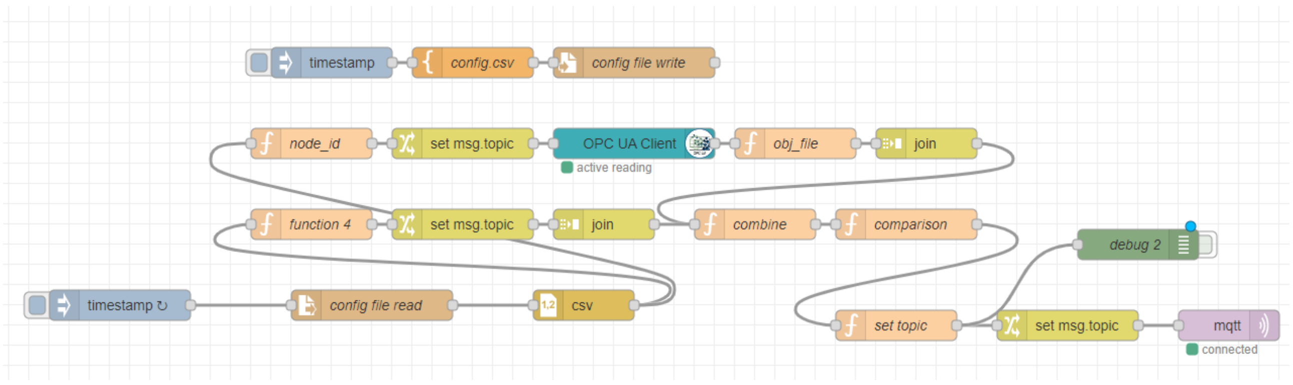
MQTT-Telegraf configuration

R. Deepak Nair
Malavika K.V
V1.0 30.12.2022

Project: **MQTT-Telegraf Configuration**
Rev: 1.0

MQTT-Telegraf Configuration using Topic Parsing Method[No Queue Part]

Node-red Part:



Output getting from this comparison function node is then structuring to mqtt topic pivoting model

Using a function node(set topic) by using this logic given below.

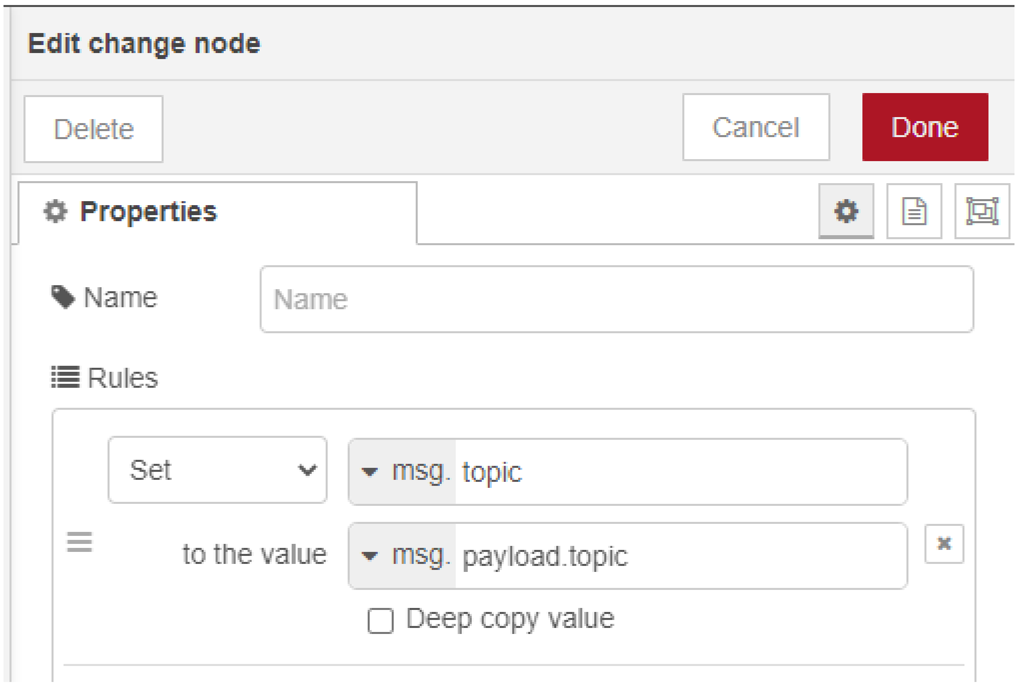
```
for (var i = 0; i < msg.payload.length; i++) {
   var  time= msg.payload[i].time
    var  b= msg.payload[i].b
    var  bd= msg.payload[i].bd
    var  d=msg.payload[i].d
    var  dd= msg.payload[i].dd
    var  dt=msg.payload[i].dt
    var  f= msg.payload[i].f
    var fd=msg.payload[i].fd
    var h=msg.payload[i].h
    var iid= msg.payload[i].iid
    var  p= msg.payload[i].p
    var u=msg.payload[i].u
    var mn= msg.payload[i].mn
    var value= msg.payload[i].value
    var qu= msg.payload[i].qu
    var text= msg.payload[i].text

  var topic = "iplon" + "/" + b + "/" + bd + "/" + d  + "/" + dd + "/" + dt + "/" +
f + "/" + fd + "/" + h + "/" + iid + "/" + p + "/" + u + "/" + "v" + "/" + value +
"/"  + qu + "/" + text
    var obj = {};
obj.payload={ topic}
node.send(obj);
}return null;
```

so, we will get output from the function node like this

```
topic: "iplon/B01/Block_1/INV1/INV1/INV/PDC/DC_POWER/server_7712/7712/Technic_solar/kW
/v/317.25/0/opcua"
```

This topic will change every time according to the for loop

And a change node is used to set msg.topic to the incoming value(msg.payload.topic)



Name: Name

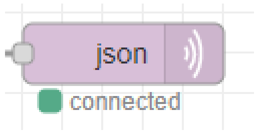
Rules

Set msg. topic

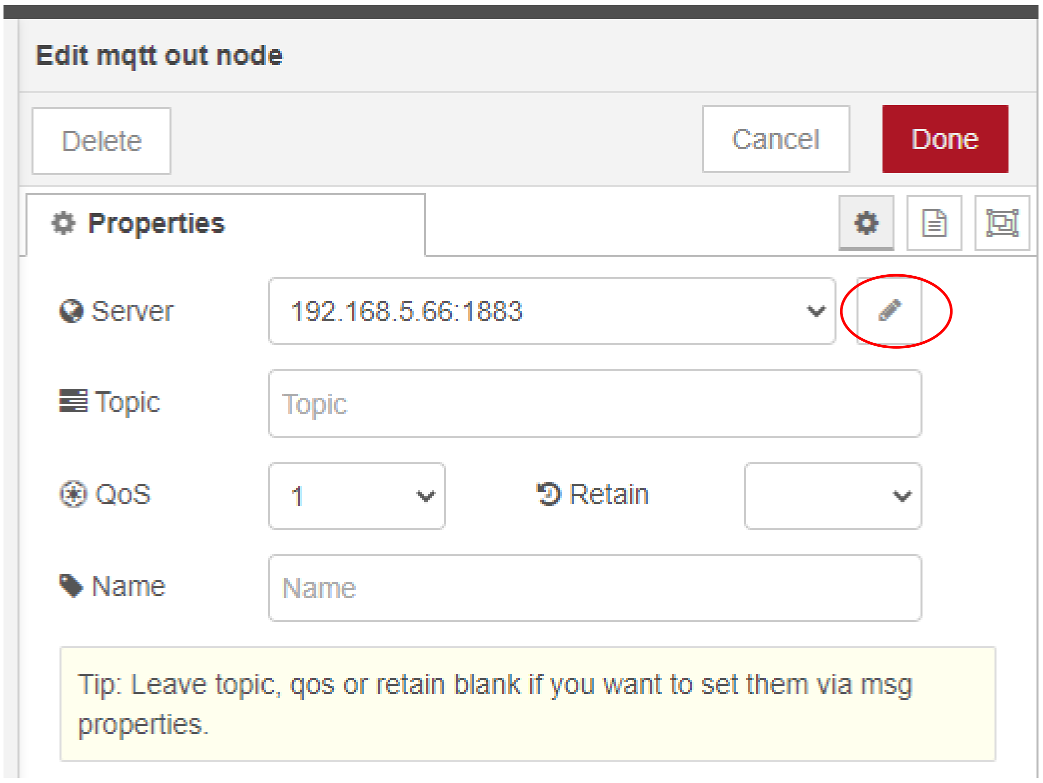
to the value msg. payload.topic

Deep copy value

This output we are giving to mqtt out node



Set up configuration of mqtt by clicking the pencil icon



Edit mqtt out node

Delete | Cancel | Done

Properties

Server: 192.168.5.66:1883

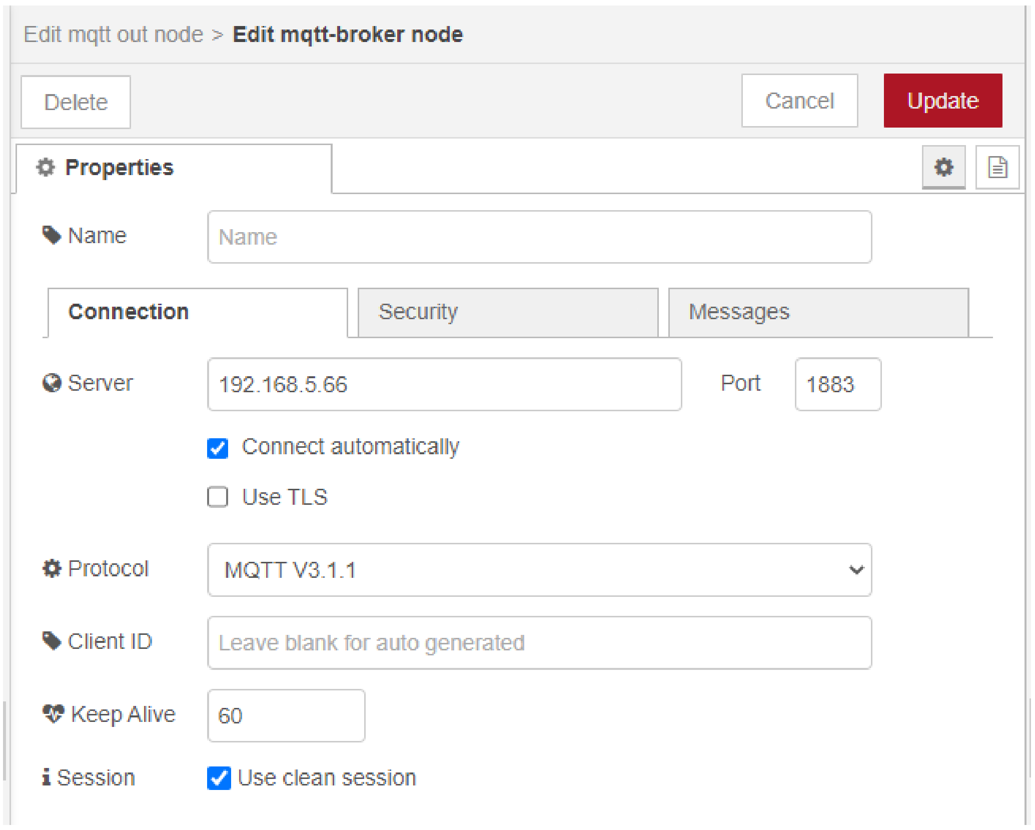
Topic: Topic

QoS: 1 Retain:

Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

In connection give sever ip,port,and select protocol



Edit mqtt out node > Edit mqtt-broker node

Delete | Cancel | Update

Properties

Name: Name

Connection | Security | Messages

Server: 192.168.5.66 Port: 1883

☑ Connect automatically

☐ Use TLS

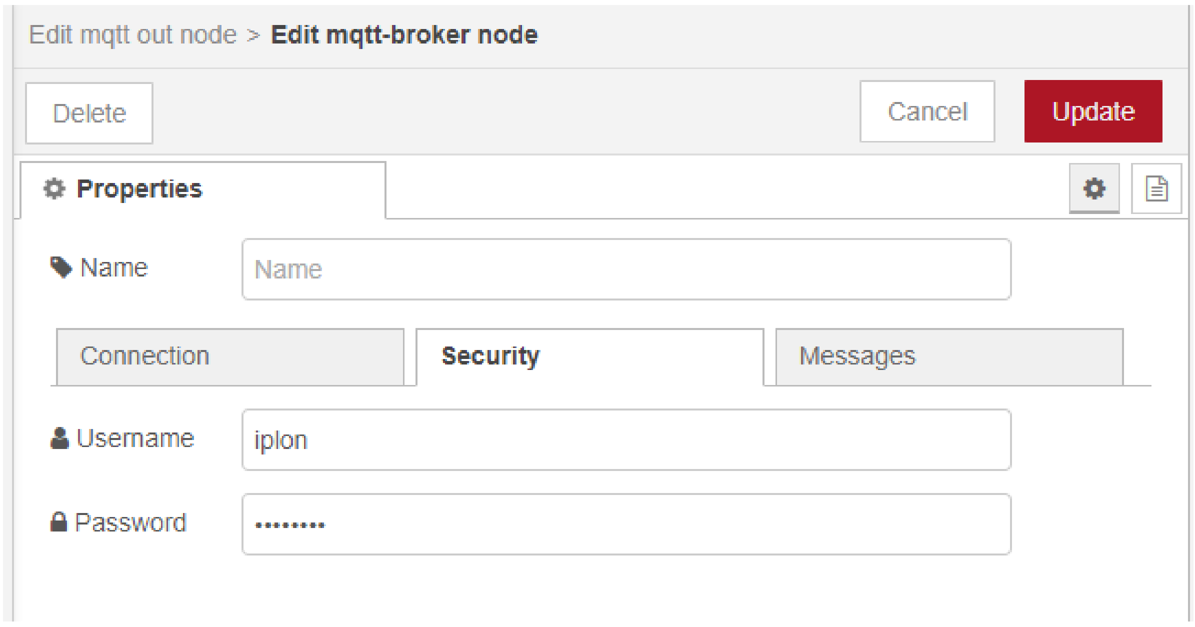
Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☑ Use clean session

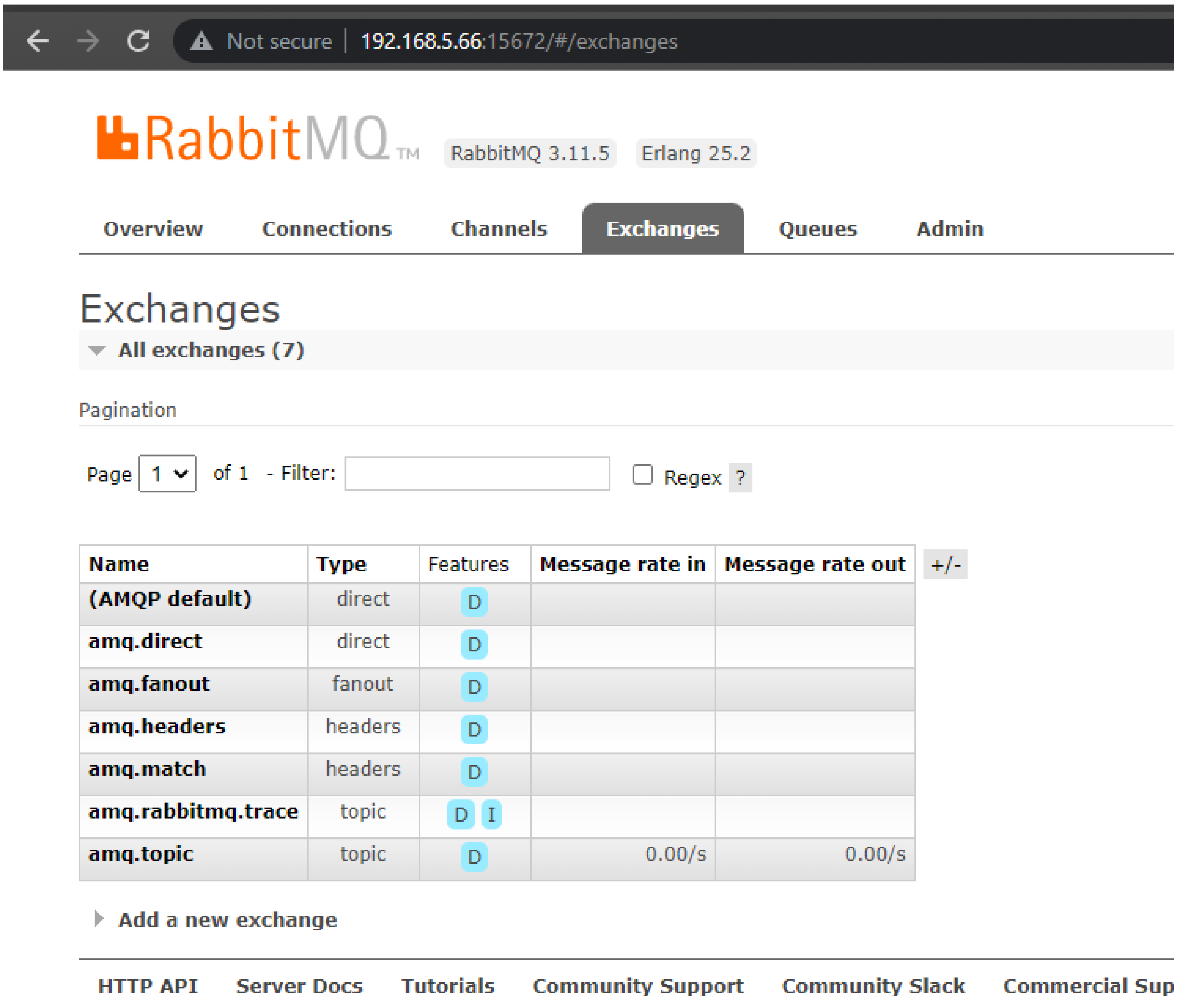
Give username and password in security



Name: Name
Username: iplon
Password: ••••••••

Rabbitmq Part:

- installation guide for ubuntu: https://linuxhint.com/install-rabbitmq-ubuntu/
- docker container installation and plugin enabling guide:https://tewarid.github.io/2019/02/15/mqtt-with-rabbitmq-and-node-red.html
- after installation we can able to log into the management interface at http://localhost:15672 using username/password iplon/iplon321,
- Now go to exchanges click on amq.topic and if the messages are publishing or not



Not secure | 192.168.5.66:15672/#/exchanges

RabbitMQ 3.11.5 Erlang 25.2

Overview Connections Channels Exchanges Queues Admin

Exchanges

All exchanges (7)

Pagination

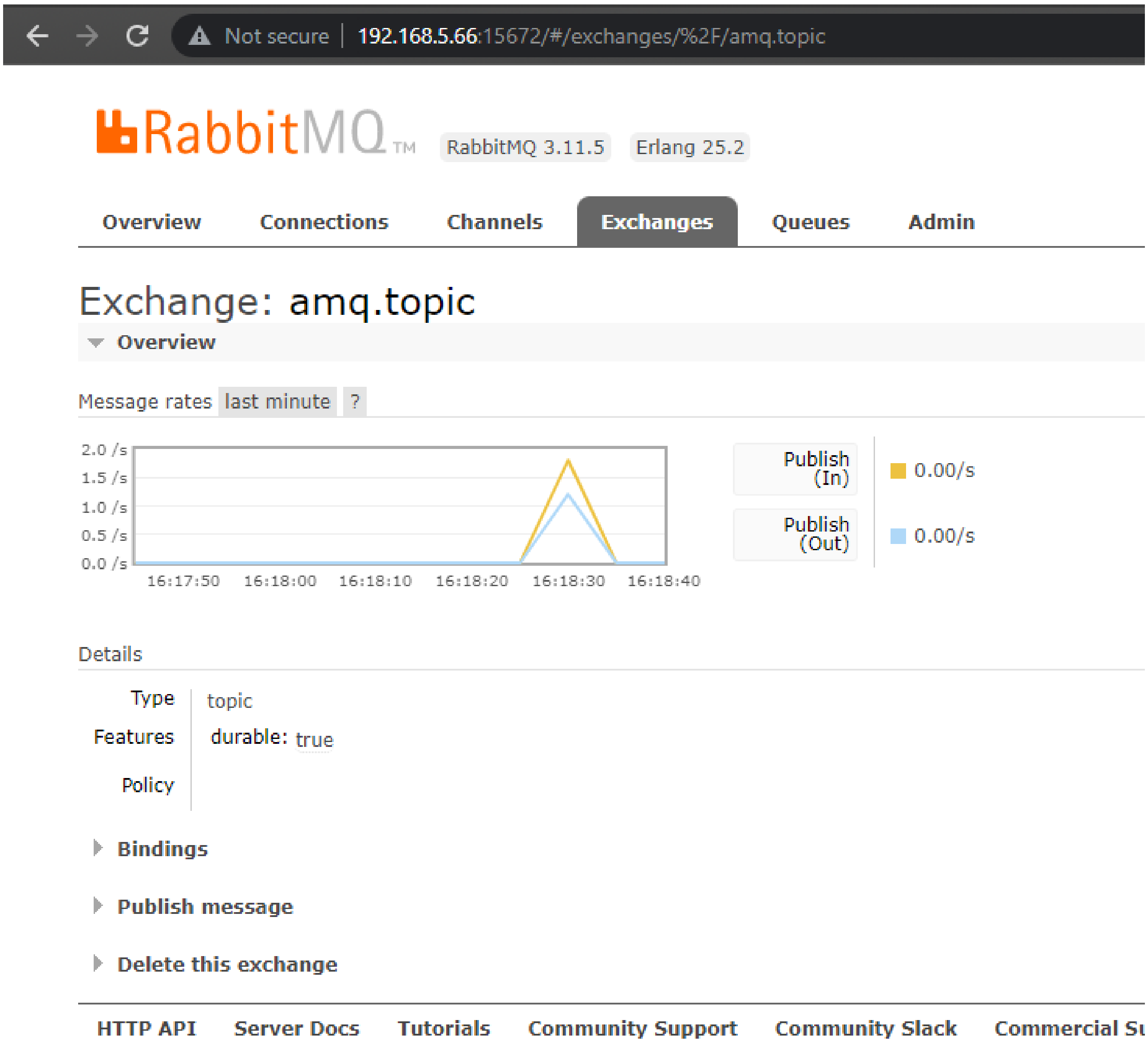
Page 1 of 1 - Filter: Regex ?

Name	Type	Features	Message rate in	Message rate out
(AMQP default)	direct	D		
amq.direct	direct	D		
amq.fanout	fanout	D		
amq.headers	headers	D		
amq.match	headers	D		
amq.rabbitmq.trace	topic	D I		
amq.topic	topic	D	0.00/s	0.00/s

Add a new exchange

HTTP API Server Docs Tutorials Community Support Community Slack Commercial Sup

- Now go to exchanges click on amq.topic and if the messages are publishing or not



Exchange: amq.topic

Overview

Message rates last minute ?

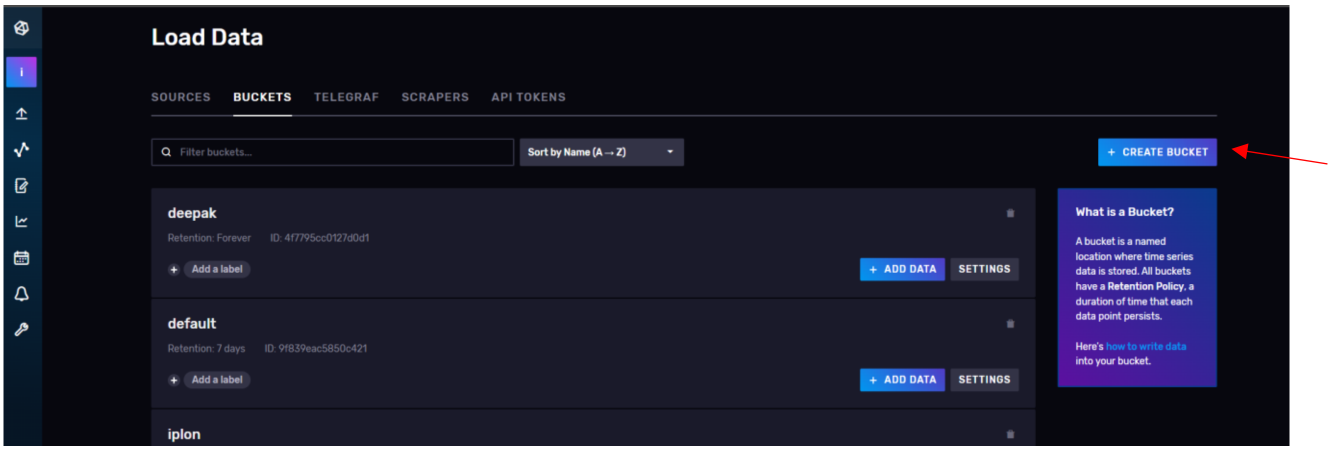
Publish (In)	🟨 0.00/s
Publish (Out)	🟦 0.00/s

Details

Type	topic
Features	durable: true
Policy	

Telegraf Part:

In first create one bucket by clicking create bucket option

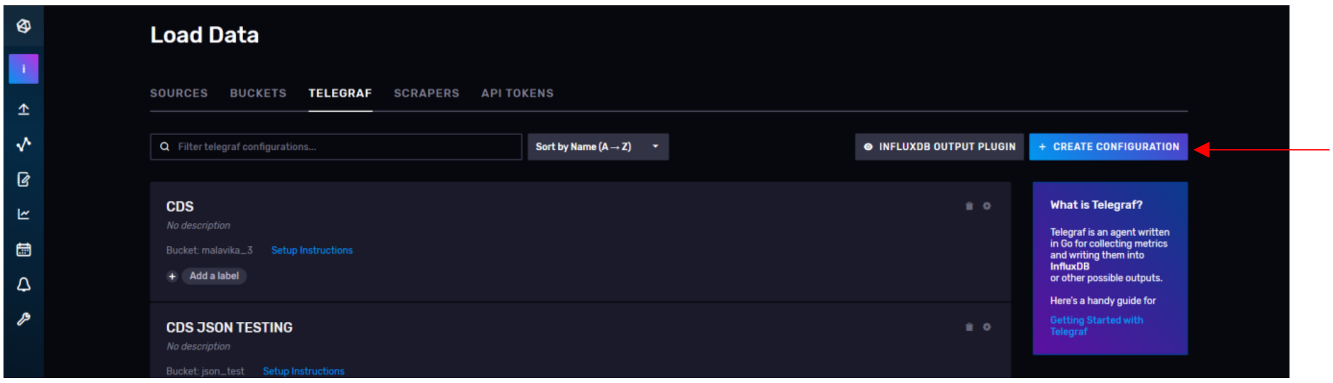


What is a Bucket?

A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.

Here's how to write data into your bucket.

Then click telegraf click create configuration

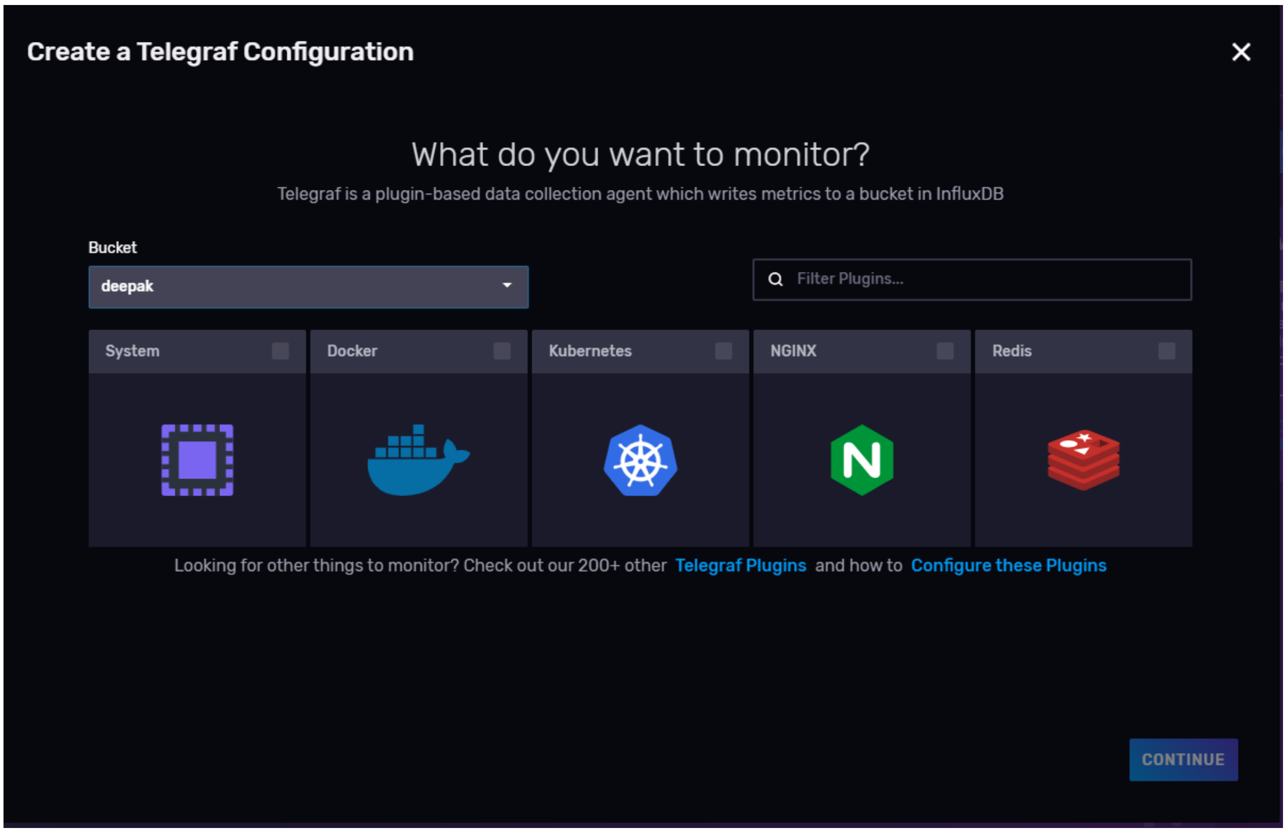


What is Telegraf?

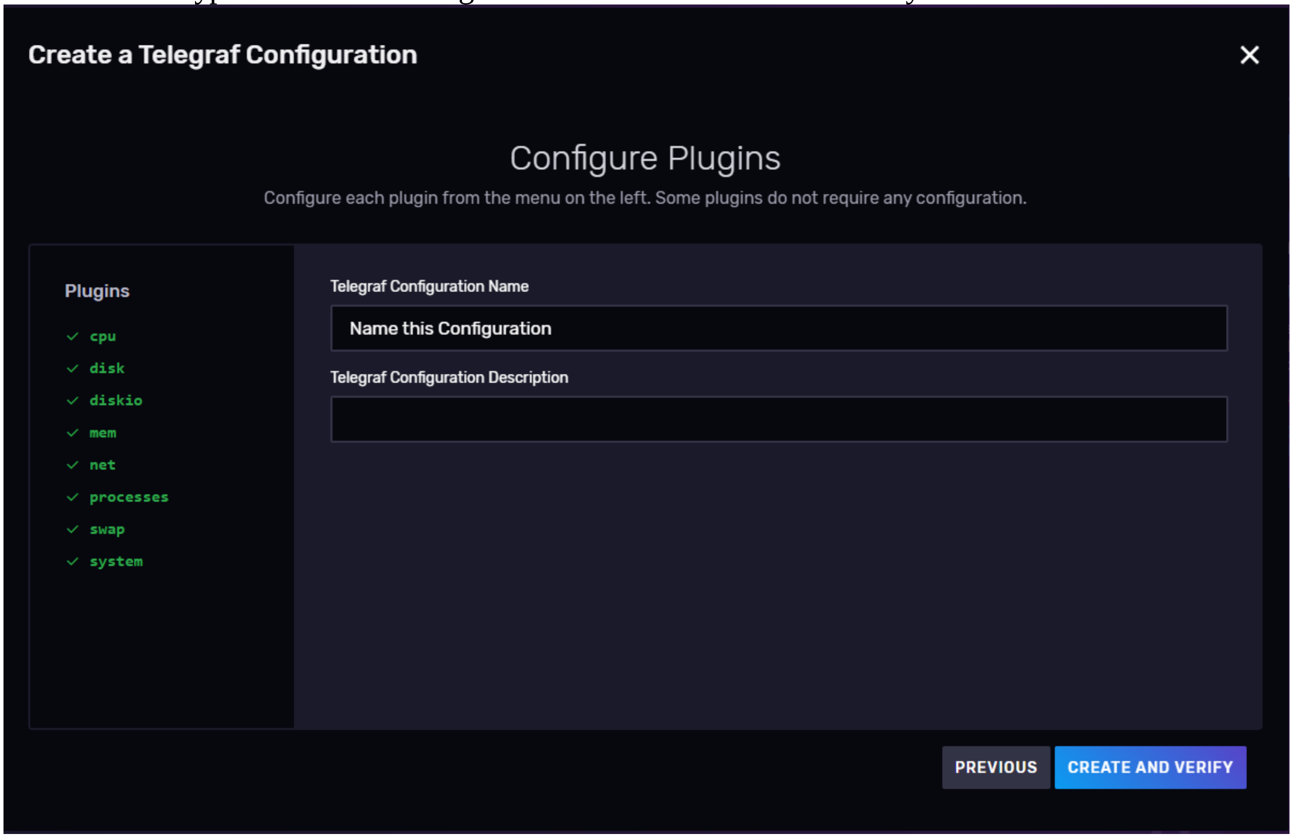
Telegraf is an agent written in Go for collecting metrics and writing them into **InfluxDB** or other possible outputs.

Here's a handy guide for Getting Started with Telegraf

One new window will open there select bucket that you created for the configuration and then click system then continue


Create a Telegraf Configuration
What do you want to monitor?
Telegraf is a plugin-based data collection agent which writes metrics to a bucket in InfluxDB
Bucket
deepak
Filter Plugins...
System
Docker
Kubernetes
NGINX
Redis
Looking for other things to monitor? Check out our 200+ other Telegraf Plugins and how to Configure these Plugins
CONTINUE

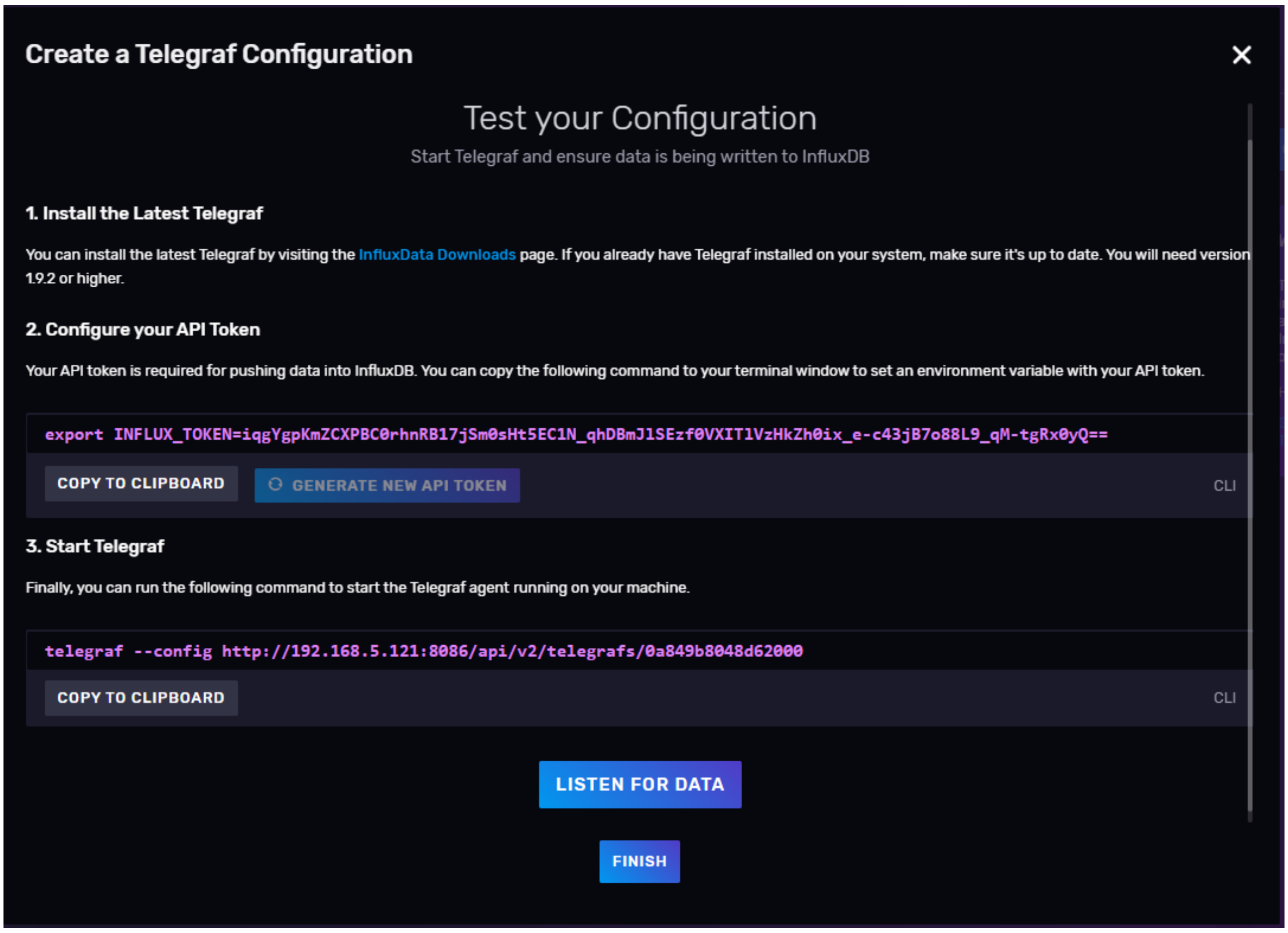
Next window type a name for configuration then click create and verify


Create a Telegraf Configuration
Configure Plugins
Configure each plugin from the menu on the left. Some plugins do not require any configuration.
Plugins
cpu
disk
diskio
mem
net
processes
swap
system
Telegraf Configuration Name
Name this Configuration
Telegraf Configuration Description
PREVIOUS
CREATE AND VERIFY

Again a new window will open just click finish in that

In this **configure your API Token** command is used to connect telegraf with influxdb

Start Telegraf command used to run the applied configuration as a service



Create a Telegraf Configuration

Test your Configuration

Start Telegraf and ensure data is being written to InfluxDB

1. Install the Latest Telegraf

You can install the latest Telegraf by visiting the InfluxData Downloads page. If you already have Telegraf installed on your system, make sure it's up to date. You will need version 1.9.2 or higher.

2. Configure your API Token

Your API token is required for pushing data into InfluxDB. You can copy the following command to your terminal window to set an environment variable with your API token.

```
export INFLUX_TOKEN=iqgYgpKmZCXPBC0rhnRB17jSm0sHt5EC1N_qhDBmJ1SEzf0VXIT1VzHkZh0ix_e-c43jB7o88L9_qM-tgRx0yQ==
```

3. Start Telegraf

Finally, you can run the following command to start the Telegraf agent running on your machine.

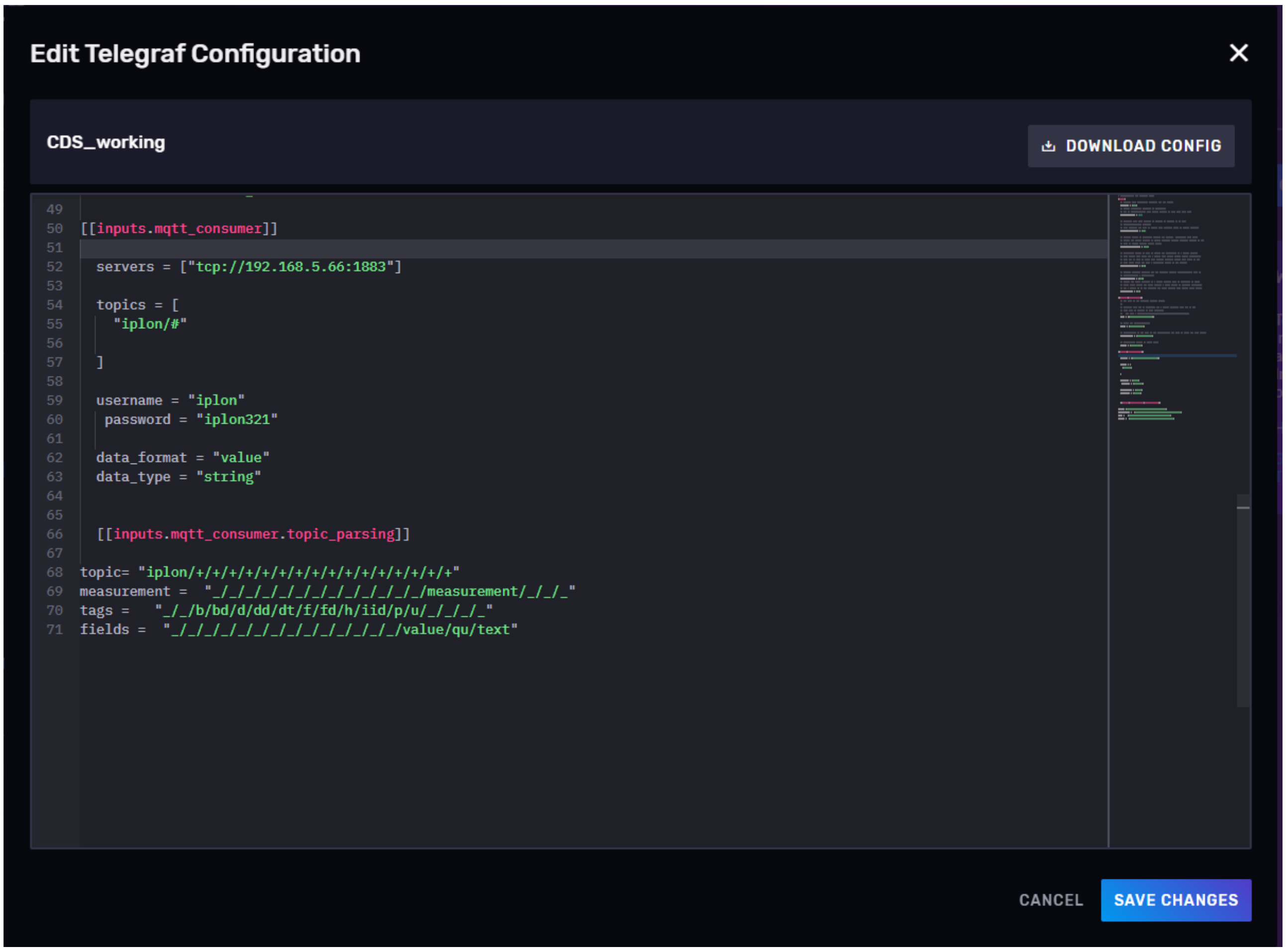
```
telegraf --config http://192.168.5.121:8086/api/v2/telegrafs/0a849b8048d62000
```

Now your configuration created check it in Telegraf configuration list page there you can see your configuration which you created just click on the configuration there you see default input and output plugins. Now you have to append the below mentioned configuration.

```
 [[inputs.mqtt_consumer]]
 ## Broker URLs for the MQTT server or cluster.  To connect to multiple
  ## clusters or standalone servers, use a separate plugin instance.
  ##   example: servers = ["tcp://localhost:1883"]
  ##            servers = ["ssl://localhost:1883"]
  ##            servers = ["ws://localhost:1883"]
  servers = ["tcp://192.168.5.66:1883"]      # serevr ip in which rabbitmq broker with mqtt protocol is running
```

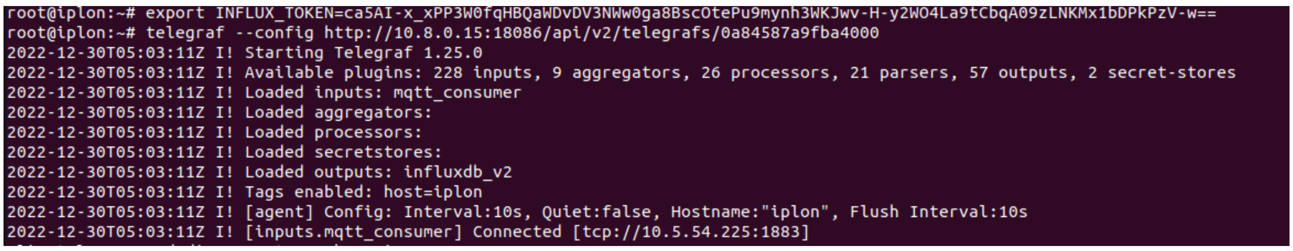
```
## Topics that will be subscribed to.
topics = [
  "iplon/#"   # user defined topic initialized from node-red

]
## The message topic will be stored in a tag specified by this value.  If set
## to the empty string no topic tag will be created.
# topic_tag = "topic"

## QoS policy for messages
##   0 = at most once
##   1 = at least once
##   2 = exactly once
##
## When using a QoS of 1 or 2, you should enable persistent_session to allow
## resuming unacknowledged messages.
# qos = 0

## Connection timeout for initial connection in seconds
# connection_timeout = "30s"

## Maximum messages to read from the broker that have not been written by an
## output.  For best throughput set based on the number of metrics within
## each message and the size of the output's metric_batch_size.
##
## For example, if each message from the queue contains 10 metrics and the
## output metric_batch_size is 1000, setting this to 100 will ensure that a
## full batch is collected and the write is triggered immediately without
## waiting until the next flush_interval.
# max_undelivered_messages = 1000

## Persistent session disables clearing of the client session on connection.
## In order for this option to work you must also set client_id to identify
## the client.  To receive messages that arrived while the client is offline,
## also set the qos option to 1 or 2 and don't forget to also set the QoS when
## publishing.
```

```
  # persistent_session = false

  ## If unset, a random client ID will be generated.
  # client_id = ""

  ## Username and password to connect MQTT server.
  username = "iplon"
   password = "iplon321"

  ## Optional TLS Config
  # tls_ca = "/etc/telegraf/ca.pem"
  # tls_cert = "/etc/telegraf/cert.pem"
  # tls_key = "/etc/telegraf/key.pem"
  ## Use TLS but skip chain & host verification
  # insecure_skip_verify = false

  ## Data format to consume.
  ## Each data format has its own unique set of configuration options, read
  ## more about them here:
  ## https://github.com/influxdata/telegraf/blob/master/docs/DATA_FORMATS_INPUT.md
  # data_format = "influx"
  data_format = "value"
  data_type = "string"

  ## Enable extracting tag values from MQTT topics
  ## _ denotes an ignored entry in the topic path
# In this part only we are adding the tags fields in the format which we are getting from node-red[do not change until or unless you are adding new tags or fields]
  [[inputs.mqtt_consumer.topic_parsing]]

topic= "iplon/+/+/+/+/+/+/+/+/+/+/+/+/+/+/+/+"
measurement =  "_/_/_/_/_/_/_/_/_/_/_/_/_/measurement/_/_/_"
tags =  "_/_/b/bd/d/dd/dt/f/fd/h/iid/p/u/_/_/_/_"
fields =  "_/_/_/_/_/_/_/_/_/_/_/_/_/_/value/qu/text"
```



```
[[inputs.mqtt_consumer]]

  servers = ["tcp://192.168.5.66:1883"]

  topics = [
    "iplon/#"

  ]

  username = "iplon"
   password = "iplon321"

  data_format = "value"
  data_type = "string"


  [[inputs.mqtt_consumer.topic_parsing]]

topic= "iplon/+/+/+/+/+/+/+/+/+/+/+/+/+/+/+"
measurement =  "_/_/_/_/_/_/_/_/_/_/_/_/_/measurement/_/_/_"
tags =   "_/_/b/bd/d/dd/dt/f/fd/h/iid/p/u/_/_/_/_"
fields =  "_/_/_/_/_/_/_/_/_/_/_/_/_/_/value/qu/text"
```

- Then click save changes
- Under telegraf configuration which we created click on setup instruction
- And then under **Configure your API** Token generate new token then run that command in the terminal.
- After run the export token command then run the telegraf configuration command in the same terminal[mostly in this configuration no error will come but if some kind of connectivity, syntax or authorization error then please check inside the configuration about url, topic, username, password and data_format.



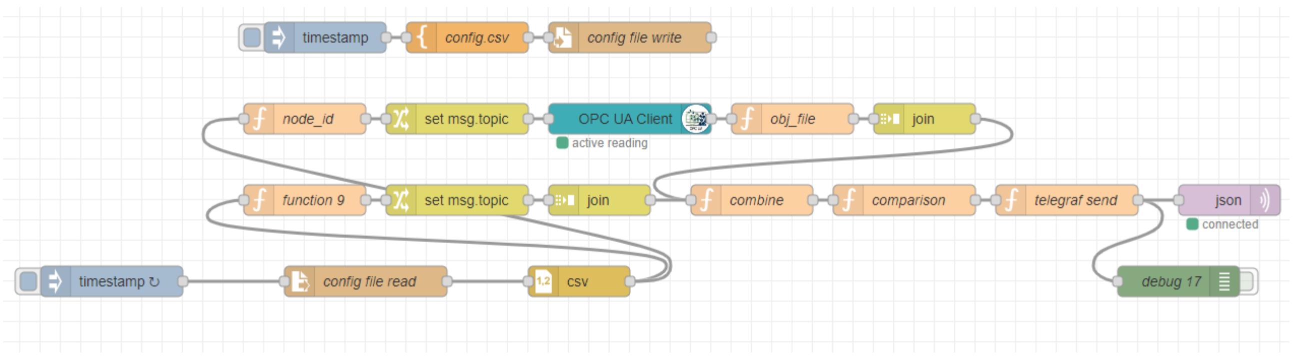
```
root@iplon:~# export INFLUX_TOKEN=ca5AI-x_xPP3W0fqHBQaWDvDV3NWw0ga8BscOtePu9mynh3WKJwv-H-y2WO4La9tCbqA09zLNKMx1bDPkPzV-w==
root@iplon:~# telegraf --config http://10.8.0.15:18086/api/v2/telegrafs/0a84587a9fba4000
2022-12-30T05:03:11Z I! Starting Telegraf 1.25.0
2022-12-30T05:03:11Z I! Available plugins: 228 inputs, 9 aggregators, 26 processors, 21 parsers, 57 outputs, 2 secret-stores
2022-12-30T05:03:11Z I! Loaded inputs: mqtt_consumer
2022-12-30T05:03:11Z I! Loaded aggregators:
2022-12-30T05:03:11Z I! Loaded processors:
2022-12-30T05:03:11Z I! Loaded secretstores:
2022-12-30T05:03:11Z I! Loaded outputs: influxdb_v2
2022-12-30T05:03:11Z I! Tags enabled: host=iplon
2022-12-30T05:03:11Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"iplon", Flush Interval:10s
2022-12-30T05:03:11Z I! [inputs.mqtt_consumer] Connected [tcp://10.5.54.225:1883]
```

- Now go to the buckets list in influxdb and validate the data

2. MQTT-Telegraf Configuration using Json_V2 Method

Node-red Part:

Output getting from this comparison function node is then giving to another function (telegraf send) there we are iterating the incoming array of output and sending that into single message object



Using a function node(telegraf send) by using this logic given below.

```
for (var i = 0; i < msg.payload.length; i++) {
    var obj = {};
    obj.payload =

    {
        Time: msg.payload[i].time,
        value: msg.payload[i].value,
        qu: msg.payload[i].qu,
        text: msg.payload[i].text,
        b: msg.payload[i].b,
        bd: msg.payload[i].bd,
        d: msg.payload[i].d,
        dd: msg.payload[i].dd,
        dt: msg.payload[i].dt,
        f: msg.payload[i].f,
        fd: msg.payload[i].fd,
        h: msg.payload[i].h,
        iid: msg.payload[i].iid,
        m: msg.payload[i].m,
        p: msg.payload[i].p,
        u: msg.payload[i].u,
        mn: msg.payload[i].mn
    }
    node.send(obj);
}

return null;
```

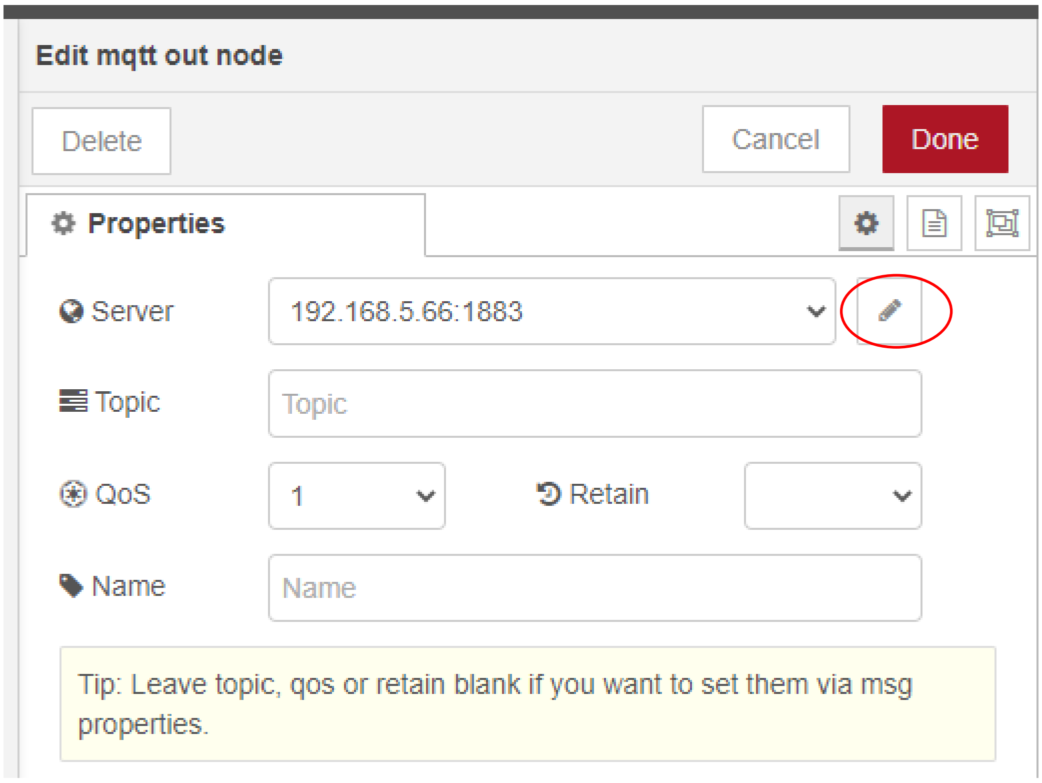
so, we will get output from the function node like this

```
12/30/2022, 4:04:32 PMnode: debug 17msg.payload : Object
object
Time: "2022-12-30T10:30:12"
value: 408
qu: 0
text: "opcua"
b: "B01"
bd: "Block_1"
d: "INV1"
dd: "INV1"
dt: "INV"
f: "PDC"
fd: "DC_POWER"
h: "server_7712"
iid: 7712
m: 1
p: "Technic_solar"
u: "kW"
mn: "v"
```

This output we are giving to mqtt out node

Double click and open the node

Set up configuration of mqtt by clicking the pencil icon

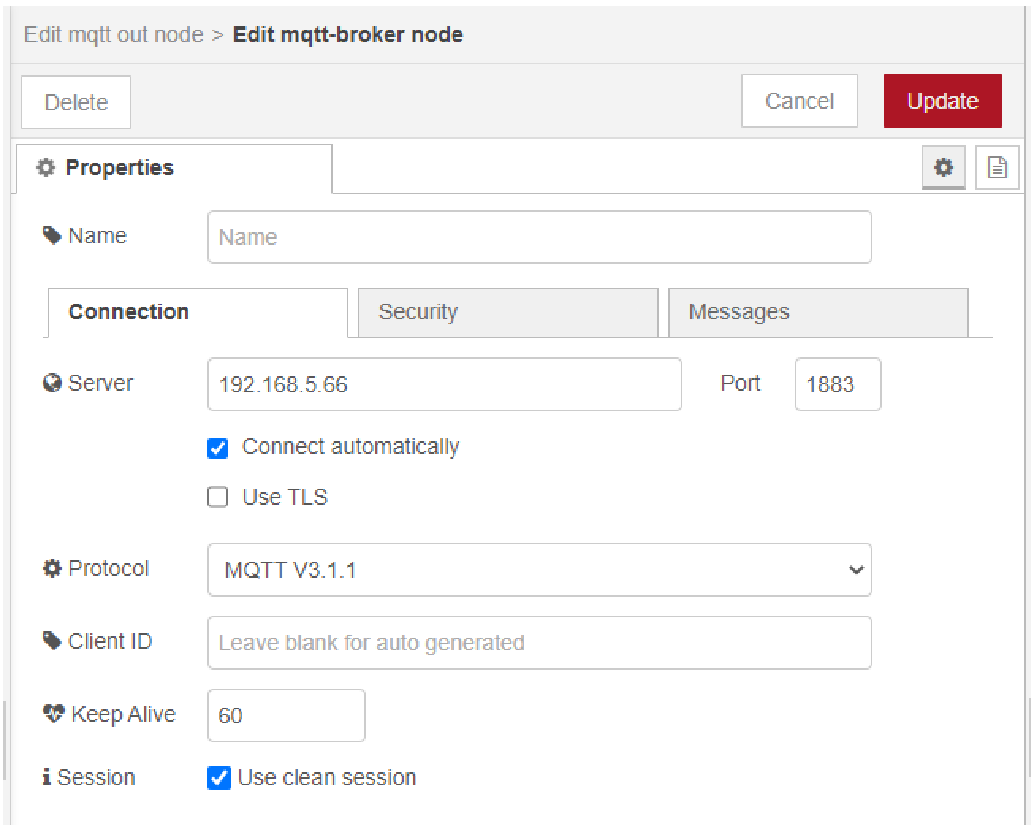


Edit mqtt out node

Server: 192.168.5.66:1883
Topic: Topic
QoS: 1
Retain:
Name: Name

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

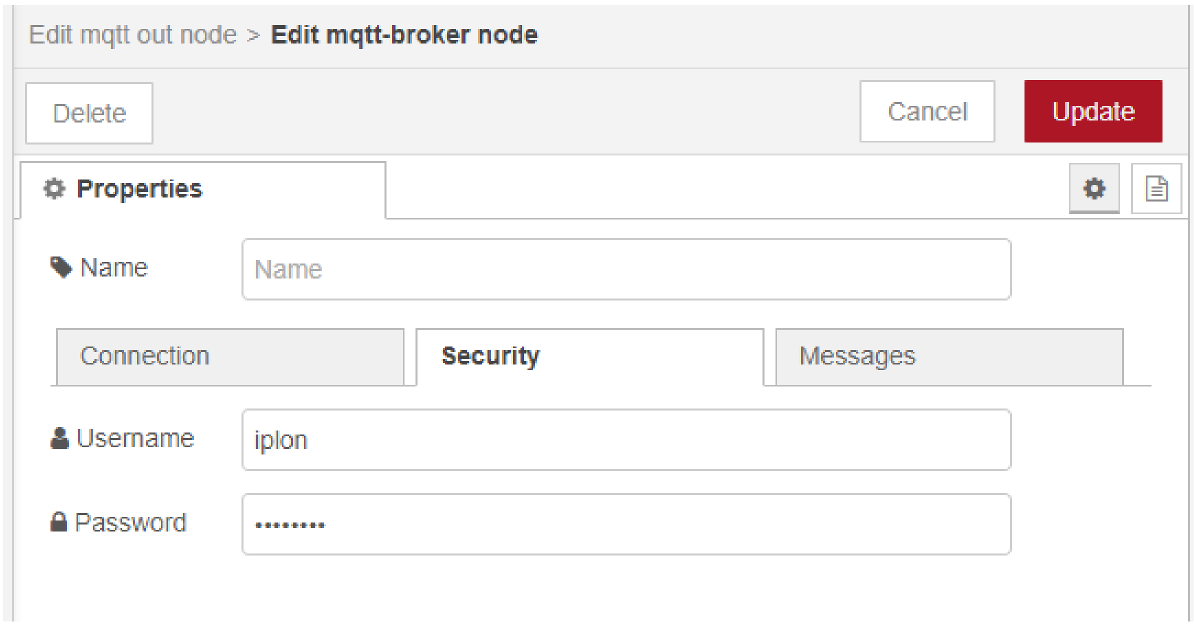
In connection give sever ip, port and select protocol



Edit mqtt out node > Edit mqtt-broker node

Name: Name
Server: 192.168.5.66 Port: 1883
☑ Connect automatically
☐ Use TLS
Protocol: MQTT V3.1.1
Client ID: Leave blank for auto generated
Keep Alive: 60
Session: ☑ Use clean session

Give username and password in security



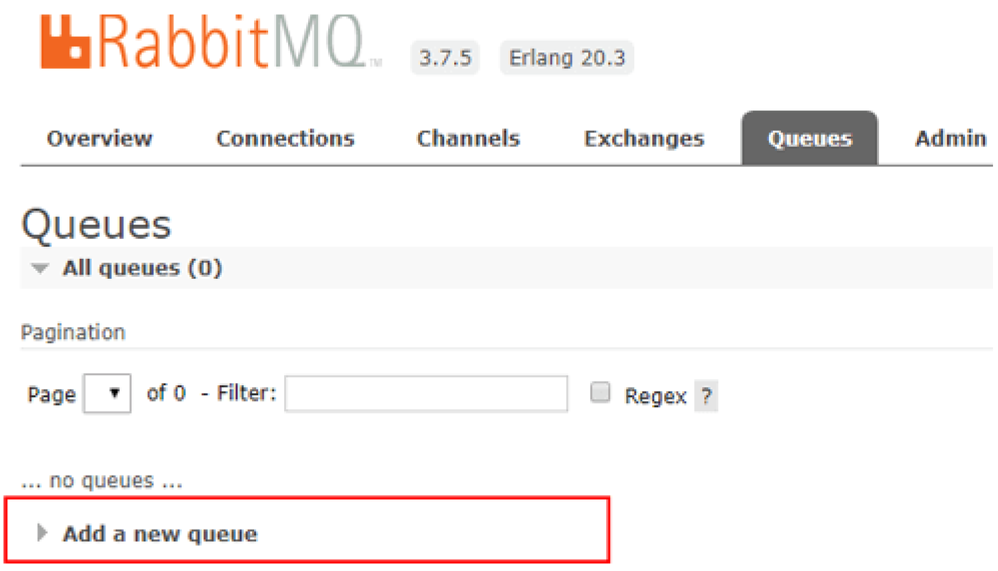
Name: Name
Username: iplon
Password: ••••••••

Rabbitmq Part:

installation guide for ubuntu: https://linuxhint.com/install-rabbitmq-ubuntu/
docker container installation and plugin enabling guide:https://tewarid.github.io/2019/02/15/mqtt-with-rabbitmq-and-node-red.html

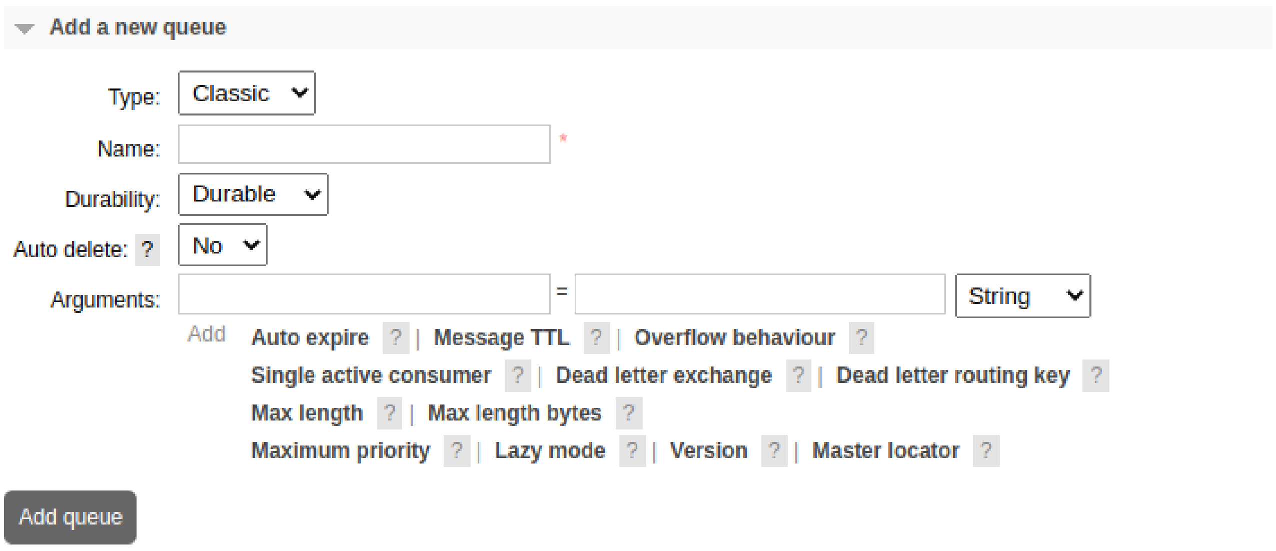
after installation we can able to log into the management interface at http://localhost:15672 using username/password guest/guest,

- Now we should create queue in rabbitmq navigate to **Queues** tab, you will see “**Add a new queue**” just click on that panel to expand like as shown below.



➢

After clicking on **Add a new queue** option, a new panel will open and that will contain a different properties to create a new queue like as shown below.


Add a new queue

Type: Classic
Name:
Durability: Durable
Auto delete: No
Arguments: = String

Add
Auto expire ? | Message TTL ? | Overflow behaviour ?
Single active consumer ? | Dead letter exchange ? | Dead letter routing key ?
Max length ? | Max length bytes ?
Maximum priority ? | Lazy mode ? | Version ? | Master locator ?
Add queue

1. queue type there is 3 type of queues available in rabbitmq

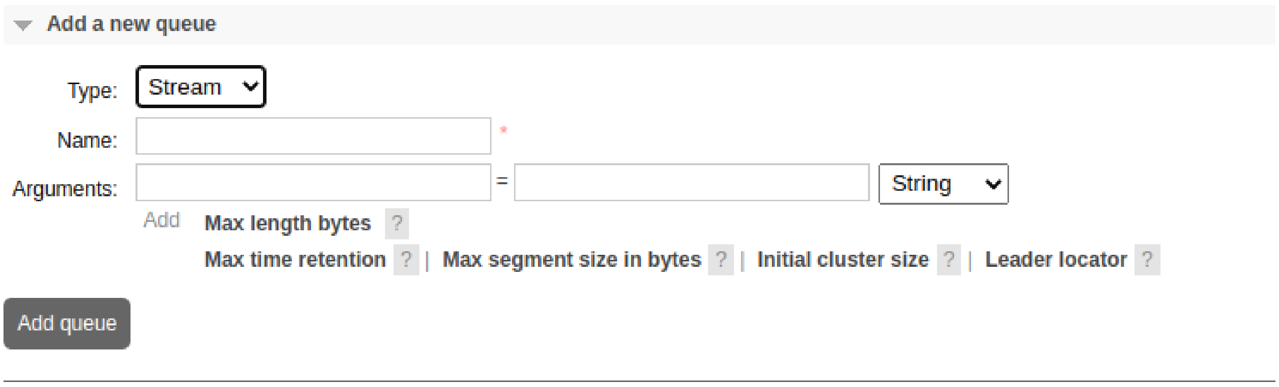
1)classic

2)quorum

3)stream

2. Name
3. Durable (the queue will survive a broker restart)

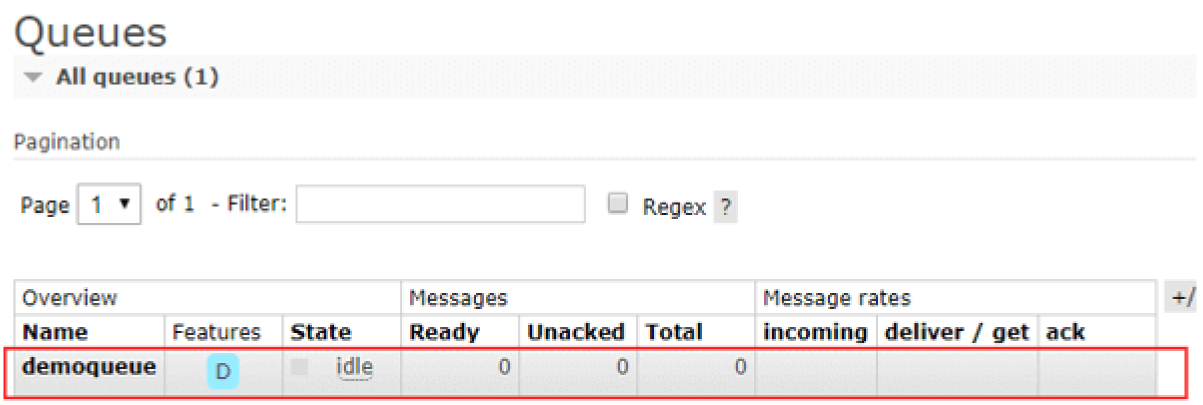
Exclusive (used by only one connection and the queue will be deleted when that connection closes)
4. Auto-delete (queue that has had at least one consumer is deleted when last consumer unsubscribes)
5. Arguments (optional; used by plugins and broker-specific features such as message TTL, queue length limit, etc)
to know more about queues,arguments settings check
https://www.rabbitmq.com/queues.html#basics
https://www.tutlane.com/tutorial/rabbitmq/rabbitmq-queues


Add a new queue
Type: Stream
Name:
Arguments: =
String
Add
Max length bytes ?
Max time retention ? | Max segment size in bytes ? | Initial cluster size ? | Leader locator ?
Add queue

2nd is name box is to give a naming to the queue

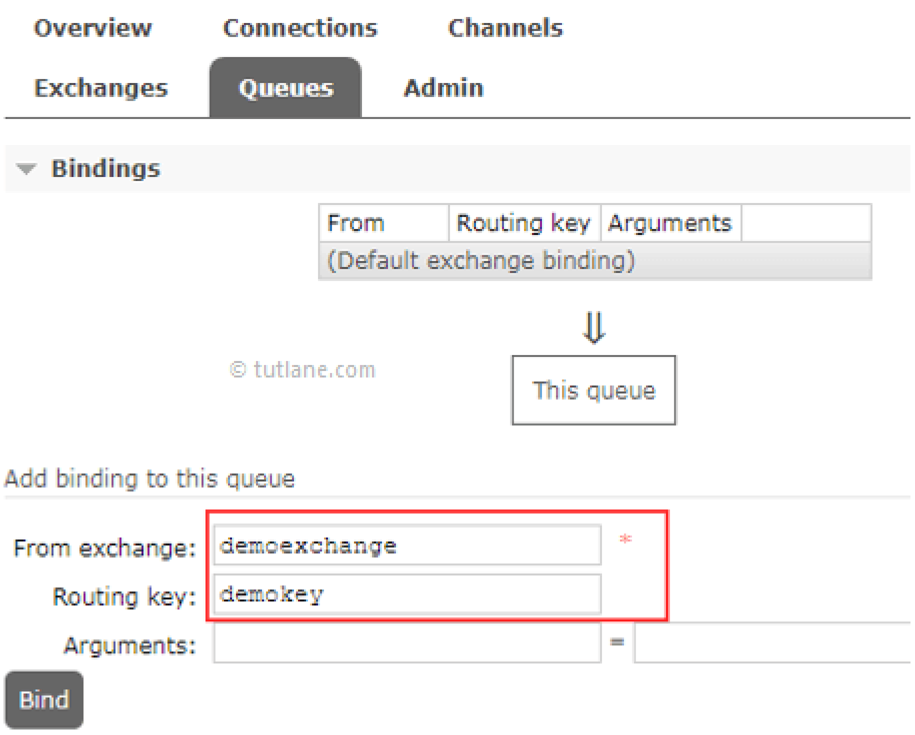
3rd **Arguments** (optional; used by plugins and broker-specific features such as message TTL, queue length limit, etc)

➢ After creating a queue, you can view queue which you have recently added, it is located just above the add queue panel like as shown below.



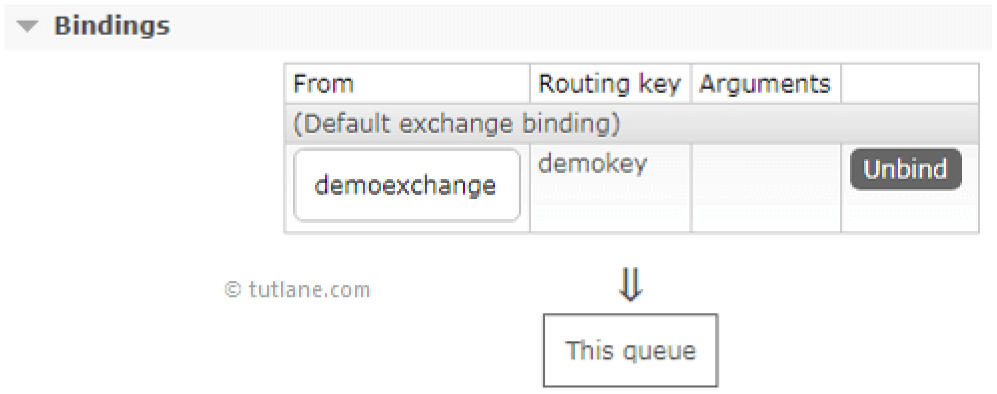
Overview			Messages			Message rates		
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
demoqueue	D	idle	0	0	0			

After click on queue (**demoqueue**) name, the **Bindings** panel will expand and next it will ask for the exchange name, enter exchange ,routing key name which we have added in node-red mqtt node setup and and give any argument like[x-dead-letter-routing-key]click on **Bind** button.



From exchange:	demoexchange
Routing key:	demokey
Arguments:	=

After click on **Bind** button, the defined exchange will be bind to our queue and that will be like as shown below.

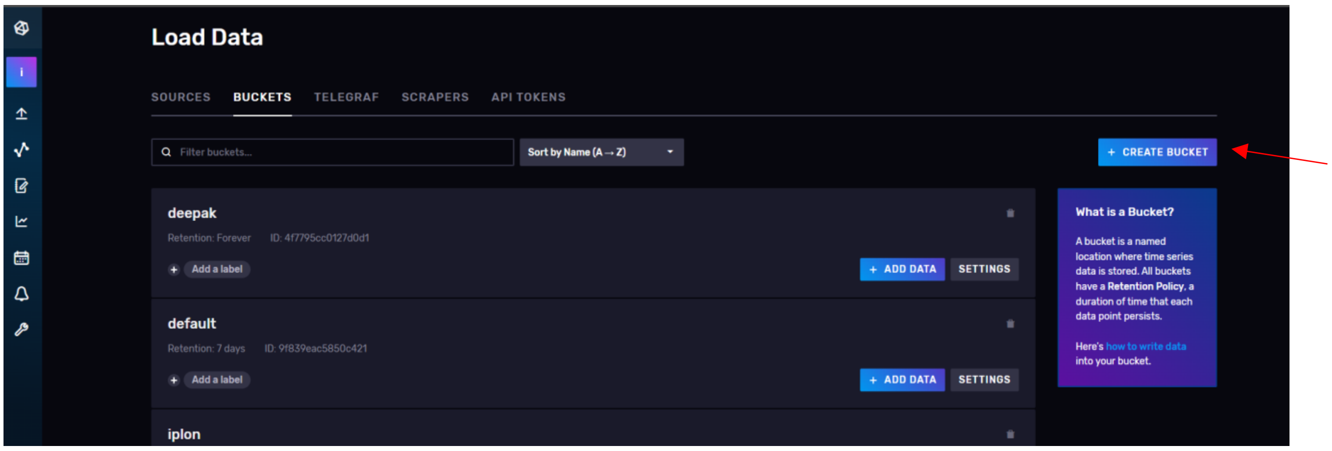


From	Routing key	Arguments	
(Default exchange binding)			
demoexchange	demokey		Unbind

After binding, in case if you want to unbind it then you can click on unbind button to remove binding.

Telegraf Part:

In first create one bucket by clicking create bucket option



deepak
Retention: Forever ID: 4f7795cc0127d0d1

default
Retention: 7 days ID: 9f839eac5850c421

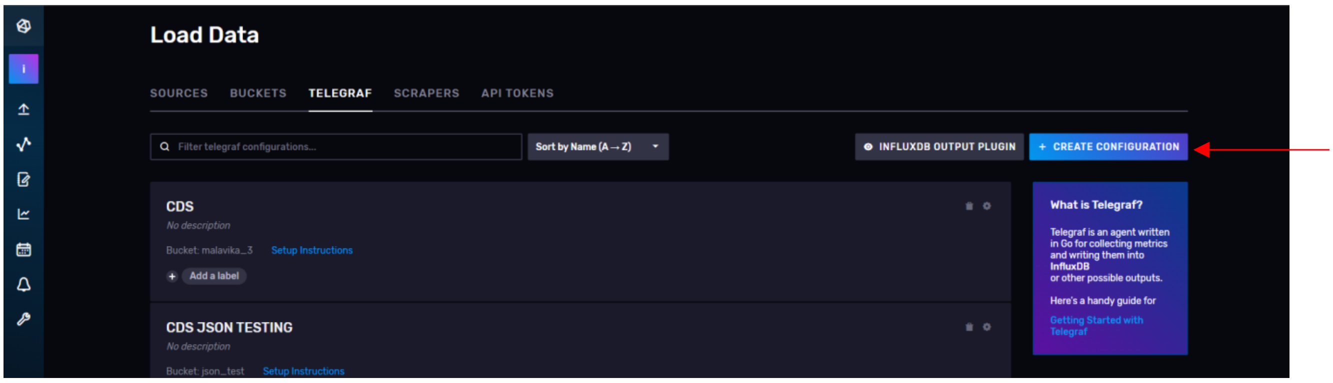
iplon

What is a Bucket?

A bucket is a named location where time series data is stored. All buckets have a **Retention Policy**, a duration of time that each data point persists.

Here's how to write data into your bucket.

Then click telegraf click create configuration



CDS
No description
Bucket: malavika_3

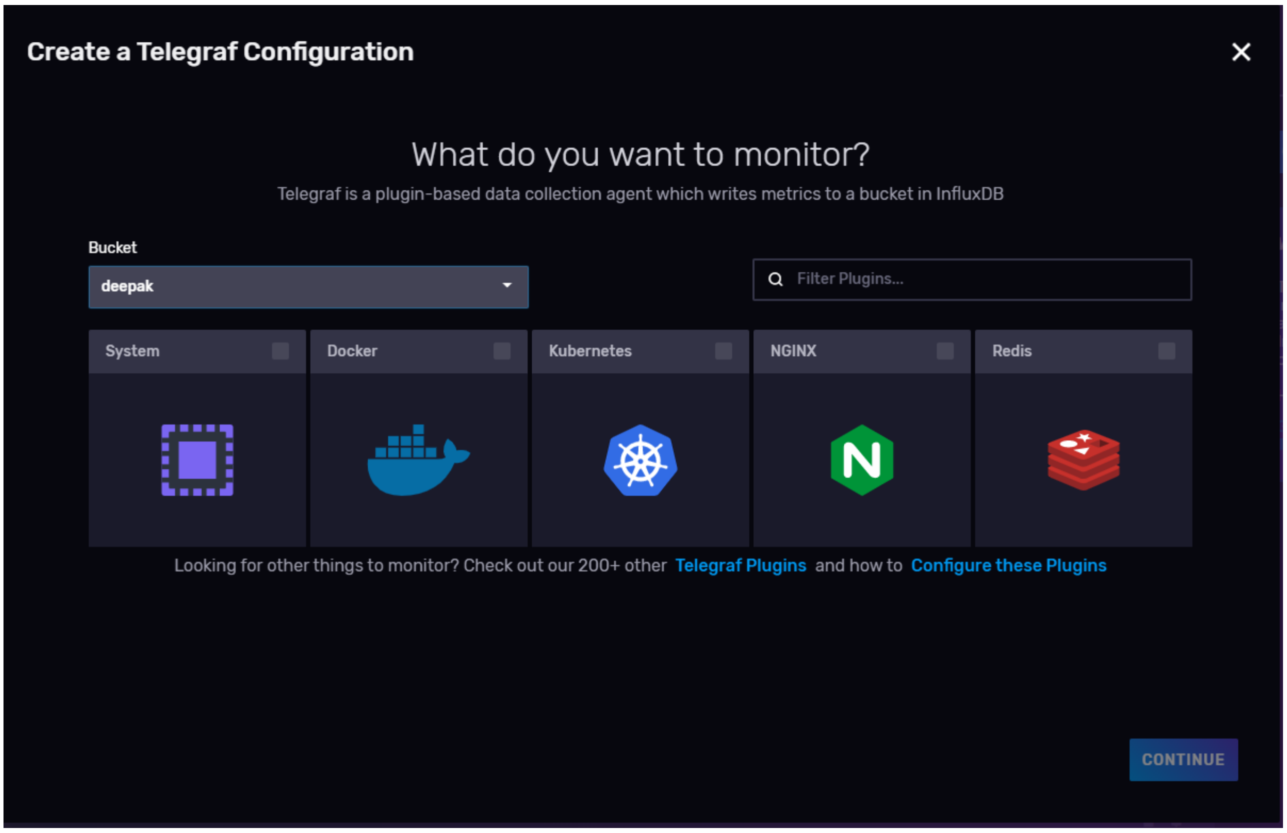
CDS JSON TESTING
No description
Bucket: json_test

What is Telegraf?

Telegraf is an agent written in Go for collecting metrics and writing them into **InfluxDB** or other possible outputs.

Here's a handy guide for Getting Started with Telegraf

One new window will open there select bucket that you created for the configuration and then click system then continue


Create a Telegraf Configuration

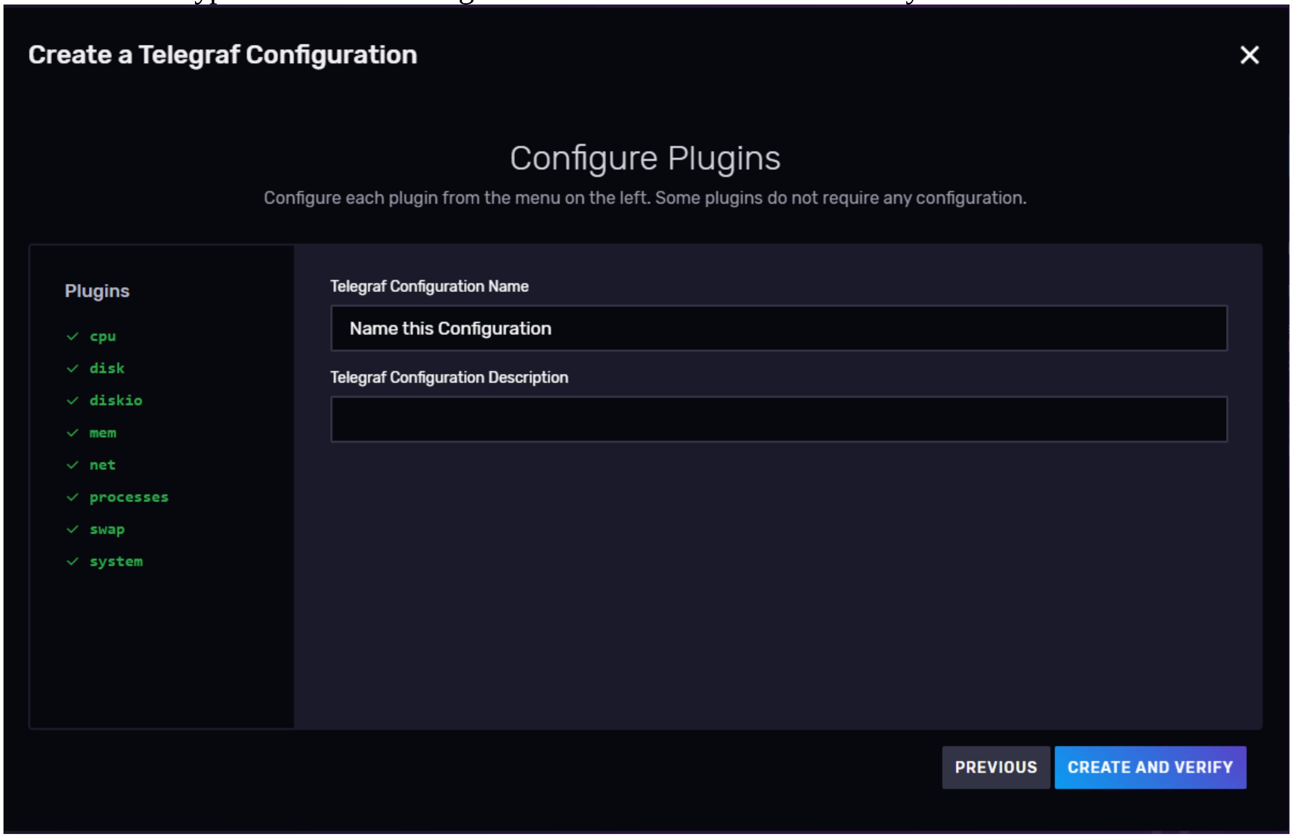
What do you want to monitor?

Telegraf is a plugin-based data collection agent which writes metrics to a bucket in InfluxDB

Bucket: deepak

Looking for other things to monitor? Check out our 200+ other Telegraf Plugins and how to Configure these Plugins

Next window type a name for configuration then click create and verify


Create a Telegraf Configuration

Configure Plugins

Configure each plugin from the menu on the left. Some plugins do not require any configuration.

Plugins
- ✓ cpu
- ✓ disk
- ✓ diskio
- ✓ mem
- ✓ net
- ✓ processes
- ✓ swap
- ✓ system

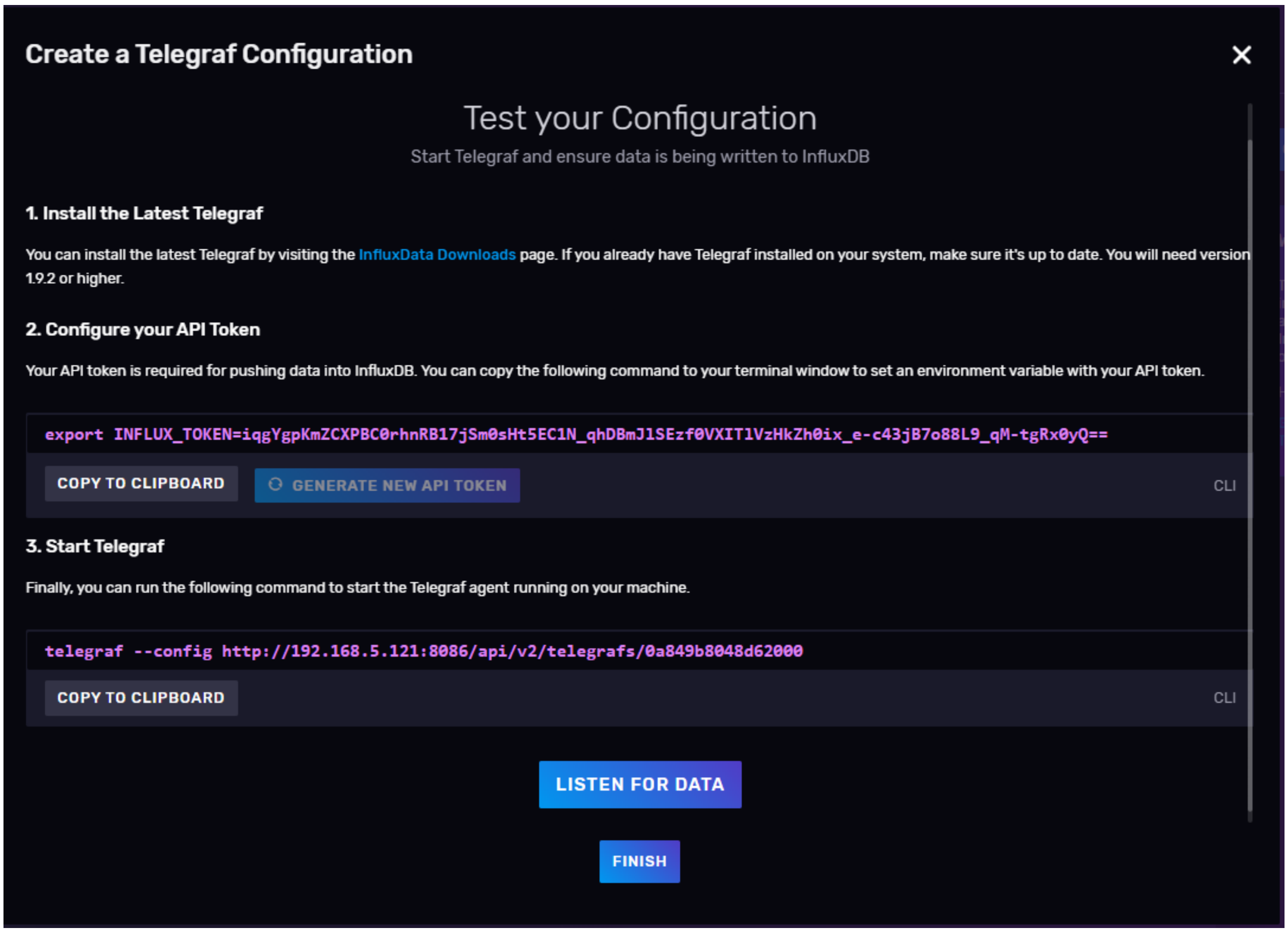
Telegraf Configuration Name: Name this Configuration

Telegraf Configuration Description:

Again a new window will open just click finish in that

In this **configure your API Token** command is used to connect telegraf with influxdb

Start Telegraf command used to run the applied configuration as a service



Create a Telegraf Configuration

Test your Configuration

Start Telegraf and ensure data is being written to InfluxDB

1. Install the Latest Telegraf

You can install the latest Telegraf by visiting the InfluxData Downloads page. If you already have Telegraf installed on your system, make sure it's up to date. You will need version 1.9.2 or higher.

2. Configure your API Token

Your API token is required for pushing data into InfluxDB. You can copy the following command to your terminal window to set an environment variable with your API token.

```
export INFLUX_TOKEN=iqgYgpKmZCXPBC0rhnRB17jSm0sHt5EC1N_qhDBmJ1SEzf0VXIT1VzHkZh0ix_e-c43jB7o88L9_qM-tgRx0yQ==
```

3. Start Telegraf

Finally, you can run the following command to start the Telegraf agent running on your machine.

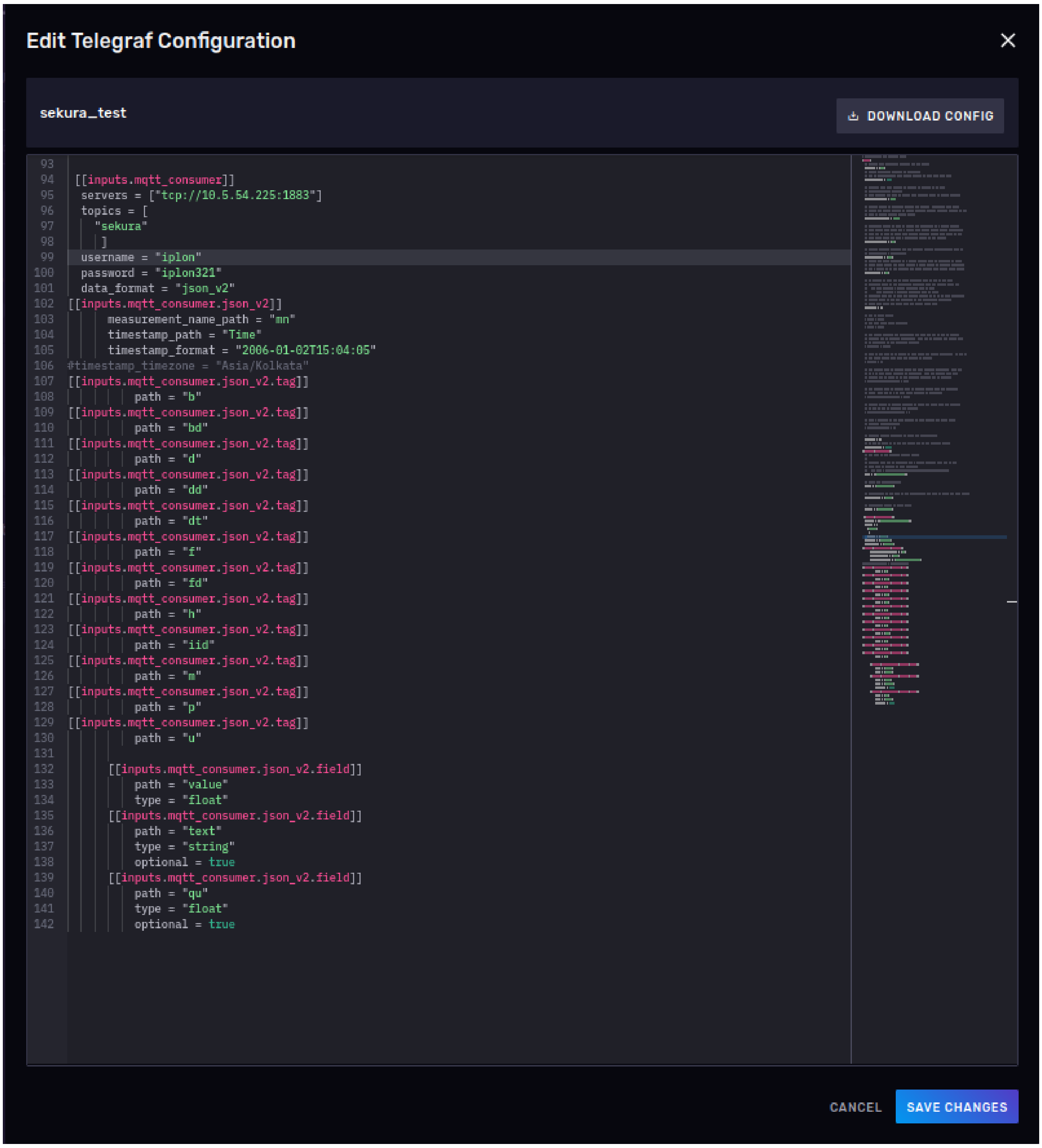
```
telegraf --config http://192.168.5.121:8086/api/v2/telegrafs/0a849b8048d62000
```

Now your configuration created check it in Telegraf configuration list page there you can see your configuration which you created just click on the configuration there you see default input and output plugins. Now you have to append the below mentioned configuration.

```
[[inputs.mqtt_consumer]]
 servers = ["tcp://10.5.54.225:1883"]
 topics = [
   "sekura"
   ]
 username = "iplon"
```

```
  password = "iplon321"
  data_format = "json_v2"
 [[inputs.mqtt_consumer.json_v2]]
     measurement_name_path = "mn"
     timestamp_path = "Time"
     timestamp_format = "2006-01-02T15:04:05"
 #timestamp_timezone = "Asia/Kolkata"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "b"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "bd"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "d"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "dd"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "dt"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "f"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "fd"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "h"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "iid"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "m"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "p"
 [[inputs.mqtt_consumer.json_v2.tag]]
        path = "u"
```

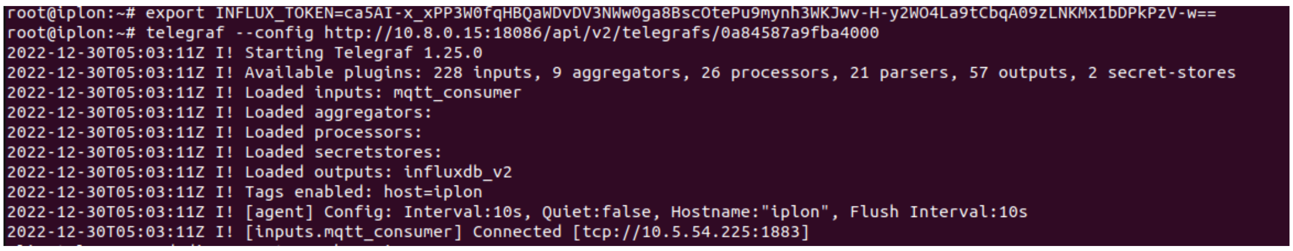
```
[[inputs.mqtt_consumer.json_v2.field]]
    path = "value"
    type = "float"
[[inputs.mqtt_consumer.json_v2.field]]
    path = "text"
    type = "string"
    optional = true
[[inputs.mqtt_consumer.json_v2.field]]
    path = "qu"
    type = "float"
    optional = true
```



```

 [[inputs.mqtt_consumer]]
  servers = ["tcp://10.5.54.225:1883"]
  topics = [
    "sekura"
     ]
  username = "iplon"
  password = "iplon321"
  data_format = "json_v2"
[[inputs.mqtt_consumer.json_v2]]
      measurement_name_path = "mn"
      timestamp_path = "Time"
      timestamp_format = "2006-01-02T15:04:05"
#timestamp_timezone = "Asia/Kolkata"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "b"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "bd"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "d"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "dd"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "dt"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "f"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "fd"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "h"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "iid"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "m"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "p"
[[inputs.mqtt_consumer.json_v2.tag]]
          path = "u"

      [[inputs.mqtt_consumer.json_v2.field]]
          path = "value"
          type = "float"
      [[inputs.mqtt_consumer.json_v2.field]]
          path = "text"
          type = "string"
          optional = true
      [[inputs.mqtt_consumer.json_v2.field]]
          path = "qu"
          type = "float"
          optional = true
```

- Then click save changes
- Under telegraf configuration which we created click on setup instruction
- And then under **Configure your API** Token generate new token then run that command in the terminal.
- After run the export token command then run the telegraf configuration command in the same terminal[mostly in this configuration no error will come but if some

kind of connectivity, syntax or authorization error then please check inside the configuration about url, topic, username, password and data_format.

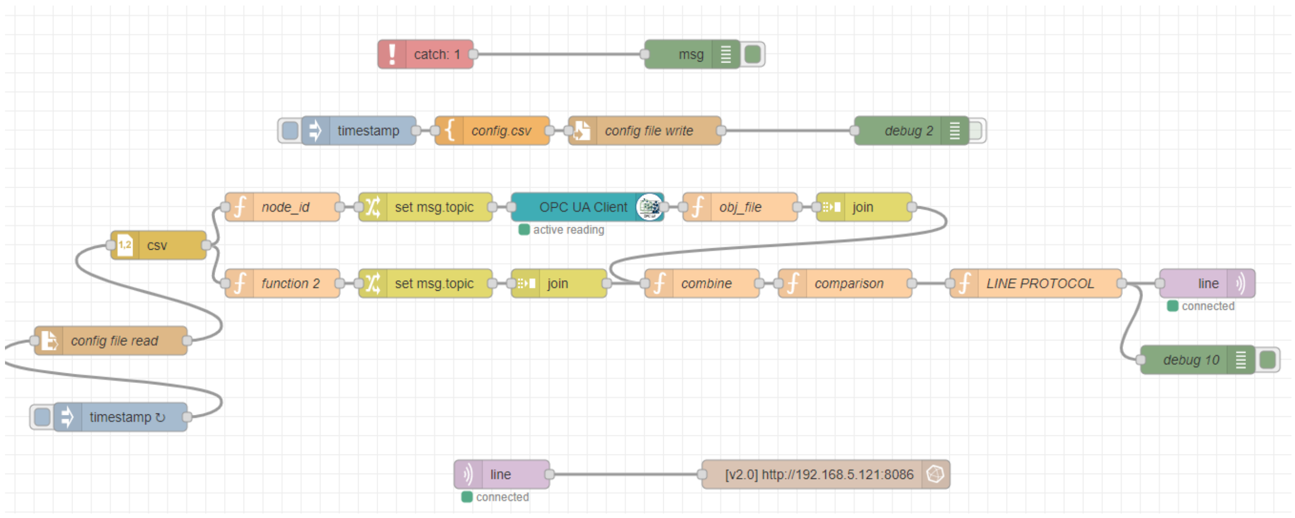


```
root@iplon:~# export INFLUX_TOKEN=ca5AI-x_xPP3W0fqHBQaWDvDV3NWw0ga8BscOtePu9mynh3WKJwv-H-y2WO4La9tCbqA09zLNKMx1bDPkPzV-w==
root@iplon:~# telegraf --config http://10.8.0.15:18086/api/v2/telegrafs/0a84587a9fba4000
2022-12-30T05:03:11Z I! Starting Telegraf 1.25.0
2022-12-30T05:03:11Z I! Available plugins: 228 inputs, 9 aggregators, 26 processors, 21 parsers, 57 outputs, 2 secret-stores
2022-12-30T05:03:11Z I! Loaded inputs: mqtt_consumer
2022-12-30T05:03:11Z I! Loaded aggregators:
2022-12-30T05:03:11Z I! Loaded processors:
2022-12-30T05:03:11Z I! Loaded secretstores:
2022-12-30T05:03:11Z I! Loaded outputs: influxdb_v2
2022-12-30T05:03:11Z I! Tags enabled: host=iplon
2022-12-30T05:03:11Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"iplon", Flush Interval:10s
2022-12-30T05:03:11Z I! [inputs.mqtt_consumer] Connected [tcp://10.5.54.225:1883]
```

- Now go to the buckets list in influxdb and validate the data

3. MQTT-Influxdb Configuration by direct method[No Telegraf Part]Node-red Part:

Output getting from this comparison function node is then giving to another function (**LINE PROTOCOL**) there we are iterating the incoming array of output structuring it into line protocol format and sending that into single message object



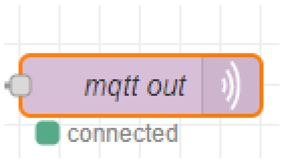
Using a function node(LINE PROTOCOL) by using this logic given below.

```
for (var i = 0; i < msg.payload.length; i++) {

    var obj = {};
    obj.payload =
        [
    {
        measurement: "v",
            fields: {
            Value: msg.payload[i].value,
            qu: msg.payload[i].qu,
            text: msg.payload[i].text
        },
        tags: {
            b: msg.payload[i].b,
            bd: msg.payload[i].bd,
            d: msg.payload[i].d,
            dd: msg.payload[i].dd,
            dt: msg.payload[i].dt,
            f: msg.payload[i].f,
            fd: msg.payload[i].fd,
            h: msg.payload[i].h,
            iid: msg.payload[i].iid,
            m: msg.payload[i].m,
            p: msg.payload[i].p,
            u: msg.payload[i].u,
            mn: msg.payload[i].mn
        },
            time: msg.payload[i].time
    },
        ]
    node.send(obj);
}
return null;
```

so, we will get output from the function node like this

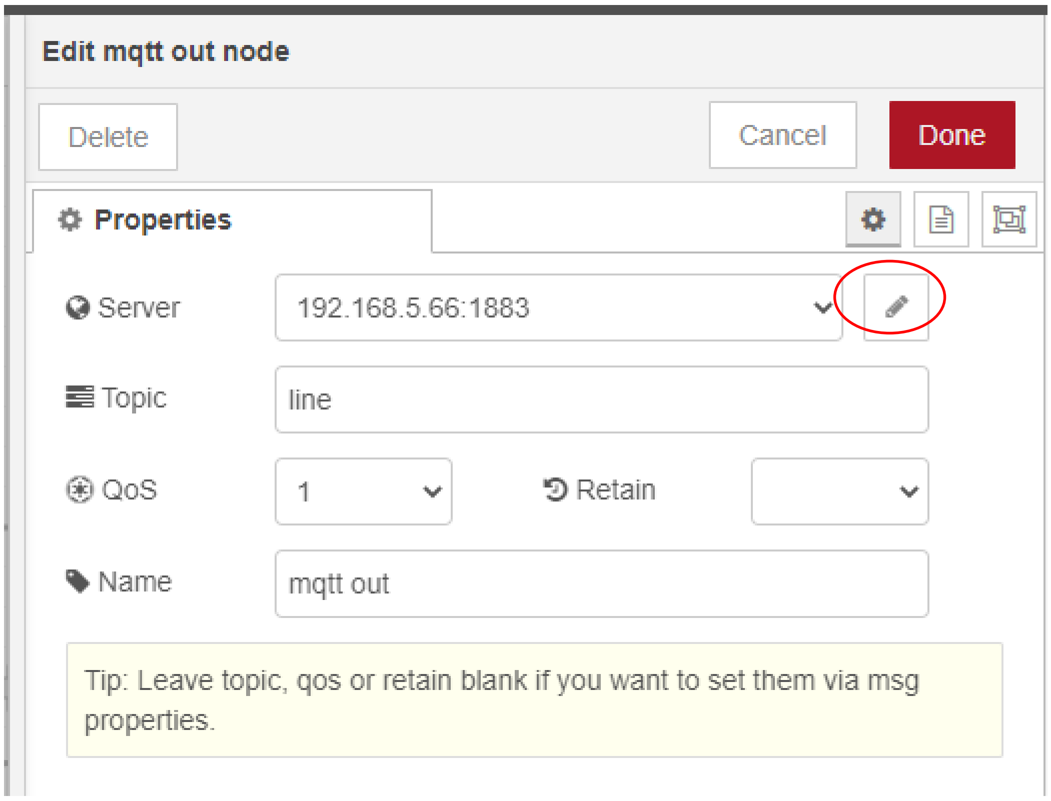
```
12/30/2022, 4:24:08 PMnode: debug 10msg.payload : array[1]
array[1]
0: object
measurement: "v"
fields: object
Value: 323.75
qu: 0
text: "opcua"
tags: object
b: "B01"
bd: "Block_1"
d: "INV1"
dd: "INV1"
dt: "INV"
f: "PDC"
fd: "DC_POWER"
h: "server_7712"
iid: 7712
m: 1
p: "Technic_solar"
u: "kW"
mn: "v"
time: "2022-12-30T10:45:25.148Z"
```

This output we are giving to mqtt out node



Double click and open the node

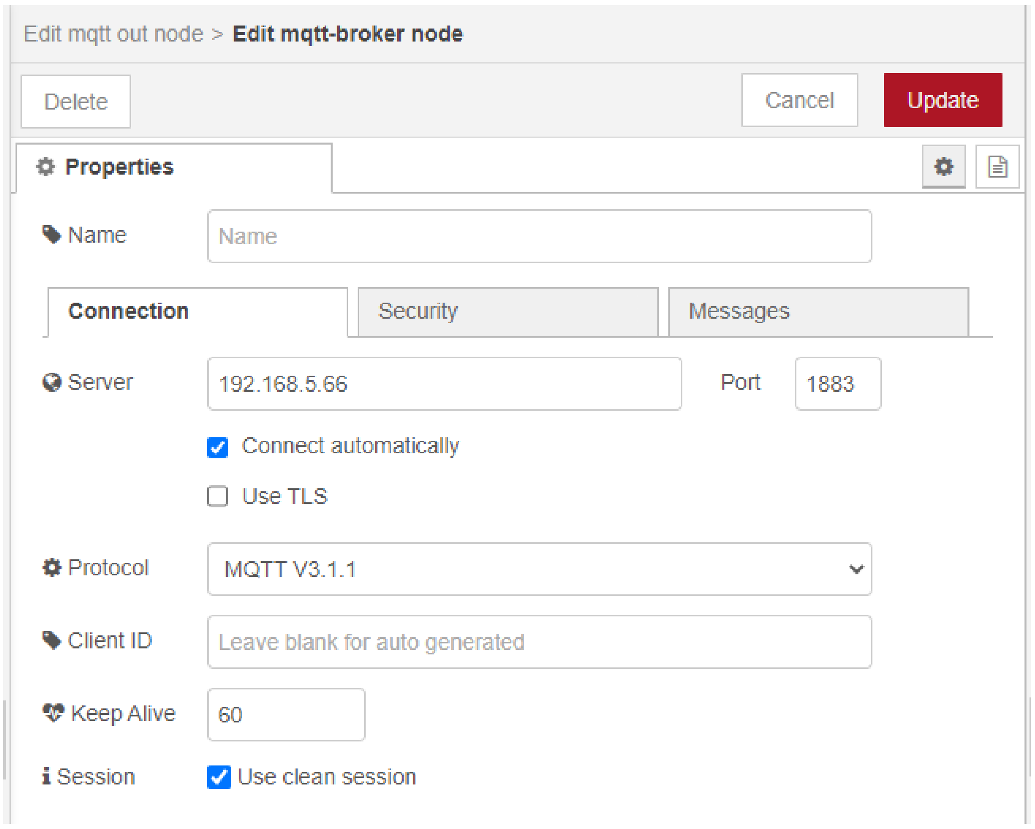
Set up configuration of mqtt by clicking the pencil icon



Server: 192.168.5.66:1883
Topic: line
QoS: 1
Retain:
Name: mqtt out

Tip: Leave topic, qos or retain blank if you want to set them via msg properties.

In connection give sever ip, port and select protocol



Name: Name

Server: 192.168.5.66 Port: 1883

☑ Connect automatically

☐ Use TLS

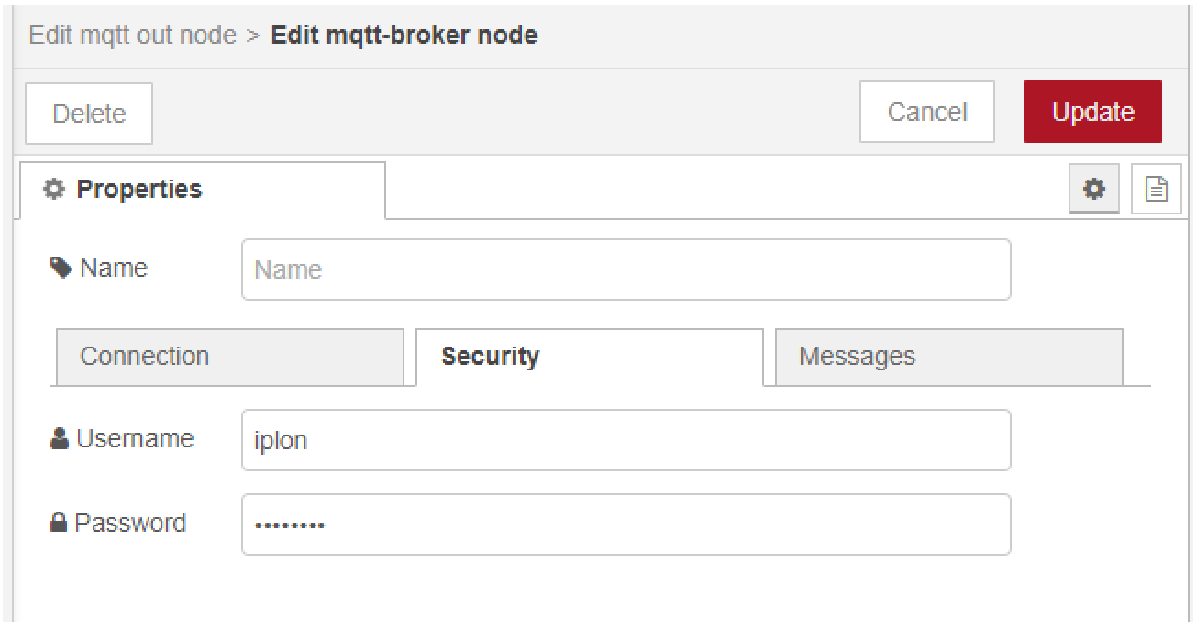
Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☑ Use clean session

Give username and password in security

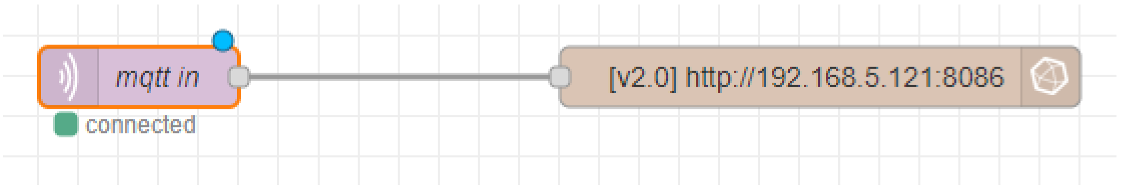


Name: Name

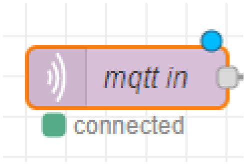
Username: iplon

Password: ••••••••

Mqtt message consume and sent that to influxdb batch node flow

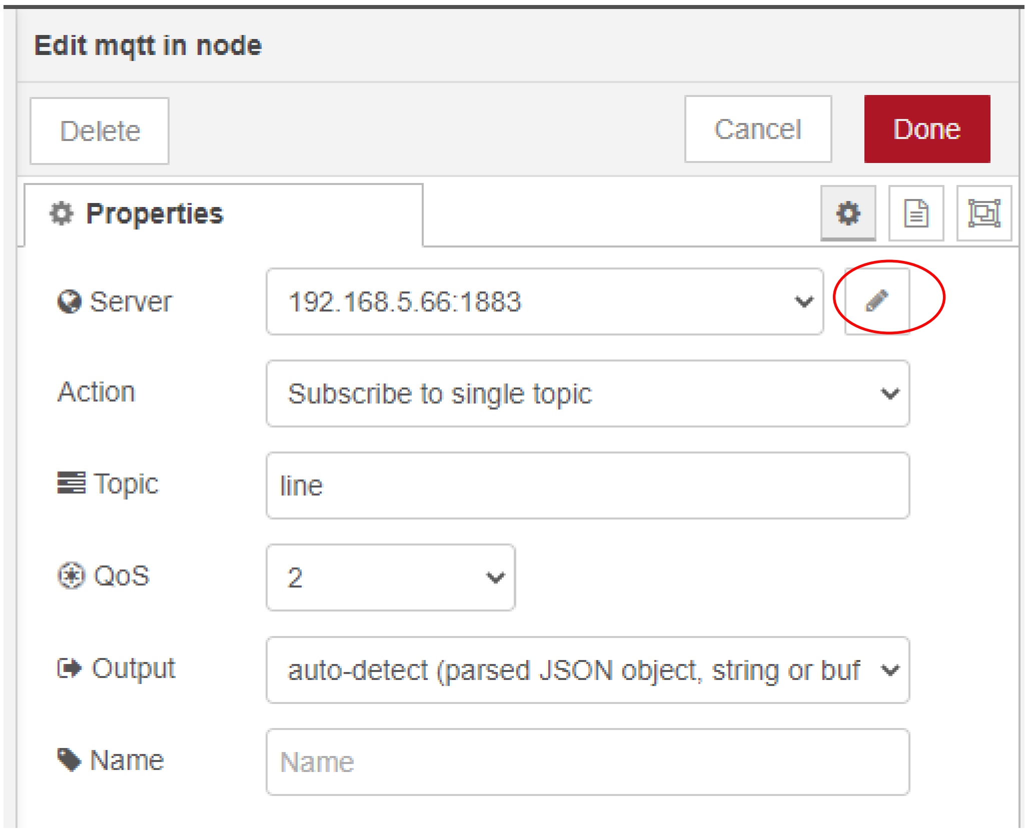


Take mqtt in node



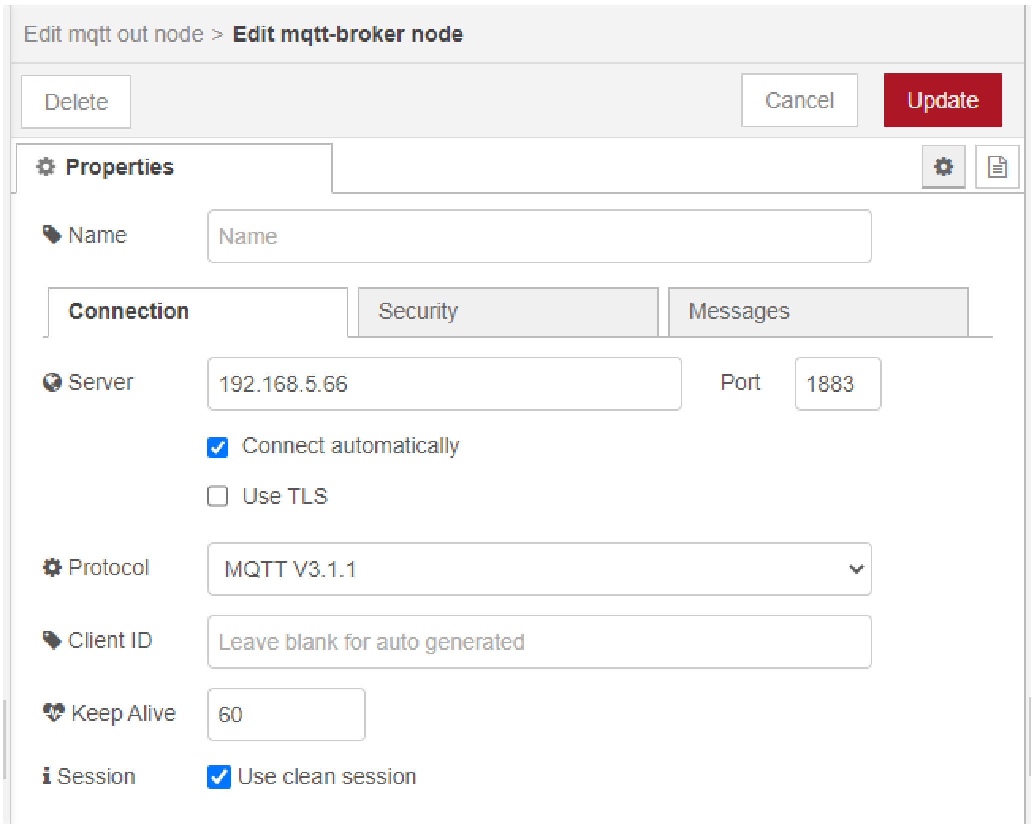
Double click and open the node

Set up configuration of mqtt by clicking the pencil icon topic is what you gave to mqtt out node topic shoud be the same what we gave in mqtt out node and rabbitmq binding part



Server: 192.168.5.66:1883
Action: Subscribe to single topic
Topic: line
QoS: 2
Output: auto-detect (parsed JSON object, string or buf
Name: Name

In connection give sever ip, port and select protocol


Edit mqtt out node > **Edit mqtt-broker node**

Delete Cancel Update

Properties

Name: Name

Connection | Security | Messages

Server: 192.168.5.66 Port: 1883

☑ Connect automatically

☐ Use TLS

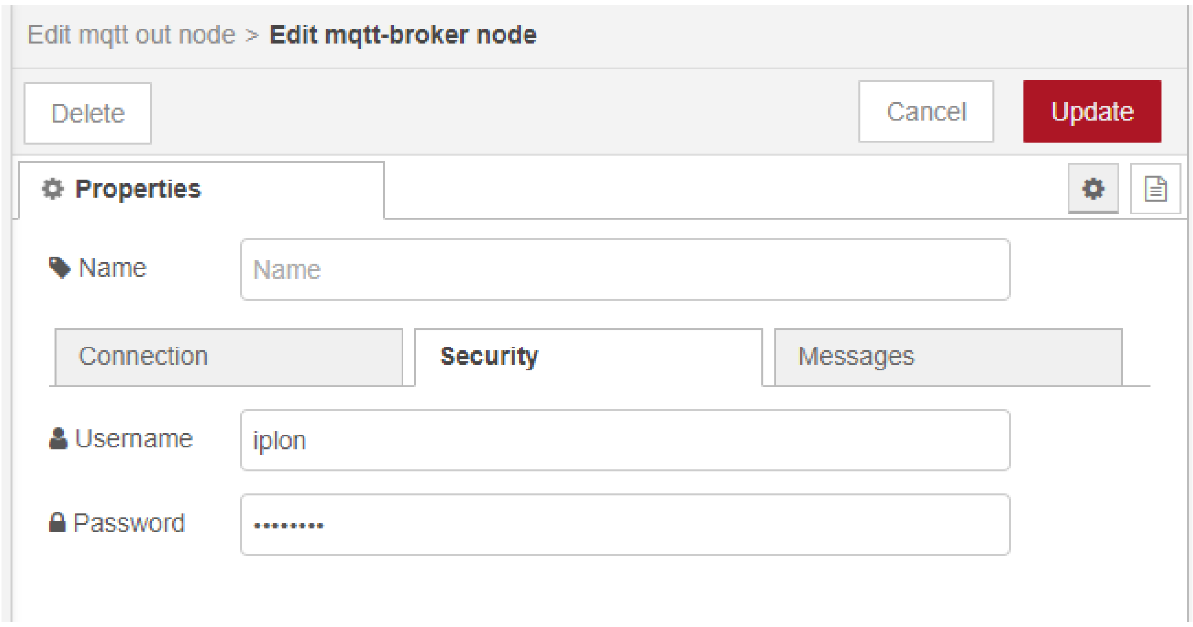
Protocol: MQTT V3.1.1

Client ID: Leave blank for auto generated

Keep Alive: 60

Session: ☑ Use clean session

Give username and password in security


Edit mqtt out node > **Edit mqtt-broker node**

Delete Cancel Update

Properties

Name: Name

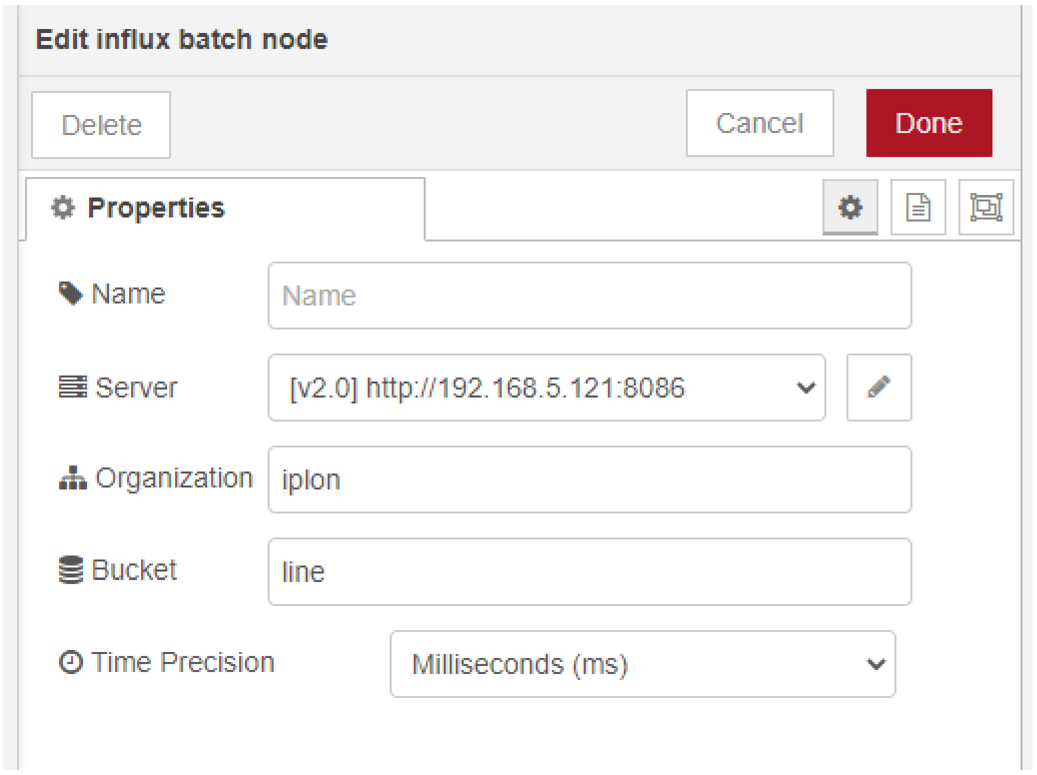
Connection | **Security** | Messages

Username: iplon

Password: ••••••••

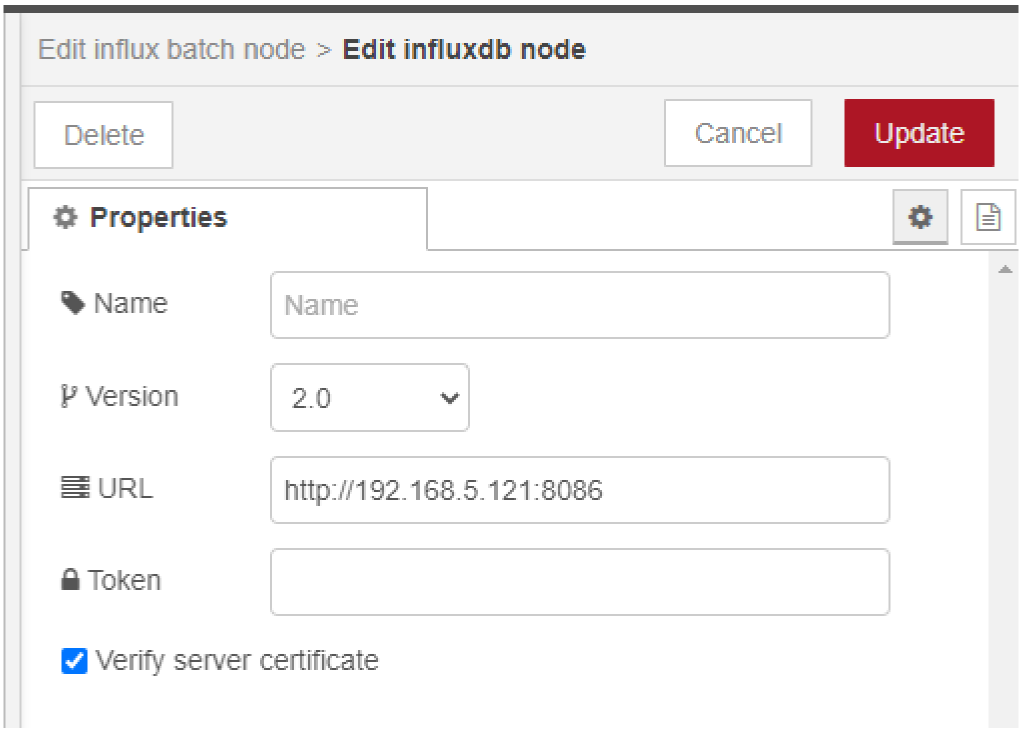
Influxdb batch node

Give server configuration like version, url and token of influxdb organization



Name: Name
Server: [v2.0] http://192.168.5.121:8086
Organization: iplon
Bucket: line
Time Precision: Milliseconds (ms)

Give organization, bucket name[which you have to create inside influxdb before fill here]



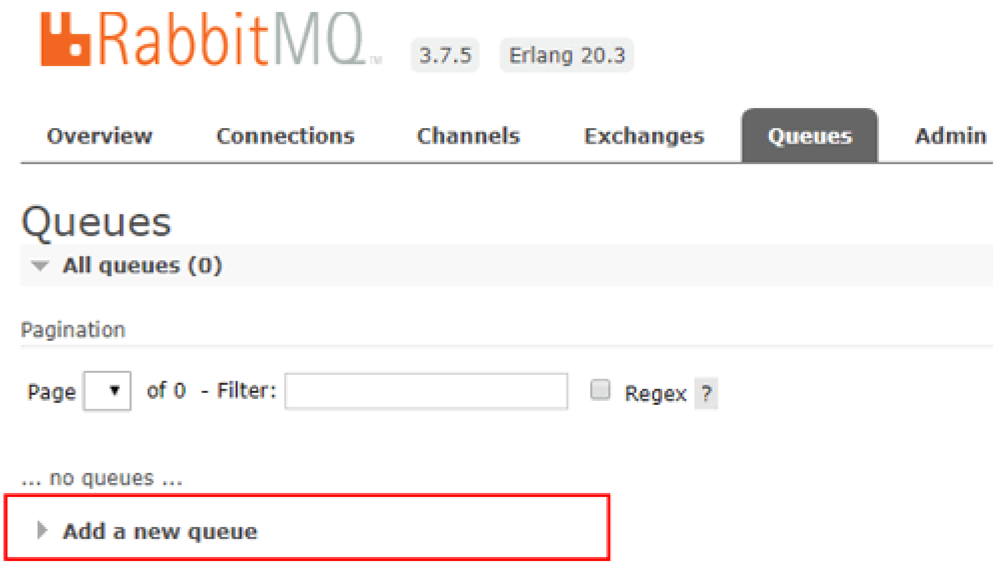
Name: Name
Version: 2.0
URL: http://192.168.5.121:8086
Token:
☑ Verify server certificate

Rabbitmq Part:

installation guide for ubuntu: https://linuxhint.com/install-rabbitmq-ubuntu/
docker container installation and plugin enabling guide:https://tewarid.github.io/2019/02/15/mqtt-with-rabbitmq-and-node-red.html

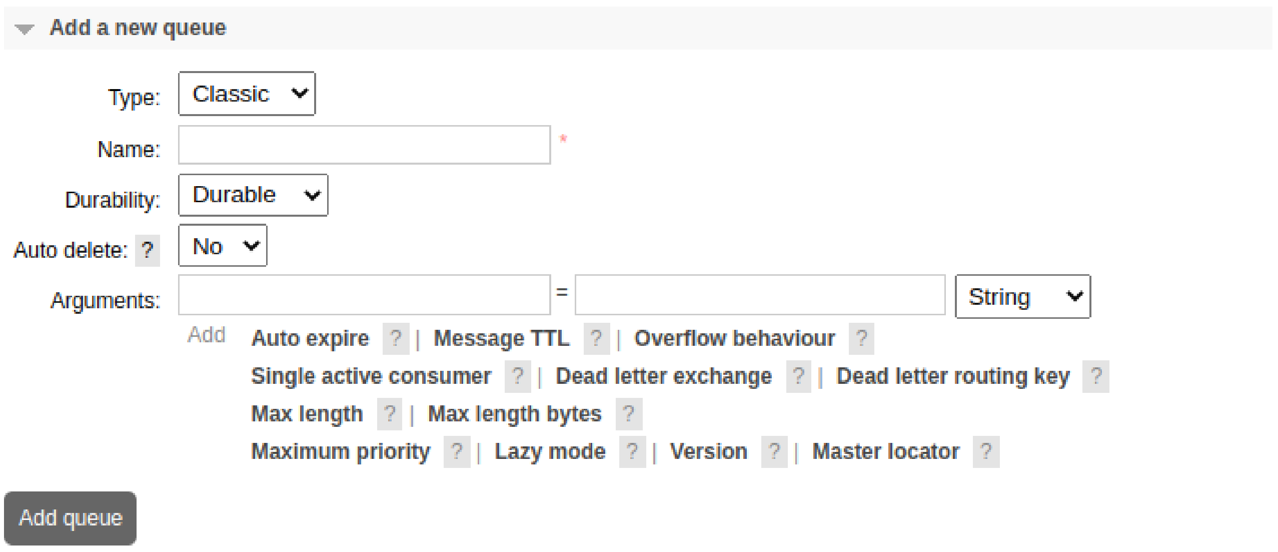
after installation we can able to log into the management interface at http://localhost:15672 using username/password guest/guest,

- Now we should create queue in rabbitmq navigate to **Queues** tab, you will see "**Add a new queue**" just click on that panel to expand like as shown below.



-

After clicking on **Add a new queue** option, a new panel will open and that will contain a different properties to create a new queue like as shown below.


Add a new queue

Type: Classic
Name:
Durability: Durable
Auto delete: ? No
Arguments: = String

Add Auto expire ? | Message TTL ? | Overflow behaviour ?
Single active consumer ? | Dead letter exchange ? | Dead letter routing key ?
Max length ? | Max length bytes ?
Maximum priority ? | Lazy mode ? | Version ? | Master locator ?

Add queue

1. queue type there is 3 type of queues available in rabbitmq

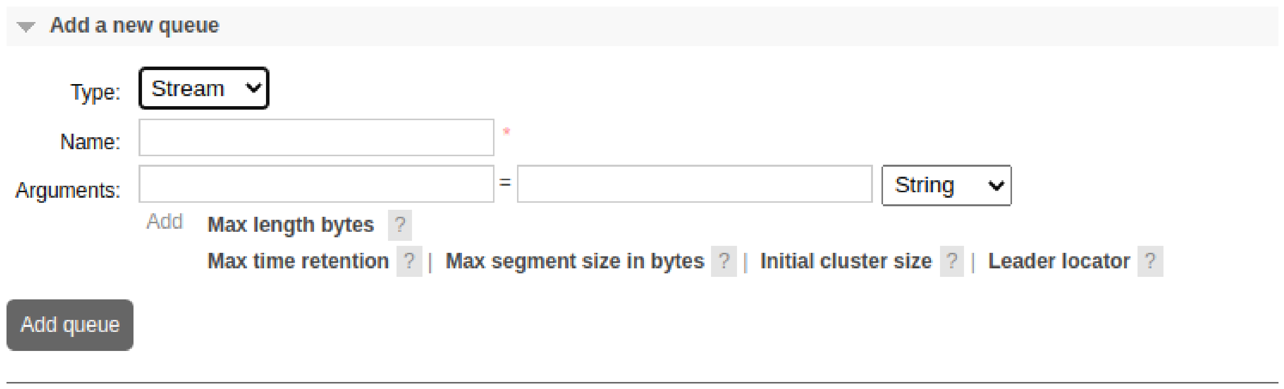
1)classic

2)quorum

3)stream

2. Name
3. Durable (the queue will survive a broker restart)
    Exclusive (used by only one connection and the queue will be deleted when that connection closes)
4. Auto-delete (queue that has had at least one consumer is deleted when last consumer unsubscribes)
5. Arguments (optional; used by plugins and broker-specific features such as message TTL, queue length limit, etc)

to know more about queues,arguments settings check
https://www.rabbitmq.com/queues.html#basics
https://www.tutlane.com/tutorial/rabbitmq/rabbitmq-queues



Add a new queue

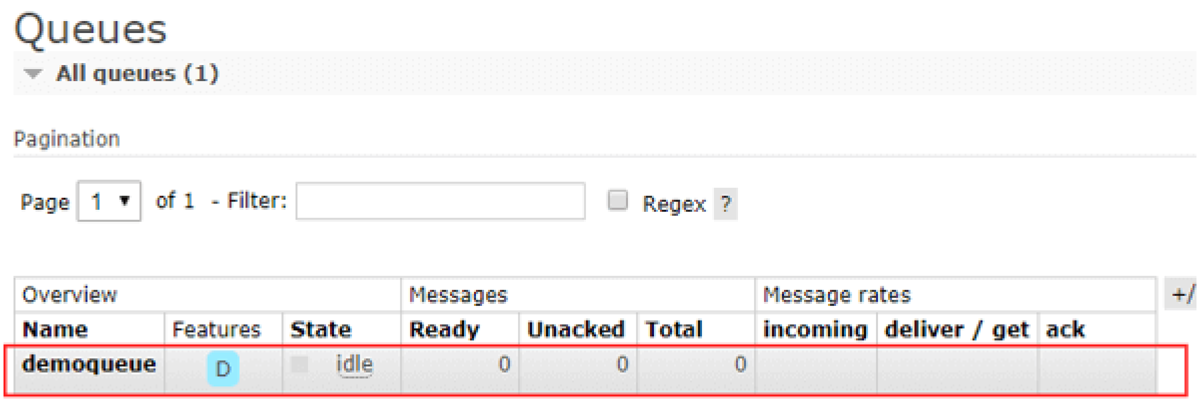
Type:	Stream
Name:	*
Arguments:	= String

Add
Max length bytes ?
Max time retention ? | Max segment size in bytes ? | Initial cluster size ? | Leader locator ?
Add queue

2nd is name box is to give a naming to the queue

3rd **Arguments** (optional; used by plugins and broker-specific features such as message TTL, queue length limit, etc)

➢ After creating a queue, you can view queue which you have recently added, it is located just above the add queue panel like as shown below.



Queues

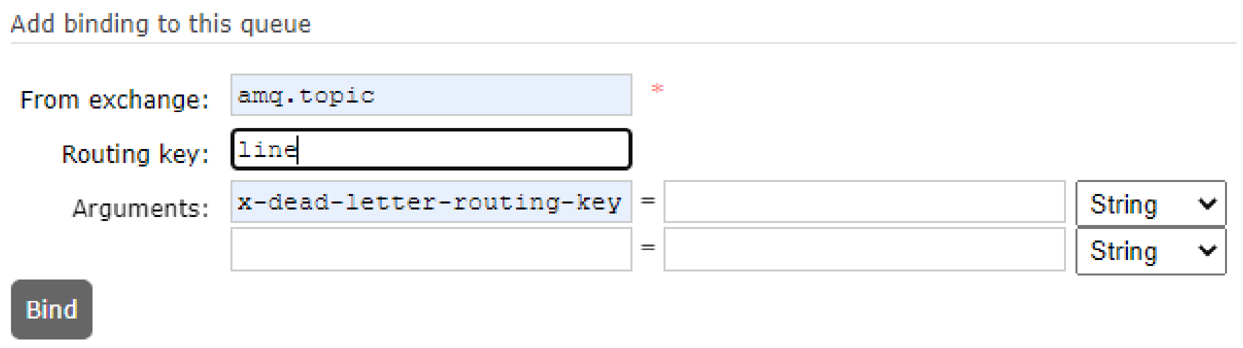
All queues (1)

Pagination

Page 1 of 1 - Filter: Regex ?

Overview			Messages			Message rates		
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack
demoqueue	D	idle	0	0	0			

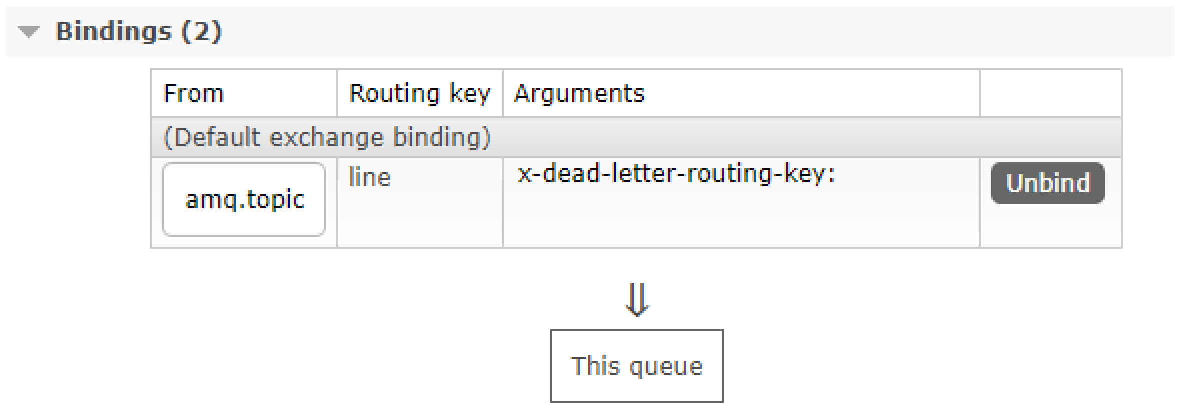
After click on queue (**demoqueue**) name, the **Bindings** panel will expand and next it will ask for the exchange name, enter exchange, routing key name which we have added in node-red mqtt node setup and any argument click on **Bind** button.



Add binding to this queue

From exchange:	amq.topic *
Routing key:	line
Arguments:	x-dead-letter-routing-key = String
	= String

After click on **Bind** button, the defined exchange will be bind to our queue and that will be like as shown below.



From	Routing key	Arguments	
(Default exchange binding)			
amq.topic	line	x-dead-letter-routing-key:	Unbind

After binding, in case if you want to unbind it then you can click on unbind button to remove binding.

INFLUXDB PART

- Now go to the buckets list in influxdb and validate the data

Note:

1. **Telegraf version must be 1.21 or higher**
2. **Make sure the topic given inside mqtt out node must be same as routing key given in rabbitmq queue and exchange**
3. **Configuration has to set as per the project requirement inside node-red, rabbitmq, telegraf, influxdb**
4. **For initial reference we have implemented complete stack running in local CDS [192.168.5.66]**