iPLON®
The Infranet Company

i_SUT

Malavika K.V

V1.0 **22.06.2022**

Project: **Diagnostic Box Configuration**
Rev: 1.0

Project: **Diagnostic Box Configuration**
Rev: 1.0

What is i-SUT??

- **System independent of iGate and Microgrid -To recreate and test the issue in Microgrid**
- **Building Virtual Plant model and logger with local ,cloud setup in node-red**
- **Checking API possibilities from different providers**
- **Building 1st Version Inverter model (sungrow 110cx)**
- **Exploring transfer protocol AMQP , RabbitMQ**

.

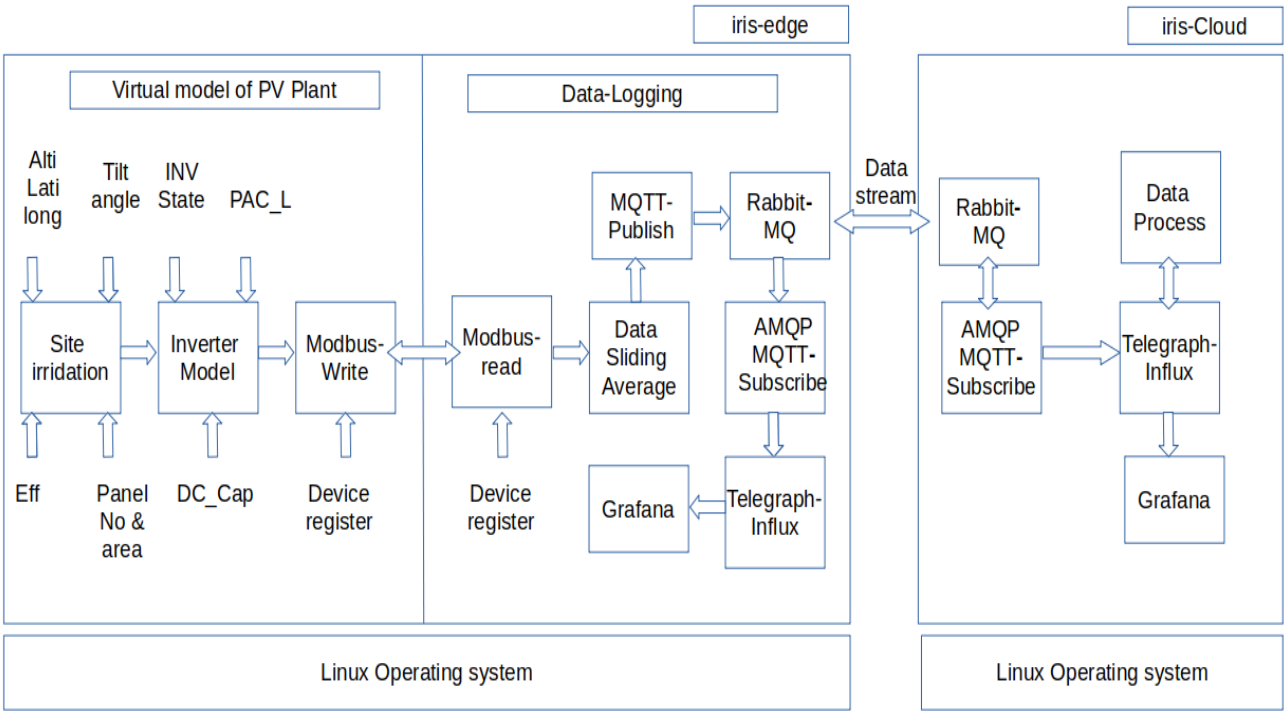


Fig – 1: Architecture of i_SUT

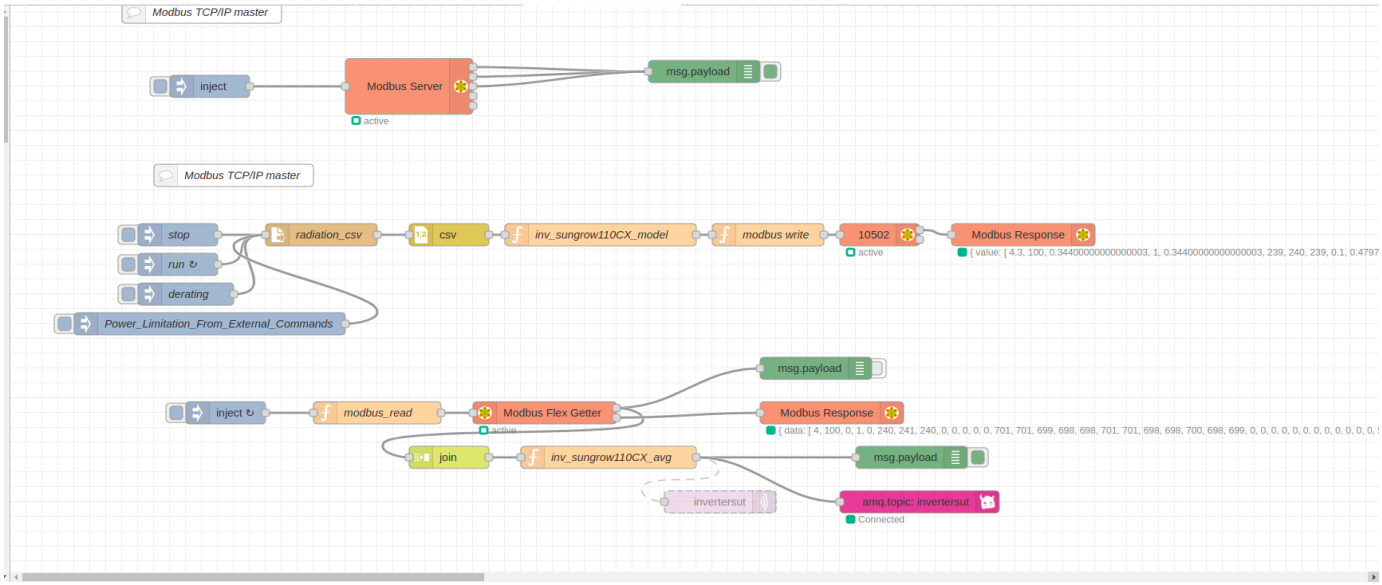
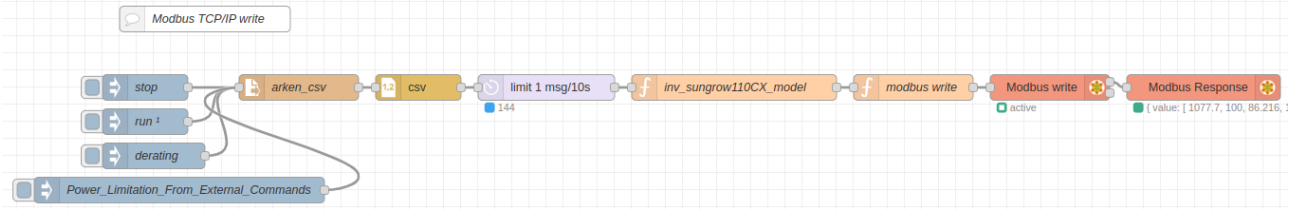


Fig – 2 Node-RED flow for i_SUT

Modbus write flow session:



- In above flow first we are setting up different operating condition inside inject node as msg.topic=stop,run,derating and Power_Limitation_From_External_Commands
- So that inverter model logic will run and give output according to msg.topic which is triggered
- After this we used a .csv file which contain 2 columns one is time of a day in 5 min intervals and another is solar radiation from Arken site.

	Standard	Standard
1	Time	Solar_Radiation (W/M2)
2	2022-09-18 00:00:00	0
3	2022-09-18 00:05:00	0
4	2022-09-18 00:10:00	0
5	2022-09-18 00:15:00	0
6	2022-09-18 00:20:00	0
7	2022-09-18 00:25:00	0
8	2022-09-18 00:30:00	0

➢ Read file node is used here to read this csv file need to fill the filename box with the path of particular csv filename

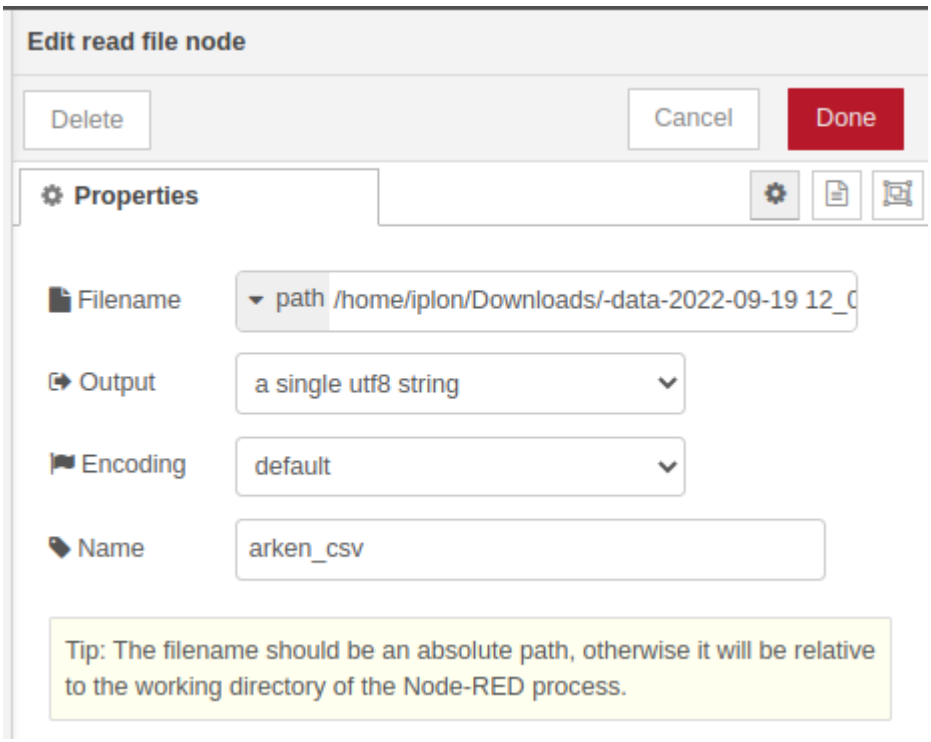


Fig – 4 read file node edit box

➢ A csv node used here to take out the csv file data into single msg per row format and also output as always sent column headers

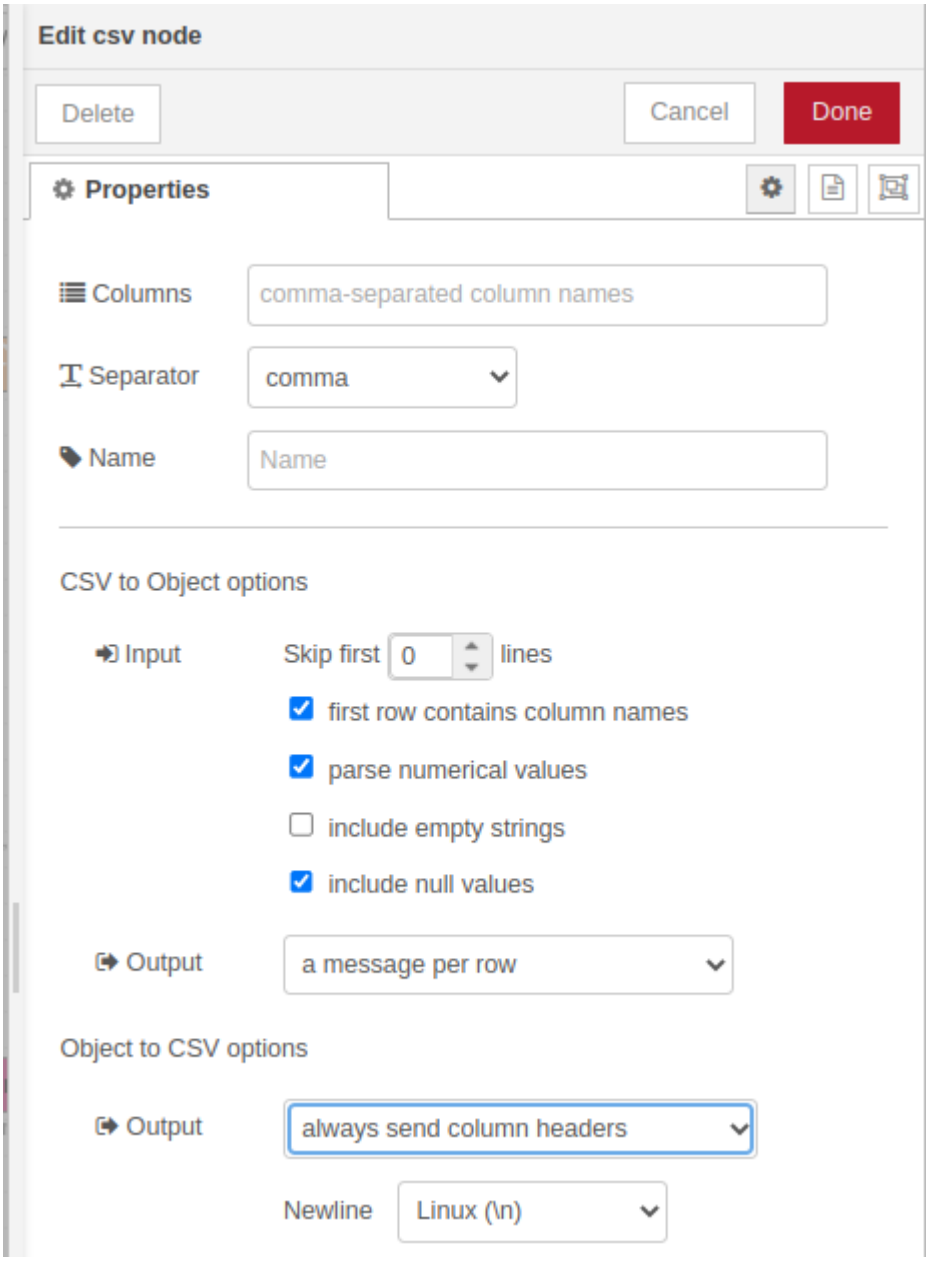


Fig – 4 csv file node edit box

- A delay node followed by csv node is used to limit each message per 10 second

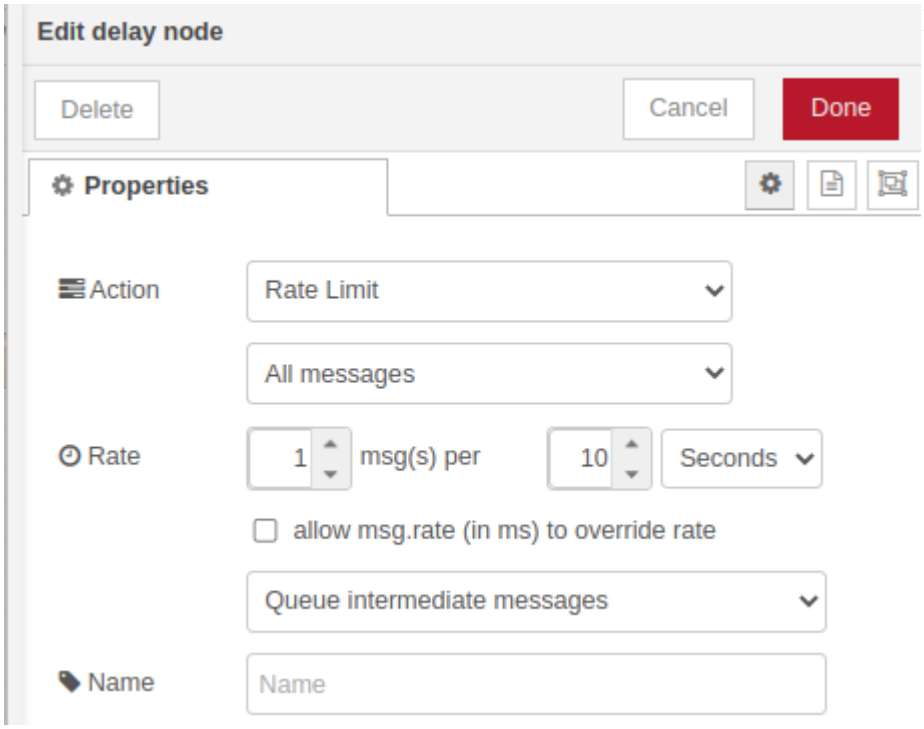


Fig – 5 delay node edit box

- So after this output will come as one message per 10 second so these single messages/10 s will go into function node as input inside the function node we wrote a javascript code for inverter simulation (inv_sungrow110CX_model in this flow)
- So by using the solar radiation from the reference csv file we are calculating the PAC,PF,SAC,UAC1,UAC2,UAC3,IAC1,IAC2,IAC3,QAC,UDC(1-12),PDC,IDC(1-12),INTERNAL_TEMPERATURE,FREQUENCY using corresponding equations

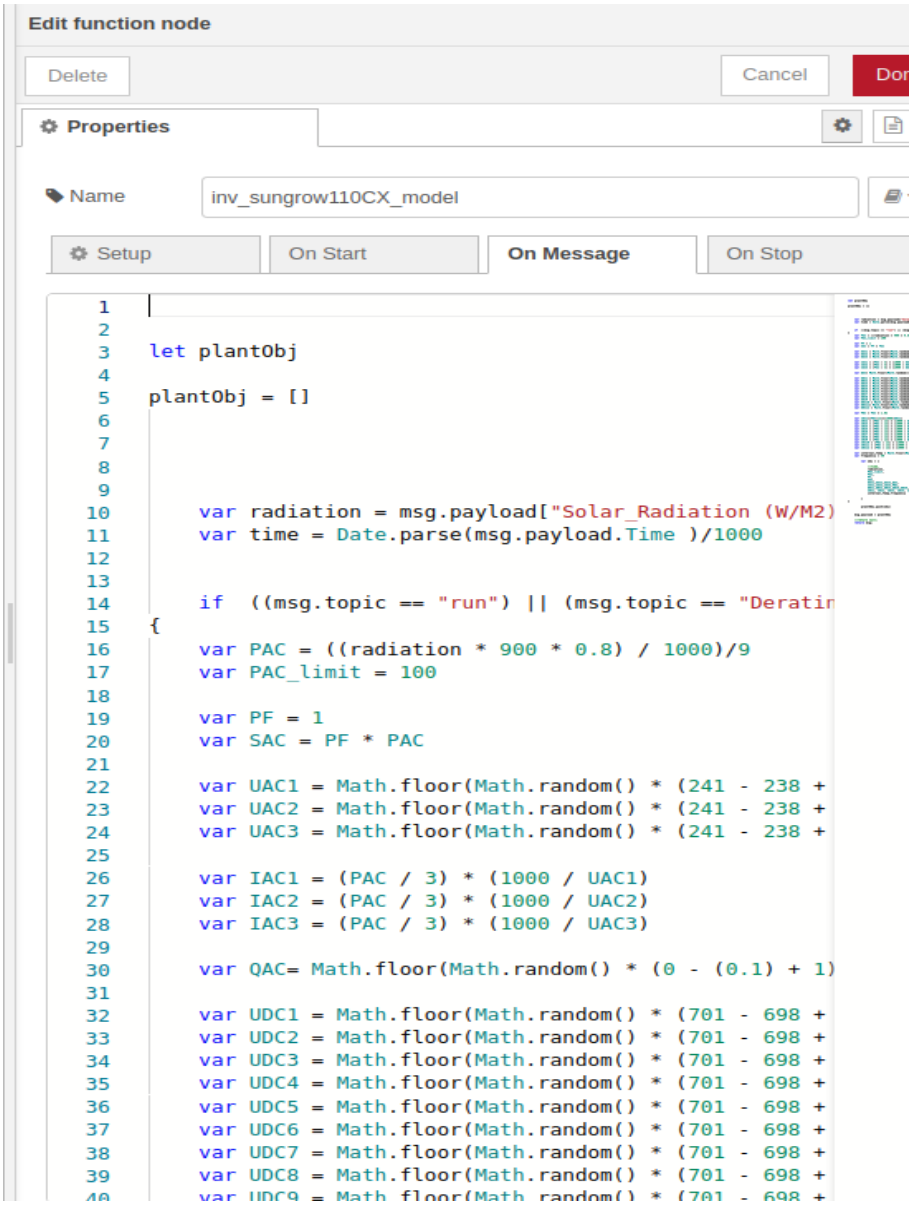


Fig – 6 function(inv_sungrow110CX_model)node edit box

➢ These output again giving into a function node which has code includes value,unit id,functon code, quantity and address of the registers to where we are writing the data to write inverter simulation outputs to modbus server

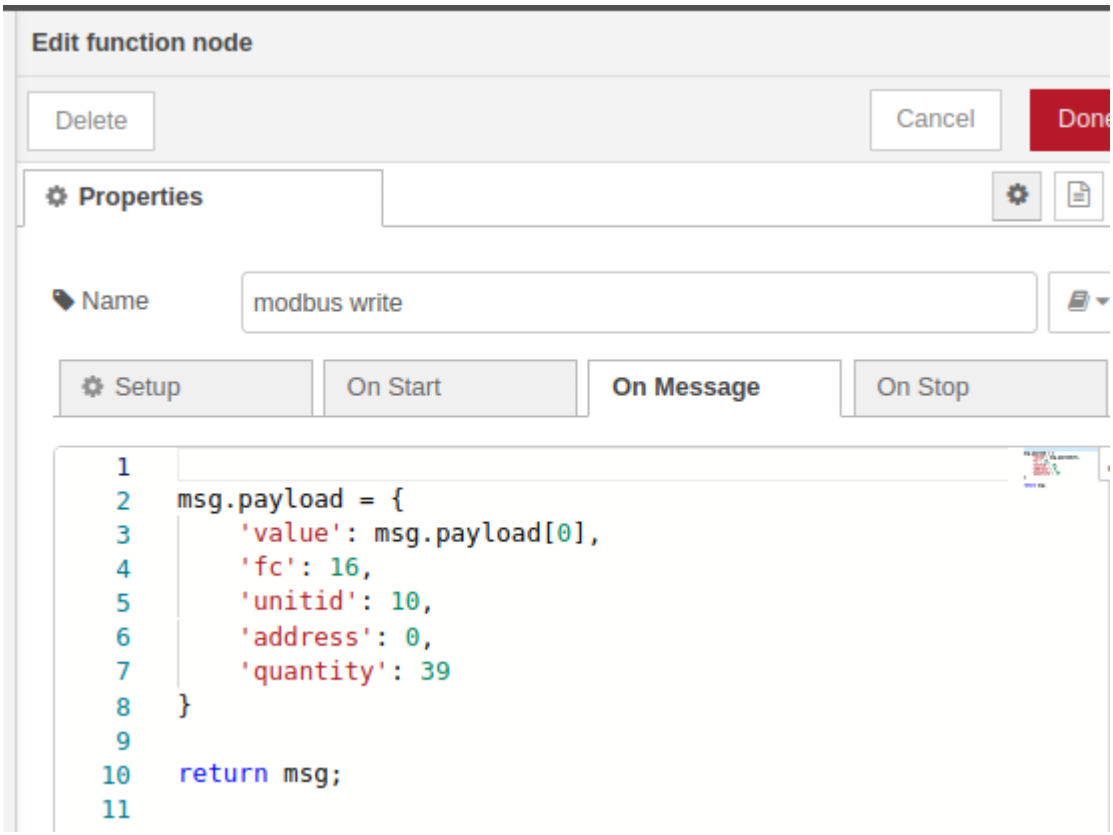
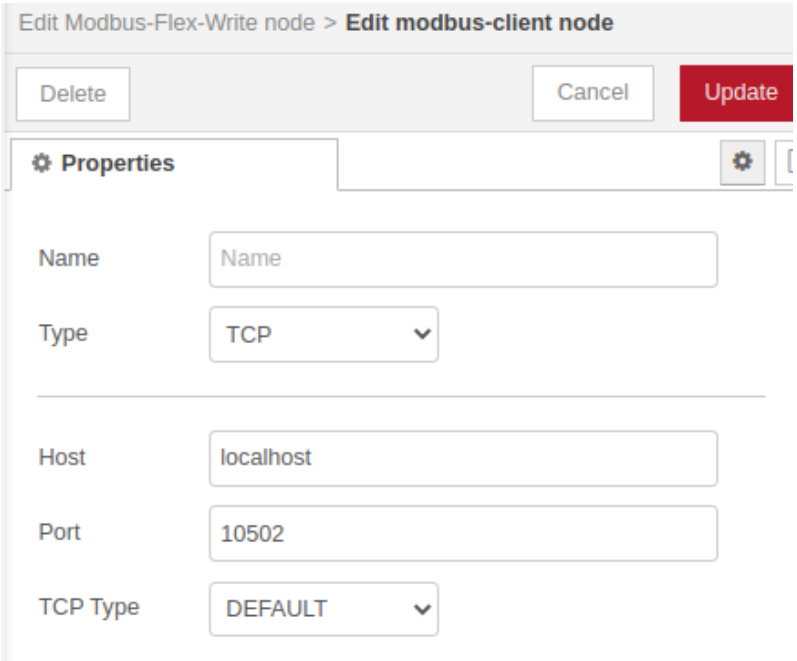


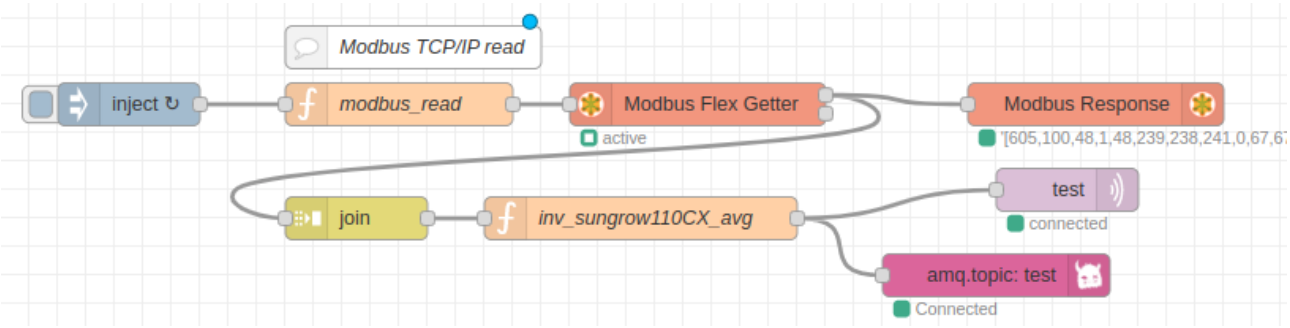
Fig – 7 function(modbus write)node edit box

- This function node outputs we are giving to modbus-flex-write node(have to install external node package node-red-contrib-modbus) where we have to give modbus server configuration details to where we are writing this inverter simulation outputs



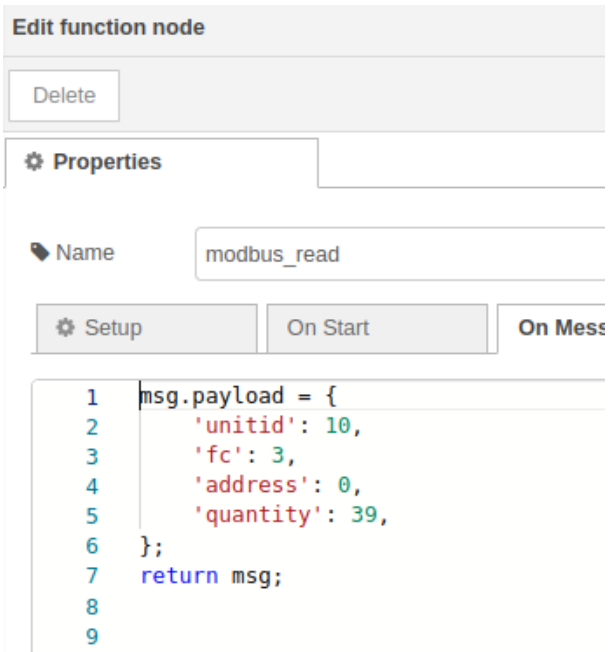
➢ Modbus response node followed by modbus-flex-write node is used just to see the modbus responses sent to modbus server

Modbus read flow session:

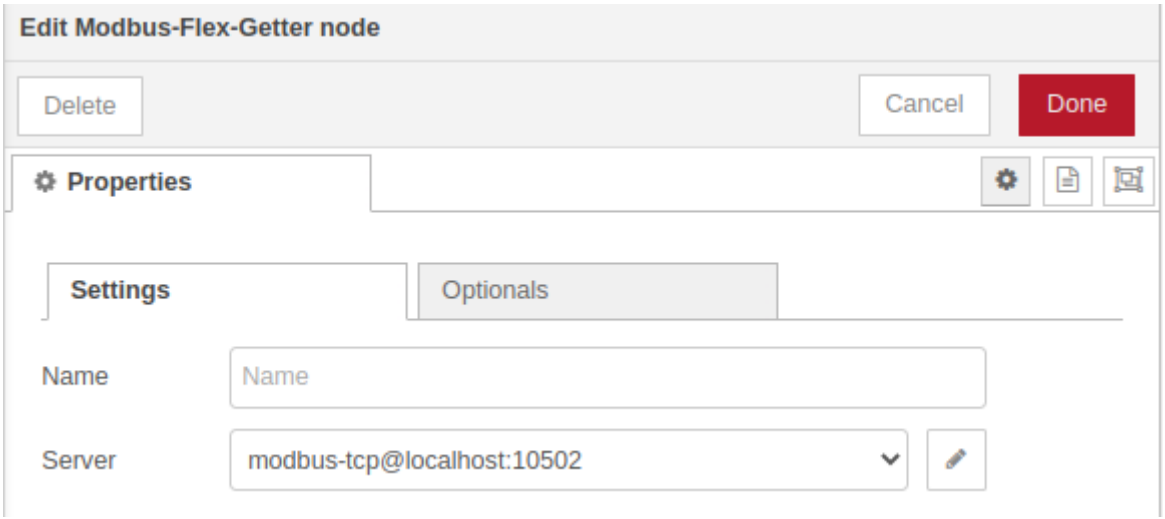


➢ In above flow first we are setting triggering interval for the flow by using inject node for every 10s

➢ A function node which has code to read data from modbus server has connected after inject node code includes unit id,functon code, quantity and address of the registers from where we are taking the data.

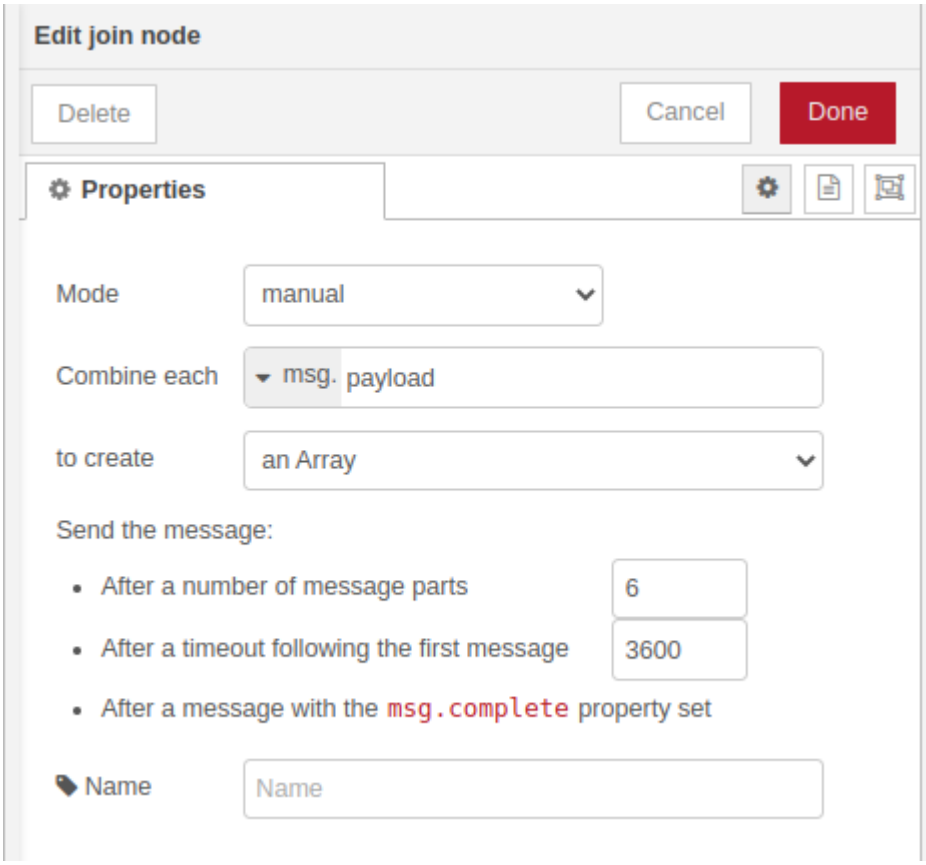


➢ This function node outputs we are giving to modbus-flex-getter node(have to install external node package node-red-contrib-modbus) where we have to give modbus server configuration details from where we are taking this inverter simulation outputs

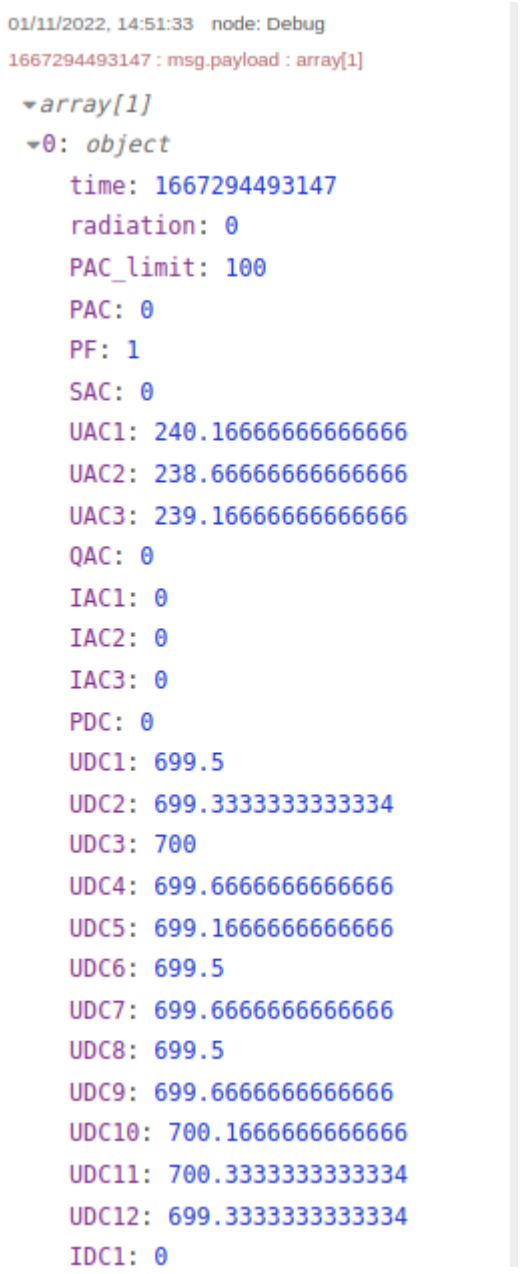


➢ Modbus response node followed by modbus-flex-getter node is used just to see the modbus responses sent to modbus server

➢ Join node used immediately after modbus-flex-getter node is to join every message coming in every 10s into one message per minute(Array of 6 messages per minute)



➢ Forwarding joined messages into a function node which has a javascript logic code to calculate average of array of 6 messages came in a minute



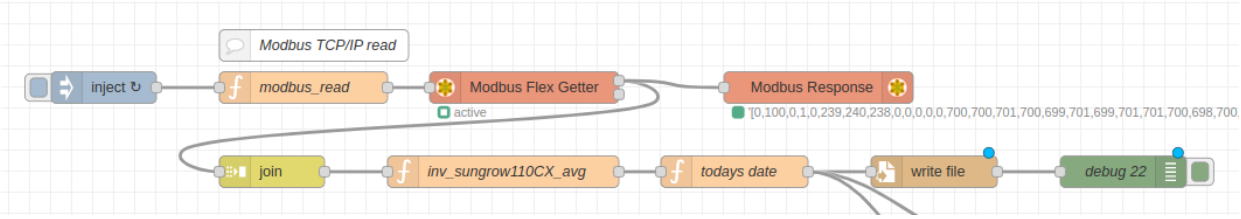
➢ output of function node connected to 2 nodes of 2 different message tranfer protocol(mqtt,amqp)

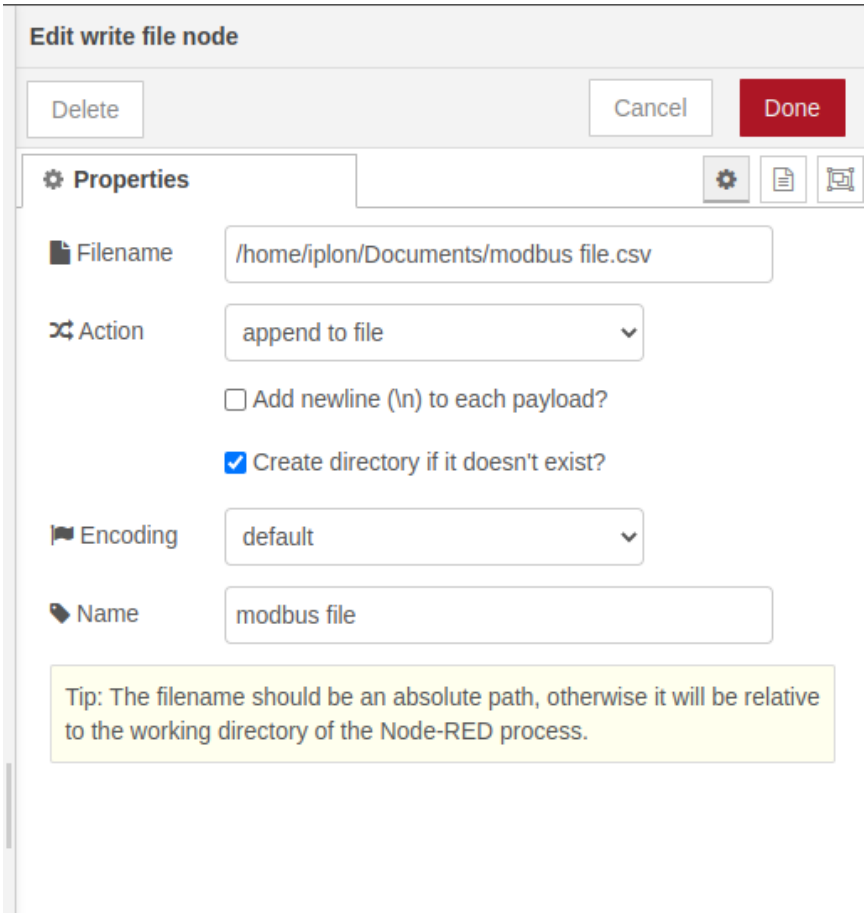
1. **mqtt out node:** mqtt used to exchange messages from one machine to other which is connected to network .To publish messages to a machine we have to

configure it inside the node. There we have to fill host, port and topic (topic is important this is the routing key for published messages)

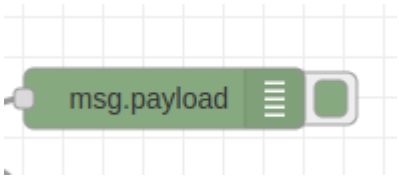
2. **amqp-out node:**Connects and sends messages to an AMQP broker(we have to install external node package @meowwolf/node-red-contrib-amqp) inside the amqp-out node first we have to configure the rabbitmq broker [host,port,credentials have to fill],then fill exchange info which we created inside the rabbitemq server or add default exchanges and give a routing key [RabbitMQ messaging system uses this to determine who will receive a copy of your message]

- Once it got connected we can se the published message in rabbitmq server
- we can save csv file of the coming output on local system for that we can use write file node inside the node edit box we have to add the path where we have to save the file and filename ie,in which name the file have to created in the given path. Here i used function node before write file node because the file name will change every minute in that cases make the space given to fill path and filename inside the write file node empty.





- Use debug nodes to take out output from each nodes which we can see in debug window.



2. RABBITMQ

RabbitMQ is a **messaging broker - an intermediary for messaging**. It gives your applications a common platform to send and receive messages, and your messages a safe place to live until received.

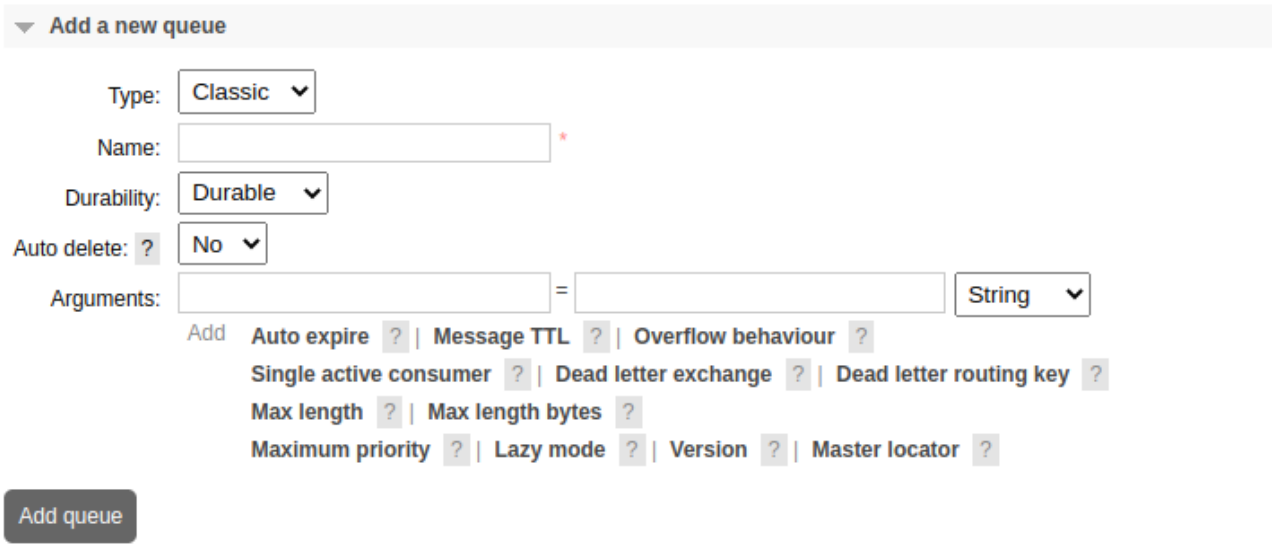
installation guide for ubuntu: https://linuxhint.com/install-rabbitmq-ubuntu/
docker container installation and plugin enabling guide:https://tewarid.github.io/2019/02/15/mqtt-with-rabbitmq-and-node-red.html

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.



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)

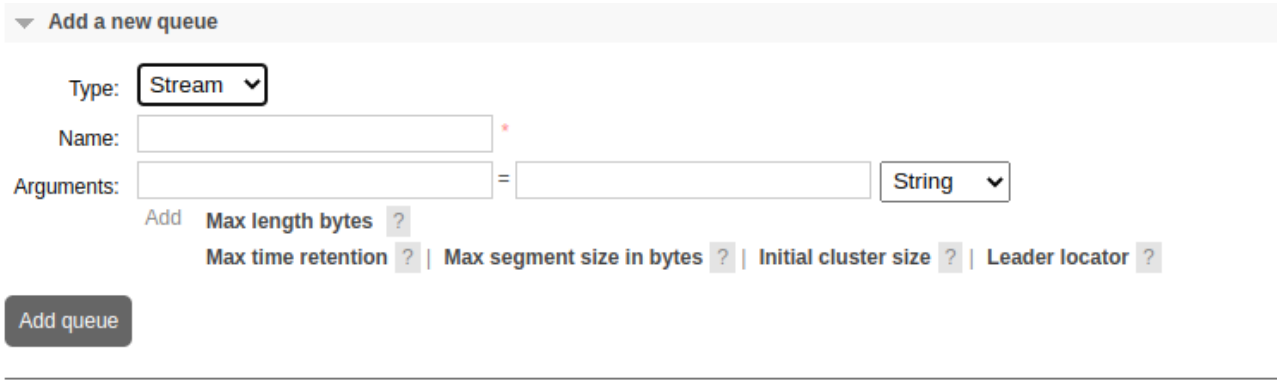
4. Auto-delete (queue that has had at least one consumer is deleted when last consumer unsubscribes)

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

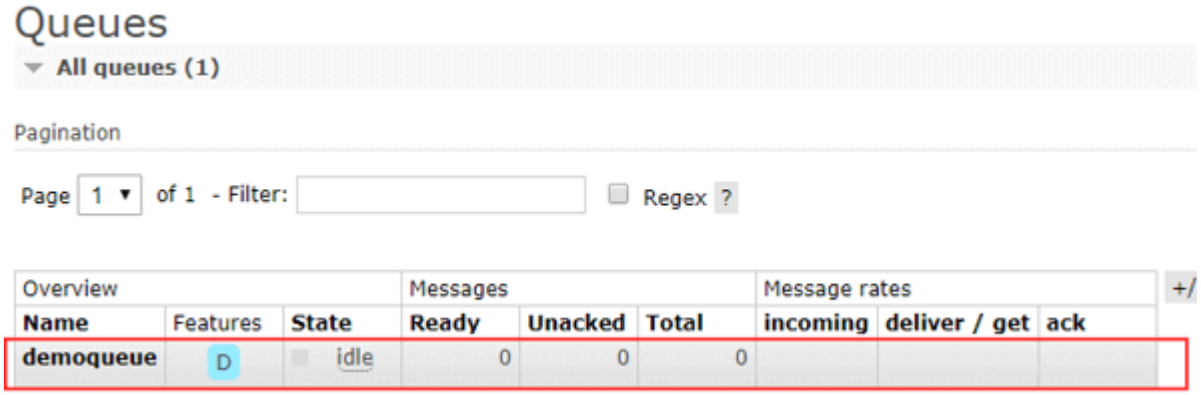
https://www.tutlane.com/tutorial/rabbitmq/rabbitmq-queues



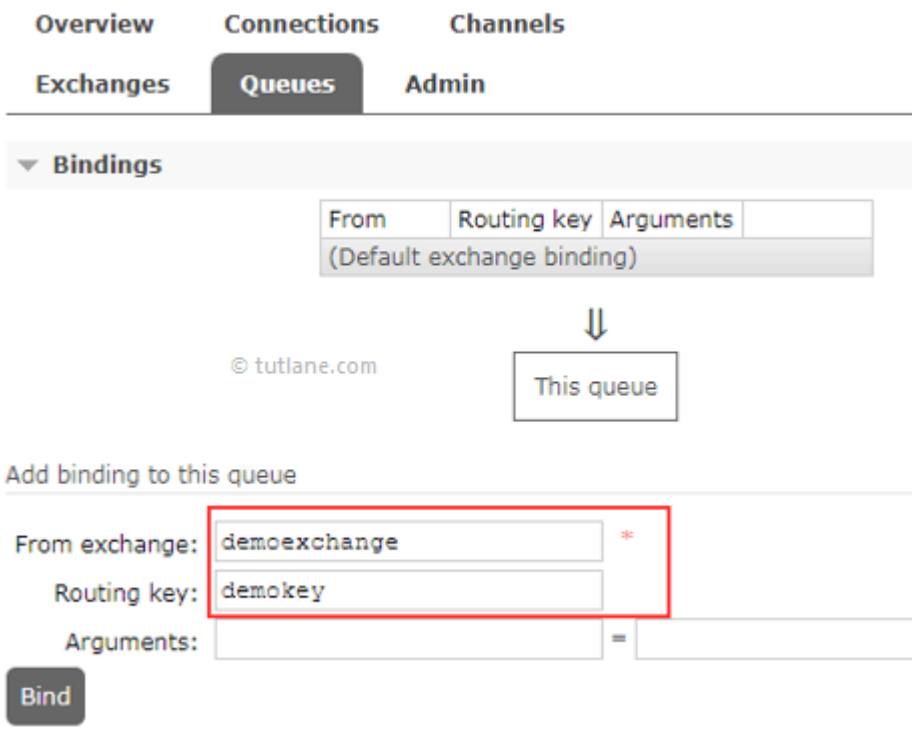
2nd is name box is to give a naming to the queue

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

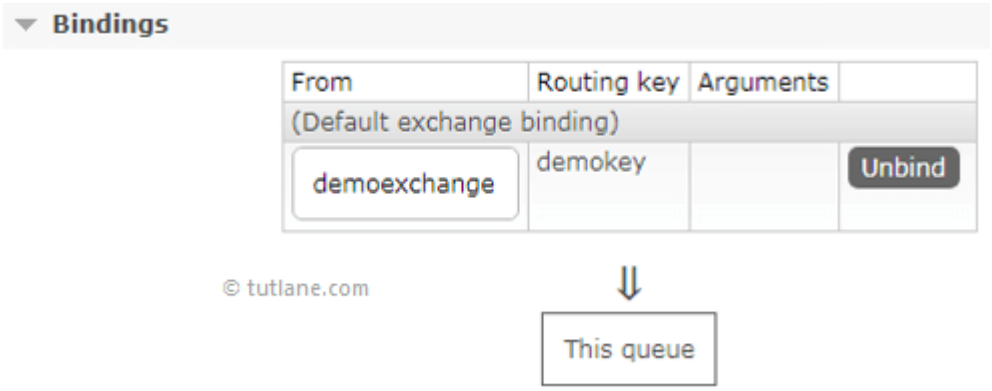
➢ After creating a queue, you can view queue which you have recently added, it is located just above the add queue panel like as shown below.



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 amqp node setup and and click on **Bind** button.

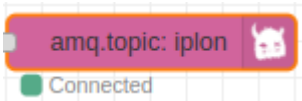


After click on **Bind** button, the defined exchange will be bind to our queue and that will be like as shown below.

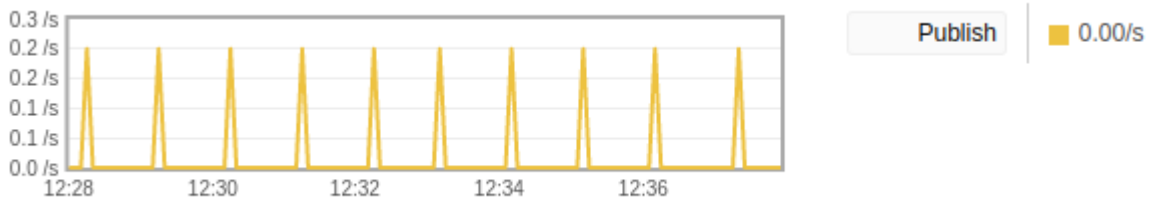


After binding, in case if you want to unbind it then you can click on unbind button to remove binding.

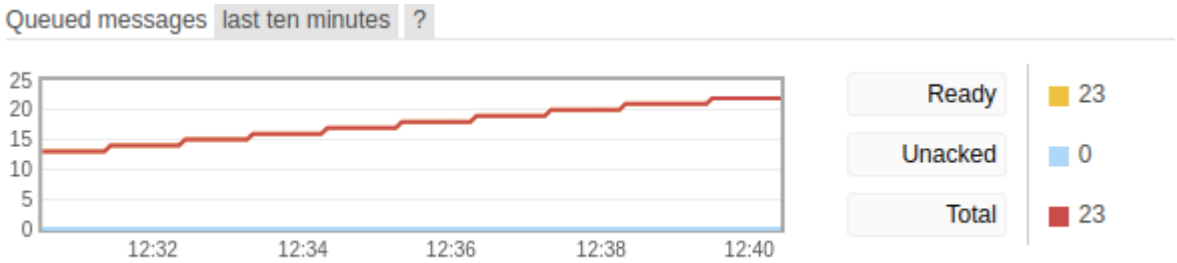
- Once node-red got connected to rabbitmq it will show connected symbol under amqp-out node like this



so sent messages published and queue status we can see in rabbitmq like this

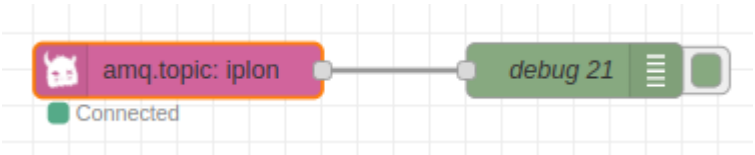


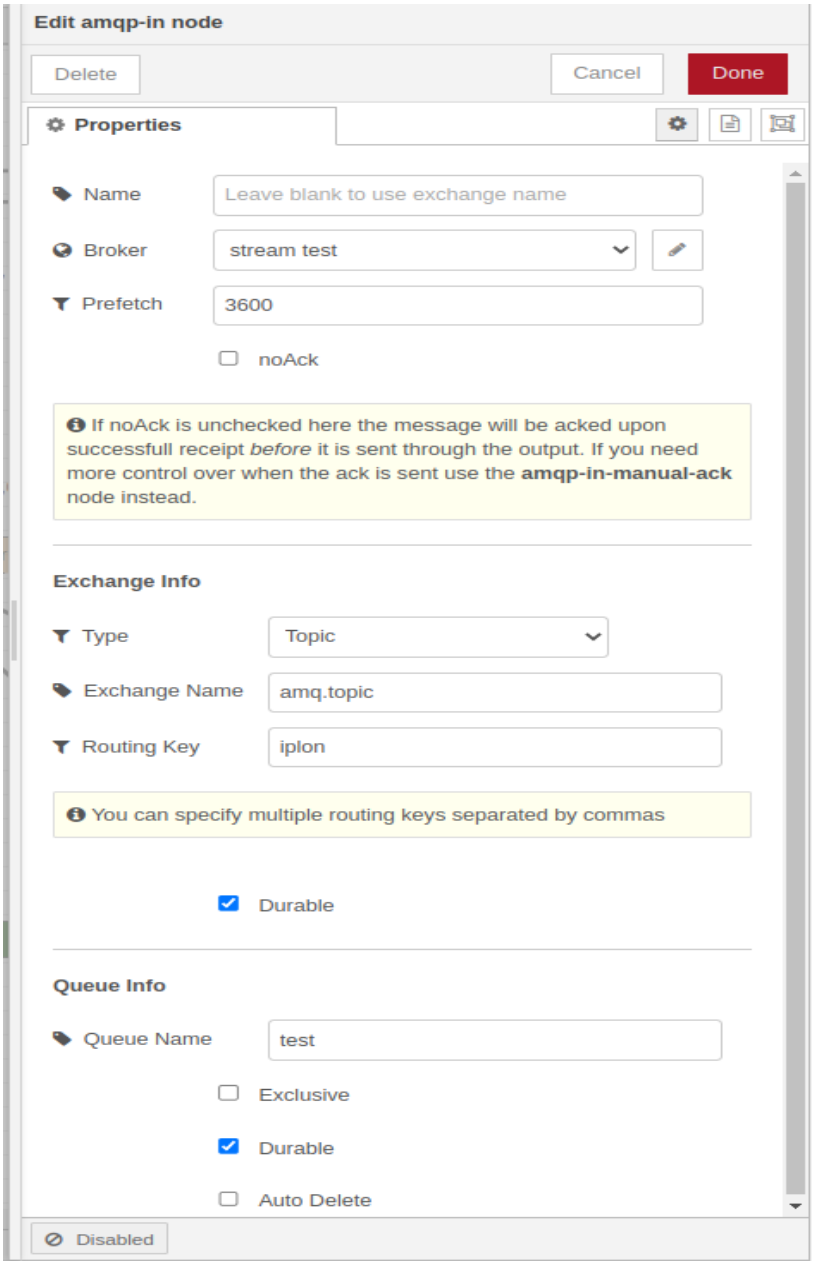
- incase there is no consumer is alive to consume that messages this will come to queue so qe can see how many messages are in queue in the graph like this



HOW TO ACKNOWLEDGE QUEUED MESSAGES :

- Use amqp-in node to consume published messages



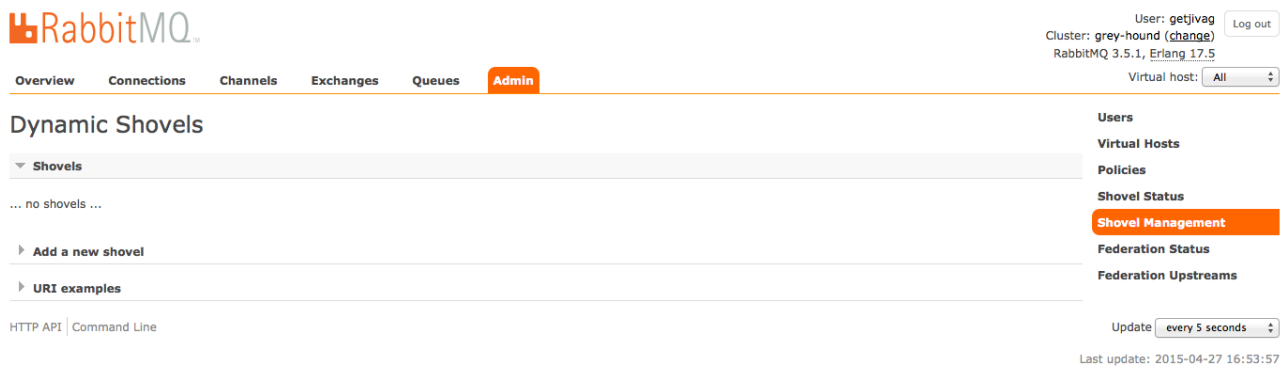


- Add the broker configurations
- Fill all the box based on your rabbitmq setup you did and add the queue name from which queue we want to consume the data from tick the durable option only like the pictue above
- Deploy the changes now we can see the published messages or queued messages in debug window.

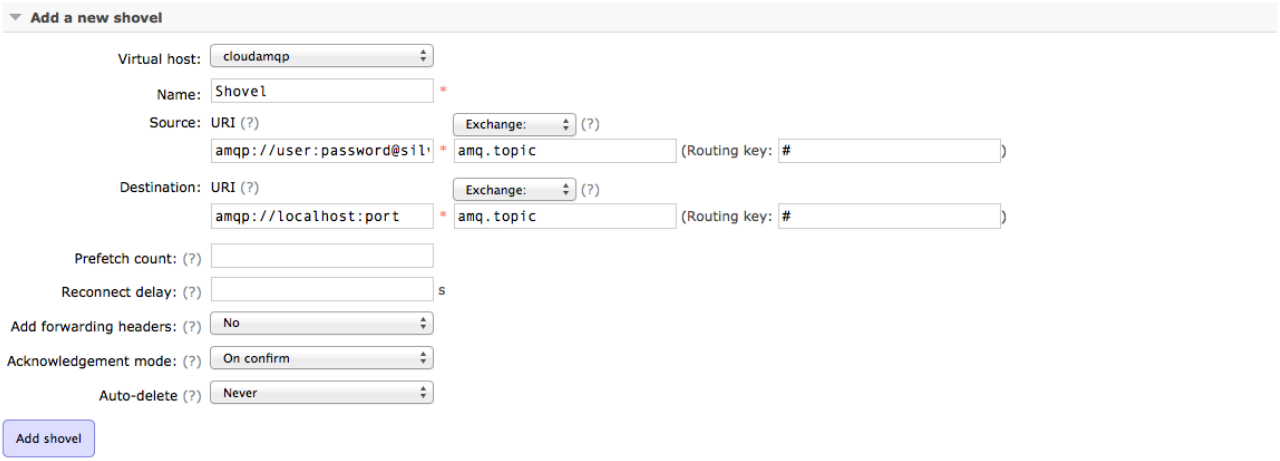
Project: **Diagnostic Box Configuration**
Rev: 1.0

RABBITMQ SHOVEL MANAGEMET PLUGIN;

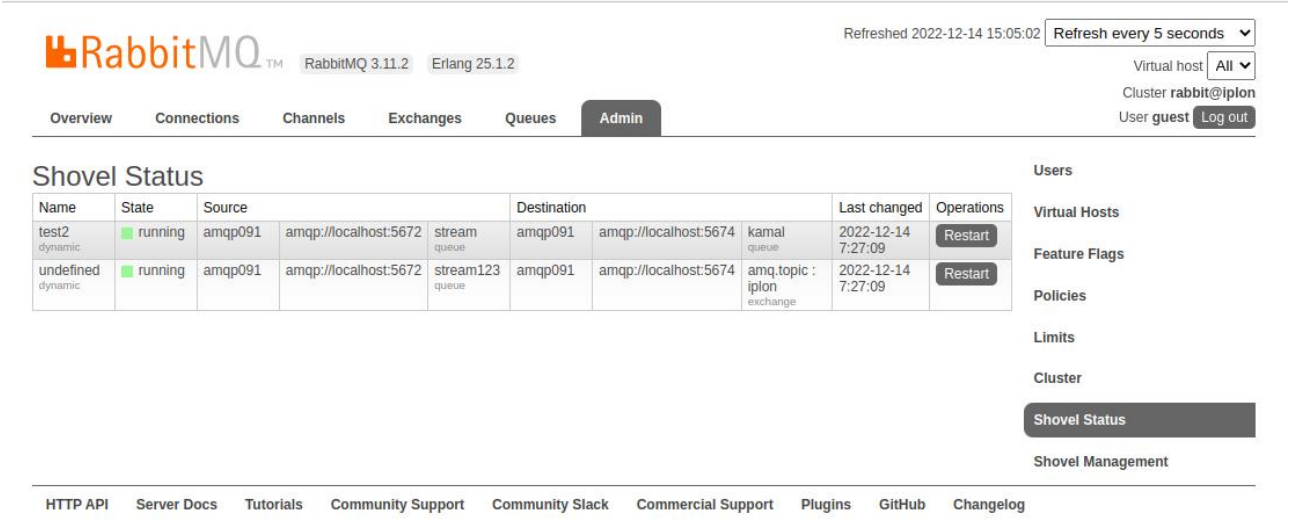
- Commant to enable shovel management plugin:**rabbitmq-plugins enable rabbitmq_shovel** run this commant in terminal inside docker container
- now shovel management and shovel status is option is available in rabbitmq management click on admin select shovel management



- Virtual host: Any vhost chosen
  Name: The name of the shovel
  Source: URI of the source instance. This should be the full CloudAMQP URI
  Destination: URI of the destination instance.



- After adding shovel we an check shovel status by clicking shovel staus if it is running it will show like this



- we can now see the same queue in different rabbitmq
- for more information check this https://www.rabbitmq.com/shovel.html

RABBITMQ CLUSTERING

For clustering we need to setup different rabbitmq broker running in same network so, first we need to create a network in docker or local etc.

Create custom bridge network — mynet

docker network create mynet

docker network inspect mynet

now create rabbitmq brokers as per the given commands

Node 1

rabbit1 is hostname

erlang cookie is defined to facilitate cluster

Port 15672 is mapped to server to access GUI for RabbitMQ management.

docker run -d --hostname rabbit1 --name myrabbit1 -p 15672:15672 -p 5672:5672 --network mynet -e RABBITMQ_ERLANG_COOKIE='rabbitcookie' rabbitmq:management

Node 2

docker run -d --hostname rabbit2 --name myrabbit2 -p 5673:5672 --link myrabbit1:rabbit1 --network mynet -e RABBITMQ_ERLANG_COOKIE='rabbitcookie' rabbitmq:management

Node 3

docker run -d --hostname rabbit3 --name myrabbit3 -p 5674:5672 — link myrabbit1:rabbit1 --link myrabbit2:rabbit2 --network mynet -e RABBITMQ_ERLANG_COOKIE='rabbitcookie' rabbitmq:management

Now the nodes are up and running but cluster is still not configured. We have to connect to each running instance to configure and restart each node. We do following steps:

- connect to running instance
- stop the instance
- join cluster
- start the instance

Node 1

docker exec -it myrabbit1 bashrabbitmqctl stop_apprabbitmqctl resetrabbitmqctl start_app

Node 2

docker exec -it myrabbit2 bashrabbitmqctl stop_apprabbitmqctl resetrabbitmqctl join_cluster --ram rabbit@rabbit1rabbitmqctl start_app

Node 3

docker exec -it myrabbit3 bashrabbitmqctl stop_apprabbitmqctl resetrabbitmqctl join_cluster --ram rabbit@rabbit1rabbitmqctl start_app

Now the nodes are up and running but cluster is still not configured. We have to connect to each running instance to configure and restart each node. We do following steps:

- ➔ connect to running instance
- ➔ stop the instance
- ➔ join cluster

➔ start the instance

Node 1

docker exec -it myrabbit1 bash

rabbitmqctl stop_app

rabbitmqctl reset

rabbitmqctl start_app

Node 2

docker exec -it myrabbit2 bash

rabbitmqctl stop_app

rabbitmqctl reset

rabbitmqctl join_cluster --ram rabbit@rabbit1

rabbitmqctl start_app

Node 3

docker exec -it myrabbit2 bash

rabbitmqctl stop_app

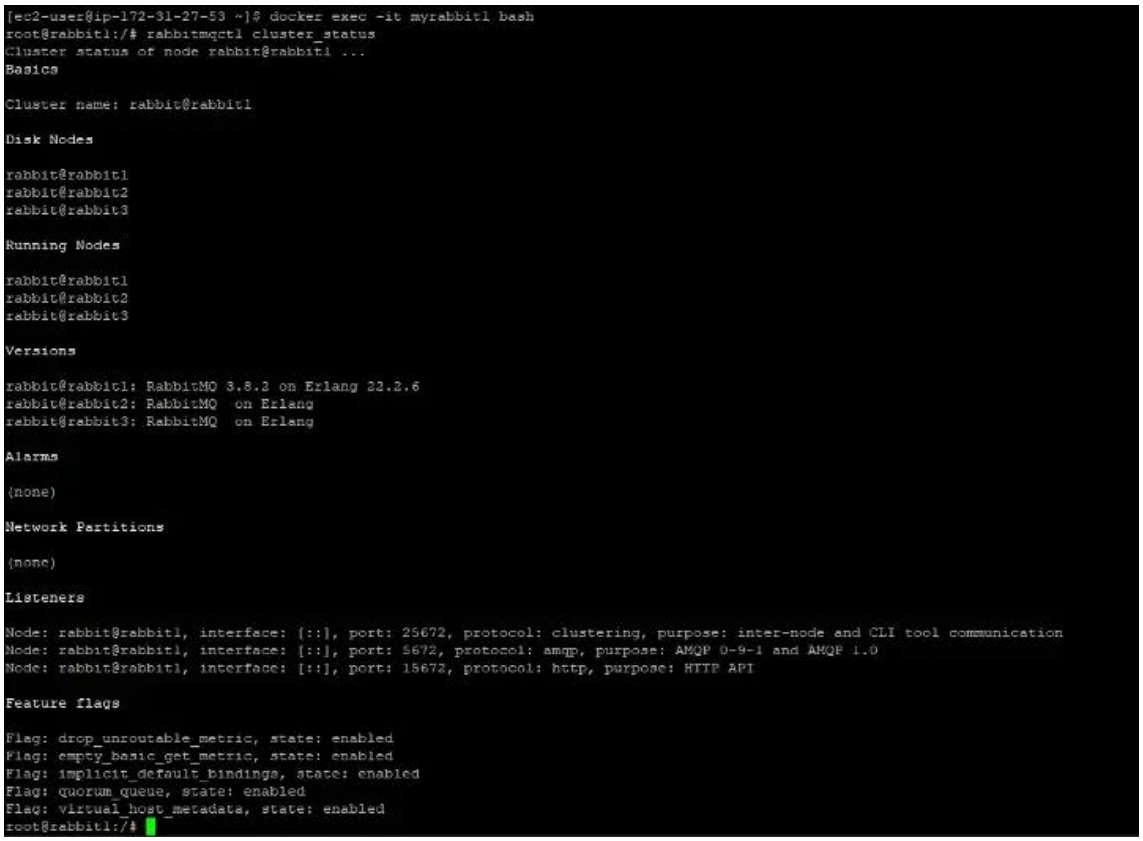
#rabbitmqctl reset

rabbitmqctl join_cluster --ram rabbit@rabbit1

rabbitmqctl start_app

connect to any one of the running instance and check the cluster status using following command

rabbitmdctl cluster_status



Now all rabbitmq brokers interconnected and created a cluster now anychanges you are making in one node will reflect in another brokers

3. Telegraf

- **Influxdb installation** : follow the step https://portal.influxdata.com/downloads/

  select the version and platform copy and paste the command in terminal

- **Telegraf installation** : follow the step https://portal.influxdata.com/downloads/

  select the version and platform copy and paste the command in terminal

OR

- Use the tar file to install all the services like influxdb,telegraf,node-red,rabbitmq,grafana-

  steps to use the file :

- $ wget https://github.com/malavikaiplon/i_sut/raw/2b41c8001fb77c9d6154c0a36514ffefab32caf2/docker-compose-grafana-influxdb.tar.gz

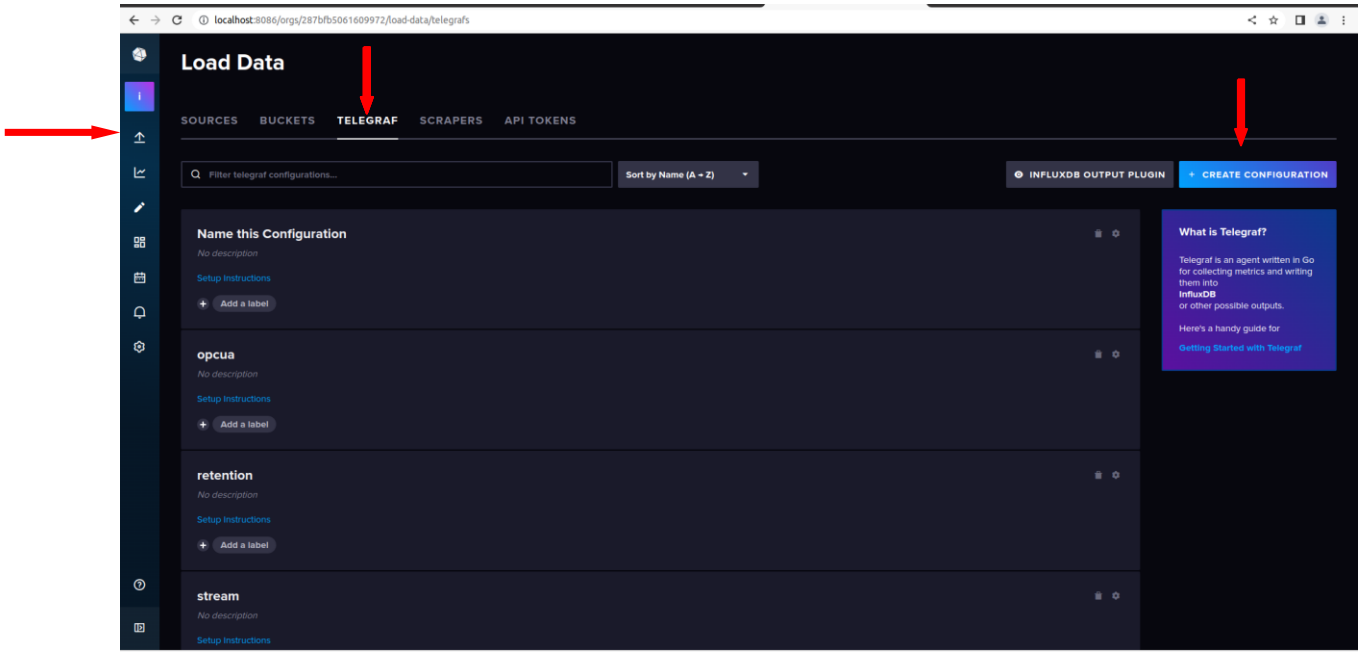
  $ tar -xvf docker-compose-grafana-influxdb.tar.gz

  $ cd docker-compose-grafana-influxdb/

  $ docker-compose up

- start influxdb docker container if it not started automatically go to the browser go to url http://localhost:8086/

- click the uparrow icon > telegraf > create configuration



- Choose existing bucket or create new bucket and search for amq consumer click on it
- a new window will open there we can setup configuration for rabbitmq

give broker url in brokers = amqp://[localhost or ip of rabbitmq broker]:[port]
eg=amqp://localhost:5672

```
brokers = ["amqp://localhost:5672/influxdb"]
```

give username and password if there is any credential you set up other than default [guest,guest]

```
username = "iplon"
password = "iplon321
```

give exchange = [type the same exchange from where we are consuming the message]

```
exchange = "telegraf"
```

give queue=[type the same queue from where we are consuming the message]give

queue = "test"

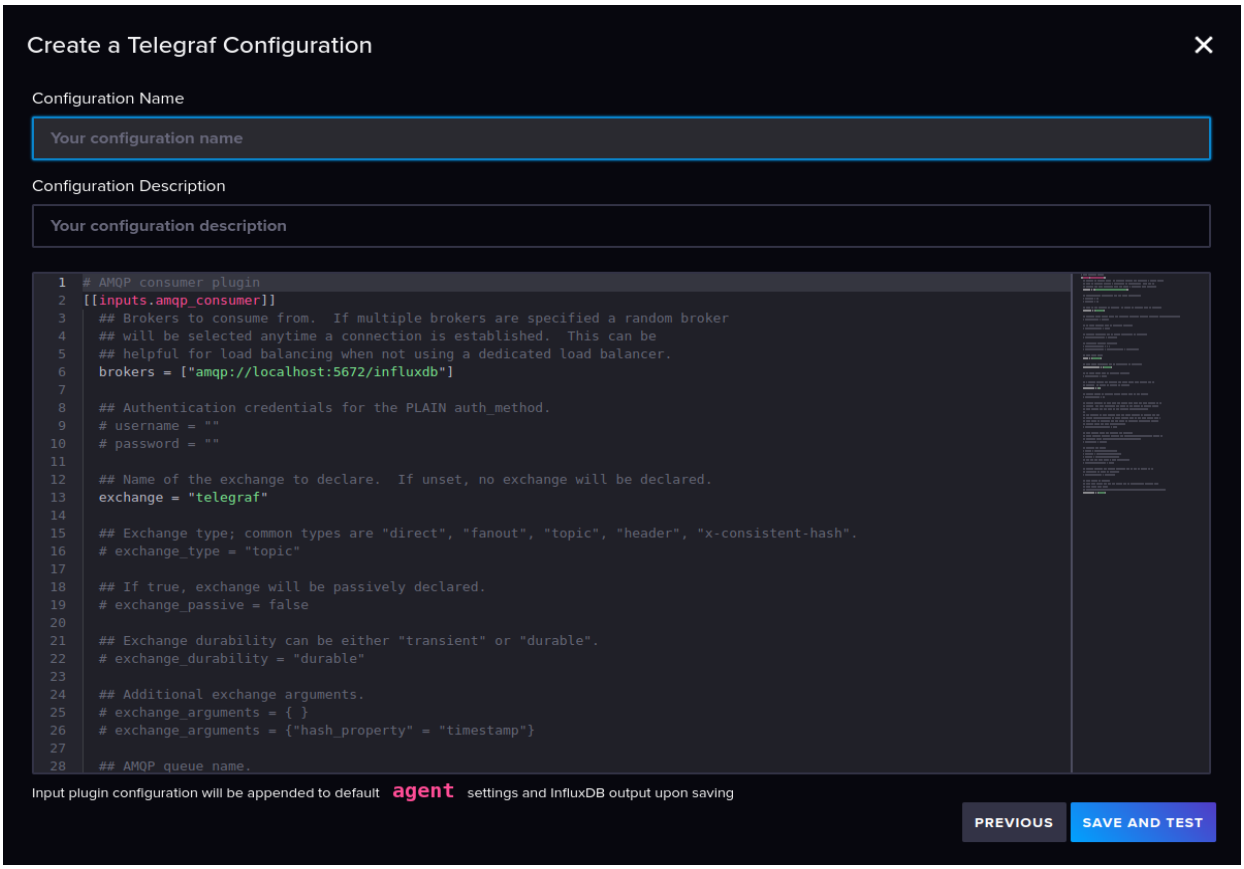
set queue durability =durable

if you are taking input from a stream queue enable

queue_passive =true

queue_passive = true

queue_durability = "durable"



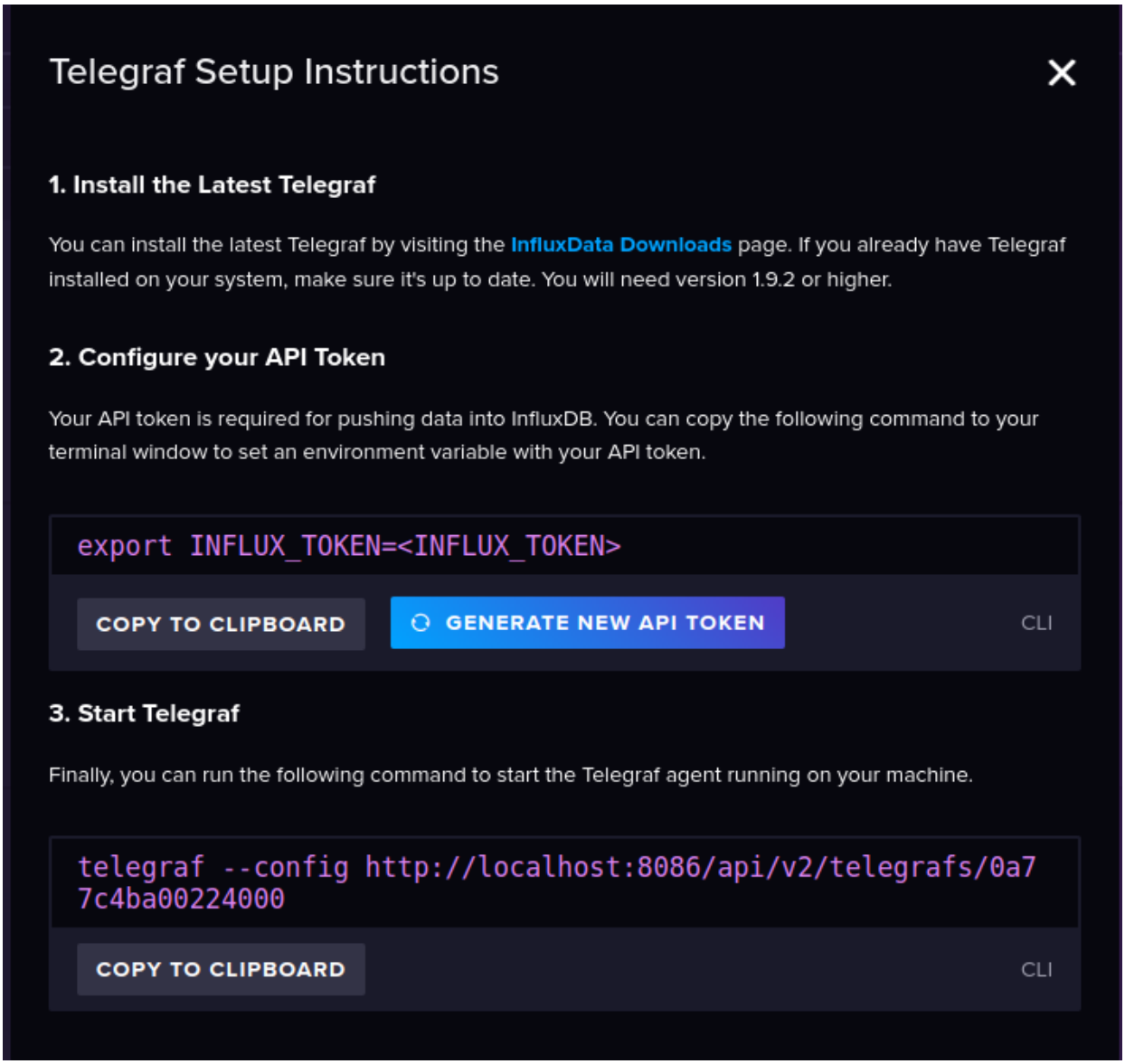
binding_key=[type the same routing key at the time of publishing message]

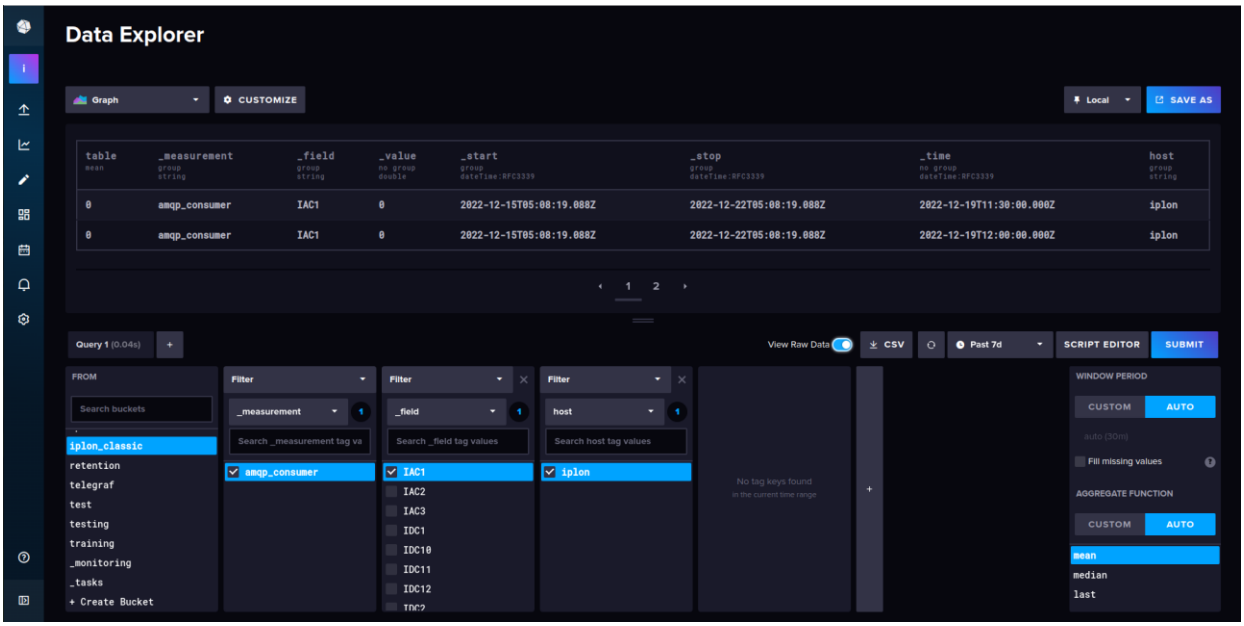
binding_key = "test"

data_format=json

data_format = "json"

- click on save and test
- One window like this will open to configure api token click on generate new ap token click copy to click copy to clipboard paste this on terminal
- To start telegraf run the second command in terminal check if there is an error it will show in logs.
- After set up the telegraf configuration open the bucket you choose for this configuration. We can see the coming data on influxdb bucket .




Data Explorer
Graph
CUSTOMIZE
Local
SAVE AS
table
_measurement
_field
_value
_start
_stop
_time
host
0
amqp_consumer
IAC1
0
2022-12-15T05:08:19.088Z
2022-12-22T05:08:19.088Z
2022-12-19T11:30:00.000Z
2022-12-19T12:00:00.000Z
iplon
1
2
Query 1 (0.04s)
View Raw Data
CSV
Past 7d
SCRIPT EDITOR
SUBMIT
FROM
Search buckets
iplon_classic
retention
telegraf
test
testing
training
_monitoring
_tasks
+ Create Bucket
Filter
_measurement
Search _measurement tag va
amqp_consumer
_field
Search _field tag values
IAC1
IAC2
IAC3
IDC1
IDC10
IDC11
IDC12
host
Search host tag values
iplon
No tag keys found
in the current time range
WINDOW PERIOD
CUSTOM
AUTO
auto (30m)
Fill missing values
AGGREGATE FUNCTION
mean
median
last