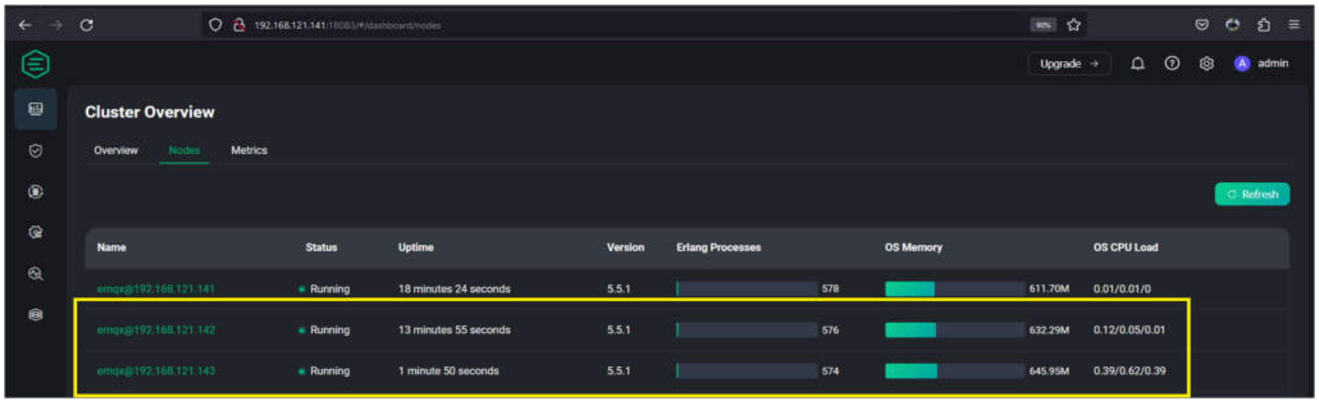
OUTPUT: Joined Node2 and Node3 to Master/Node1


192.168.121.141:18083/#/dashboard/nodes

Cluster Overview

Overview Nodes Metrics

Name	Status	Uptime	Version	Erlang Processes	OS Memory	OS CPU Load
emqx@192.168.121.141	Running	29 minutes 51 seconds	5.5.1	577	614.16M	0.01/0.01/0
emqx@192.168.121.142	Running	4 minutes 16 seconds	5.5.1	577	650.97M	0.07/0.14/0.08


192.168.121.141:18083/#/dashboard/nodes

Cluster Overview

Overview Nodes Metrics

Name	Status	Uptime	Version	Erlang Processes	OS Memory	OS CPU Load
emqx@192.168.121.141	Running	18 minutes 24 seconds	5.5.1	578	611.70M	0.01/0.01/0
emqx@192.168.121.142	Running	13 minutes 55 seconds	5.5.1	576	632.29M	0.12/0.05/0.01
emqx@192.168.121.143	Running	1 minute 50 seconds	5.5.1	574	645.95M	0.39/0.62/0.39

3. Load Balancing by HAProxy

3.1 Installation of HAProxy

```
# installation and service command
sudo yum install -y haproxy
sudo systemctl enable haproxy
sudo systemctl start haproxy
sudo systemctl status haproxy
```

3.2 Configure HAProxy as LB

```
# Open the haproxy.cfg and make a backup of default config
# and create a new haproxy.cfg file and paste the config provided below
cd /etc/haproxy/ && mv haproxy.cfg default-haproxy.cfg
Vi haproxy.cfg
```

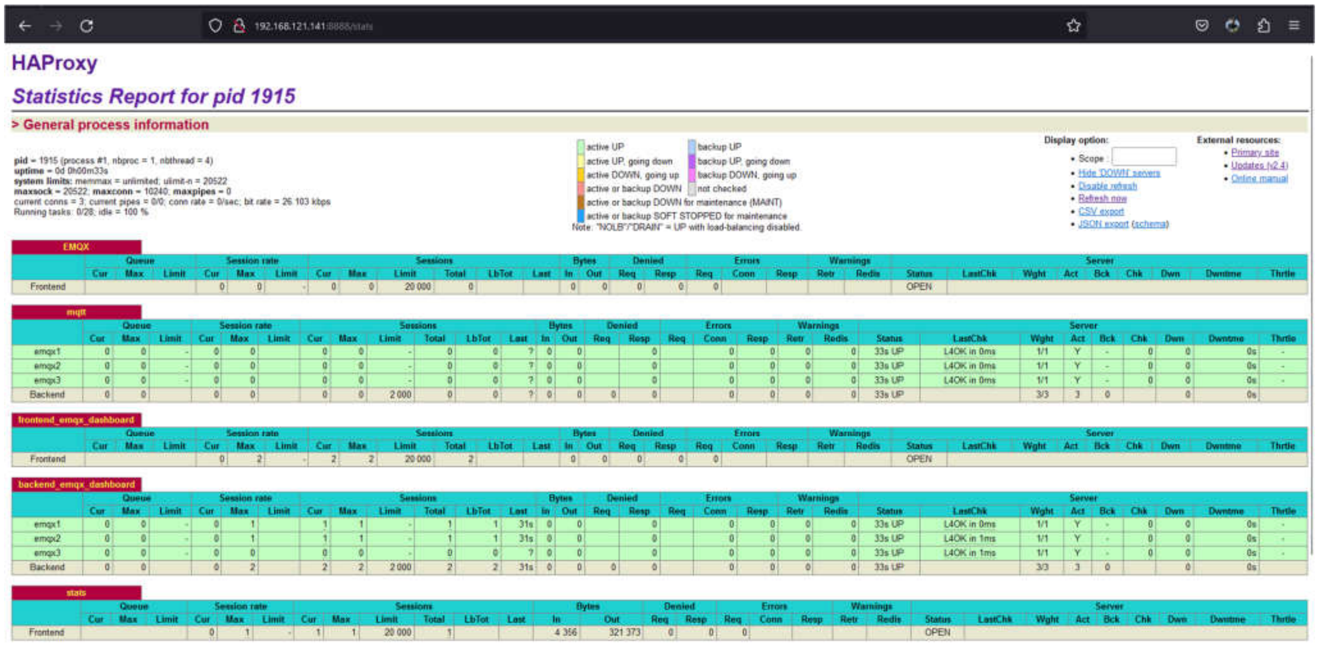
```
# -------------------- Haproxy config Start from here --------------------
global
  log 127.0.0.1 local3 info
  daemon
  maxconn 10240

defaults
  log global
  mode tcp
  option tcplog
  #option dontlognull
  timeout connect 10000
  # timeout > mqtt's keepalive * 1.2
  timeout client 240s
  timeout server 240s
  maxconn 20000

# -------------------- Loadtest Websocket --------------------
frontend EMQX
  bind *:18831
  mode tcp
  tcp-request inspect-delay 10s
  tcp-request content reject unless { req.payload(0,0),mqtt_is_valid }
  default_backend mqtt

backend mqtt
  mode tcp
  stick-table type string len 32 size 1000k expire 30m
  stick on req.payload(0,0),mqtt_field_value(connect,client_identifier)

  server emqx1 192.168.121.141:1883 check
  server emqx2 192.168.121.142:1883 check
  server emqx3 192.168.121.143:1883 check

# -------------------- Dashboaed --------------------
frontend frontend_emqx_dashboard
    bind *:18083
    option tcplog
    mode tcp
    default_backend backend_emqx_dashboard

backend backend_emqx_dashboard
    mode tcp
    balance roundrobin
    server emqx1 192.168.121.141:18083 check
    server emqx2 192.168.121.142:18083 check
    server emqx3 192.168.121.143:18083 check

# -------------------- Haproxy stats --------------------
frontend stats
  mode http
  bind *:8888
  stats enable
  stats uri /stats
  stats refresh 10s
# -------------------- Haproxy config end here --------------------
```

To apply and restart the HaProxy

```
# -------------------- Check HAProxy Config and restrt the service --------------------
haproxy -c -f haproxy.cfg
systemctl restart haproxy
```

Complete Configurations: [LINK]()

OUTPUT: ***visit:: http://haproxy_IP:8888/stats***



192.168.121.141:8888/stats

HAProxy

Statistics Report for pid 1915

> General process information

pid = 1915 (process #1, nbproc = 1, nbthread = 4)
uptime = 0d 0h00m33s
system limits: memmax = unlimited; ulimit-n = 20522
maxsock = 20522; **maxconn** = 10240; **maxpipes** = 0
current conns = 3; current pipes = 0/0; conn rate = 0/sec; bit rate = 26.103 kbps
Running tasks: 0/28; idle = 100 %

🟩 active UP	🟦 backup UP
🟨 active UP, going down	🟪 backup UP, going down
🟧 active DOWN, going up	🟪 backup DOWN, going up
🟥 active or backup DOWN	⬜ not checked
🟫 active or backup DOWN for maintenance (MAINT)	
🟦 active or backup SOFT STOPPED for maintenance	

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export
- JSON export (schema)

External resources:
- Primary site
- Updates (v2.4)
- Online manual

EMQX

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	20 000	0			0	0	0	0	0					OPEN								

mqtt

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
emqx1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
emqx2	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
emqx3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
Backend	0	0		0	0		0	0	2 000	0	0	?	0	0	0	0		0	0	0	0	33s UP		3/3	3	0		0	0s	

frontend_emqx_dashboard

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	2	2	20 000	2			0	0	0	0	0					OPEN								

backend_emqx_dashboard

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
emqx1	0	0	-	0	1		1	1	-	1	1	31s	0	0		0		0	0	0	0	33s UP	L4OK in 0ms	1/1	Y	-	0	0	0s	-
emqx2	0	0	-	0	1		1	1	-	1	1	31s	0	0		0		0	0	0	0	33s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-
emqx3	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	0	0	33s UP	L4OK in 1ms	1/1	Y	-	0	0	0s	-
Backend	0	0		0	2		2	2	2 000	2	2	31s	0	0	0	0		0	0	0	0	33s UP		3/3	3	0		0	0s	

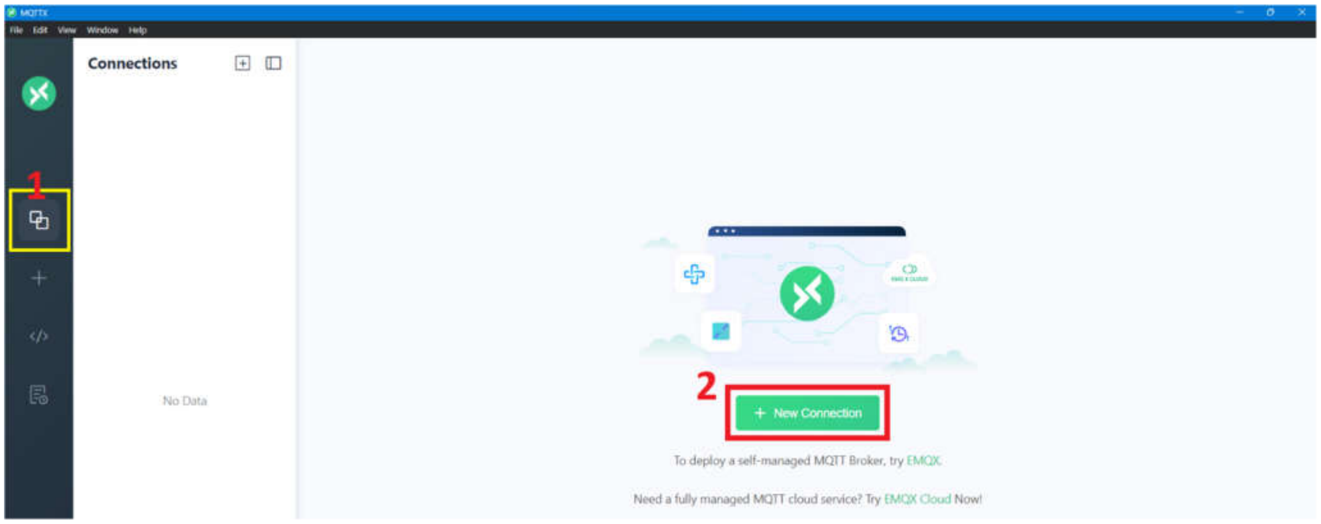
stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1	-	1	1	20 000	1			4 356	321 373	0	0	0					OPEN								

4. MQTTX Load testing

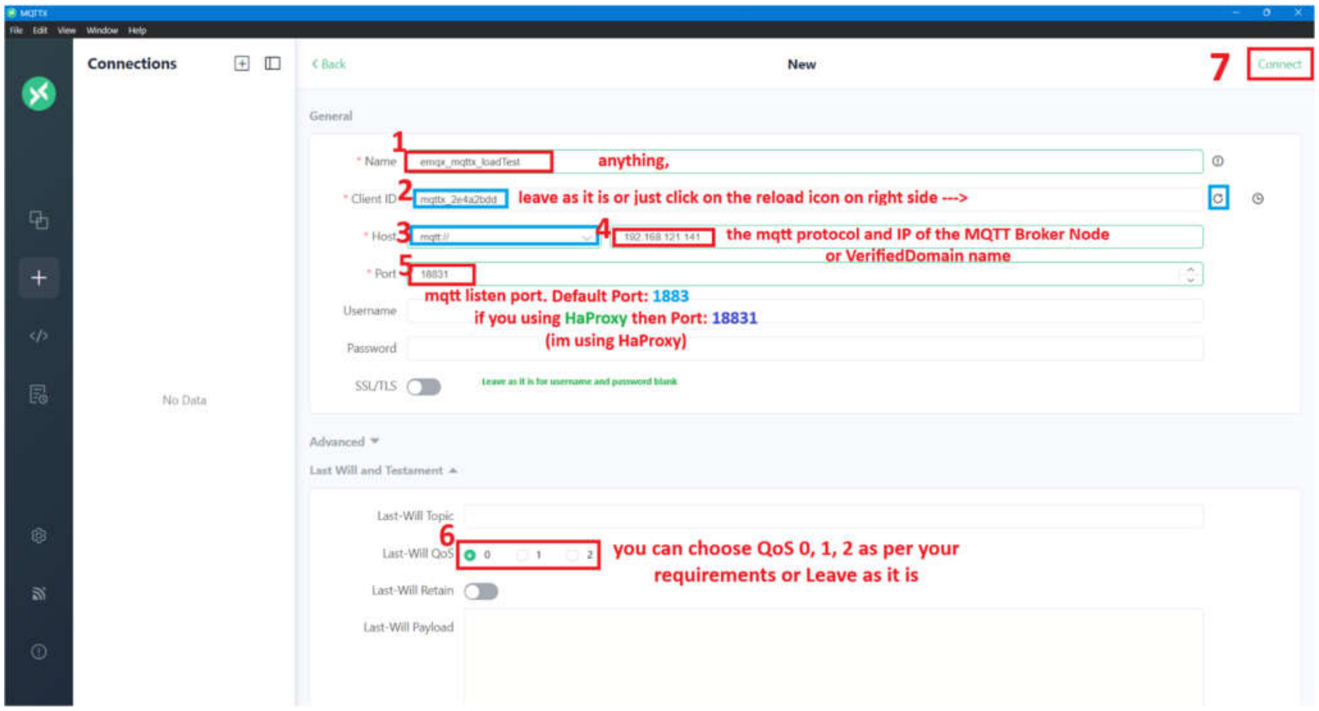
Download the MQTTX Software from official website (LINK), and install on your machine. After installation completed run the MQTTX software and do the following steps; Tutorial-LINK

Create a Connection



To deploy a self-managed MQTT Broker, try EMQX.

Need a fully managed MQTT cloud service? Try EMQX Cloud Now!



1 Name: emqx_mqttx_loadTest — anything,

2 Client ID: mqttx_2e4a2bdd — leave as it is or just click on the reload icon on right side --->

3 Host: mqtt:// 4 192.168.121.141 — the mqtt protocol and IP of the MQTT Broker Node or VerifiedDomain name

5 Port: 18831 — mqtt listen port. Default Port: 1883 if you using HaProxy then Port: 18831 (im using HaProxy)

Username

Password

SSL/TLS — Leave as it is for username and password blank

Last-Will Topic

6 Last-Will QoS: [✓ 0] 1 2 — you can choose QoS 0, 1, 2 as per your requirements or Leave as it is

Last-Will Retain

Last-Will Payload

When we click on '**Connect**' button at right top side then it will automatically open a new window and shows '**Connected**' figure-conncected, if there is any issue it shows the following window; figure-conncect

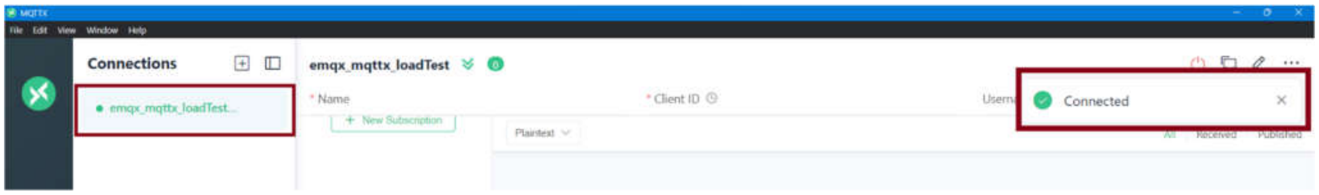
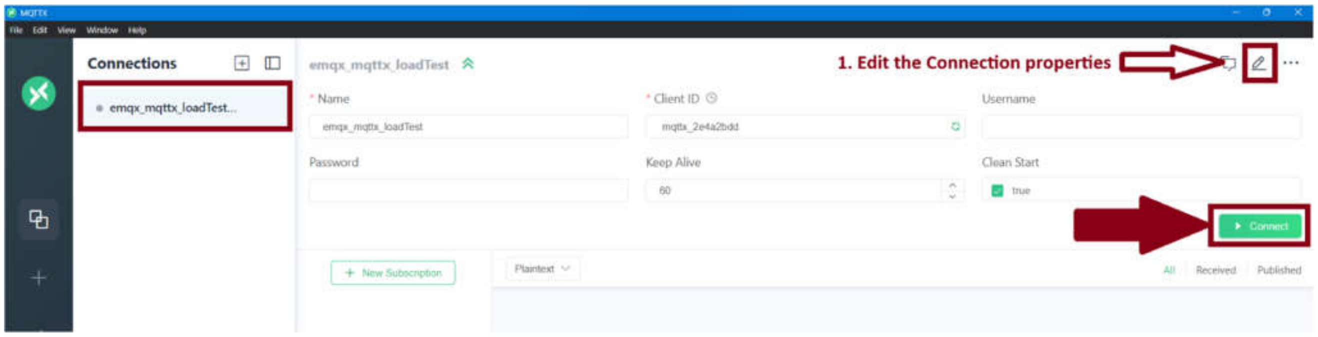


figure-conncected

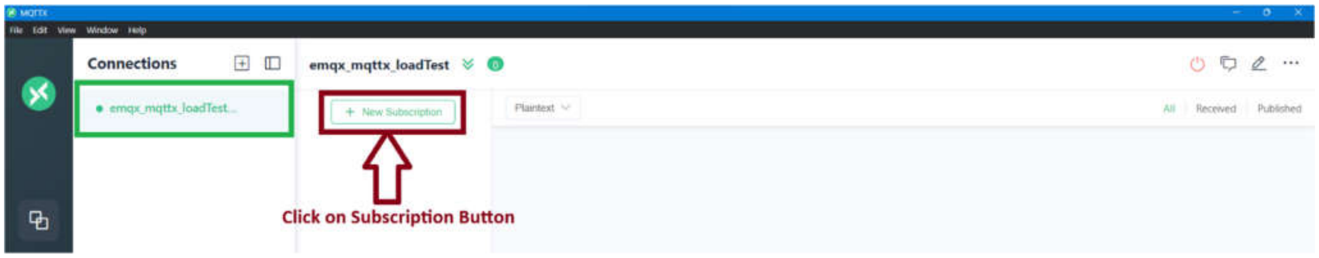


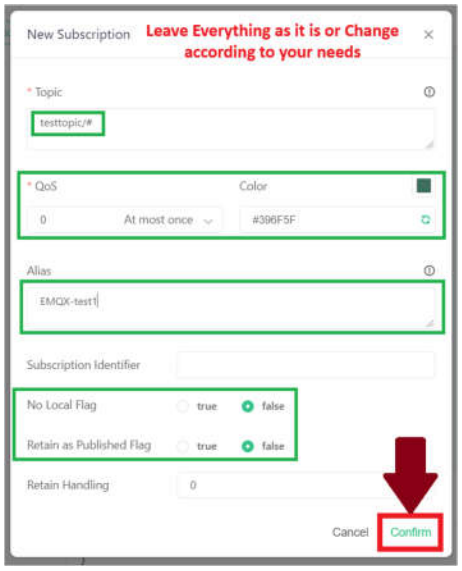
Name: emqx_mqttx_loadTest
Client ID: mqttx_2e4a2bdd
Username:
Password:
Keep Alive: 60
Clean Start: true

figure-conncect

if the above window appears, then you have to check your **NodeIP/DomainName** or **Protocol** and **Port** by click on ‘**Edit Icon’** then click on ‘**Connect’** or it will be connected automatically.

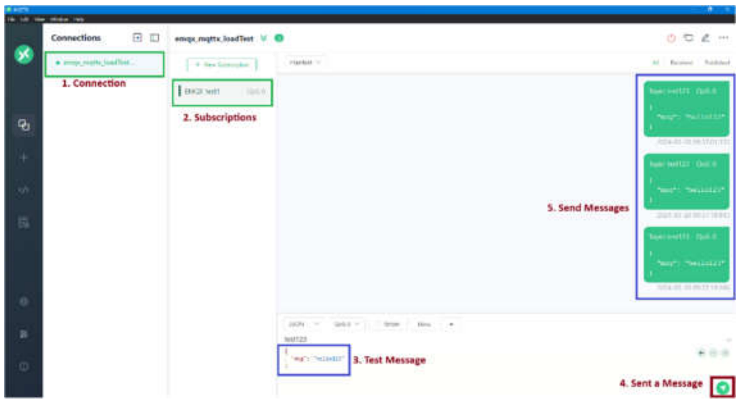
Creating a Subscription | Read More: LINK

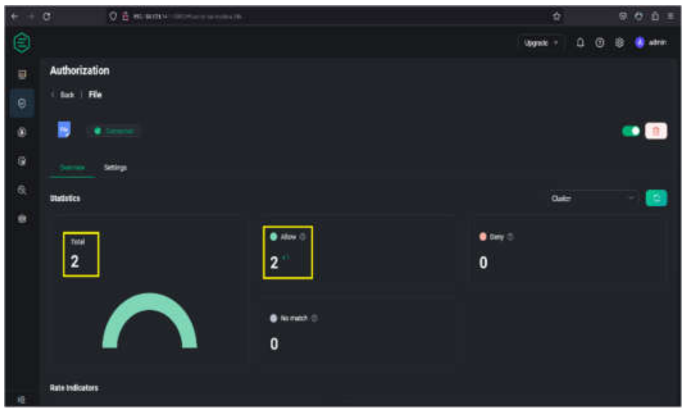




Topic: testtopic/#
QoS: 0 At most once
Color: #396F5F
Alias: EMQX-test1
Subscription Identifier:
No Local Flag: true [✓ false]
Retain as Published Flag: true [✓ false]
Retain Handling: 0

Load Testing or Message test and Check on Web Dashboard





5. Load Testing via golan (Download File: LINK)

5.1 Creating loadtest.go file

(replace with your HaProxy /Node1 IP: `192.168.121.141`) || Official Load Test Docs: LINK

```go
package main // start go file
import (
    "fmt"
    "os"
    "os/signal"
    "strconv"
    "sync"
    "time"

    MQTT "github.com/eclipse/paho.mqtt.golang"
)
const (
    brokerAddress = "tcp://192.168.121.141:18831" // Update with your EMQX broker address
    clientPrefix  = "client-"
    topic         = "test/topic"
    qos           = 2
    numClients    = 1000 // Number of clients to simulate
)
func main() {
    wg := &sync.WaitGroup{}
    wg.Add(numClients)
    for i := 0; i < numClients; i++ {
        go func(clientID int) {
            defer wg.Done()
            opts := MQTT.NewClientOptions()
            opts.AddBroker(brokerAddress)
            opts.SetClientID(clientPrefix + strconv.Itoa(clientID))
            client := MQTT.NewClient(opts)
            if token := client.Connect(); token.Wait() && token.Error() != nil {
                fmt.Printf("Client %d: Error connecting: %v\n", clientID, token.Error())
                return
            }
            defer client.Disconnect(250)

            for {
                token := client.Publish(topic, byte(qos), false, "Hello from client
"+strconv.Itoa(clientID))
                token.Wait()
                if token.Error() != nil {
                    fmt.Printf("Client %d: Error publishing message: %v\n", clientID, token.Error())
                    return
                }
                time.Sleep(1 * time.Second) // Adjust this delay as needed
            }
        }(i)
    }
    // Capture CTRL+C signal to gracefully shutdown
    c := make(chan os.Signal, 1)
    signal.Notify(c, os.Interrupt)
    go func() {
        <-c
        fmt.Println("\nShutting down...")
        wg.Wait()
        os.Exit(0)
    }()
    fmt.Println("Press CTRL+C to exit")
    wg.Wait()
} // end of go file
```

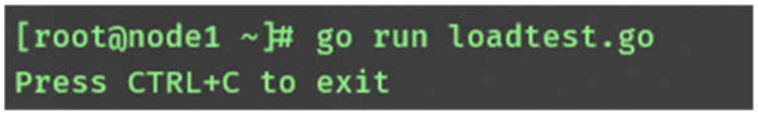
5.2 Go Inatallation:

```
yum install -y go                                 # go language installation
go version                                        # go version check
go mod init loadtest.go                           # .go file must be present in a pwd
go get github.com/eclipse/paho.mqtt.golang        # import paho module for mqtt test
go get github.com/gorilla/websocket
go get golang.org/x/net/proxy
```

5.3 Load testing

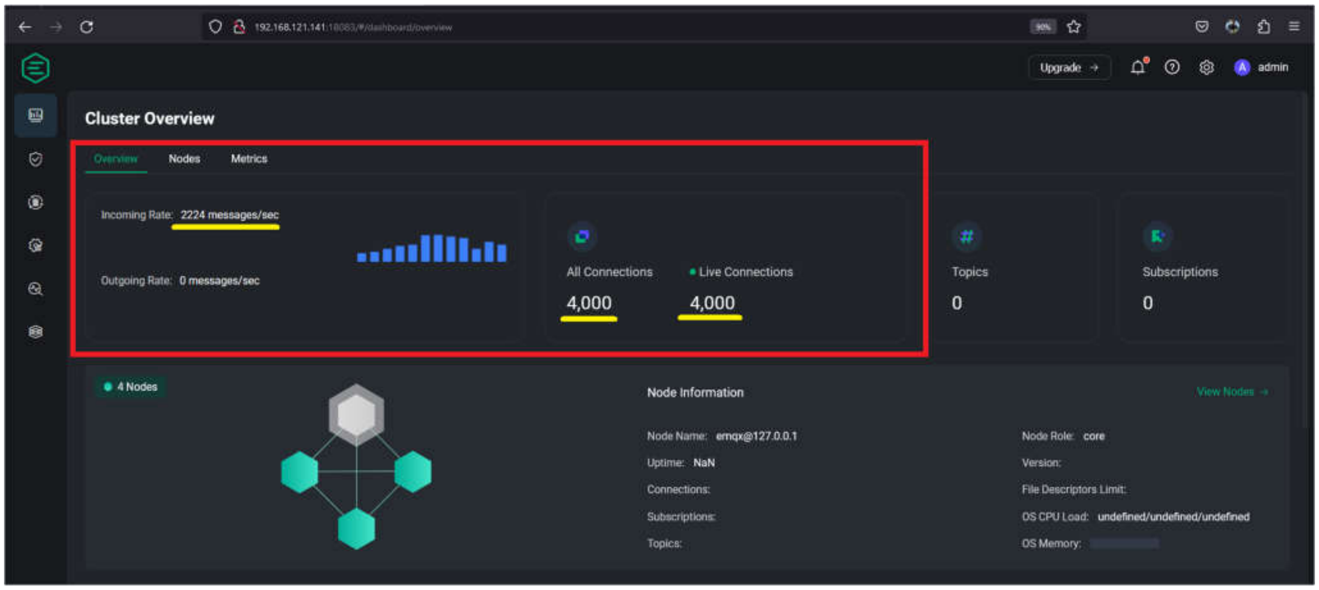
```
go run loadtest.go                                # load testing using .go file
```

OUTPUT:



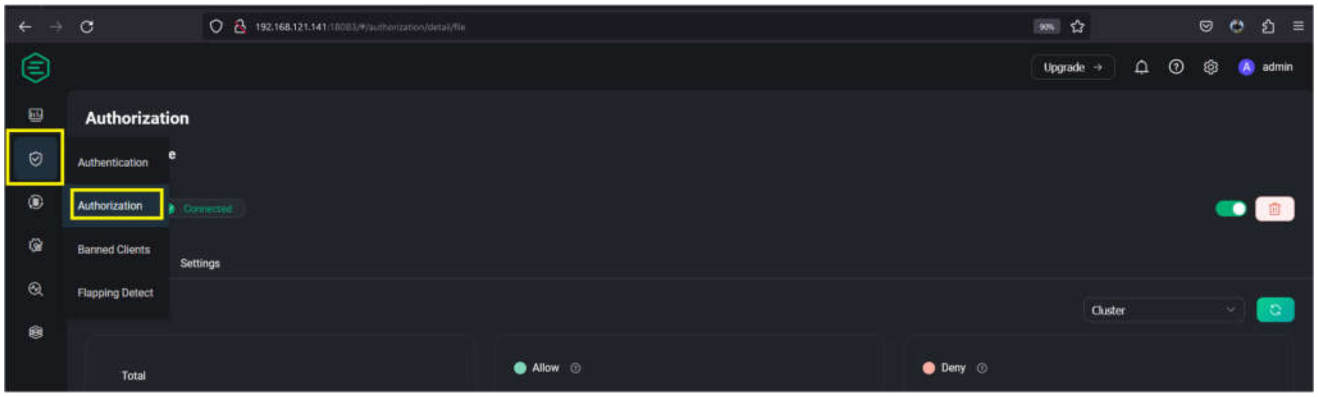
```
[root@node1 ~]# go run loadtest.go
Press CTRL+C to exit
```

check in Web Dashboard (Click on **Monitor Icon**).

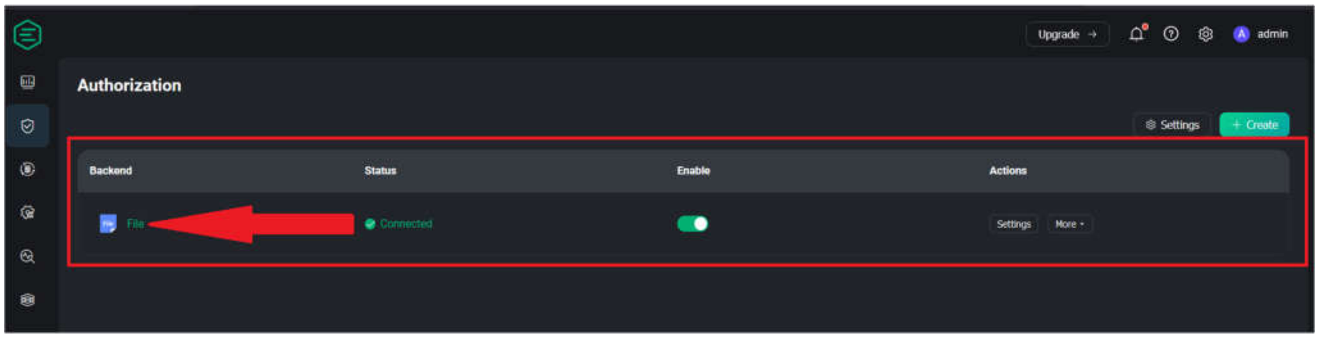


```
192.168.121.141:18083/#/dashboard/overview
Cluster Overview
Overview   Nodes   Metrics
Incoming Rate:  2224 messages/sec
Outgoing Rate:  0 messages/sec
All Connections  4,000     Live Connections  4,000
Topics  0
Subscriptions  0
4 Nodes
Node Information
Node Name:  emqx@127.0.0.1          Node Role:  core
Uptime:  NaN                        Version:
Connections:                        File Descriptors Limit:
Subscriptions:                      OS CPU Load:  undefined/undefined/undefined
Topics:                             OS Memory:
```

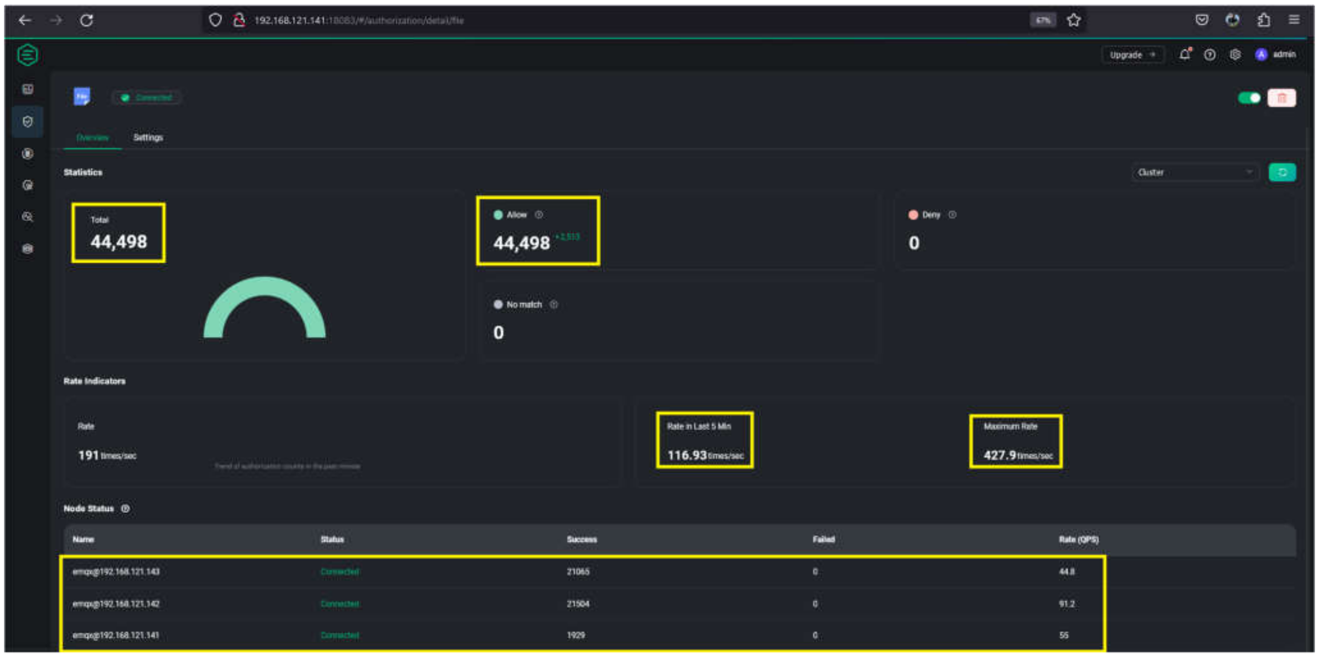
Authorization check (click on **Shield Icon** --> **Authorization** --> **File**).



```
192.168.121.141:18083/#/authorization/detail/file
Authorization
Connected
Settings
Cluster
Allow
Deny
Total
```



Backend	Status	Enable	Actions
File	● Connected		Settings More



Total
44,498

Allow
44,498 +2,513

Deny
0

No match
0

Rate Indicators

Rate
191 times/sec

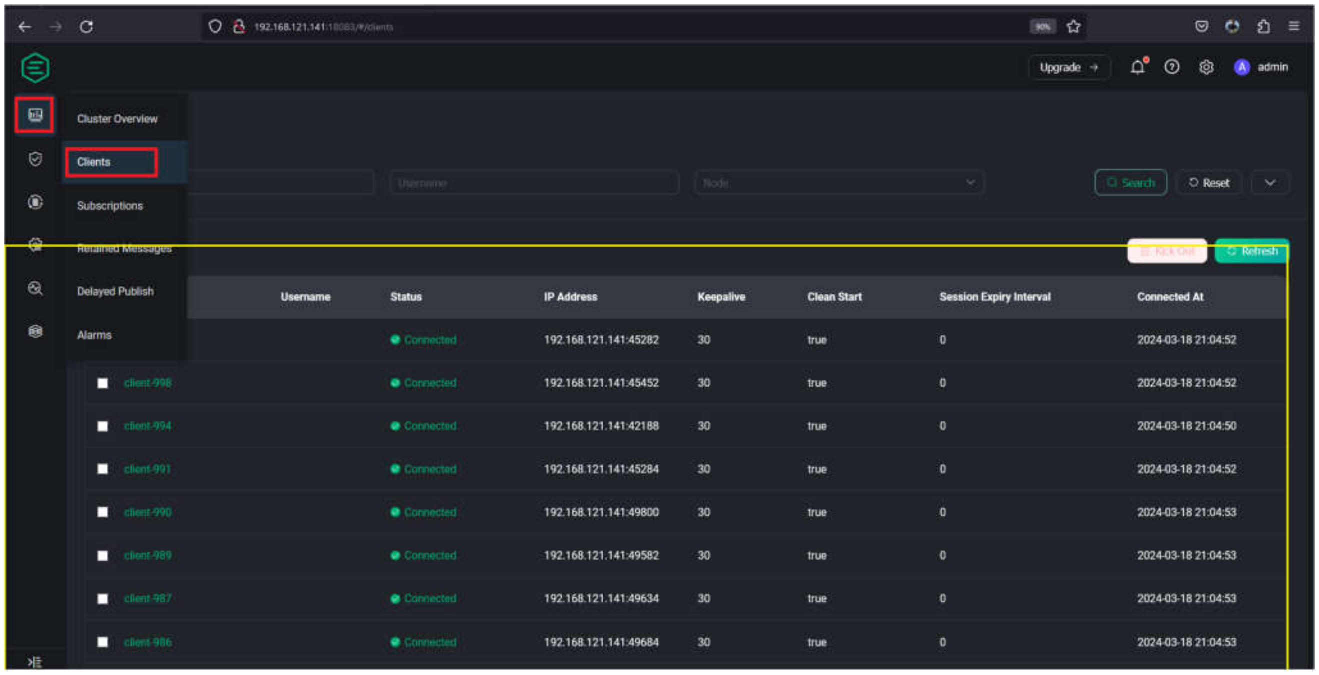
Rate in Last 5 Min
116.93 times/sec

Maximum Rate
427.9 times/sec

Node Status

Name	Status	Success	Failed	Rate (QPS)
emqx@192.168.121.143	Connected	21065	0	44.8
emqx@192.168.121.142	Connected	21504	0	91.2
emqx@192.168.121.141	Connected	1929	0	55

Clients Check (Click on **Monitor Icon --> Clients**), This shown realtime connected clients.



	Username	Status	IP Address	Keepalive	Clean Start	Session Expiry Interval	Connected At
		● Connected	192.168.121.141:45282	30	true	0	2024-03-18 21:04:52
client-998		● Connected	192.168.121.141:45452	30	true	0	2024-03-18 21:04:52
client-994		● Connected	192.168.121.141:42188	30	true	0	2024-03-18 21:04:50
client-991		● Connected	192.168.121.141:45284	30	true	0	2024-03-18 21:04:52
client-990		● Connected	192.168.121.141:49800	30	true	0	2024-03-18 21:04:53
client-989		● Connected	192.168.121.141:49582	30	true	0	2024-03-18 21:04:53
client-987		● Connected	192.168.121.141:49634	30	true	0	2024-03-18 21:04:53
client-986		● Connected	192.168.121.141:49684	30	true	0	2024-03-18 21:04:53

6. Uninstallation of EMQX

```
# ----------------- Complete Uninstall ------------
sudo yum remove emqx -y
sudo rm -rf /etc/emqx
sudo rm -rf /var/lib/emqx
sudo rm -rf /var/log/emqx

# ----------------- Service Delete ------------
sudo rm /etc/systemd/system/emqx.service
sudo systemctl daemon-reload

# ----------------- User and Group Delete ------------
sudo userdel emqx
sudo groupdel emqx
```

7. Using Docker Container

official Docs : LINK || official Docker Setup link : LINK

```
# Latest Version of Docker Inatallation
sudo yum install -y yum-utils
sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
sudo yum -y install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

sudo usermod -aG docker $(whoami)
sudo systemctl start docker
sudo systemctl enable --now docker
newgrp docker
```

```
# EMQX Docker Image Pull
docker pull emqx/emqx:5.5.1
```

```
# Creating a volume path for EMQX to persist Config/logs... after pod/container die
mkdir -p /opt/emqx/data /opt/emqx/log

cd /opt/emqx/

# Give a permission to the folder, docker create a EMQX conf files on it
chmod -R 777 /opt/emqx/
```

```
# run a Single container of emqx
docker run -d --name emqx \
  -p 1883:1883 -p 8083:8083 \
  -p 8084:8084 -p 8883:8883 \
  -p 18083:18083 \
  -v $PWD/data:/opt/emqx/data \
  -v $PWD/log:/opt/emqx/log \
  emqx/emqx:5.5.1

# to check all container of docker
docker ps -a
```

Read More Cluster Create: LINK

```
#docker network create for cluster
docker network create emqx-net
```

```
# Docker container 1: Master node
docker run -d \
    --name emqx1 \
    -e "EMQX_NODE_NAME=emqx@node1.emqx.com" \
    --network emqx-bridge \
    --network-alias node1.emqx.com \
    -p 1883:1883 \
    -p 8083:8083 \
    -p 8084:8084 \
    -p 8883:8883 \
    -p 18083:18083 \
    emqx/emqx:5.5.1
```

```
# Docker container 2: slave node1
docker run -d \
    --name emqx2 \
    -e "EMQX_NODE_NAME=emqx@node2.emqx.com" \
    --network emqx-net \
    --network-alias node2.emqx.com \
    emqx/emqx:5.5.1

# Docker container 2: join a Master Node, second command is run inside the container
docker exec -it emqx2 \
    emqx ctl cluster join emqx@node1.emqx.com
```

```
# Docker container 3: slave node2
docker run -d \
    --name emqx3 \
    -e "EMQX_NODE_NAME=emqx@node3.emqx.com" \
    --network emqx-net \
    --network-alias node3.emqx.com \
    emqx/emqx:5.5.1

# Docker container 3: join a Master Node, second command is run inside the container
docker exec -it emqx3 \
    emqx ctl cluster join emqx@node1.emqx.com
```

8. Using Docker Compose

```
# ----------Docker-Compose Latst version Installation----------
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" \
    -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

Create a docker-compose.yaml file and copy paste the following lines of code on it.

```
# Creating a docker-compose file
vi docker-compose.yml
```

```
version: '3'

services:
  emqx1:
    image: emqx:5.5.1
    container_name: emqx1
    environment:
    - "EMQX_NODE_NAME=emqx@node1.emqx.io"
    - "EMQX_CLUSTER__DISCOVERY_STRATEGY=static"
    - "EMQX_CLUSTER__STATIC__SEEDS=[emqx@node1.emqx.io,emqx@node2.emqx.io]"
    healthcheck:
      test: ["CMD", "/opt/emqx/bin/emqx", "ctl", "status"]
      interval: 5s
      timeout: 25s
      retries: 5
    networks:
      emqx-bridge:
        aliases:
        - node1.emqx.io
    ports:
      - 1883:1883
      - 8083:8083
      - 8084:8084
      - 8883:8883
      - 18083:18083
    volumes:
      - $PWD/emqx1_data:/opt/emqx/data

  emqx2:
    image: emqx:5.5.1
    container_name: emqx2
    environment:
    - "EMQX_NODE_NAME=emqx@node2.emqx.io"
    - "EMQX_CLUSTER__DISCOVERY_STRATEGY=static"
    - "EMQX_CLUSTER__STATIC__SEEDS=[emqx@node1.emqx.io,emqx@node2.emqx.io]"
    healthcheck:
      test: ["CMD", "/opt/emqx/bin/emqx", "ctl", "status"]
      interval: 5s
      timeout: 25s
      retries: 5
    networks:
      emqx-bridge:
        aliases:
        - node2.emqx.io
    volumes:
      - $PWD/emqx2_data:/opt/emqx/data

networks:
  emqx-bridge:
    driver: bridge
```

```
docker-compose up -d
docker exec -it emqx1 sh -c "emqx ctl cluster status"
```

Web Dashboard check: http://Node_VM_IP:18083
Defafult-userName: admin
Default-Password: public

Node: load tesing and all configurations same as Normal Node and Cluster mention above documentation

9. Performence Tunning

```
# ------------------------------ Performance tunning (All Node)---------------------------
sysctl -w fs.file-max=2097152
sysctl -w fs.nr_open=2097152
echo 2097152 > /proc/sys/fs/nr_open
ulimit -n 2097152

echo "fs.file-max = 2097152">>/etc/sysctl.conf
echo "fs.file-max = 2097152">>/etc/sysctl.conf
echo "DefaultLimitNOFILE=2097152">>/etc/systemd/system.conf


vi /usr/lib/systemd/system/emqx.service
LimitNOFILE=2097152


echo -e "*      soft   nofile      2097152
*      hard   nofile      2097152">>/etc/security/limits.conf

systemctl restart emqx
systemctl daemon-reload


### ------------- Network -----------------------------------------
sysctl -w net.core.somaxconn=32768
sysctl -w net.ipv4.tcp_max_syn_backlog=16384
sysctl -w net.core.netdev_max_backlog=16384
sysctl -w net.ipv4.ip_local_port_range='1024 65535'
sysctl -w net.core.rmem_default=262144
sysctl -w net.core.wmem_default=262144
sysctl -w net.core.rmem_max=16777216
sysctl -w net.core.wmem_max=16777216
sysctl -w net.core.optmem_max=16777216
#sysctl -w net.ipv4.tcp_mem='16777216 16777216 16777216'
sysctl -w net.ipv4.tcp_rmem='1024 4096 16777216'
sysctl -w net.ipv4.tcp_wmem='1024 4096 16777216'
sysctl -w net.nf_conntrack_max=1000000
sysctl -w net.netfilter.nf_conntrack_max=1000000
sysctl -w net.netfilter.nf_conntrack_tcp_timeout_time_wait=30
sysctl -w net.ipv4.tcp_max_tw_buckets=1048576

# Enabling following option is not recommended. It could cause connection reset under NAT
# sysctl -w net.ipv4.tcp_tw_recycle=1
# sysctl -w net.ipv4.tcp_tw_reuse=1
sysctl -w net.ipv4.tcp_fin_timeout=15
vi /etc/emqx/emqx.conf
## Sets the maximum number of simultaneously existing ports for this system
node.max_ports = 2097152
  # node {
  #   name = "emqx@10.13.194.69"
  #   cookie = "emqxsecretcookie"
  #   data_dir = "/var/lib/emqx"
  #   max_ports = 2097152   # this line needed to be add
  # }
echo -e "## TCP Listener
#listeners.tcp.$name.acceptors = 64
#listeners.tcp.$name.max_connections = 1024000" >> /etc/emqx/emqx.conf

systemctl restart emqx
```