

LivingFog documentation

<http://www.fogguru.eu/livingfog>

11/04/2021



LivingFog description	2
The design of software stack	2
Software stack running on kubernetes clusters	5
Software stack on Central Server PC	6
Installation guide	7
Preparing the raspberry Pis	7
Creating Kubernetes Cluster	9
Installing Chirpstack software and other software stacks	11
Configuring LoRa gateways	13
Installing Chirpstack gateway bridge on the LoRa gateways	17
Configure Chirpstack gateway bridge on the LoRa gateways	18
Add sensors to Chirpstack Application server	19
LivingFog Maintenance	23
Platform monitoring	23
Manual debugging using SSH	26
Use case: IoT Fablab	28
Hardware	28
LoRa Gateways and Fog clusters	29
Sensors	30
Data description	31

This documentation describes the LivingFog platform, including the platform description, the deployment process with our open-sourced project, the basic maintenance of LivingFog, and a use case of IoT Fablab.

LivingFog description

The design of software stack

The whole system can basically split into 3 levels of abstractions: LoRa sensors and gateways, Raspberry Pi (RPI) Clusters and Central PC server. No specific software is installed on LoRa gateways and sensors, but they need to be configured.

Raspberry Pi OS is installed on each RPi as a preparation. Then, Kubernetes 1.18 is installed and configured on 5 to 10 RPis as a Fog node. The whole structure is shown in Figure 1.

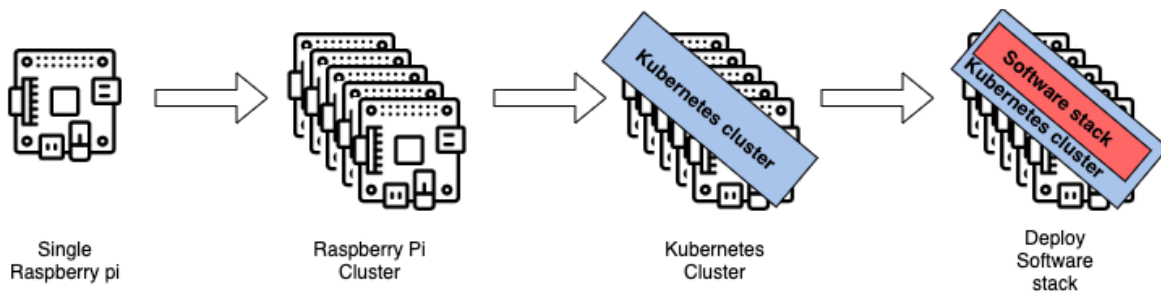


Figure 1: Fog node cluster structure

We used Kubernetes: open-source container orchestration system for automating and deploying our softwares. General Kubernetes structure is shown in Figure 2. One of the machines (Raspberry pi in this case) should be set as a master node, while the remaining ones can join it as a worker. Kubernetes installation and preparation step is explained in the installation section below.

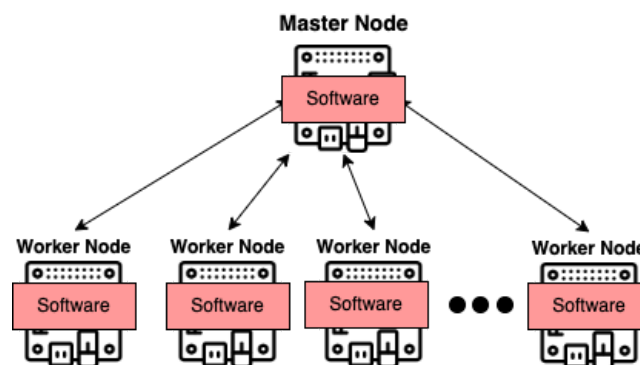


Figure 2: Kubernetes cluster structure

From the hardware point of view, the LivingFog platform can be visualized as shown in Figure 3. The whole structure concerns the sensors connected to specific gateways, gateway devices connected to specific Raspberry Pi (kubernetes) clusters, and Raspberry Pi clusters are connected to Central Server PC.

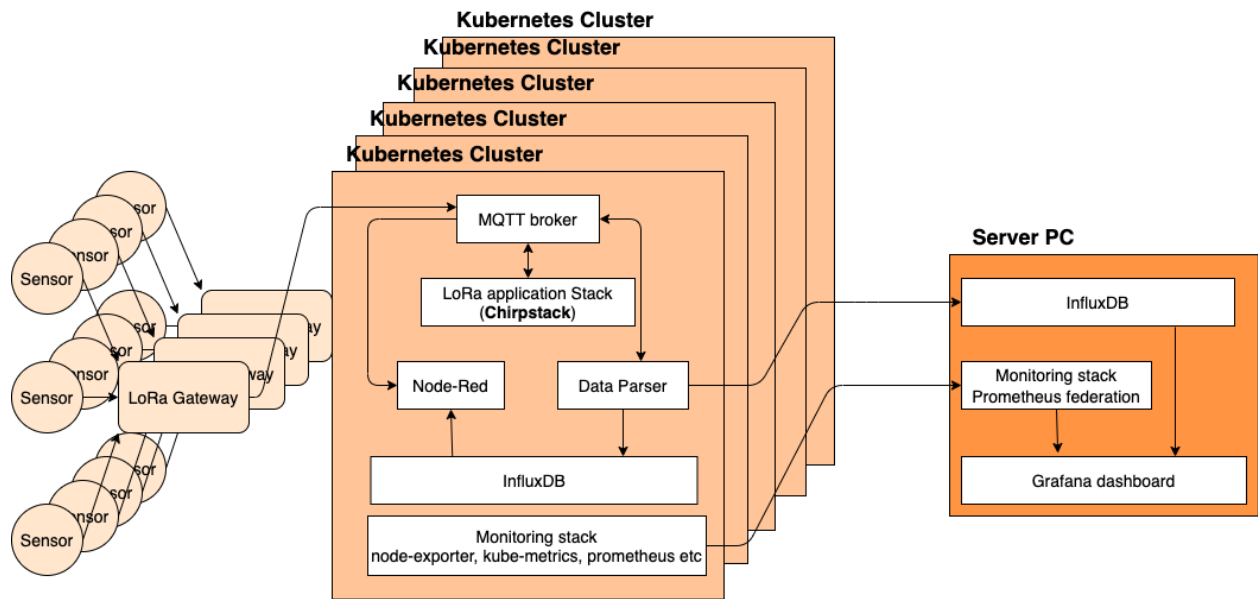


Figure 3: Overview of the LivingFog Platform

There are a number of specific Softwares and stacks deployed and configured on raspberry pi (kubernetes) clusters and Central Server PC.

Figure 4 shows the internal software structure, relation and data flow between the systems and components. Each software component is described down below.

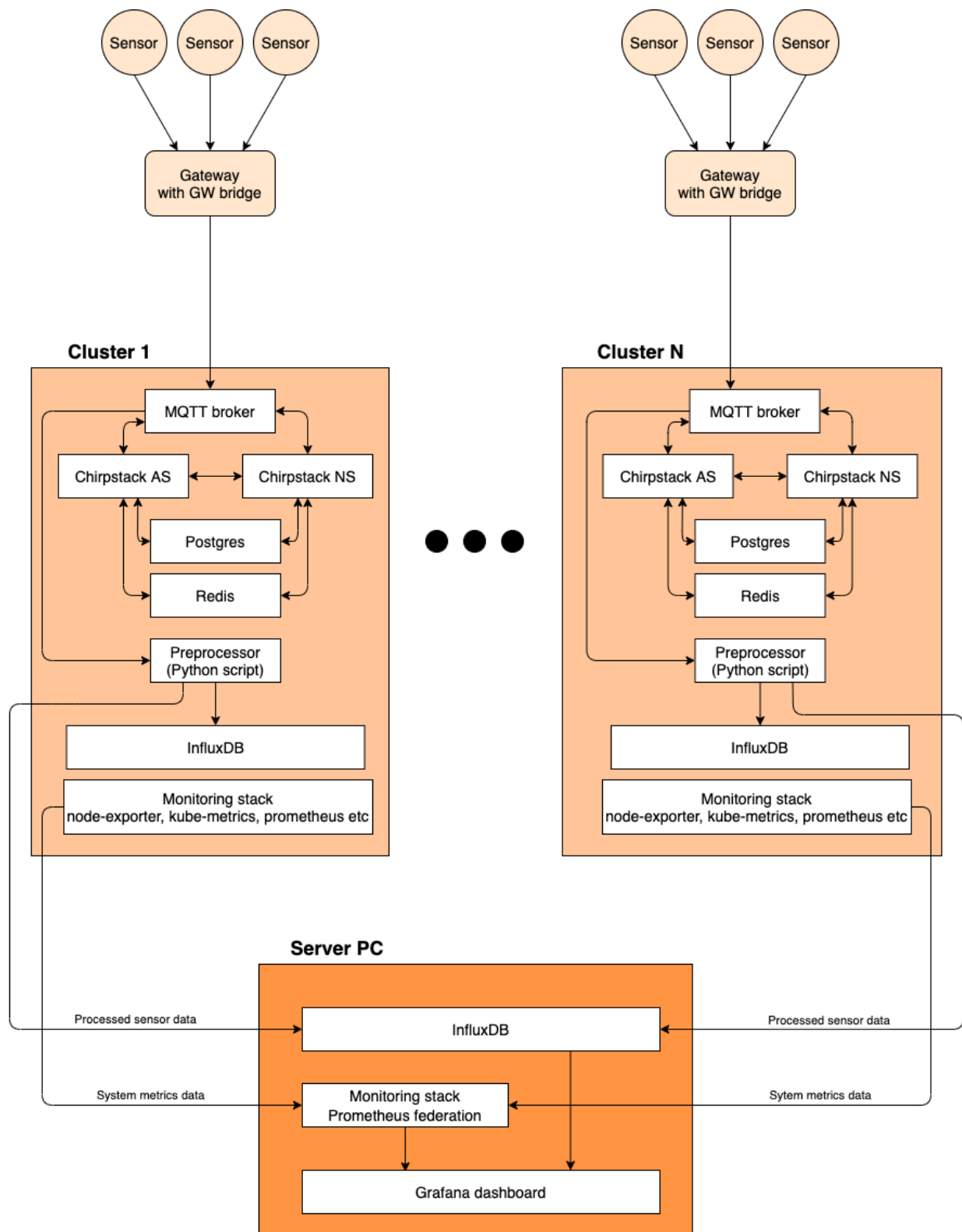


Figure 4: Software structure and relation between the systems.

Software stack running on kubernetes clusters

After the Kubernetes cluster creation, all the necessary softwares is installed. Software and kubernetes code components are linked here:

<https://github.com/FogGuru/livingfog/tree/main/chirpstack-kubernetes>.

All the specific configurations for each component are mainly written in configMap.yml files in their specific module.

Softwares installed on each cluster are:

- Chirpstack Application Server - The ChirpStack Application Server is a LoRaWAN Application Server, compatible with the ChirpStack Network Server. It provides a web-interface and APIs for management of users, organizations, applications, gateways and devices.

Received uplink data is forwarded to one or multiple configured integrations.

- Chirpstack Network Server - The ChirpStack Network Server is a LoRaWAN Network Server, responsible for managing the state of the network. It has knowledge of device activations on the network and is able to handle join-requests when devices want to join the network.

When data is received by multiple gateways, the ChirpStack Network Server will de-duplicate this data and forward it as one payload to the ChirpStack Application Server. When an application-server needs to send data back to a device, the ChirpStack Network Server will keep these items in queue, until it is able to send to one of the gateways.

- Chirpstack Gateway Bridge - The ChirpStack Gateway Bridge sits between the Packet Forwarder and MQTT broker. It transforms the Packet Forwarder format (like the Semtech UDP Packet Forwarder protocol) into a data-format used by the ChirpStack components.

The Gateway bridge is not installed on RPi clusters. It's installed and configured on each Gateway device. More detailed explanation about configuration is in the Installation section.

- MQTT broker - MQTT is a publish/subscribe messaging system. In our system, it acts as a main messaging system, it's in control of handling all the major data flow between Chirpstack Gateway Bridge, Chirpstack Application and Network Servers, and data parser python code.
- Postgres SQL - Acts as a main database for storing information about the devices connected to the Chirpstack Application server.
- Redis - ChirpStack Network Server uses Redis for storing device-session data and non-persistent data like distributed locks, deduplication sets and meta-data.

- Preprocessor Python code - Series of data parser scripts written on python and deployed as a docker image. It's role is to parse incoming hexadecimal numbers from Chirpstack to MQTT, interpret them into readable values according to the dataframe of each sensor, and forward them back into MQTT broker and InfluxDB. More detailed explanation of the data parser is in the Data Description section.
- InfluxDB - We use InfluxDB to store historical measurements data coming from the sensors.
- Monitoring stack - Its purpose is to monitor each machines (RPi) and kubernetes cluster and kubernetes namespaces. Main components are Prometheus, Kube-state-metrics (node-port) and node-exporter. It's sending the metrics to the central Prometheus running on Server PC.

Software stack on Central Server PC

Server PC is acting as a centralized monitoring machine. All the software components are linked here: <https://github.com/FogGuru/livingfog/tree/main/central-server>.

Softwares installed on the PC are:

- InfluxDB - We store all the historical sensor measurements coming from multiple RPi clusters on centralized influxDB as a backup. It's also used as a data source for creating sensor dashboards on Grafana.
- Monitoring stack - Prometheus Federation collecting all metrics from each RPi device and clusters. Also used as a data source for creating monitoring dashboard on Grafana
- Grafana Dashboard - Open-source, interactive dashboard for managing, visualizing and analyzing everything.

Grafana dashboard can be found at <http://192.168.9.71:3002/> with the username: admin Password: FogGuru2020

We have prepared 2 main types of dashboards, namely: Sensor data and platform monitoring. There are 12 types of Sensor data dashboards and it can be found in *la-marina-sensors* folder in a dashboard. Also there are 2 types of platform monitoring dashboards and it can be found in *LivingFog platform monitoring* folder.

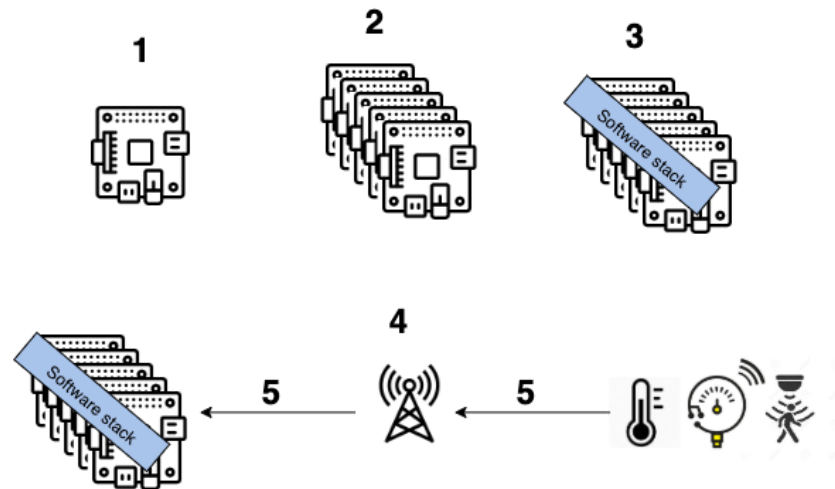
The system monitoring is explained in system management section below.

Software stacks on either RPi clusters or Server PC are deployed as a Kubernetes deployments, services and pods.

Installation guide

This section covers the installation and configuration of lora gateway software stack.

Lora gateway software stack architecture was explained in the section above.



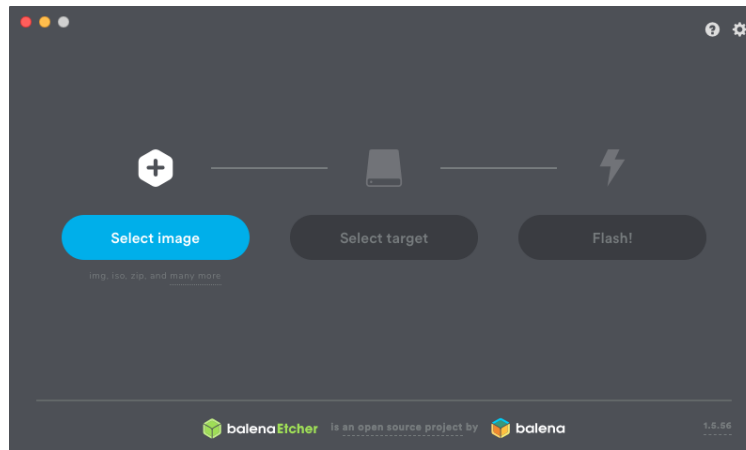
Installation and configuration of lora gateway software stack consists of 5 main steps:

1. Preparation of each raspberry pi device in order to create the cluster.
2. Create Kubernetes cluster on every Pi cluster by running the provided ansible script.
3. Installation of actual chirpstack gateway software and other software stack on a cluster.
4. Configuration of LoRa gateway device to connect to cluster.
5. Enable the connection connection between the cluster, LoRa gateway and sensor devices using Chirpstack application server user interface (UI).

Preparing the raspberry Pis

It's possible to acquire a pre-built raspberry pi cluster such as [Picocluster](#), but the pre-installation step must be carried out on each Raspberry Pi (RPI).

1. If you're starting from installing the OS on RPi, download the latest raspbian buster [image](#) and burn the image using [etcher](#) or similar tools. If you're using OS pre-installed RPi, skip steps 1 and 2 and go to step 3.
 - a. Run etcher to burn images:



b. Or run etcher from cli (MAC) as admin to burn images by:

```
$ sudo /Applications/balenaEtcher.app/Contents/MacOS/balenaEtcher
```

2. Allow ssh in SD cards

```
$ touch /Volumes/boot/ssh
```

3. SSH into each Pis

Note: you have to have access to the same network that you're accessing to the cluster.

```
$ ssh pi@raspberrypi.local
```

4. Add the following text inside the file `/boot/cmdline.txt` with sudo

```
cgroup_enable=memory
```

5. Update the `/etc/network/interfaces` file for setting specific IP address. For example:

```
auto eth0
iface eth0 inet static
address 192.168.1.10
netmask 255.255.255.0
gateway 192.168.1.1
dns-nameservers 192.168.1.1 8.8.8.8
```

6. Update the `/etc/hosts` with your cluster's of IP addresses

```
192.168.1.10 pc0
192.168.1.11 pc1
192.168.1.12 pc2
...
192.168.1.N pcN
```

7. Edit the hostname and password by editing ``raspi-config``

```
$ sudo raspi-config
```

8. Reboot

```
$ sudo reboot
```

9. Lastly, enable passwordless connection by copying your authorized key from your machine

```
$ ssh-copy-id pi@name
```

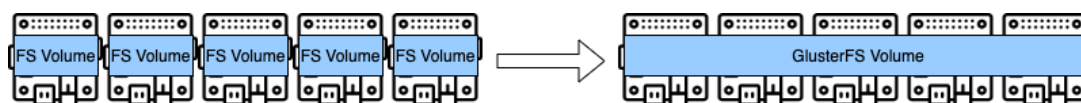
The Pi is now ready to run ansible scripts.

Creating Kubernetes Cluster

Make sure you have installed Ansible 2.4.0+ on your host machine.

1. Creating GlusterFS volume

We use glusterfs on the pico clusters as the file system on which Kubernetes creates PersistentVolumes.



Before running Ansible, make sure if there are any gluster volumes running by using the following command on the master node.

```
$ sudo gluster volume status
```

If there are any gluster volumes, first stop and delete them. For example, run the following command to stop and delete the volumes ``mosquitto_volume``, ``postgres_volume`` and ``influxdb_volume``:

```
$ sudo gluster volume stop mosquitto_volume
$ sudo gluster volume stop postgres_volume
$ sudo gluster volume stop influxdb_volume
$ sudo gluster volume delete mosquitto_volume
$ sudo gluster volume delete influxdb_volume
$ sudo gluster volume delete postgres_volume
```

Also, make sure that the glusterfs volume replicas are set to ``3`` when running on PicoCluster10, and ``2`` when running on PicoCluster5. Edit all instances of ``replicas`` in file ``roles/glusterfs/master/tasks/main.yml``

2. Configuring Host files

Add the system information gathered above into a file called `hosts.ini`. For example:

```
[master]
192.16.35.12
[node]
192.16.35.[10:11]
[kube-cluster:children]
master
node
```

If you're working with ubuntu, add the following properties to each host
`ansible_python_interpreter='python3'`:

```
[master]
192.16.35.12 ansible_python_interpreter='python3'
[node]
192.16.35.[10:11] ansible_python_interpreter='python3'
[kube-cluster:children]
master
node
```

Before continuing, edit `group_vars/all.yml` to your specified configuration. For example, we choose to run `flannel` instead of calico, and thus:

```
# Network implementation('flannel', 'calico')
network: flannel
```

Note: Depending on your setup, you may need to modify `cni_opts` to an available network interface. By default, `kubeadm-ansible` uses `eth1`. Your default interface may be `eth0`.

Note: Edit the "master_vpn" variable in `all.yml` file. The value should be equal to the IP address of the master node in the VPN interface.

3. After going through the setup, run the `site.yml` playbook:

```
$ ansible-playbook -i hosts.ini site.yml
```

```
...
==> master1: TASK [addon : Create Kubernetes dashboard deployment]
*****
==> master1: changed: [192.16.35.12 -> 192.16.35.12]
==> master1:
==> master1: PLAY RECAP
*****
```

```

==> master1: 192.16.35.10      : ok=18   changed=14   unreachable=0
failed=0
==> master1: 192.16.35.11      : ok=18   changed=14   unreachable=0
failed=0
==> master1: 192.16.35.12      : ok=34   changed=29   unreachable=0
failed=0

```

4. Verify cluster is fully running using kubectl:

```
$ export KUBECONFIG=~/.admin.conf
```

```
$ kubectl get node
```

NAME	STATUS	AGE	VERSION
master1	Ready	22m	v1.6.3
node1	Ready	20m	v1.6.3
node2	Ready	20m	v1.6.3

```
$ kubectl get po -n kube-system
```

NAME	READY	STATUS	RESTARTS	AGE
etcd-master1	1/1	Running	0	23m
...				

5. Changing the password

If you need to change the default password of the user picocluster, run from `kubeadm-ansible` directory:

```
$ ansible-playbook -i hosts.ini change-password.yml --extra-vars
newpassword=NEWPASSWORD
```

The cluster is running a Kubernetes cluster and ready for installing the software stack. Check [this repository](#) out for more information on installing and debugging.

Installing Chirpstack software and other software stacks

In order to deploy chirpstack and software stack with kubernetes:

1. Replace your cluster ip with [myclusterIP] value

myclusterIP is the host IP address, acts as a manager in GlusterFS

```
$ export myclusterIP=192.168.1.10
```

2. Generate GlusterFS Endpoints files

Edit the IP address range in the `generate_glusterfs_endpoints.sh` file first using a text editor

```
for i in {11..19} -> change your cluster address range here
do
echo "- addresses:
  - ip: 192.168.9.$i
  ports:
```

3. Run the glusterFS generation script

```
$ ./generate_glusterfs_endpoints.sh
```

4. Deploy the stack using the script

Run following to deploy all software stack

```
$ sh ./deploy_all.sh
```

5. After running the deployment, wait for them to fully deployed, then make sure all the pods, services and deployments are running

```
$ kubectl get pods
$ kubectl get deployments
$ kubectl get services
```

6. After the pod creation, create the databases:

```
$ sh ./postgres/create_db.sh
$ sh ./influxdb/create_users.sh
```

7. Deleting

If you wish to delete the stack, run following to delete all:

```
$ sh ./delete_all.sh
```

8. Exposing to external IPs

Kubernetes services are exposing external IPs to access from outside, you can enable/disable exposing IP addresses at the end of following files:

```
port:8080 /chirpstack-application-server/service.yml
port:8000 /chirpstack-network-server/service.yml
port:9090 /monitoring/prometheus.yml
port:3000 /monitoring/grafana.yml
port:1883 /mosquitto/service.yml
port:8086 /influxdb/service.yml
port:1880 /nodered/service.yml
```

End of the file looks like

```
externalIPs:
- $myclusterIP
```

Configuring LoRa gateways

LoRa gateways are configured via the web interface.

1. Open the web interface of the LoRa gateway using its IP address (E.g., <https://192.168.9.2/>)
2. Ignore the error regarding the certificate. You should see the log in screen
3. Login using the user name and password
4. Go to the LoRaWAN section

← → ↻ ⚠ Not secure | 192.168.9.2/lora/network

MULTITECH mPower™ Edge Intelligence Conduit - Application Enablement Platform
MTCDTIP-L4E1-267A Firmware 5.3.0

Home
Save and Apply
LoRaWAN®
Network Settings
Setup
Cellular
Wireless
Firewall
SMS
Tunnels
Administration
Status & Logs
Commands
Apps
Help

LORAWAN NETWORKING ?

LoRa Mode

PACKET FORWARDER	Packet Forwarder	4.0.1-r32.0	RUNNING
	Network Server	2.4.12	DISABLED
	Lens Server	2.4.12	DISABLED
	Basic Station	2.0.5-1-r2.0	DISABLED

LoRa Card Information

Gateway EUI	00-80-00-00-A0-00-5E-14
Frequency Band	868
FPGA Version	31

[Upgrade](#)

LoRa Packet Forwarder Configuration [Manual Configuration](#)

Gateway Info

UUID	8E6655D1-60BA-F575-8DE2-EC5E9F86EC5D
Serial Number	20813434

SX1301

Channel Plan	Additional Channels 1 (MHz)
EU868	867.5

Duty Cycle

<input type="checkbox"/> Enable Duty-Cycle	Duty-Cycle Period
	Duty-Cycle Ratio

Basics

<input checked="" type="checkbox"/> Public	Intervals
	Keep Alive Interval (s)
	10

5. Select "PACKET FORWARDER" for LoRa Mode
6. Select EU868 for Channel plan
7. Set Upstream and Downstream ports

Duty-Cycle Ratio

Basics

☒ Public

Gateway ID Source
Manual

Gateway ID
00800000A0005E14

Packet Forwarder Path
/opt/lorawan/lorawan_pkt_fwd

Intervals

Keep Alive Interval (s)
10

Stat Interval (s)
20

Push Timeout (ms)
100

Autoquit Threshold
60

Server

Network
Manual

Server Address
127.0.0.1

Upstream Port
1782

Downstream Port
1782

Forward CRC

☐ Forward CRC Disabled

☒ Forward CRC Error

☒ Forward CRC Valid

Beacon Configuration

☐ Enable Beacons

☐ Disable Beacon Frequency Hopping

Beacon Frequency (MHz)
869.525

Beacon Power (dBm)
27

Info Descriptor
0

Beacon Latitude (°)
0

Beacon Longitude (°)
0

Submit **Reset To Default**

Copyright © 1995 - 2021 by Multi-Tech Systems, Inc. - All rights reserved

8. Leave the rest as is and click on "Submit" to save
9. Go to Setup -> Network Interfaces

← → ↻ ⚠ Not secure | 192.168.9.2/setup/network/interfaces

MULTITECH mPower™ Edge Intelligence Conduit - Application Enablement Platform
MTCDTIP-L4E1-267A Firmware 5.3.0

Home
Save and Apply
LoRaWAN ®
Setup
 Network Interfaces
 WAN Configuration
 Global DNS

NETWORK INTERFACES CONFIGURATION ? **Reset To Default**

Name	Direction	Type	IP Mode	IP Address	Bridge	Options
eth0	WAN IPv4	ETHER	Static	192.168.9.2/24		
ppp0	WAN IPv4	PPP	PPP			
wlan0	WAN IPv4	WIFI_AS_WAN	DHCP Client			
wlan1	LAN	WIFI_AP	--	--	br0	
br0	LAN IPv4	BRIDGE	Static	192.168.2.1/24	br0	

10. Click on the edit icon to the right of the "eth0" interface

11. Configure as shown in the screenshot with the IP address of the LoRa gateway and click on "Submit" to save.

The screenshot shows the Multitech mPower Edge Intelligence Conduit web interface. The browser address bar displays "192.168.9.2/setup/network/interfaces/eth0". The page title is "mPower™ Edge Intelligence Conduit - Application Enablement Platform". The left sidebar contains navigation links: Home, Save and Apply, LoRaWAN @, Setup (highlighted), and Network Interfaces (highlighted). The main content area is titled "NETWORK INTERFACE CONFIGURATION - ETH0". It contains a "Direction" dropdown menu set to "WAN". Below this is the "IPv4 Settings" section, which includes a "Mode" dropdown menu set to "Static". The "IP Address" field is set to "192.168.9.2", the "Mask" field is set to "255.255.255.0", the "Gateway" field is set to "192.168.9.1", the "Primary DNS Server" field is set to "192.168.9.1", and the "Secondary DNS Server" field is set to "8.8.8.8". At the bottom of the form are "Submit" and "Cancel" buttons.

12. Go to Administration -> Access Configuration, and check the boxes for SSH and ICMP access as shown in the screenshot below.

← → ↻ ⚠ Not secure | 192.168.9.2/administration/access-configuration

MULTITECH mPower™ Edge Intelligence Conduit - Application Enablement Platform
MTCDTIP-L4E1-267A Firmware 5.3.0

Home
Save and Apply
LoRaWAN ®
Setup
Cellular
Wireless
Firewall
SMS
Tunnels
Administration
User Accounts
Self-Diagnostics (beta)
Access Configuration
RADIUS Configuration
X.509 Certificate
X.509 CA Certificates
Remote Management
Notifications
Web UI Customization
Firmware Upgrade
Package Management
Save/Restore
Debug Options
Usage Policy
Support

ACCESS CONFIGURATION ?

Web Server

HTTP Redirect to HTTPS	HTTPS	Authorization
<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Via WAN	Session Timeout (minutes)
<input checked="" type="checkbox"/> Via LAN	Port	60
<input checked="" type="checkbox"/> Via WAN	443	
Port		
80		

[Show ↓](#)

HTTPS Security

SSH Settings

<input checked="" type="checkbox"/> Enabled	Port	<input checked="" type="checkbox"/> Via LAN	<input checked="" type="checkbox"/> Via WAN
	22		

[Show ↓](#)

SSH Security

Reverse SSH Tunnel

<input type="checkbox"/> Enabled	Server	Remote Port
		2222
Username	Authentication Method	Password
	Password	

ICMP Settings

<input checked="" type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Respond to LAN	<input checked="" type="checkbox"/> Respond to WAN
---	--	--

SNMP Settings

<input checked="" type="checkbox"/> Via LAN	<input type="checkbox"/> Via WAN
---	----------------------------------

Modbus Slave

<input type="checkbox"/> Enabled	<input checked="" type="checkbox"/> Via LAN	Port
		1502

13. Click Submit to save

14. Finally, click on "Save and Apply" on the left to save the configuration. If prompted to restart the gateway, please do.

Installing Chirpstack gateway bridge on the LoRa gateways

1. First, login to the LoRa gateway over ssh

Eg. `ssh fogguru@192.168.9.2`

2. Update the opkg cache:

`sudo opkg update`

3. Download the latest chirpstack-gateway-bridge .ipk package from:
<https://artifacts.chirpstack.io/vendor/multitech/conduit/>.
 Example (assuming you want to install
 chirpstack-gateway-bridge_3.10.0-r1_arm926ejste.ipk):
 wget https://artifacts.chirpstack.io/vendor/multitech/conduit/chirpstack-gatewa
4. Install it using the opkg package-manager utility. Example (assuming the same .ipk
 file):
 opkg install chirpstack-gateway-bridge_3.10.0-r1_arm926ejste.ipk

Configure Chirpstack gateway bridge on the LoRa gateways

The Chirpstack gateway bridge on each LoRa gateway needs to be configured so that it can connect to the fog clusters.

1. First, login to the LoRa gateway over ssh
 Eg. ssh fogguru@192.168.9.2
2. Open the configuration file for editing
 sudo nano /var/config/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml
3. Update the udp port selected for the LoRa packet forwarder
 udp_bind = "0.0.0.0:1782"
4. Edit the information about MQTT server (E.g., to connect to MQTT on 192.168.9.10)
 # Generic MQTT authentication.
 [integration.mqtt.auth.generic]
 # MQTT server (e.g. scheme://host:port where scheme is tcp, ssl or ws)
 server="tcp://192.168.9.10:1883"

 # Connect with the given username
 username="USERNAME"

 # Connect with the given password
 password="PASSWORD"

5. Save the file and exit
6. Restart the Chirpstack gateway bridge service

```
sudo /etc/init.d/chirpstack-gateway-bridge restart
```

Add sensors to Chirpstack Application server

Chirpstack Application server is available on all fog clusters. However, at the moment sensors are added on three of them.

To open Chirpstack Application server, from your browser open one of the URL of chirpStack Application, i.e., <http://192.168.9.10:8080> with username and password.

After you log in, you will see the screen below

ID	Name	Service-profile	Description
6	indoor-environment-app	service-profile-01	Indoor environment sensor app
3	people-counter-app	service-profile-01	people counting app
4	smart-water-ions-app	service-profile-01	Smart water ions app
2	traffic-counter-app	service-profile-01	Traffic counter app
5	weather-station-app	service-profile-01	Weather station app
1	wind-sensor-app	service-profile-01	Wind sensor app

Rows per page: 10 ▼ 1-6 of 6 < >

To add new sensors,

- First create a device profile. To do so, click on “Device-profiles” on the left, and then click on the “CREATE” button.

ChirpStack

Search organization, application, gateway or device

admin

Network-servers
Gateway-profiles
Organizations
All users
API keys

chirpstack

Org. settings
Org. users
Service-profiles
Device-profiles
Gateways
Applications
Multicast-groups

Device-profiles / Create

GENERAL JOIN (OTAA / ABP) CLASS-B CLASS-C CODEC TAGS

Device-profile name *

new-sensor

A name to identify the device-profile.

Network-server *

network-server-01

The network-server on which this device-profile will be provisioned. After creating the device-profile, this value can't be changed.

LoRaWAN MAC version *

1.0.2

The LoRaWAN MAC version supported by the device.

LoRaWAN Regional Parameters revision *

A

Revision of the Regional Parameters specification supported by the device.

Max EIRP *

0

Maximum EIRP supported by the device.

Geolocation buffer TTL (seconds)

0

The time in seconds that historical uplinks will be stored in the geolocation buffer.

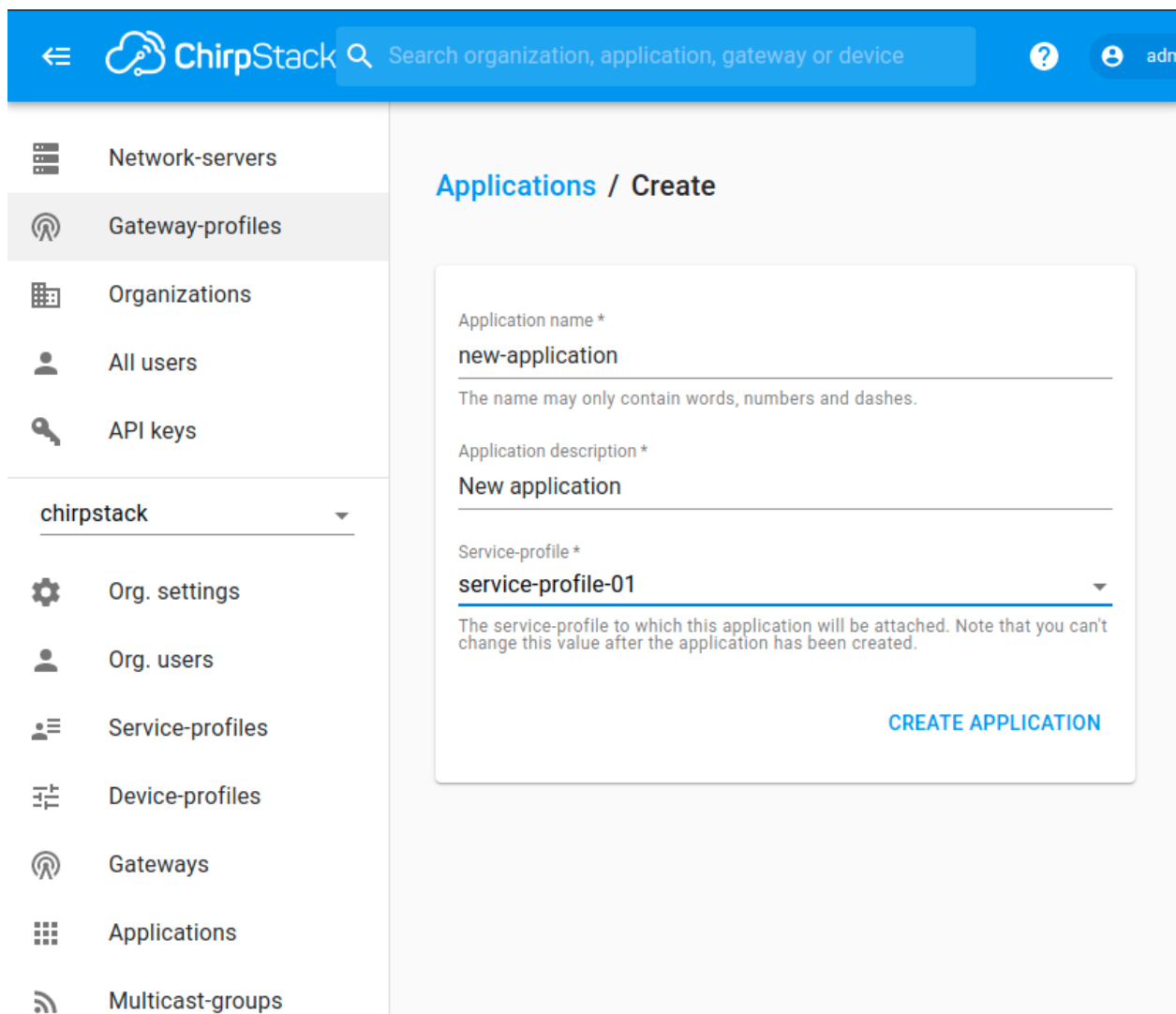
Geolocation minimum buffer size

0

The minimum buffer size required before using geolocation (when enabled in the Service Profile). Using multiple uplinks for geolocation can increase the accuracy of the geolocation results.

CREATE DEVICE-PROFILE

- Complete the fields in the “GENERAL” tab
 - If the sensor uses OTAA for joining the LoRa network (check with the device manual), go to “JOIN OTAA/ABP) tab and check the “Device supports OTAA” checkbox
 - Click on “CREATE DEVICE PROFILE” to finish
-
- Create an “application” by going to Applications -> “CREATE”
 - Complete the fields, and click on “CREATE APPLICATION” to complete



The screenshot shows the ChirpStack web interface. The top navigation bar is blue with the ChirpStack logo, a search bar, and user information. The left sidebar contains a list of navigation items: Network-servers, Gateway-profiles (highlighted), Organizations, All users, API keys, and a dropdown for 'chirpstack'. Below this is a list of organization settings: Org. settings, Org. users, Service-profiles, Device-profiles, Gateways, Applications, and Multicast-groups. The main content area is titled 'Applications / Create' and contains a form with the following fields: 'Application name *' with the value 'new-application' and a note 'The name may only contain words, numbers and dashes.'; 'Application description *' with the value 'New application'; and 'Service-profile *' with a dropdown menu showing 'service-profile-01' and a note 'The service-profile to which this application will be attached. Note that you can't change this value after the application has been created.' A 'CREATE APPLICATION' button is located at the bottom right of the form.

ChirpStack Search organization, application, gateway or device ? adm

Network-servers

Gateway-profiles

Organizations

All users

API keys

chirpstack

Org. settings

Org. users

Service-profiles

Device-profiles

Gateways

Applications

Multicast-groups

Applications / Create

Application name *

new-application

The name may only contain words, numbers and dashes.

Application description *

New application

Service-profile *

service-profile-01

The service-profile to which this application will be attached. Note that you can't change this value after the application has been created.

[CREATE APPLICATION](#)

Add a device to the application

- Click on the newly created application, then create on "CREATE" under the "Devices" tab

The screenshot shows the ChirpStack web interface. The top navigation bar is blue with the ChirpStack logo, a search bar, and user information. The left sidebar contains a menu with icons and labels for various system components. The main content area displays the 'Create' form for a new device, with tabs for 'GENERAL', 'VARIABLES', and 'TAGS'. The 'GENERAL' tab is active, showing fields for 'Device name', 'Device description', 'Device EUI', and 'Device-profile'. The 'Device EUI' field is highlighted with a blue background. A checkbox for 'Disable frame-counter validation' is present, with a note explaining its security implications. A 'CREATE DEVICE' button is at the bottom right of the form.

ChirpStack Search organization, application, gateway or device

Applications / indoor-environment-app / Devices /

Create

GENERAL VARIABLES TAGS

Device name *

sensor-name

The name may only contain words, numbers and dashes.

Device description *

Description of sensor

Device EUI *

00 04 a3 0b 00 ef d7 17 MSB ↻

Device-profile *

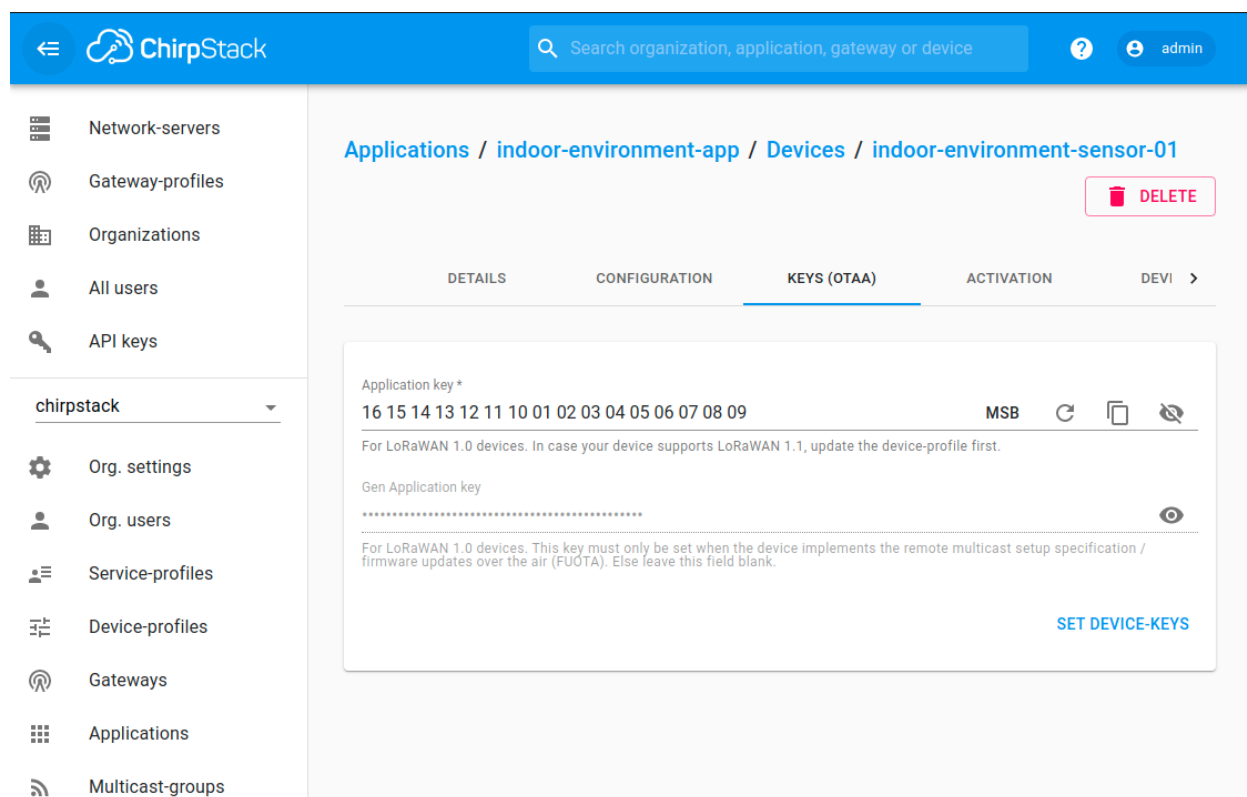
wind-sensor-profile

☐ Disable frame-counter validation

Note that disabling the frame-counter validation will compromise security as it enables people to perform replay-attacks.

CREATE DEVICE

- Click on “CREATE DEVICE” after completing the form. You may need to get the information about the sensor from the vendor.
- You will be asked to enter the Keys for the device. Fill the device key you got from the vendor and click on “SET DEVICE-KEYS”



LivingFog Maintenance

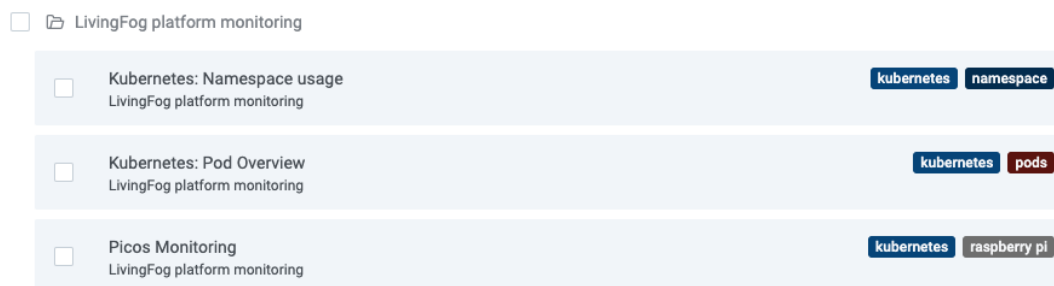
Platform monitoring

Most straightforward way of monitoring or checking the system health is to access dashboards running on Server PC. After configuring the VPN on your machine, you're able to access the system monitoring dashboards.

To access dashboards, type the URL of grafana, i.e., <http://192.168.9.71:3002/> on your browser, with the credentials of username and password.

There are 2 main types of dashboards:

1. Sensor data: 12 types of Sensor data dashboards and it can be found in *la-marina-sensors* folder in a dashboards management.
2. Platform monitoring dashboards: 3 types of platform monitoring dashboards and it can be found in *LivingFog* platform monitoring folder



a. Kubernetes: Namespace usage

This dashboard is mainly used for tracking the usage for hackathon participants. But it can be used for keeping track of the resource usage of your new namespace.

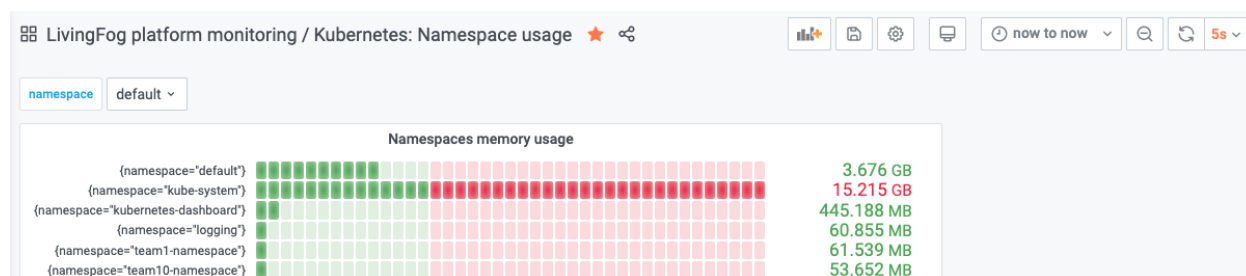


Figure 7: Kubernetes namespace usage overview

b. Kubernetes: Pod overview

In LivingFog platform, kubernetes pods/deployments/services are the applications deployed on the clusters.

This dashboard helps keeping track of pods and their activities. It shows the resource usage and the status of the pods running on specific clusters.

For example, if you notice some pods are failing, you might want to restart that specific pod manually using **ssh** connection and **kubernetes** command.

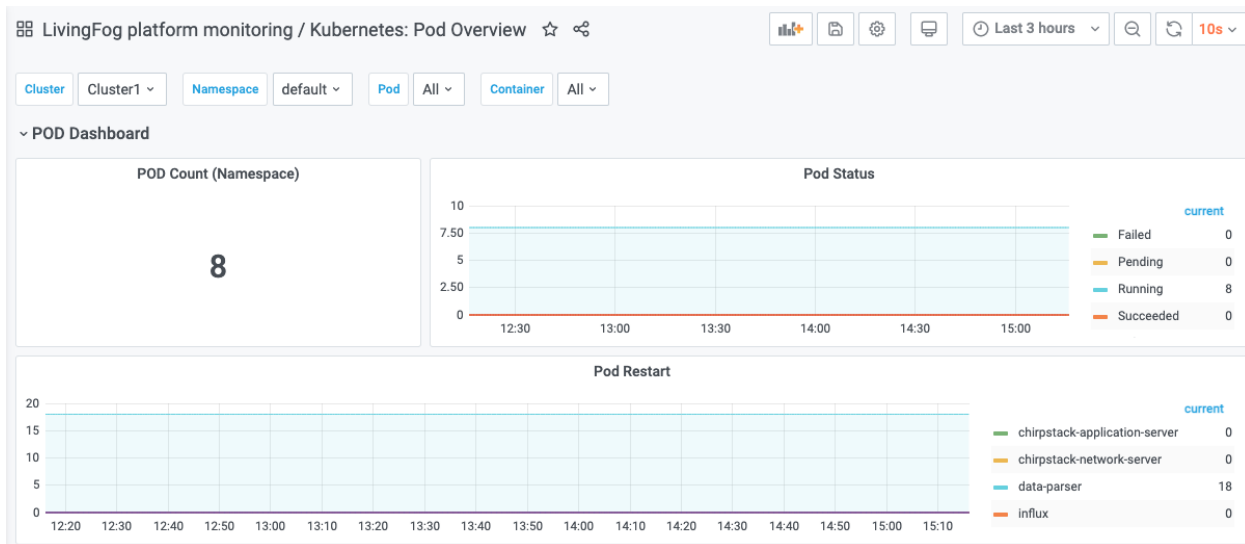


Figure 8: Kubernetes pods status

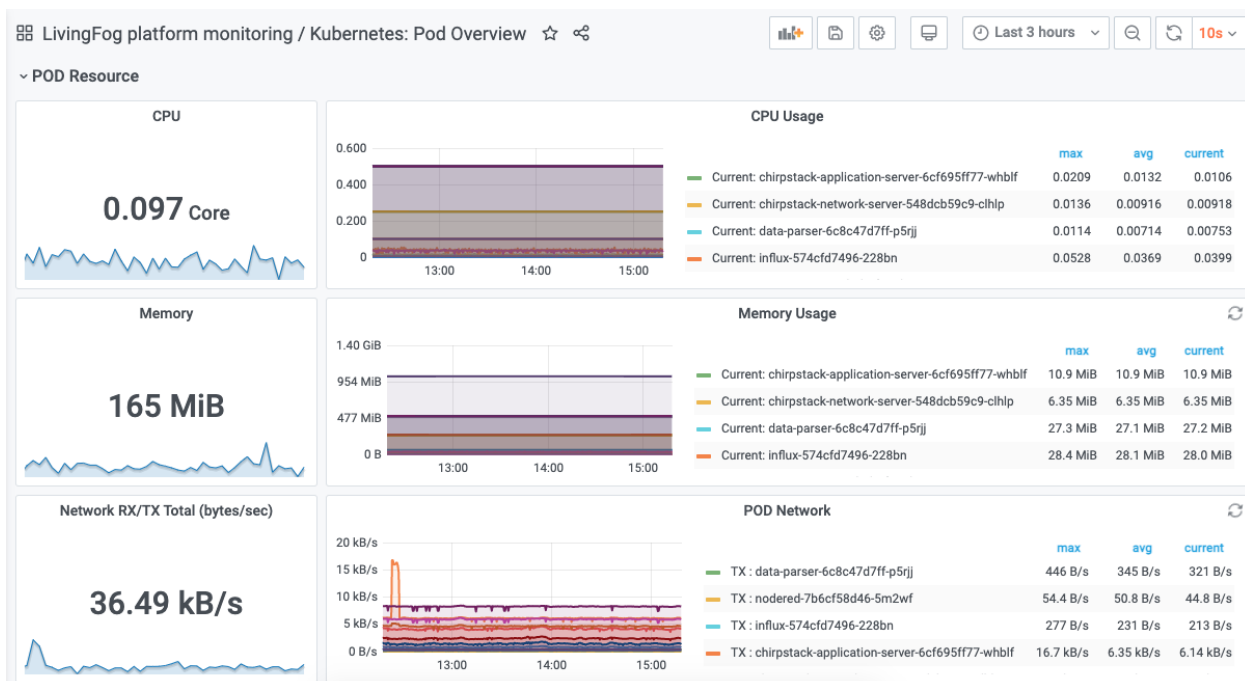


Figure 9: Kubernetes pods resource usage by cluster

c. Picos monitoring

In the LivingFog platform, Fog clusters are built on Raspberry Pi machines. Picos Monitoring dashboard will help monitor the activities of the machines.

You can check the performance and the status of the clusters by selecting the **Cluster** and **Node**.

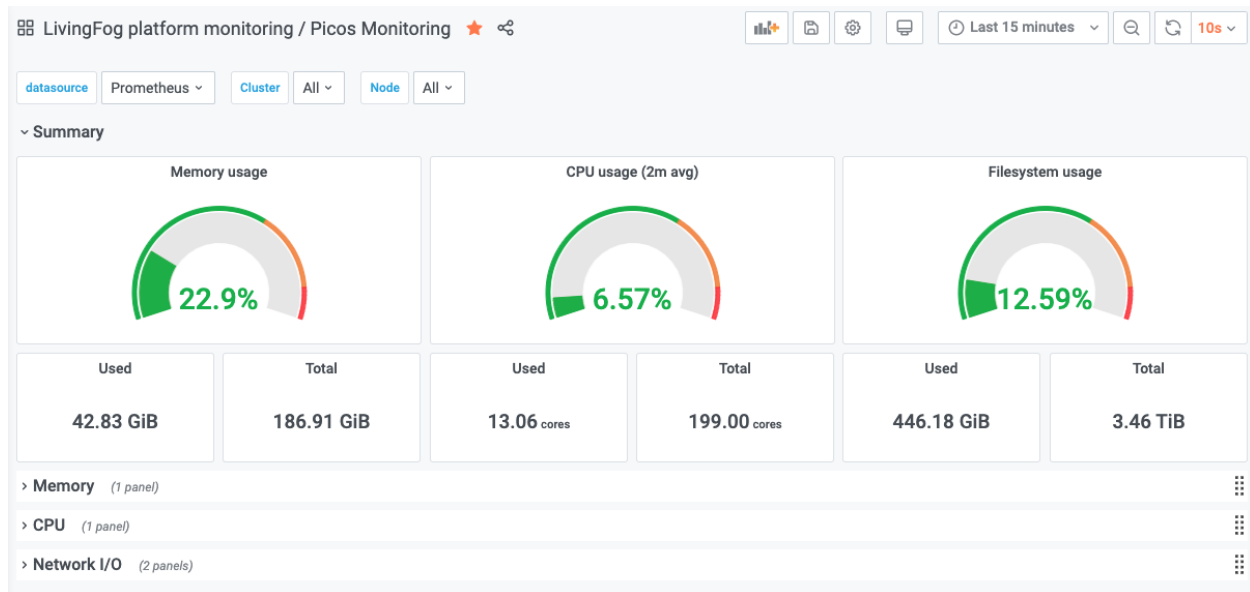


Figure 10: Raspberry Pi machines resource usage by cluster

Note: If you want to add a new dashboard, you can add it using Prometheus and InfluxDB data sources, which are already provided.

Note: There are **no alert systems** implemented yet. Alerts are added and configured in the Alert Tab of any dashboard graph panel, letting you build and visualize an alert using existing queries. To persist your alert rule changes remember to save the dashboard.

Manual debugging using SSH

All the debugging and stopping/starting the services running on the machines should be done using SSH. Otherwise, you have to be physically present on site to check the platform.

By configuring the VPN on your machine, you're also able to connect to most of the hardware using SSH connection. Table 3 shows available machines and their SSH credentials.

Log in to desired machine using SSH by:

```
$ ssh [username]@[IP_address]
[username]@[IP_address]'s password: [password]
```

For example:

```
$ ssh picocluster@192.168.9.10
```

picocluster@192.168.9.10's password:

Software stacks on either RPi clusters or Server PC are deployed as Kubernetes deployments, services and pods. After logging in using SSH to the certain machine, you will be able to use kubernetes commands to check if the pods are running.

For example:

```
picocluster@pc0:~ $ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
chirpstack-application-server-6cf695ff77-whblf	1/1	Running	0	4d5h
chirpstack-network-server-548dc9c9-clhlp	1/1	Running	0	4d5h
data-parser-6c8c47d7ff-p5rjj	1/1	Running	15	12d
influx-574cfd7496-228bn	1/1	Running	0	12d
mosquitto-6c88b5b6f4-t2kpj	1/1	Running	0	12d
nodored-7b6cf58d46-5m2wf	1/1	Running	0	26d
postgres-7747f9b7bf-j72qn	1/1	Running	0	4d5h
redis	1/1	Running	0	26d
yourpod	1/1	Running	2	41d

Same for starting/stopping pods/services/deployments. To start/stop/restart the pods/services/deployments:

You can run the following commands one by one to start the deployments on RPi

```
$ kubectl apply -f ./mosquitto/mosquitto-glusterfs-endpoint.yaml
$ kubectl apply -f ./mosquitto/storage.yaml
$ kubectl apply -f ./mosquitto/configmap.yaml
$ kubectl apply -f ./mosquitto/deployment.yaml
$ envsubst < ./mosquitto/service.yaml | kubectl apply -f -

$ kubectl apply -f ./influxdb/influxdb-glusterfs-endpoint.yaml
$ kubectl apply -f ./influxdb/storage.yaml
$ kubectl apply -f ./influxdb/deployment.yaml
$ envsubst < ./influxdb/service.yaml | kubectl apply -f -

$ kubectl apply -f ./postgres/
$ kubectl apply -k redis/.

$ kubectl apply -f ./chirpstack-network-server/configMap.yaml
$ kubectl apply -f ./chirpstack-network-server/deployment.yaml
$ envsubst < ./chirpstack-network-server/service.yaml | kubectl apply -f -
```

```

$ kubectl apply -f ./chirpstack-application-server/configMap.yml
$ kubectl apply -f ./chirpstack-application-server/deployment.yml
$ envsubst < ./chirpstack-application-server/service.yml | kubectl apply -f -
-

$ kubectl apply -f ./monitoring/configmap.yml
$ kubectl apply -f ./monitoring/kube-state-metrics.yml
$ kubectl apply -f ./monitoring/node-exporter.yml
$ kubectl apply -f ./monitoring/rbac.yml
$ envsubst < ./monitoring/grafana.yml | kubectl apply -f -
$ envsubst < ./monitoring/prometheus.yml | kubectl apply -f -

$ kubectl apply -f ./nodered/deployment.yml
$ envsubst < ./nodered/service.yml | kubectl apply -f -

```

You can run the following commands to stop each deployments RPis

```

$ kubectl delete -f ./mosquitto/
$ kubectl delete -f ./influxdb/
$ kubectl delete -f ./postgres/
$ kubectl delete -k redis/.
$ kubectl delete -f ./chirpstack-network-server/
$ kubectl delete -f ./chirpstack-application-server/
$ kubectl delete -f ./monitoring/
$ kubectl delete -f ./nodered/
$ kubectl delete pvc mosquitto postgres-pv-claim postgresinit-pv-claim
$ kubectl delete pv mosquitto-pv-volume $ postgres-pv-volume
postgresinit-pv-volume

```

Use case: IoT Fablab

FogGuru has applied the LivingFog platform in IoT Fablab in Las Naves, where a number of IoT heterogeneous sensors are installed in La Marina de València to measure and process data about water quality, wind, sea wave, outdoor and indoor environment, people counter, and traffic.

Hardware

The hardware used in IoT Fablab are listed as follows:

LoRa Gateways and Fog clusters

No	Device	Model	IP Address / IP Address Range	Location
1	LoRa Gateway 1	Multitech MTCDTIP-L4E1-267A-868 LTE Cat 4 AEP Conduit IP67	192.168.9.2	Varadero
2	LoRa Gateway 2	Multitech MTCDTIP-L4E1-267A-868 LTE Cat 4 AEP Conduit IP67	192.168.9.3	Tinglado 2
3	LoRa Gateway 3	Multitech Conduit - MTCDT-L4E1-247A-868-EU-GB	192.168.9.4	Tinglado 5
4	LoRa Gateway 5	Multitech Conduit - MTCDT-L4E1-247A-868-EU-GB	192.168.9.6	Tinglado 2
5	Pico Cluster 1	Pico 10H cluster RPi4, assembled cube + 320GB of SD storage	192.168.9.10 - 192.168.9.19	Varadero
6	Pico Cluster 2	Pico 10H cluster RPi4, assembled cube + 320GB of SD storage	192.168.9.20 - 192.168.9.29	Varadero
7	Pico Cluster 3	Pico 10H cluster RPi4, assembled cube + 320GB of SD storage	192.168.9.30 - 192.168.9.39	Varadero
8	Pico Cluster 4	Pico 10H cluster RPi4, assembled cube + 320GB of SD storage	192.168.9.40 - 192.168.9.49	Varadero
9	Pico Cluster 5	Pico 10H cluster RPi4, assembled cube + 320GB of SD storage	192.168.9.50 - 192.168.9.59	Varadero
10	Pico Cluster 6	Pico 5 Raspberry PI4 4GB + 160 GB of SD storage	192.168.9.60 - 192.168.9.64	Varadero
11	Pico Cluster 7	Pico 5 Raspberry PI4 4GB + 160 GB of SD storage	192.168.9.65 - 192.168.9.69	Varadero
12	Desktop server	Dell Optiplex 9020 Desktop PC Computer Intel Core i7-4770 3.40 Ghz 32GB Ram 240GB SSD + 1Tb SSHD	192.168.9.71	Varadero

Sensors

No	sensor name	Model	Location (latitude, longitude)
1	PEOPLE COUNTER 1	Parametric Radar People Counter with LoRaWAN® for Outdoor Applications PCR2-EU868-OD	39.46274, -0.32237
2	PEOPLE COUNTER 2	Parametric Radar People Counter with LoRaWAN® for Outdoor Applications PCR2-EU868-OD	39.4614, -0.33023
3	PEOPLE COUNTER 3	Parametric Radar People Counter with LoRaWAN® for Outdoor Applications PCR2-EU868-OD	39.4602, -0.33219
4	PEOPLE COUNTER 4	Parametric Radar People Counter with LoRaWAN® for Outdoor Applications PCR2-EU868-OD	39.45685, -0.32967
5	WIND SENSOR 1	DecentLab DL-ATM22	39.46097, -0.32424
6	WIND SENSOR 2	DecentLab DL-ATM22	39.46102, -0.33087
7	WIND SENSOR 3	DecentLab DL-ATM22	39.45655, -0.32962
8	SMART WATER	Libelium Smart Water LoRaWAN	39.45878, -0.33034
9	SMART WATER ION	Libelium Smart Water Ion LoRaWAN	39.46105, -0.32553
10	TRAFFIC COUNTER 1	Parametric TCR-LS LoRaWAN	39.46273, -0.32237
11	TRAFFIC COUNTER 2	Parametric TCR-LS LoRaWAN	39.46253, -0.32424
12	TRAFFIC COUNTER 3	Parametric TCR-LS LoRaWAN	39.45969, -0.33236
13	TRAFFIC COUNTER 4	Parametric TCR-LS LoRaWAN	39.45654, -0.33011
14	TRAFFIC COUNTER 5	Parametric TCR-LS LoRaWAN	39.45568, -0.32786
15	TRAFFIC COUNTER 6	Parametric TCR-LS LoRaWAN	39.45946, -0.33265
16	TRAFFIC COUNTER 7	Parametric TCR-LS LoRaWAN	39.45639, -0.33044
17	TRAFFIC COUNTER 8	Parametric TCR-LS LoRaWAN	39.45564, -0.33004
18	TRAFFIC COUNTER 9	Parametric TCR-LS LoRaWAN	39.45571, -0.32895

19	TRAFFIC COUNTER 10	Parametric TCR-LS LoRaWAN	
20	WEATHER STATION	Libelium-Gill-EX-Machina Smart Weather Forecast LoRaWAN Solution Kit	39.46125, -0.32293
21	INDOOR ENVIRONMENT SENSOR 1	enLink Air Wireless Air Quality Sensor	39.4611, -0.3242
22	INDOOR ENVIRONMENT SENSOR 2	enLink Air Wireless Air Quality Sensor	39.46086, -0.33106
23	INDOOR ENVIRONMENT SENSOR 3	enLink Air Wireless Air Quality Sensor	39.45595, -0.32859
24	Sea current and wave sensor	Nortek AWAC 1MHz	39.45947, -0.30955

Data description

Taking the sensors installed in La Marina as examples, Figure 5 shows the flow of data collection in our fog platform. First, all the sensors and Lora gateways are configured and installed in La Marina. Next, each sensor sends collected data through LoRaWAN to the Chirpstack that it joins during the initialization. The uploaded sensor data is in json format following LoRaWan protocol. Then a data parser decodes the data into hexadecimal numbers and interprets them into readable values according to the dataframe of each sensor. Last, the readable sensor data are sent to MQTT with a unique topic and also used to update the database in influxDB. Both the real-time data in MQTT and historical data in influxDB will be used to design fog applications in hackathon and Fablab.

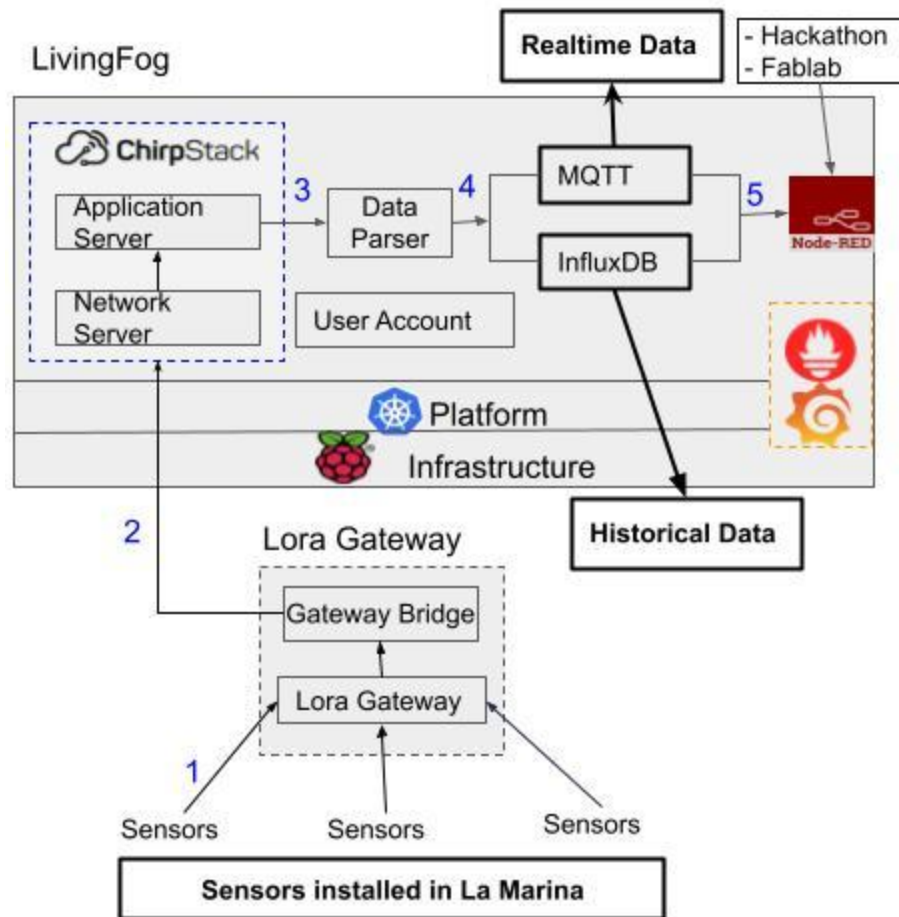


Figure 5: The flow of data collection in LivingFog.

The locations of sensors and gateways are shown as Figure 6 and are available on Google Maps

<https://www.google.com/maps/d/u/0/viewer?hl=en&mid=1qNpP2YknZCjX8HSA3X9bH20DiGkwpuOx&ll=39.45943089803211%2C-0.3219128849847097&z=16>



Figure 6: The locations of sensors and lora gateways

Table 1 shows the sensors and the measured parameters.

No.	Device name	Parameters
1	Sea wave sensor	Sea wave height
		Current speed
		Pressure
		Temperature
2	Outdoor environment sensors	Average wind direction
		Average wind speed
		Precipitation total
		Precipitation intensity
		Solar radiation
		Barometric pressure
		Absolute humidity
		Relative humidity
		Air density
		Air temperature
3	Wind sensor	Wind speed
		Wind direction
		Maximum wind speed
		Air temperature
		Tilt angle X orientation

		Tilt angle Y orientation
		North wind speed
		East wind speed
		Battery voltage
4	Indoor environment sensors	Temperature
		Humidity
		Light level
		VOC's
		bVOC
		Barometric Pressure
		CO2
		O2
		PM1.0, 2.5, 4.0,10.5
5	People counter	Right To Left
		Left To Right
		Left To Right SUM
		Right To Left SUM
		SBX BATT
		SBX PV
		DIFF
		TEMP
6	Water quality sensor	Water pH
		Dissolved oxygen
		Water Conductivity
		ORP(P&S!SOCKETE)
		Water Temperature
		SilverIons
		ChlorideIons
		FluorideIons
7	Traffic sensor	SBX BATT
		SBX PV
		TEMP
		Left CNT (the following parameters are for levels from 0 to 3
		Left AVG
		Right CNT
		Right AVG

Table 1: The sensors deployed at La Marina and parameters measured

Furthermore, for each type of sensor, their detailed information about the sensor type, frequency, MQTT topics, and sample data are provided in Table 2.

Sensor type	Sensor data	Frequency/s	MQTT topics
wind	"Windspeed": "0.28", "Windspeed_unit": "m/s", "Winddirection": "246.9", "Winddirection_unit": "°", "Maximumwindspeed": "0.35", "Maximumwindspeed_unit": "m/s", "Airtemperature": "12.3", "Airtemperature_unit": "°C", "TiltangleXorientation": "-2.8", "TiltangleXorientation_unit": "°", "TiltangleYorientation": "-3.0", "TiltangleYorientation_unit": "°", "Northwindspeed": "-0.11", "Northwindspeed_unit": "m/s", "Eastwindspeed": "-0.25", "Eastwindspeed_unit": "m/s", "Batteryvoltage": "2.87", "Batteryvoltage_unit": "V"	60	wind/73-1 wind/73-2 wind/73-3
people_counter	"RightToLeft": 0, "LeftToRight": 0, "LeftToRight_SUM": 327, "RightToLeft_SUM": 99, "SBX_BATT": 0, "SBX_PV": 0, "DIFF": 228, "TEMP": 21	60	people_counter/7 people_counter/7 people_counter/8 people_counter/4
smart_water	"SequenceNumber": 150, "Length": 29, "Batterylevel": "95", "Batterylevel_unit": "%", "WaterpH": "9.77", "WaterpH_unit": "nan", "Disolvedoxygen": "84.9", "Disolvedoxygen_unit": "%", "WaterConductivity": "-6486.1", "WaterConductivity_unit": "µS/cm", "ORP(P&S!SOCKETE)": "0.337", "ORP(P&S!SOCKETE)_unit": "voltage", "WaterTemperature": "15.04", "WaterTemperature_unit": "°C"	10	smart_water/49
smart_water_lon	"SequenceNumber": 253, "Length": 24, "Batterylevel": "98", "Batterylevel_unit": "%", "Ammonium": "0.0", "Ammonium_unit": "ppm", "NitriteIons": "0.0", "NitriteIons_unit": "ppm", "ChlorideIons": "0.0", "ChlorideIons_unit": "ppm", "WaterTemperature": "14.9", "WaterTemperature_unit": "°C"	10	smart_water_lon/
outdoor_env	"SequenceNumber": 11, "Length": 48, "Averagewinddirection": "270", "Averagewinddirection_unit": "°", "Averagewindspeed": "0.01", "Averagewindspeed_unit": "m/s", "Precipitationtotal": "0.0", "Precipitationtotal_unit": "mm", "Precipitationintensity": "0.0", "Precipitationintensity_unit": "mm", "Solarradiation": "0", "Solarradiation_unit": "W/m²", "Barometricpressure": "1017.8",	10	outdoor_env/75

Table 2. The example of sensor data collected

- database name: sensor_data

```
> select * from people_counter limit 20
name: people_counter
time      DIFF LeftToRight LeftToRight_SUM RightToLeft RightToLeft_SUM SBX_BATT SBX_PV TEMP topic
-----
1614265516539043072 248 0      292      0      44      0      0      29 people_counter/70
1614265576532470016 248 0      292      0      44      0      0      29 people_counter/70
1614265636533863936 248 0      292      0      44      0      0      29 people_counter/70
1614265696556083968 248 0      292      0      44      0      0      29 people_counter/70
1614265756562814976 248 0      292      0      44      0      0      28 people_counter/70
1614265816564773120 248 0      292      0      44      0      0      28 people_counter/70
1614265876593307904 248 0      292      0      44      0      0      28 people_counter/70
1614265936764171008 248 0      292      0      44      0      0      28 people_counter/70
1614265996616588032 248 0      292      0      44      0      0      29 people_counter/70
```

For sea wave data, it stores in a postgresQL. The information of the database is as follows:

- Database host and port: 192.168.9.71:5432
- Database name: marina_test
- Sea current profile table: marina_profile_080321
- Sea wave table: marina_awac_wave_data_080321_2