

# CLOUD WORKSHOP

---

2102544 IoT Fundamentals

10 April 2024

# Workshop outcomes

- Basic understanding of fundamental **Linux command line** operations.
- Capability to **build and deploy** applications using **Docker**.
- Ability to establish connections to a Raspberry Pi via **SSH** and **VNC Viewer**.
- Ability to **access services** running on the Raspberry Pi from a laptop.
- Familiarity with basic **Kubernetes** concepts.
- Ability to configure deployment parameters and deploy services on the **Cloud** using **Rancher**.
- Competence in developing a simple **REST API** server.
- Understanding of how to **integrate various service components** for cohesive functionality.

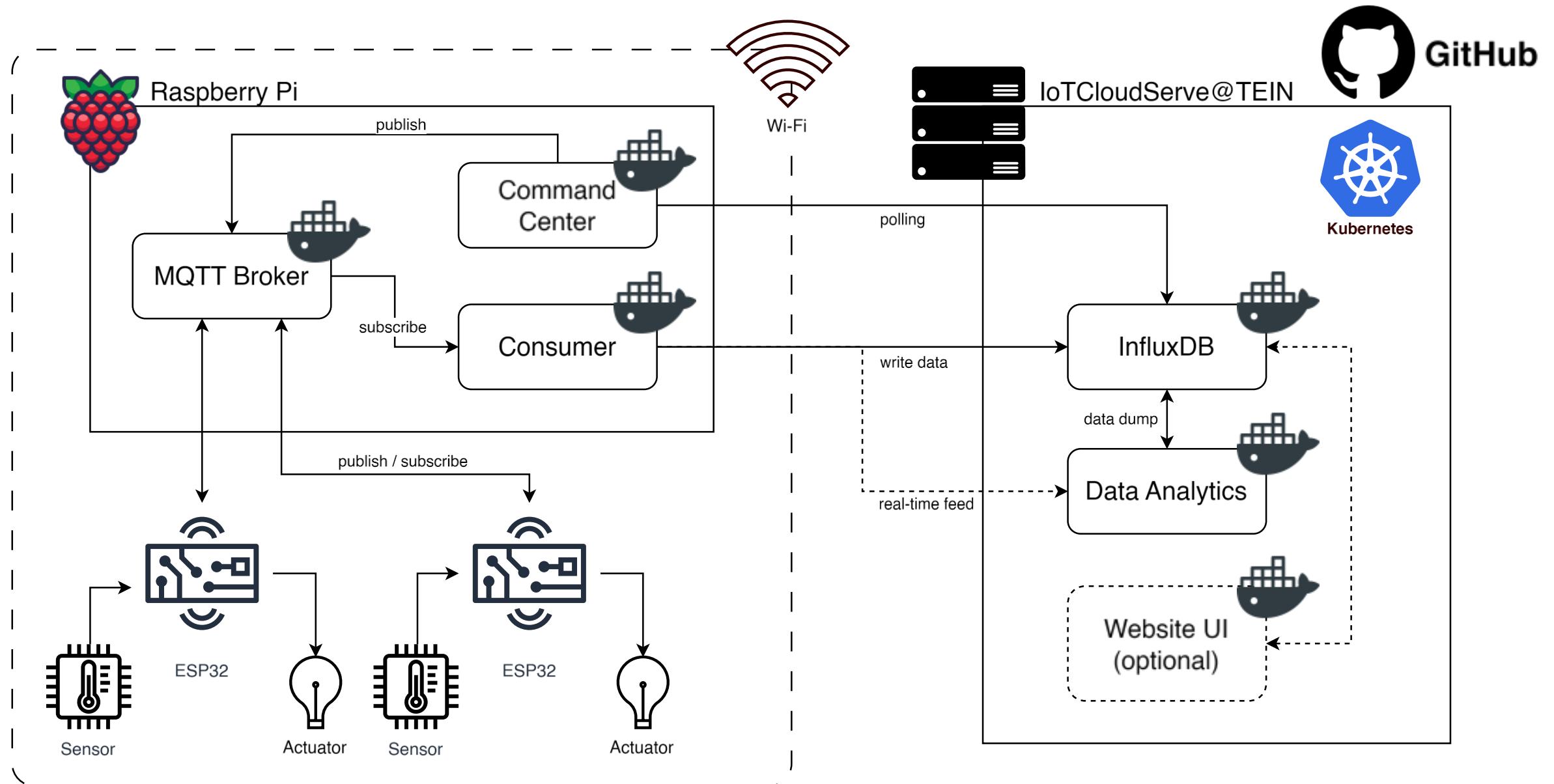
# Before the workshop!

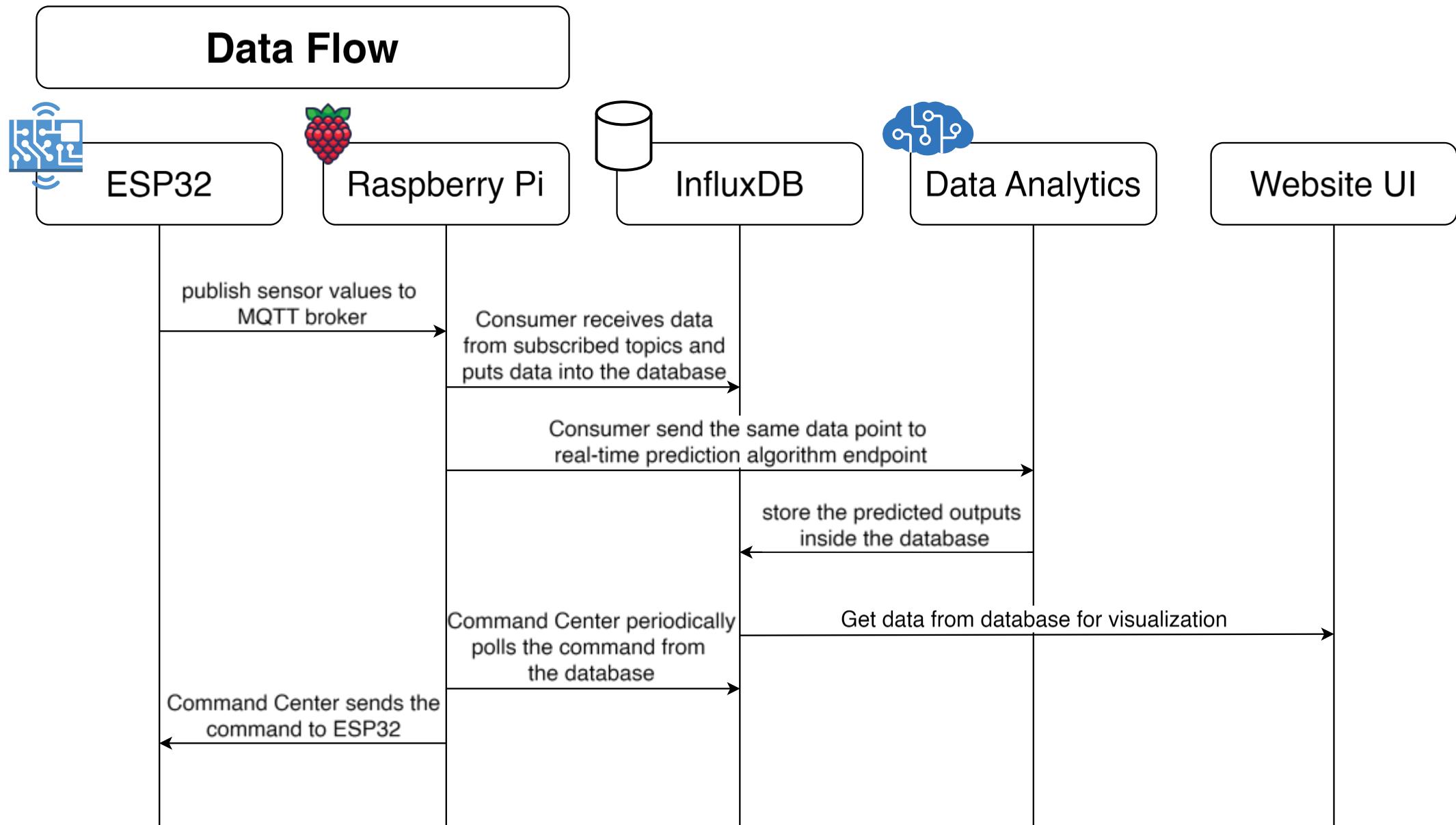
## Please prepare the following:

- Install Git: <https://github.com/git-guides/install-git>
- Install VS code: <https://code.visualstudio.com/download>
- Install Docker Desktop: <https://docs.docker.com/get-docker/>
- Register an account for the Docker Hub: <https://hub.docker.com/>
- Install VNC Viewer: <https://www.realvnc.com/en/connect/download/viewer/>
- Install Postman: <https://www.postman.com/downloads/>

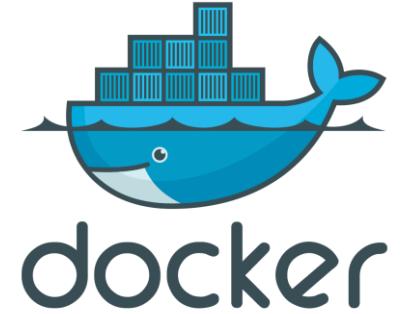
You're free to use other software variants if you prefer, but please note that during the workshop, I'll be demonstrating based on the software list provided.

Therefore, I recommend installing the software listed to ensure you can follow along seamlessly.





# Docker



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

- An open platform for developing, shipping, and running applications.
- Docker provides the ability to package and run an application in a loosely isolated environment called a **container**.
- **Containers** are lightweight and contain everything needed to run the application

# What is a Container and how to run it? Hands-on Guide

Let's follow the guide from

<https://docs.docker.com/guides/walkthroughs/what-is-a-container/>

<https://docs.docker.com/guides/walkthroughs/run-a-container/>

<https://docs.docker.com/guides/walkthroughs/run-hub-images/>

<https://docs.docker.com/guides/walkthroughs/publish-your-image/>

# What is a Container and how to run it?

The screenshot shows the Docker Desktop application interface. On the left, there's a sidebar with icons for Images, Containers, Volumes, and Compose. The main area is titled 'Containers' with a 'Give feedback' link. It displays 'Container CPU usage' (0.00% / 800%) and 'Container memory usage' (8.29MB / 7.43GB). A search bar and a 'Only show running containers' toggle are also present. A table lists one container:

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
welcome-to-docker bb9038771765	docker/welcome-to-docker:latest	Running	0%	8088:80	33 seconds ago	[...]

At the bottom, status indicators show RAM 2.23 GB, CPU 0.50%, and a user is signed in. A notification bar at the bottom right indicates a 'New version available' with 3 notifications.

**Learning center**

**What is a container?**  
Estimated time: 5 mins

**Containers on Docker Desktop**

The best way to learn about containers is to first see it in action. We have created a welcome container for you.

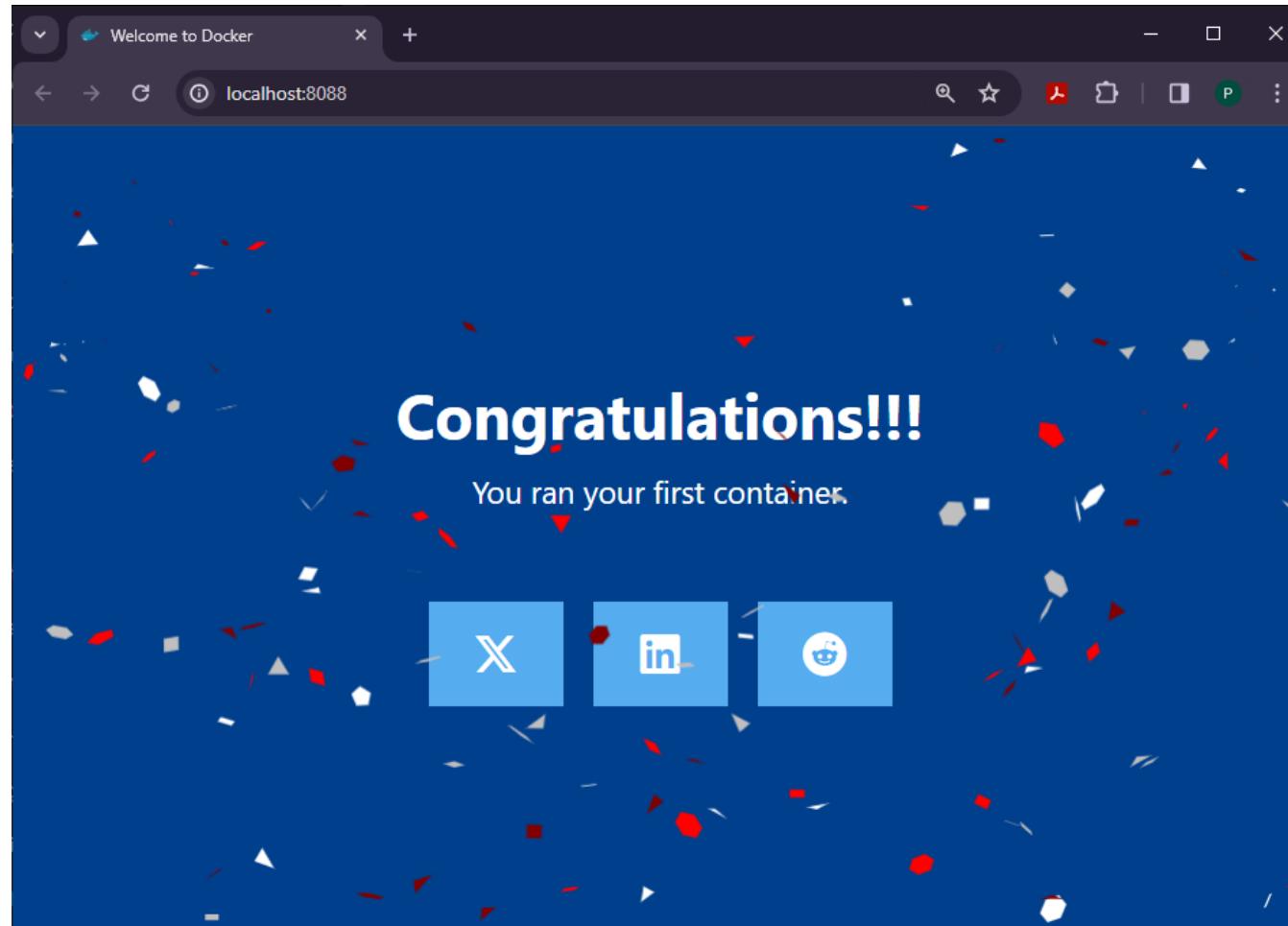
You can check it out in the [Containers tab](#) (`welcome-to-docker`).

[Back](#) [Next](#)

- View the frontend**
- Explore your container**
- Stop your container**
- 5 What's next

Showing 1 item

# What is a Container and how to run it?



# What is a Container and how to run it?

Open VS code > Add folder to workspace  
> git clone <https://github.com/docker/welcome-to-docker>

View the details inside a *Dockerfile*

# What is a Container and how to run it?

## Build your image

- > Open the command prompt at the cloned directory
- > docker build -t welcome-to-docker .
- > docker images

## Run your image

- > docker run -d -p 8088:3000 --name welcome-to-docker welcome-to-docker
- > docker ps

# What is a Container and how to run it?

## Publish your image

- 1) Log in to Docker Hub with Docker Desktop
- 2) Change image name to be  
`<Username>/<Image_name>:<tag>`  
`> docker tag welcome-to-docker USERNAME/welcome-to-docker:v1`  
`> docker push`
- 3) Go to Docker Hub and find your image

# What is a Container and how to run it?

## Remove images

- > docker images
- > docker rmi <Image\_name>

## Pull image from Docker Hub

- > docker pull USERNAME/welcome-to-docker:v1
- > docker images

# Docker Command Summary

Docker CLI Cheat sheet: <https://www.docker.com/resources/cli-cheat-sheet/>

## **Build an Image from a Dockerfile**

docker build -t <image\_name>

## **List local images**

docker images

## **Run a container with and publish a container's port(s) to the host.**

docker run -p  
<host\_port>:<container\_port>  
<image\_name>

## **To list currently running containers:**

docker ps

## **List all docker containers (running and stopped):**

docker ps --all

## **Login into Docker**

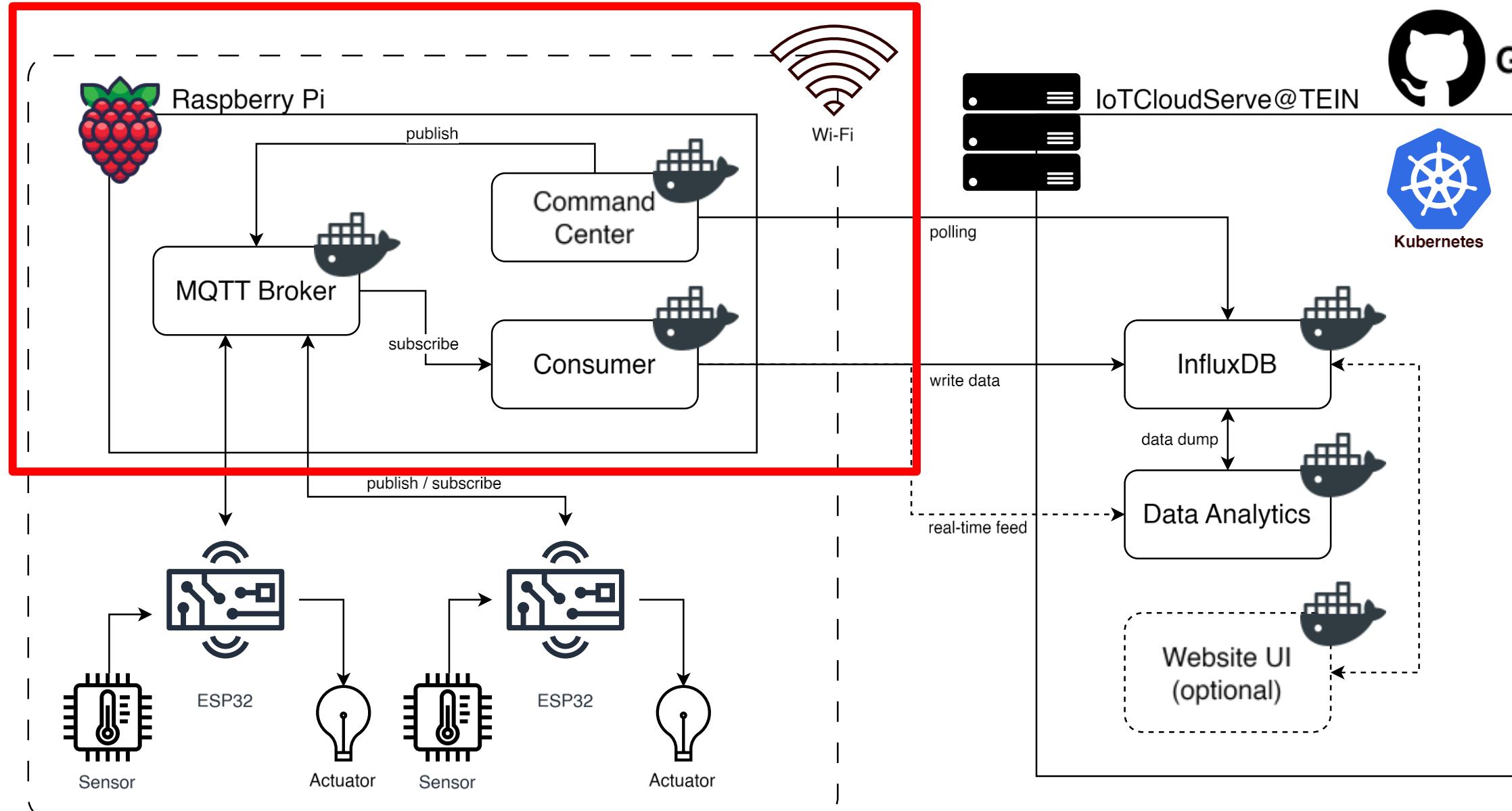
docker login -u <username>

## **Publish an image to Docker Hub**

docker push <username>/<image\_name>

## **Pull an image from a Docker Hub**

docker pull <image\_name>



# Establishing LAN Connection

\*We need **LAN** for using SSH and VNC Viewer

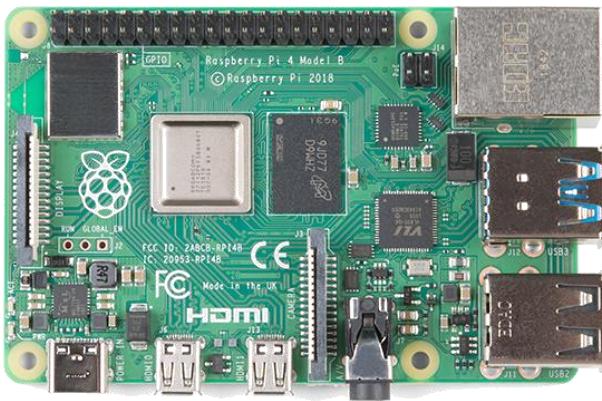
You'll have several options for connecting network for Raspberry Pi

1. Connect Pi and Laptop with the same Wi-Fi
  - A. Use monitor to connect Wi-Fi on Raspberry Pi
  - B. In today's session, connect Raspberry Pi to your mobile hotspot so that Raspberry Pi remembers the SSID and Password.
2. Use LAN Cable to connect with the Ethernet port

# Connect Wireless LAN Network



[This Photo](#) by Unknown Author is licensed under  
[CC BY-NC](#)



**Group 1-3**

TP-LINK\_E0BE  
14061653

**Group 4-6**

TP-Link\_735E  
15412904

**Group 7-9**

TP-Link\_8A28  
42695318

# Connect Raspberry Pi via SSH

\*We need **LAN** for connecting via SSH

\*\* Change the number according to your group number

> ssh pi-group-1@raspberrypi1.local

*Note: pi-group-1 is your **username** of pi and raspberrypi1 is your **hostname***

Enter the password: iotfun2023

Now you can access Raspberry Pi using the command line

> sudo raspi-config

Go to Interface Options > Enable VNC

# Connect Raspberry Pi via SSH

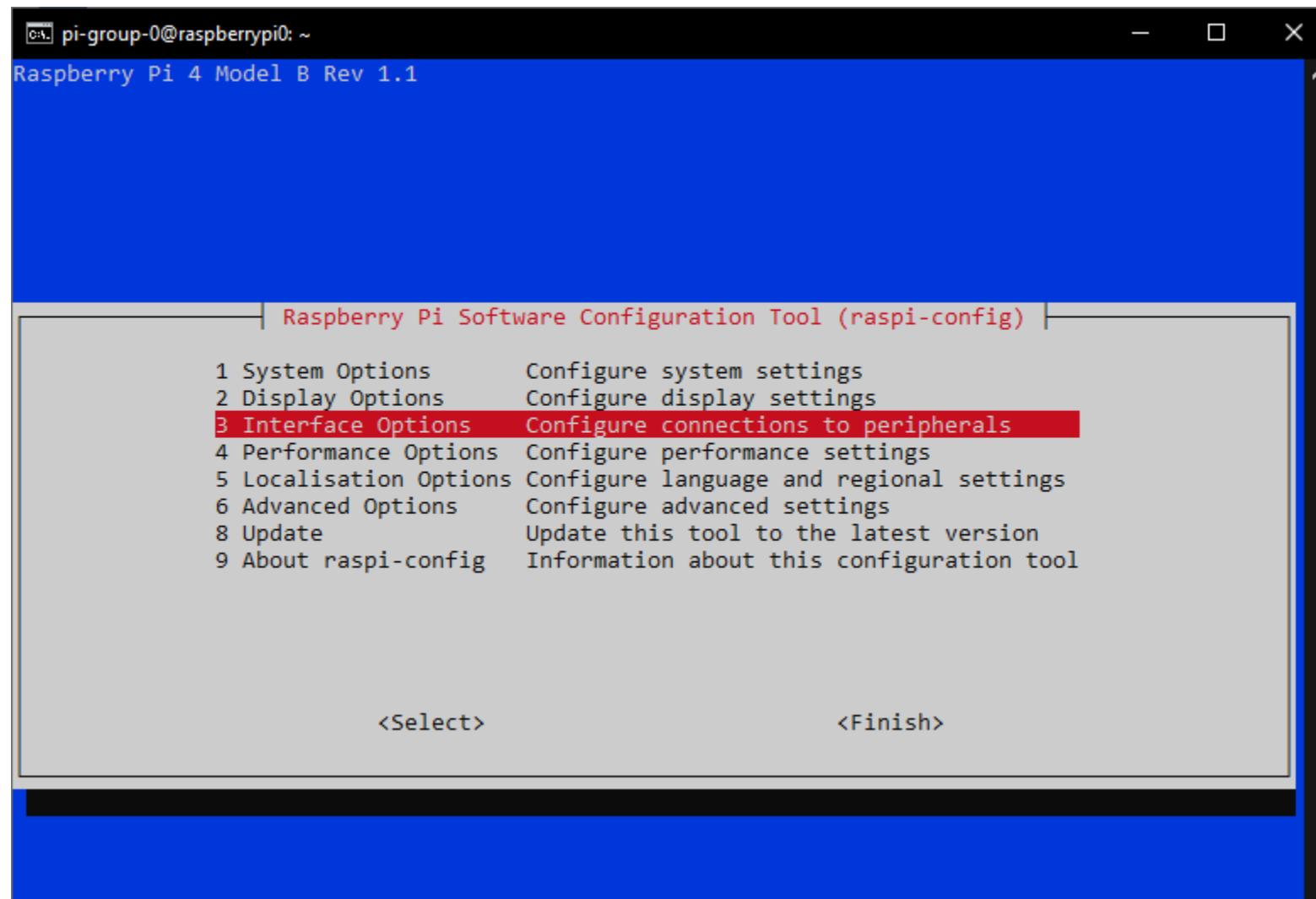
```
pi-group-0@raspberrypi0: ~
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\66891>ssh pi-group-0@raspberrypi0.local
The authenticity of host 'raspberrypi0.local (2001:44c8:4531:ddaa:82df:e376:e38c:4929)' can't b
e established.
ECDSA key fingerprint is SHA256:3UoT1/UXpZM+QrejwK8ABbi0zmiMFqUWynuKYyk+INY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'raspberrypi0.local,2001:44c8:4531:ddaa:82df:e376:e38c:4929' (ECDSA)
to the list of known hosts.
pi-group-0@raspberrypi0.local's password:
Linux raspberrypi0 6.6.20+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.20-1+rpt1 (2024-03-07) aarch64

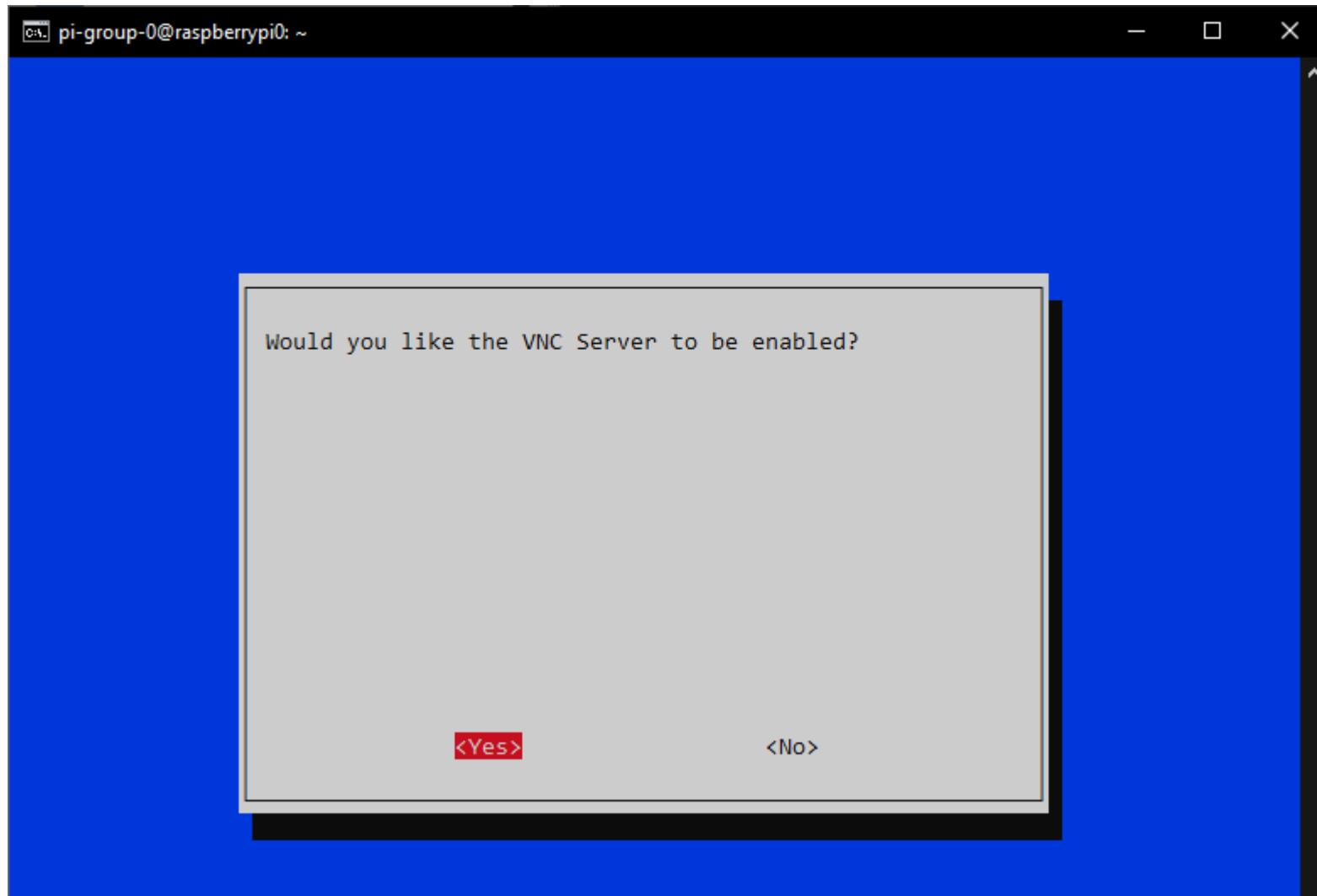
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 15 22:12:59 2024
pi-group-0@raspberrypi0:~ $ sudo raspi-config
Created symlink /etc/systemd/system/multi-user.target.wants/wayvnc.service → /lib/systemd/syste
m/wayvnc.service.
pi-group-0@raspberrypi0:~ $
```

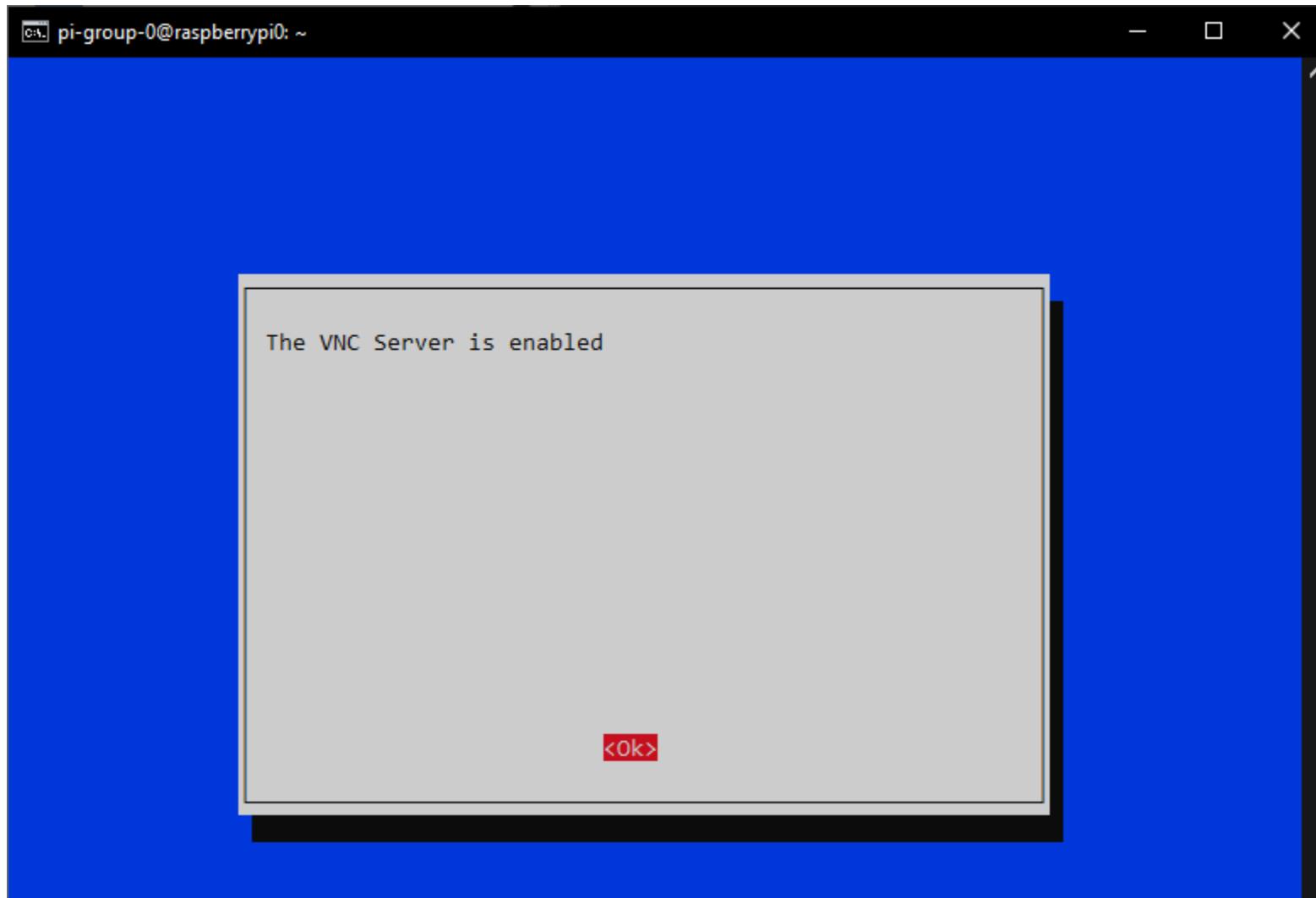
# Connect Raspberry Pi via SSH



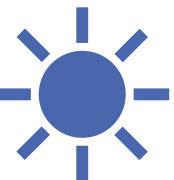
# Connect Raspberry Pi via SSH



# Connect Raspberry Pi via SSH



# Download VNC Viewer



## Download RealVNC® Viewer

Download RealVNC® Viewer to the device you want to control from.

For the best experience, install RealVNC Viewer and RealVNC Server  
together using the [RealVNC Connect Setup app](#).

Desktop

Mobile



Windows



macOS



Linux

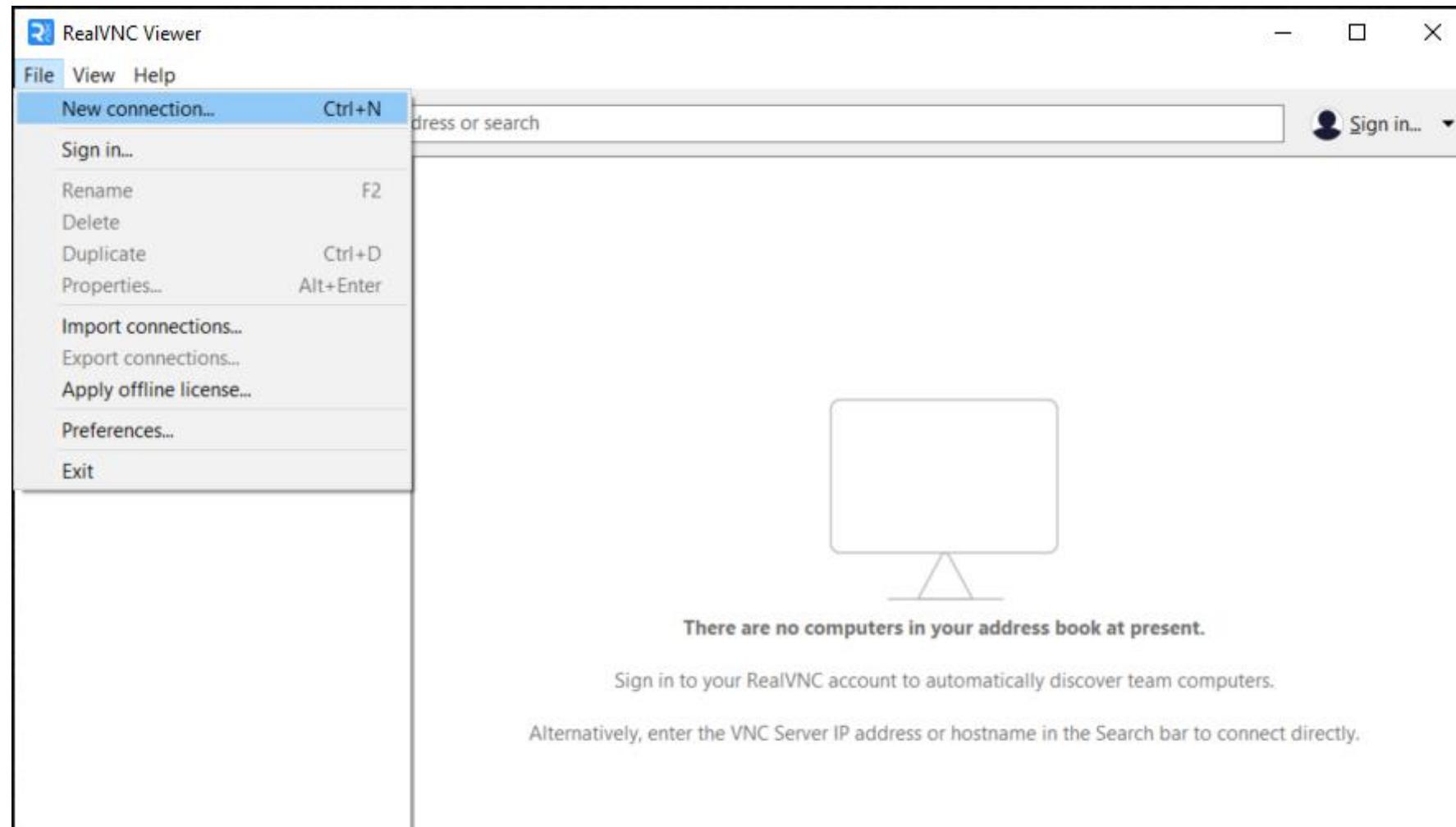


Raspberry Pi

\*We need LAN for using VNC Viewer

<https://www.realvnc.com/en/connect/download/viewer/>

# VNC Viewer

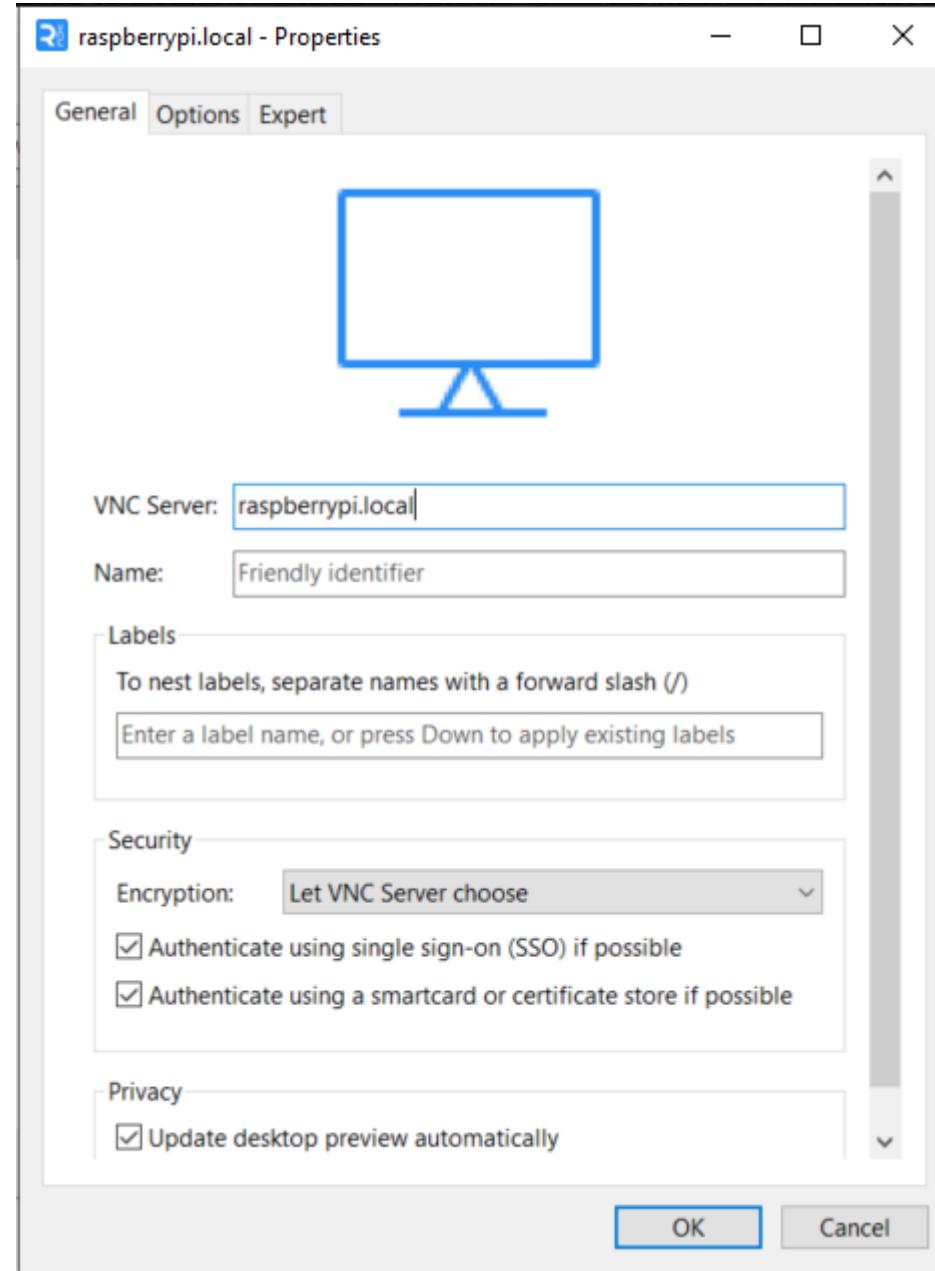


# VNC Viewer

## VNC Server:

- [Hostname].local
- or IP Address

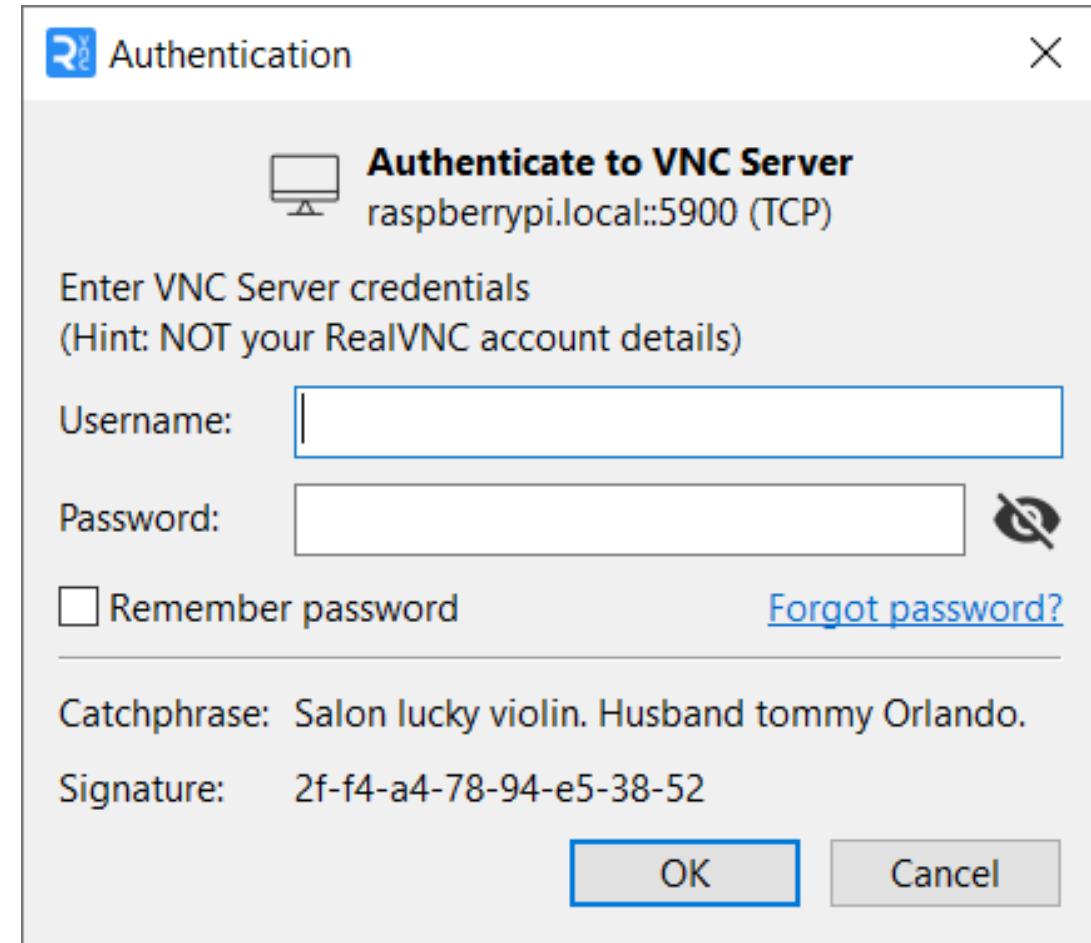
*Note: pi-group-1 is your **username** and raspberrypi1 is your **hostname***



# VNC Viewer

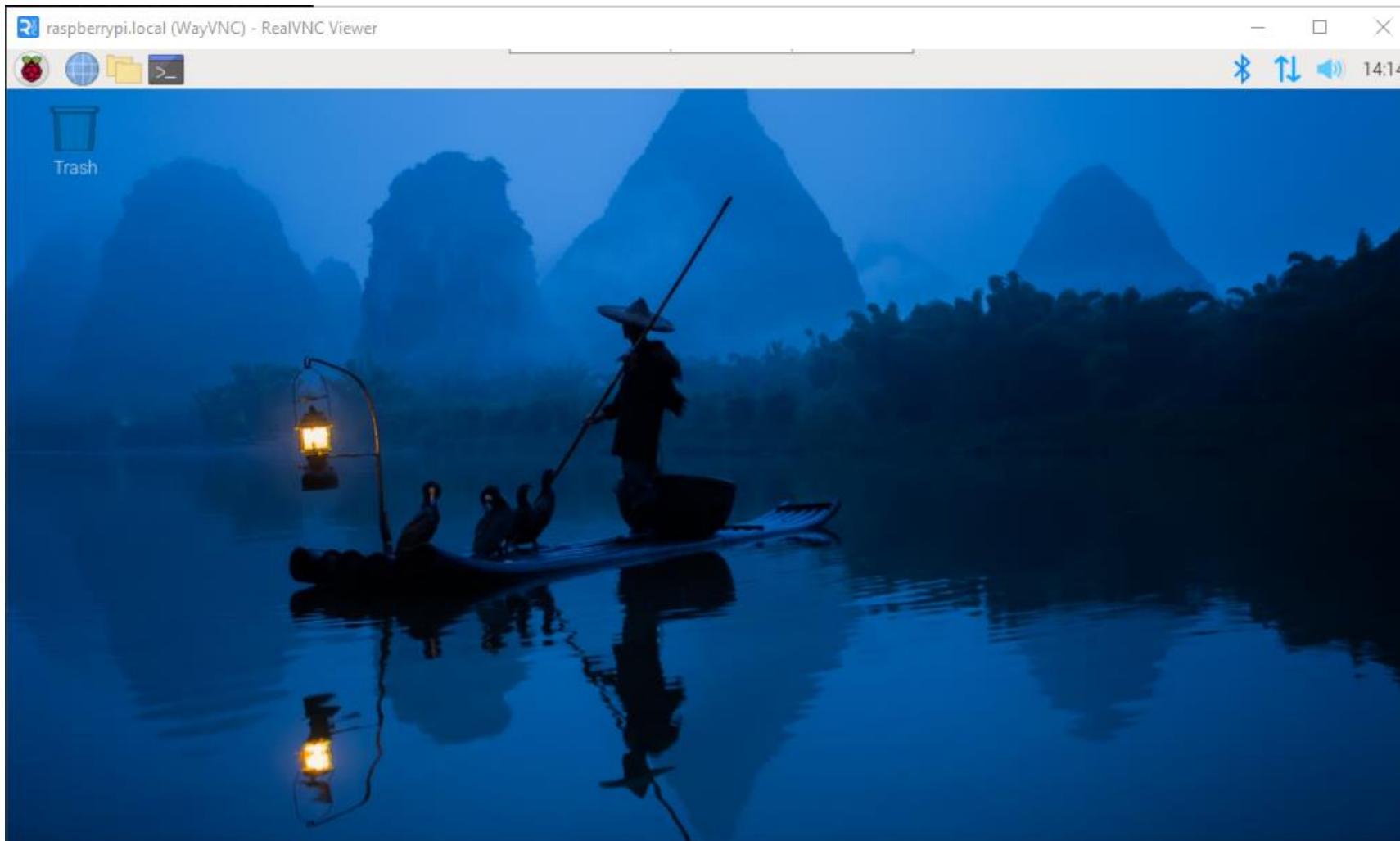
**Username:** [Pi username]  
**Password:** iotfun2023

*Note: pi-group-1 is your **username**  
and raspberrypi1 is your **hostname***

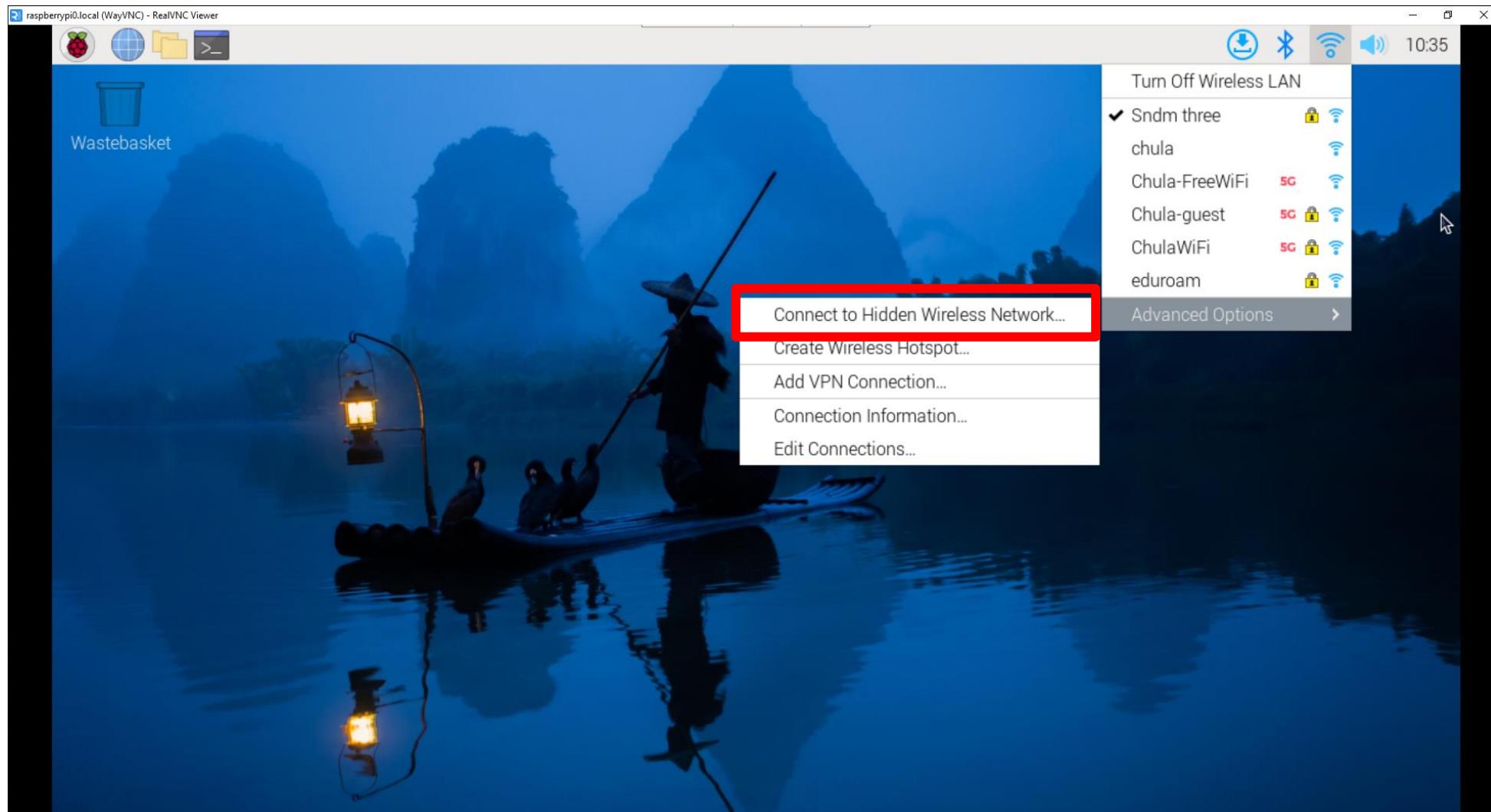


# VNC Viewer

You can now access to Pi screen



# Connect to your Mobile Hotspot



# Connect to your Mobile Hotspot



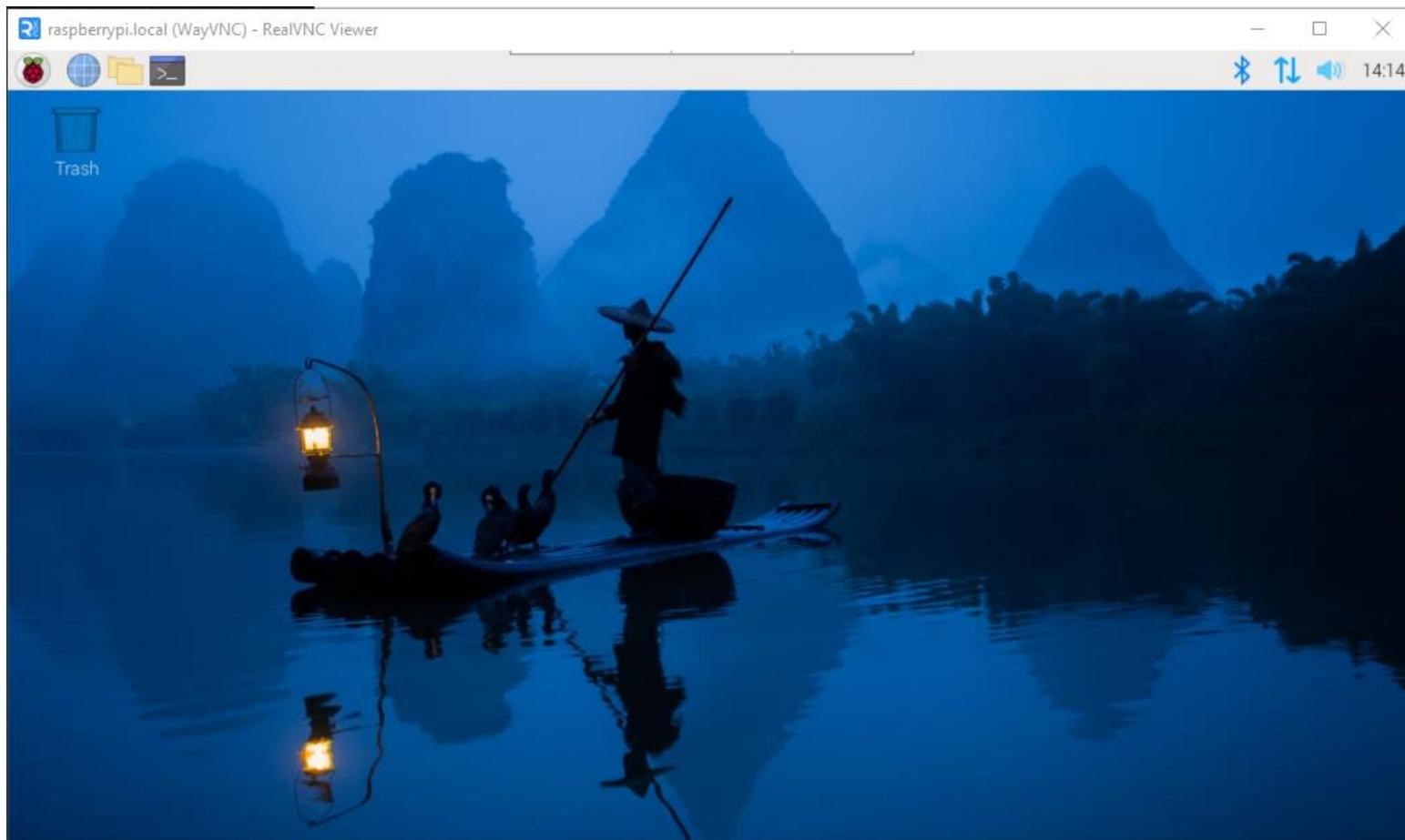
This Photo by Unknown Author is  
licensed under [CC BY-SA-NC](#)

The image shows a 'Hidden Wi-Fi network' connection dialog. At the top, it says 'Connect to Hidden Wi-Fi Network'. Below that, there's a radio tower icon. The main area has a heading 'Hidden Wi-Fi network' and a sub-instruction 'Enter the name and security details of the hidden Wi-Fi network you wish to connect to.' On the right, there's a photo of a Raspberry Pi with a Wi-Fi signal icon next to it, and a note: 'This Photo by Unknown Author is licensed under [CC BY-SA](#)'. The dialog contains fields for 'Connection' (set to 'New...'), 'Network name' (containing 'Your SSID'), 'Wi-Fi security' (set to 'WPA & WPA2 Personal'), and 'Password' (containing '.....'). A red box highlights the 'Network name', 'Wi-Fi security', and 'Password' fields. Below these fields is a checkbox labeled 'Show password'. At the bottom are 'Cancel' and 'Connect' buttons.



# VNC Viewer

You can now access to Pi screen  
with your hotspot



# VNC Viewer

You can switch back to the network from the given Wi-Fi

Please ensure to disable your hotspot, allowing the Raspberry Pi to attempt an automatic reconnection to the default Wi-Fi network.

**Note:** For those who don't need to connect to the Raspberry Pi, please use **ChulaWiFi** because the router capacity is limited.

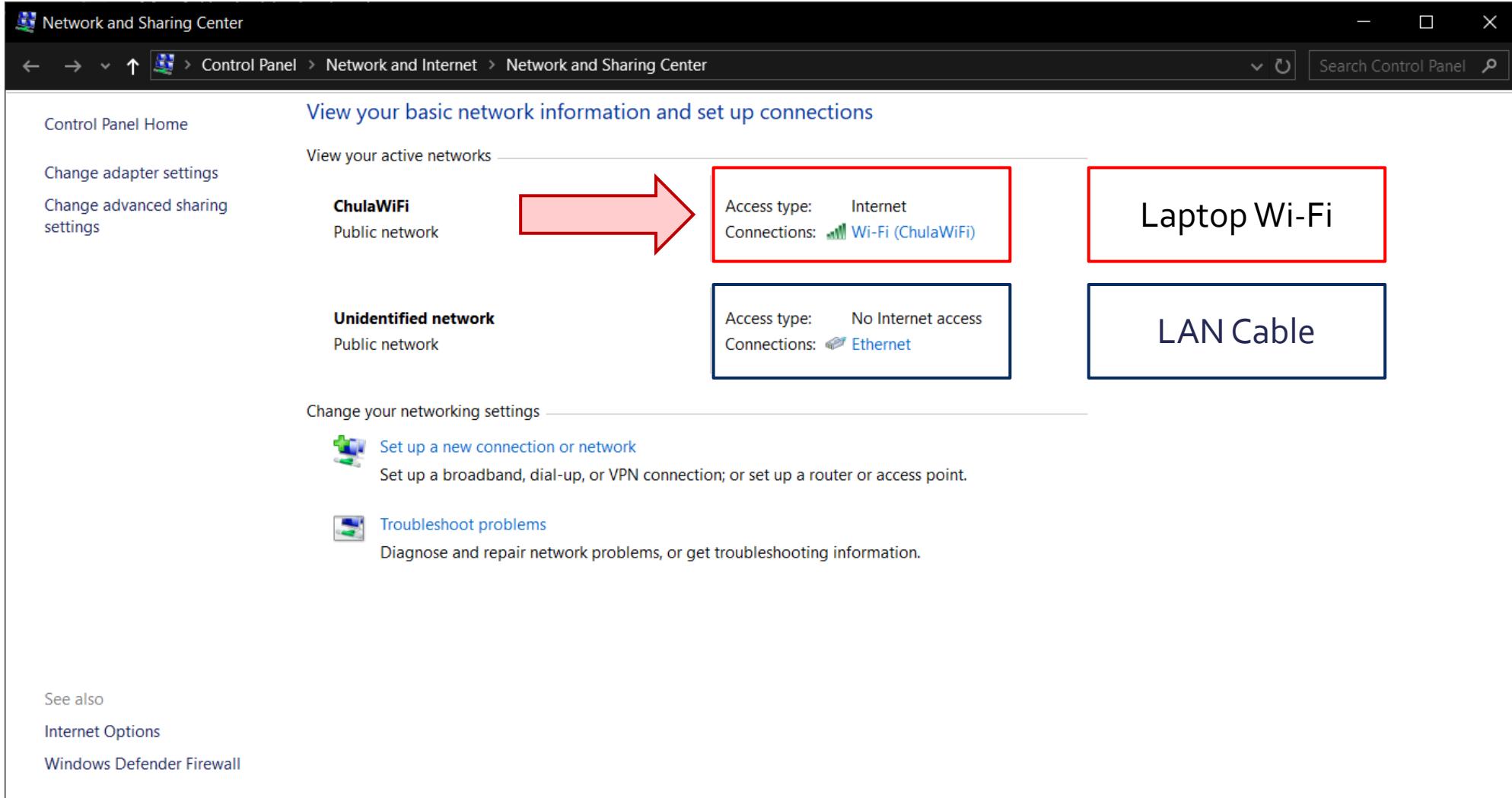
**Group 1-3**  
TP-LINK\_E0BE

**Group 4-6**  
TP-Link\_735E

**Group 7-9**  
TP-Link\_8A28

# (Optional) Setup Network for Raspberry Pi

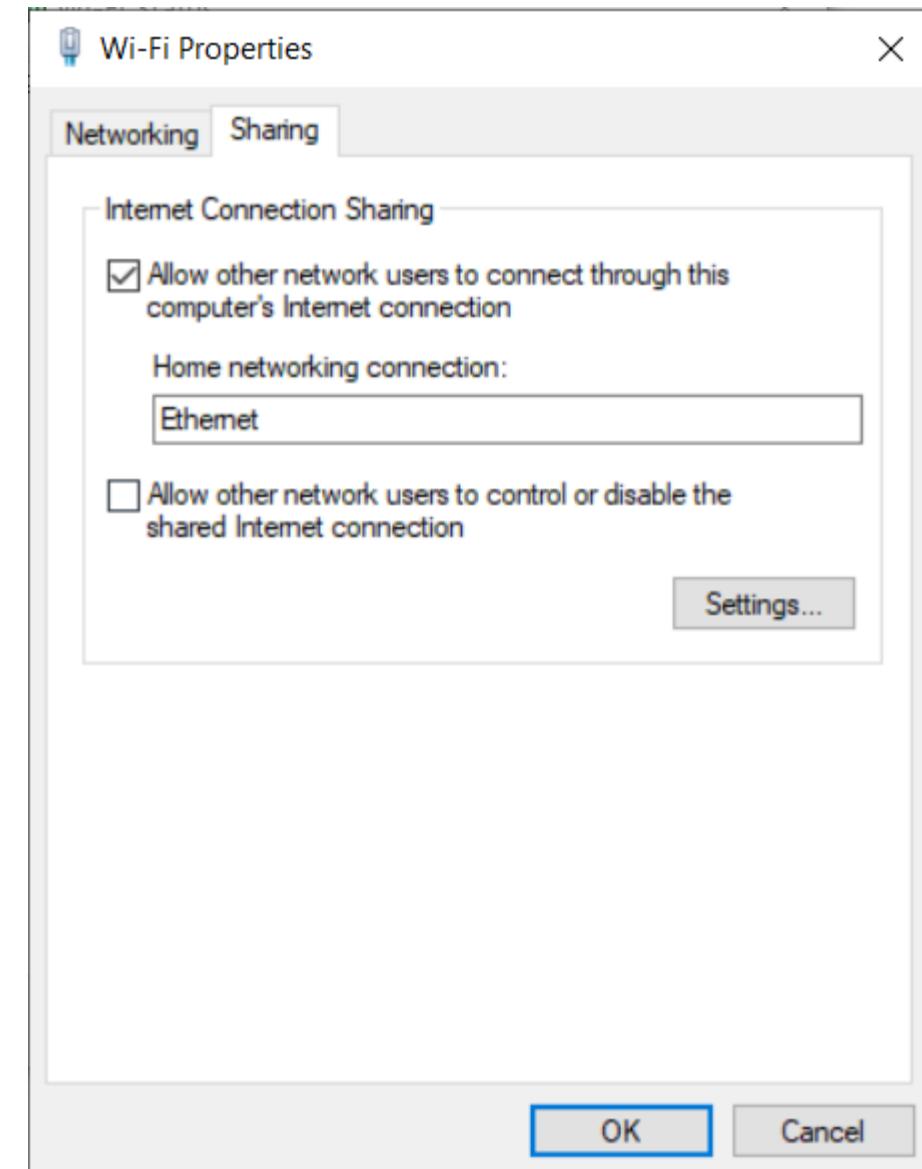
## For those who connect LAN using LAN Cable only!



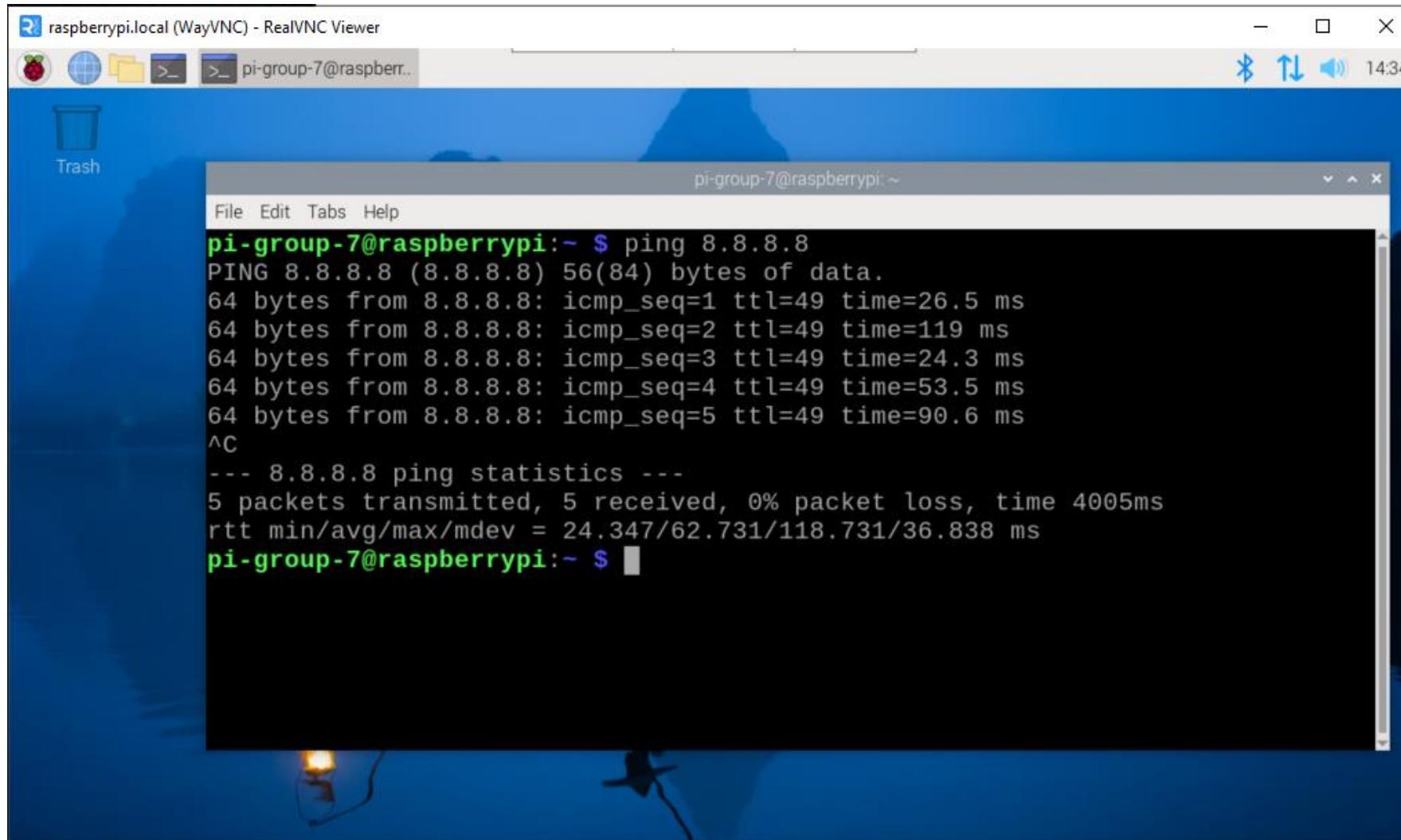
# (Optional) Setup Network for Raspberry Pi

For those who connect LAN using LAN Cable only!

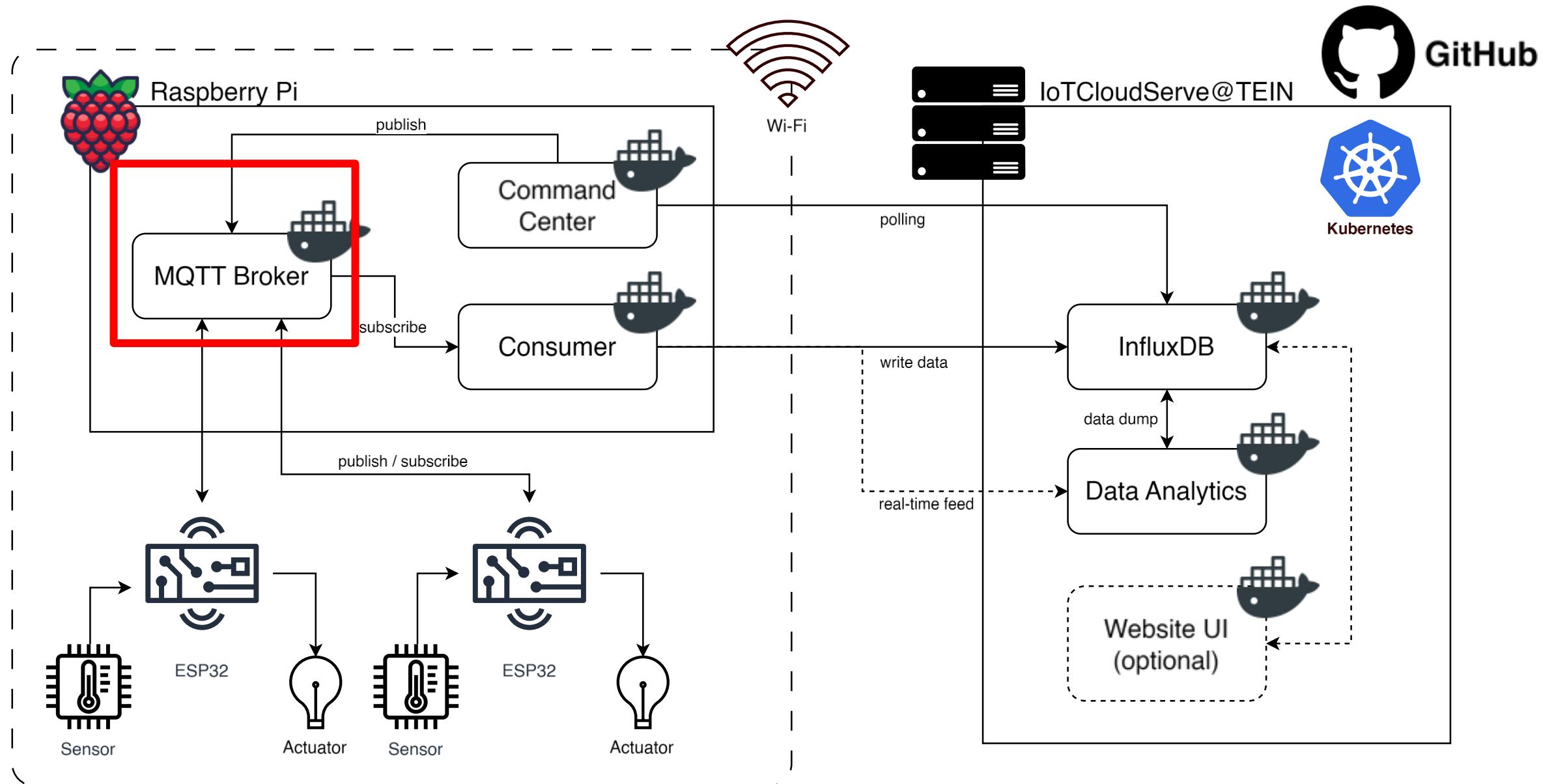
You allow an Ethernet Port to connect through Laptop Wi-Fi



# Setup Network for Raspberry Pi



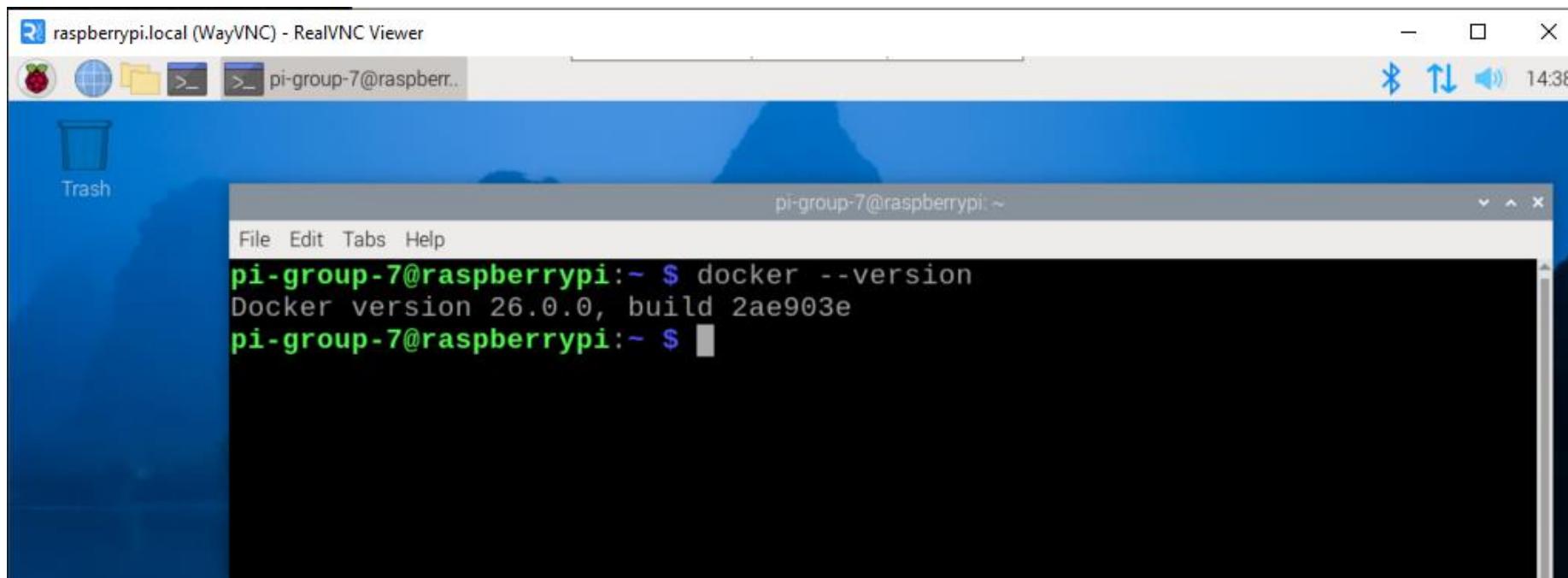
Pi now can  
connect to  
the internet



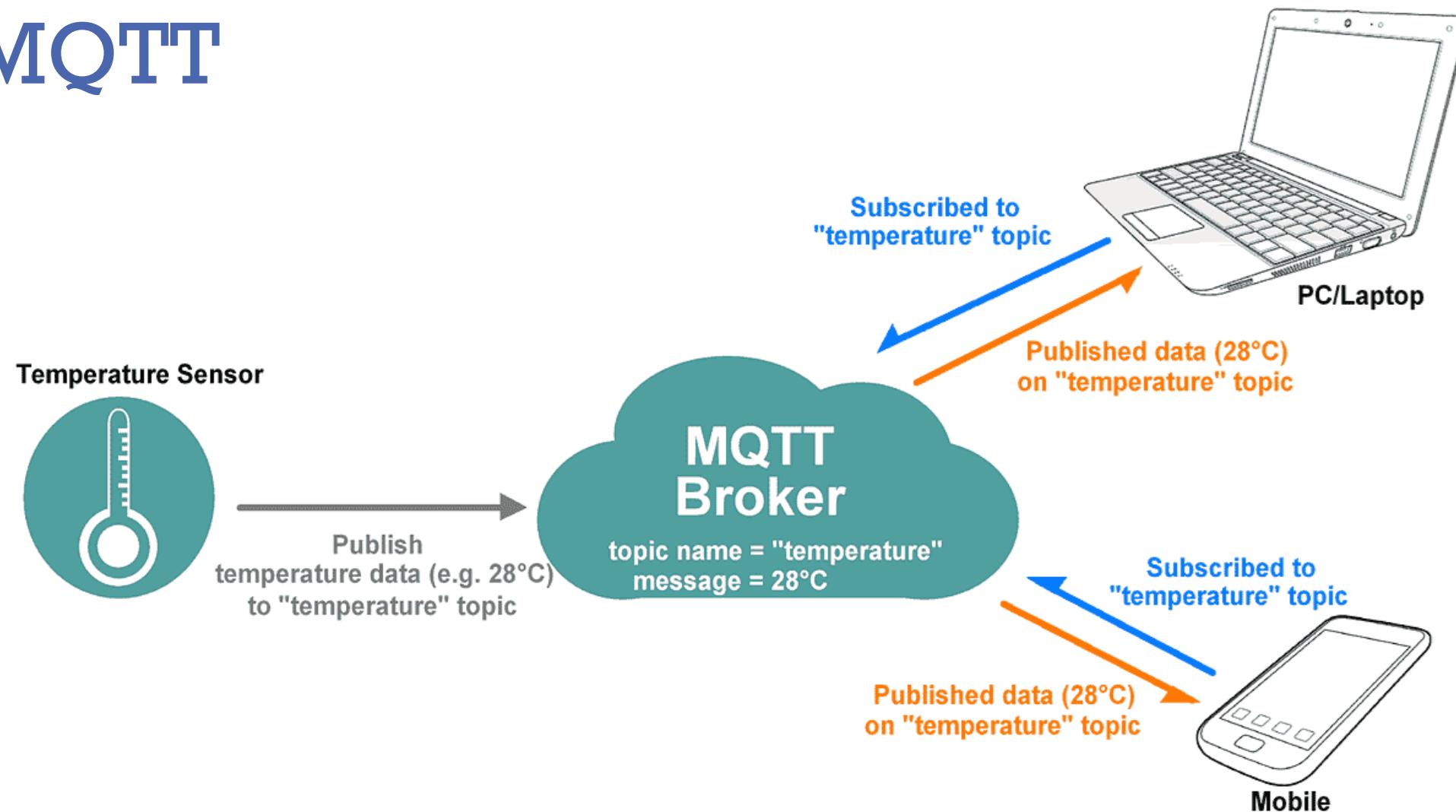
# Install Docker on Raspberry Pi

<https://raspberrytips.com/docker-on-raspberry-pi/>

```
curl -sSL https://get.docker.com | sh  
sudo usermod -aG docker $USER
```



# MQTT



<https://medium.com/@jaydev.dave93/what-is-mqtt-protocol-c6aocafffa8c>

# Deploy MQTT Broker

<https://www.emqx.com/en>

```
> sudo docker run -p 1883:1883  
-p 18083:18083 emqx
```

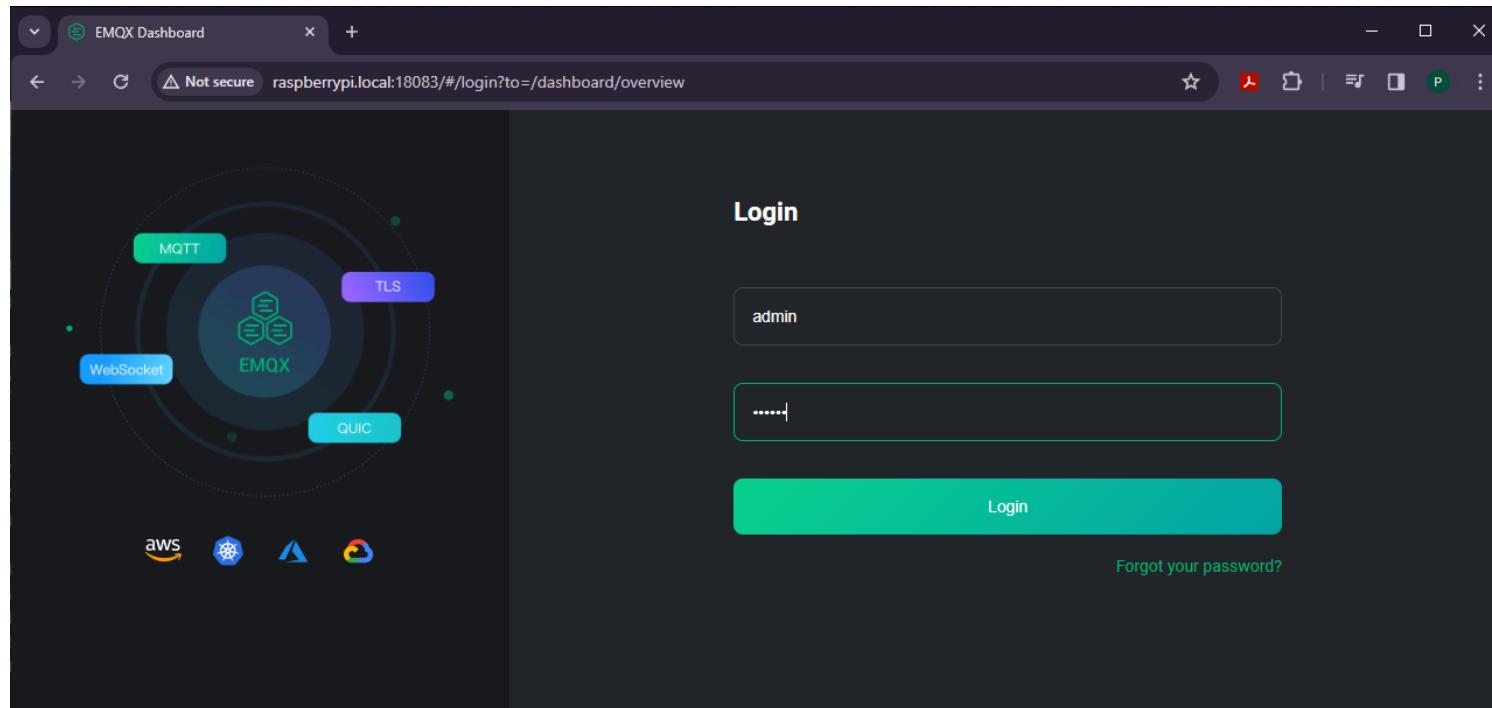
We will use port 1883 for the MQTT connection and port 18083 for the MQTT Broker management

# Deploy MQTT Broker

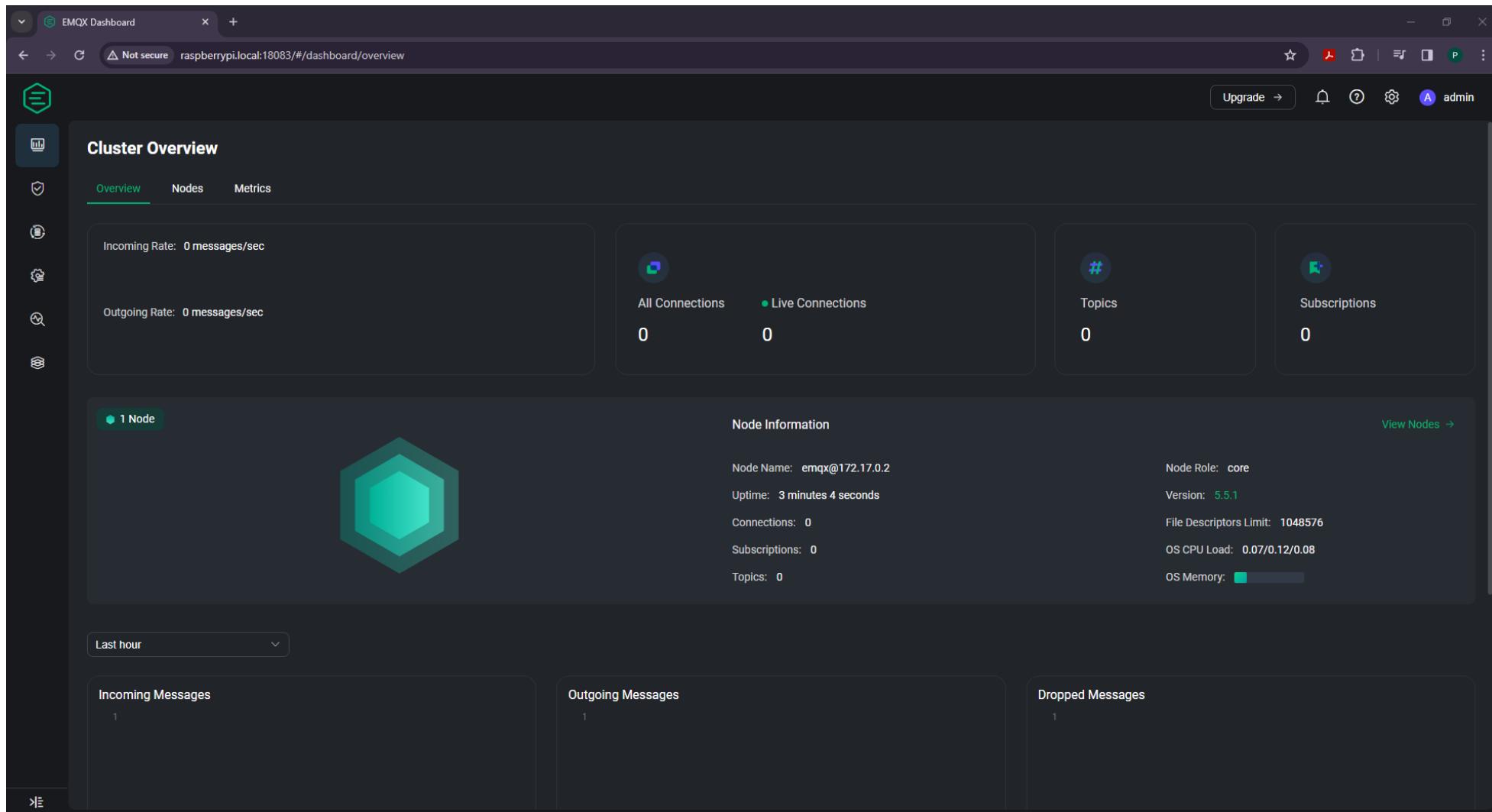
Go to “raspberrypi**1**.local:18083” and login

Username: admin

Password: public



# Deploy MQTT Broker



# Test MQTT Broker using Postman

To verify the functionality of your broker, you can use Postman for testing.

1. Connect to “raspberrypi**1**.local” at port 1883
2. Subscribe to the topic “@msg/data”
3. Try publishing a message on that topic and see the result

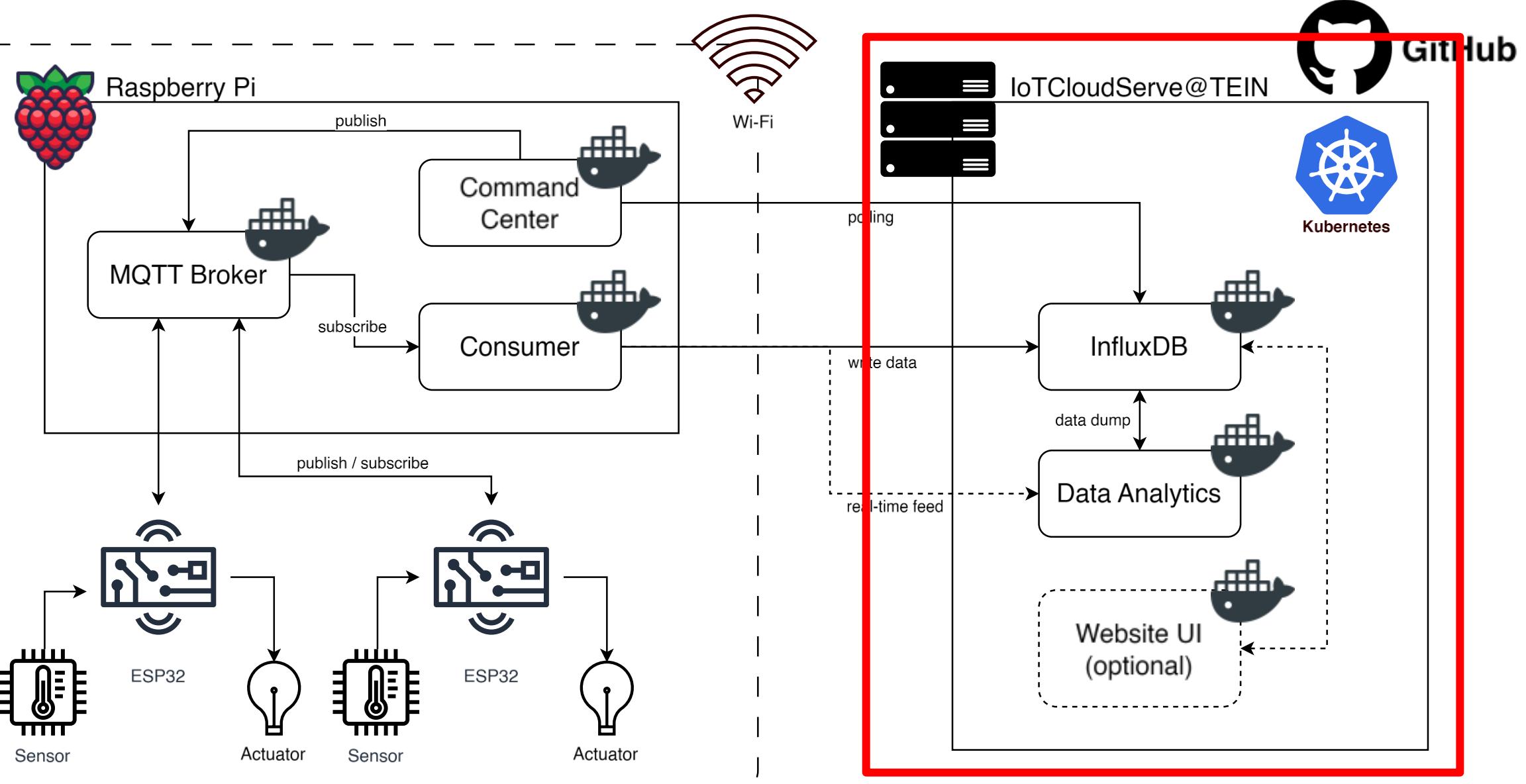
# Test MQTT Broker using Postman

The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, API Network, a search bar, and various tool buttons like Invite, Settings, and Upgrade. The main workspace is titled "My Workspace". On the left sidebar, there are sections for Collections, Environments, and History. Under Collections, the "iot-cloud-workshop-mqtt" collection is expanded, showing sub-items: GET Status, GET Get Data, POST Post Data, and a sub-collection "iot-cloud-workshop-mqtt" which contains "RaspberryPi". The "RaspberryPi" item is currently selected and highlighted with a red border. The central workspace displays the "RaspberryPi" endpoint details. It shows the URL "iot-cloud-workshop-mqtt / RaspberryPi" and the version "V3" connected to "raspberrypi0.local". Below this, the "Request" and "Response" tabs are visible. The "Response" tab shows a message list with two entries: "Subscribed to @msg/data" at 15:30:54 and "Connected to broker" at 15:30:53. At the bottom of the interface, there are status indicators for "Online" (green), "Find and replace" (blue), "Console" (grey), and several utility buttons: Postbot, Runner, Start Proxy, Cookies, Trash, and Help.

# Test MQTT Broker using Postman

The screenshot shows the Postman application interface with the following details:

- Header:** Home, Workspaces, API Network, Search Postman, Invite, Settings, Notifications, Upgrade.
- Left Sidebar (My Workspace):**
  - Collections: iot-cloud-workshop (selected), iot-cloud-workshop-mqtt, RaspberryPi.
  - Environments: raspberrypi0.local (selected).
  - History: iot-cloud-workshop-predict, Magel, MHE Project, MQTT-NETPIE.
- Toolbar:** GET Get Data, POST Post E, GET Status, Raspbe, POST predi, GET status, iot-cloud, +, No environment.
- Request Section:** Topic: iot-cloud-workshop-mqtt / RaspberryPi, Version: V3, Environment: raspberrypi0.local, Save, Disconnect, Info.
- Response Section:** Subscribed to 1 topic, Connected.
- Messages Tab:** Search, All Messages, Clear Messages.
  - Down arrow: @msg/data {"data":42} 15:31:45
  - Up arrow: @msg/data {"data":42} 15:31:45
  - Info icon: Subscribed to @msg/data 15:30:54
  - Checkmark icon: Connected to broker 15:30:53
- Bottom Navigation:** Online, Find and replace, Console, Postbot, Runner, Start Proxy, Cookies, Trash, Help.



# Kubernetes

<https://kubernetes.io/docs/concepts/overview/>



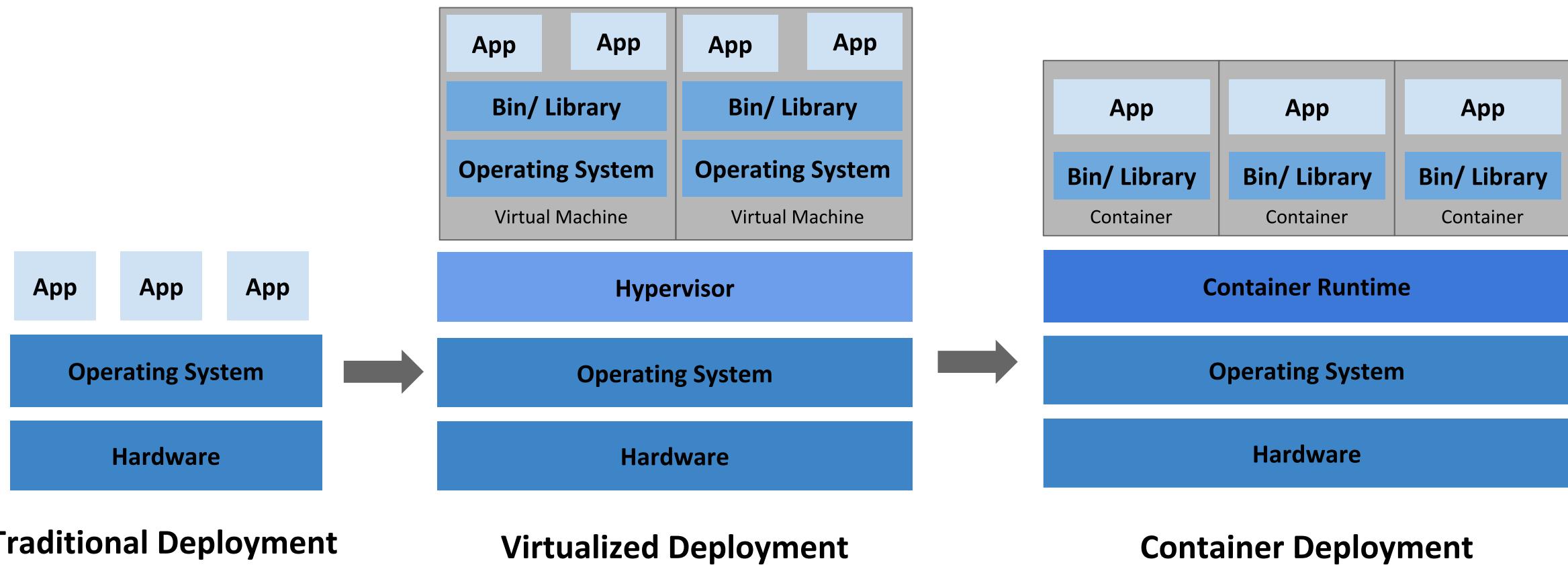
kubernetes

This Photo by Unknown Author is licensed under [CC BY](#)

Kubernetes is a portable, extensible, open-source **platform for managing containerized workloads and services**, that facilitates both declarative configuration and automation.

# Kubernetes

<https://kubernetes.io/docs/concepts/overview/>



# Why you need Kubernetes?

<https://kubernetes.io/docs/concepts/overview/>

- **Service discovery and load balancing**
  - expose a container using the DNS name or using their IP address
  - distribute the network traffic so that the deployment is stable
- **Automated rollouts and rollbacks**
  - create new containers for your deployment, remove existing containers and adopt all their resources to the new container

# Why you need Kubernetes?

<https://kubernetes.io/docs/concepts/overview/>

- **Self-healing**
  - restarts containers that fail
  - replaces containers
  - kills containers that don't respond to the health check
- **Secret and configuration management**
  - store and manage sensitive information, such as passwords or keys
  - You can deploy and update secrets and application configuration without rebuilding your container images

# Rancher

<https://ranchermanager.docs.rancher.com/v2.0-v2.4/getting-started/introduction/overview>



[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

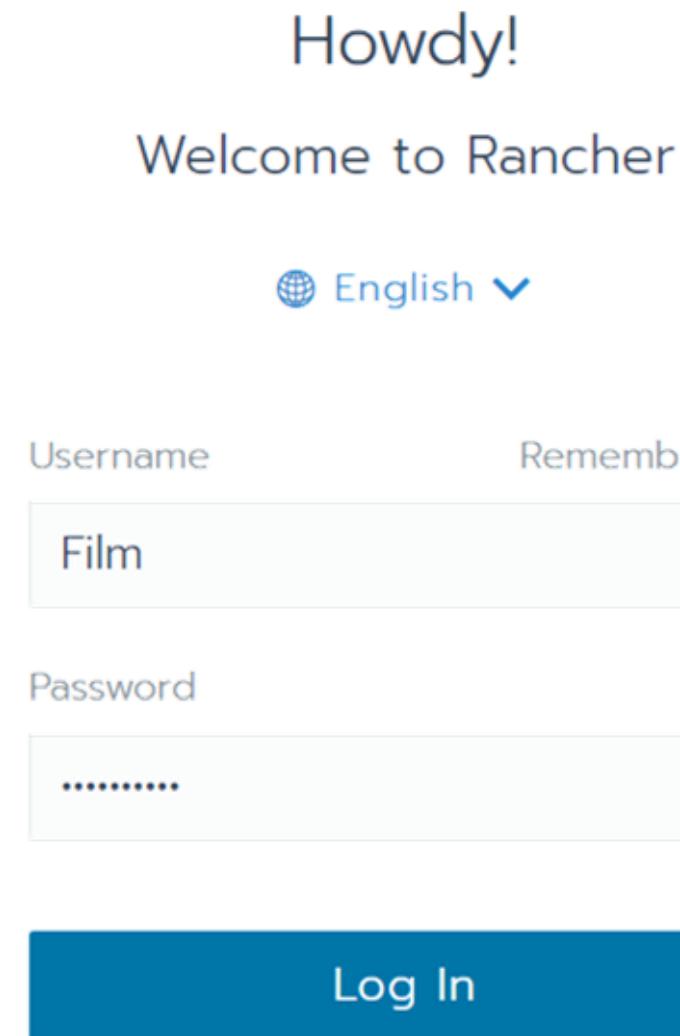
Rancher is a container management platform built for organizations that deploy containers in production. Rancher makes it easy to run Kubernetes everywhere

# Rancher

Go to <https://www.iotcloudserve.net/login>

Username: [Username]

Password: iotfun2023



The image shows a screenshot of the Rancher login interface. At the top right, it says "Howdy!" and "Welcome to Rancher". Below that is a language selection dropdown showing "English". The main area has two input fields: "Username" with "Film" typed in, and "Password" with several dots indicating the password. A "Remember" checkbox is also present. At the bottom is a large blue "Log In" button.

Howdy!

Welcome to Rancher

English ▾

Username

Remember

Film

Password

.....

Log In

# Rancher

The screenshot shows the Rancher web application interface. At the top, there is a dark header bar with the Rancher logo, a search bar containing "iotcloudserve.net/g/clusters", and various browser control icons. Below the header is a navigation bar with links for Global, Clusters (which is highlighted in blue), Apps, Users, Settings, Security, and Tools.

The main content area is titled "Clusters". On the right side of this title is a "Add Cluster" button. Below the title is a toolbar with a "Delete" button and a "Search" input field. The main table has columns for State, Cluster Name, Provider, Nodes, CPU, and RAM. A filter for State is set to "Active". The table contains one row for a cluster named "iotcloudserve" which is "Imported" from "v1.20.15+k3s1". It has 0 nodes, 0.3/24 Cores usage, and 0% RAM usage. To the right of this row is a vertical ellipsis button.

State	Cluster Name	Provider	Nodes	CPU	RAM
<input type="checkbox"/> Active	iotcloudserve	Imported v1.20.15+k3s1	0	0.3/24 Cores 1%	0.1/93.7 GiB 0%

At the bottom of the page, there is a footer bar with links for v2.3.11-rc2, Help & Docs, Forums, Slack, File an Issue, English (with a dropdown arrow), and Download CLI (with a dropdown arrow).

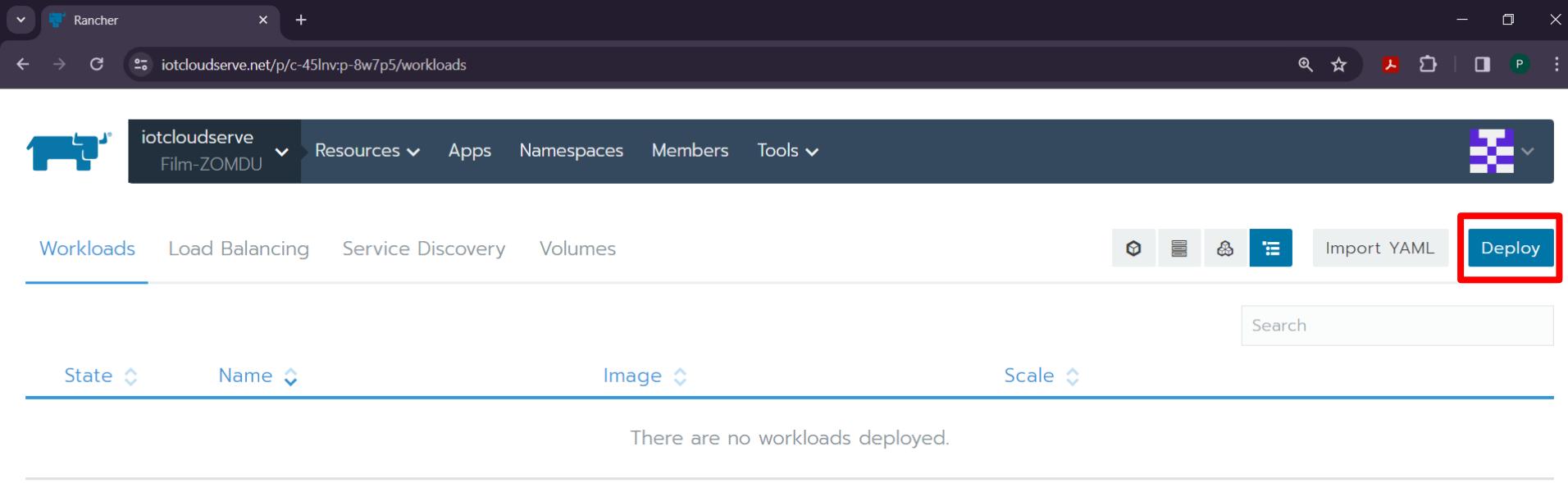
# Rancher

The screenshot shows the Rancher web interface with a dark theme. At the top, there is a navigation bar with tabs for Cluster, Nodes, Storage, Projects/Namespaces (which is highlighted with a red box), Members, and Tools. Below the navigation bar, the main content area is titled "Projects/Namespaces". It features a search bar and filter options for "State" (dropdown), "Namespace Name" (dropdown), and "Created" (dropdown). A table lists a single namespace entry:

Project: Film-ZOMDU	Add Namespace
<input type="checkbox"/> Active film-zomdu	8:24 PM <input type="button" value="More"/>

At the bottom of the page, there is a footer with links for v2.3.11-rc2, Help & Docs, Forums, Slack, File an Issue, English (dropdown), and Download CLI (dropdown).

# Rancher – Deploy InfluxDB



The screenshot shows the Rancher web interface for managing workloads. The browser title bar reads "Rancher" and the URL is "iotcloudserve.net/p/c-45Inv;p-8w7p5/workloads". The top navigation bar includes a logo, the cluster name "iotcloudserve Film-ZOMDU", and links for Resources, Apps, Namespaces, Members, and Tools. On the right side of the top bar is a cluster icon. Below the navigation is a toolbar with icons for Create, Import YAML, and Deploy, where the Deploy button is highlighted with a red box. The main content area has tabs for Workloads, Load Balancing, Service Discovery, and Volumes, with "Workloads" selected. It features filters for State, Name, Image, and Scale, and a search bar. A message at the bottom states "There are no workloads deployed." At the bottom of the page are links for v2.3.11-rc2, Help & Docs, Forums, Slack, File an Issue, English, and Download CLI.

v2.3.11-rc2 Help & Docs Forums Slack File an Issue

English Download CLI

# Rancher – Deploy InfluxDB

Docker image: [https://hub.docker.com/\\_/influxdb](https://hub.docker.com/_/influxdb)

 **influxdb** Docker Official Image • 1B+ • 1.9K  
InfluxDB is the open source time series database built for real-time analytic workloads.

`docker pull influxdb` Copy

[Overview](#) [Tags](#)

**Note:** the description for this image is longer than the Hub length limit of 25000, so has been trimmed. The full description can be found at <https://github.com/docker-library/docs/tree/master/influxdb/README.md>. See also [docker/hub-feedback#238](#) and [docker/roadmap#475](#).

## Quick reference

- Maintained by:  
[InfluxData](#)
- Where to get help:  
[the Docker Community Slack](#), [Server Fault](#), [Unix & Linux](#), or [Stack Overflow](#)

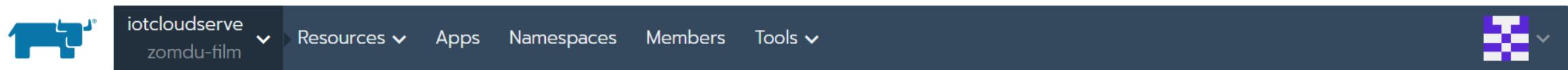
## Recent Tags

[alpine](#) [2.7.5-alpine](#) [2.7-alpine](#) [2-alpine](#)  
[1.9.13-meta-alpine](#) [1.9.13-data-alpine](#) [1.9-meta-alpine](#)  
[1.9-data-alpine](#) [1.8.10-alpine](#) [1.8-alpine](#)

## About Official Images

Docker Official Images are a curated set of Docker open source and drop-in solution repositories.

# Rancher – Deploy InfluxDB



## Edit Workload

Name

film-influxdb

Add a Description

Image name from Docker Hub (you can  
also use version tag e.g. influxdb:2.7.5)

Workload Type

Scalable deployment of 1 pod

Docker Image \*

influxdb



Namespace

film-zomdu-test

Port Mapping

Port Name

influxdb

Publish the container port \*

8086

Protocol

TCP

As a

NodePort (On every node)

The port on the host machine where  
the incoming traffic from outside  
the container will be directed to

+ Add Port

The port on which the containerized  
application is listening inside the container

On listening port \*

Random



# Rancher – Deploy InfluxDB

Each user is allocated 5 CPUs and 5 GB of memory. The reservation and limit for memory and CPU of each workload can be adjusted based on your application's requirements, but the total allocation must not exceed this quota.

Memory Reservation

128 MiB

Memory Limit

No Limit  
 Limit to 2048 MiB

CPU Reservation

100 milli CPUs

CPU Limit

No Limit  
 Limit to 1000 milli CPUs

NVIDIA GPU Reservation

e.g. 1 GPUs

# Rancher – Deploy InfluxDB

The screenshot shows the Rancher web interface for managing Kubernetes workloads. At the top, the navigation bar includes the Rancher logo, the current namespace ('iotcloudserve Film-ZOMDU'), and links for Resources, Apps, Namespaces, Members, and Tools. A cluster icon with a purple checkmark is also present.

The main area displays a workload named 'film-influxdb'. The top card provides basic details: Namespace: film-zomdu, Image: influxdb:2.7.5, and Workload Type: Deployment. Below this, another card shows Endpoints: n/a, Config Scale: 1 (Ready Scale: 1), and creation details: Created: 8:40 PM, Pod Restarts: 0.

A large button labeled 'Expand All' is located below these cards. The expanded view shows a section for 'Pods' with a sub-section for 'Pods in this workload'. It includes buttons for 'Download YAML' and 'Delete'. The table headers for the pod list are State, Name, Image, and Node. One pod is listed: 'film-influxdb-59cc8db8b5-g56j4' (State: Running, Image: influxdb:2.7.5, Node: 10.42.2.111). A three-dot menu icon is at the end of this row.

# Rancher – Deploy InfluxDB

You need to assign a domain name for the workload to access it.

The screenshot shows the Rancher web interface. At the top, there is a dark header bar with the Rancher logo, the project name "iotcloudserve Film-ZOMDU", and navigation links for Resources, Apps, Namespaces, Members, and Tools. On the far right of the header is a cluster icon. Below the header, the main content area has tabs for Workloads, Load Balancing, Service Discovery, and Volumes. The "Load Balancing" tab is currently selected and highlighted in blue. To the right of these tabs are two buttons: "Import YAML" and a blue "Add Ingress" button, which is enclosed in a red rectangular box. Below the tabs, there is a search bar labeled "Search". At the bottom of the main content area, there are four filter buttons: "State" (with a dropdown arrow), "Name" (with a dropdown arrow), "Targets", and "Created" (with a dropdown arrow). A message "There are no ingress rules defined" is displayed in the center of the main content area.

# Rancher – Deploy InfluxDB

Domain Name list for each group

- iot-group**1**-service1.iotcloudserve.net
- iot-group**1**-service2.iotcloudserve.net
- iot-group**1**-service3.iotcloudserve.net

You will get three names for each group.  
If extra names are required, please contact Film.

# Rancher – Deploy InfluxDB

Hostname for InfluxDB: **iot-group1-service1.iotcloudserve.net**

Add Ingress

Name	Add a Description	Namespace *	Add to a new namespace
film-zomdu-influxdb		film-zomdu	<input type="button" value="▼"/>

Rules

- Automatically generate a `.xip.io` hostname
- Specify a hostname to use
- Use as the default backend

Request Host

film-zomdu-influxdb.iotcloudserve.net

Target Backend  Service

Workload

Path

e.g. /foo

Target

film-influxdb

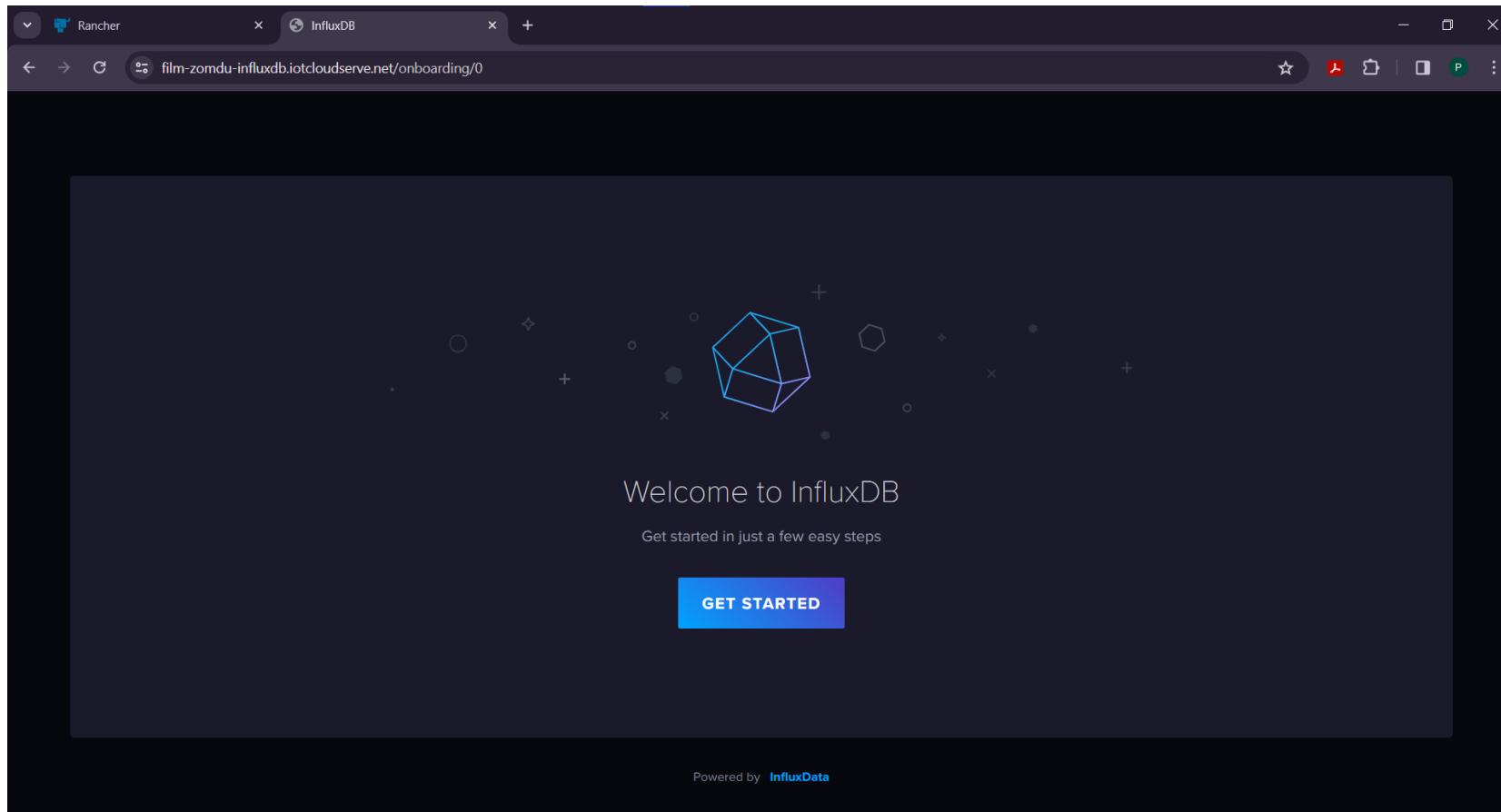
Port \*

8086

-

# InfluxDB

Go to `iot-group1-service1.iotcloudserve.net`  
You should see the InfluxDB UI



# InfluxDB

Configure a user account for InfluxDB. Ensure to securely store the credentials as they will be required for accessing InfluxDB.

INFLUXDB\_USERNAME  
INFLUXDB\_PASSWORD  
INFLUXDB\_ORG  
INFLUXDB\_BUCKET  
INFLUXDB\_TOKEN  
INFLUXDB\_URL

# Rancher – Deploy InfluxDB

If you forget to copy the token, you can generate the token here

The screenshot shows the Rancher interface for managing API tokens. On the left is a vertical sidebar with icons for Sources, Buckets, Telegraf, Scrapers, and API Tokens. The main area has a dark header with "Load Data" and tabs for SOURCES, BUCKETS, TELEGRAF, SCRAPERS, and API TOKENS (which is underlined). Below the header are search and sort filters: "Filter Tokens..." and "Sort by Description (A → Z)". A blue button labeled "+ GENERATE API TOKEN" is visible. The main content area displays a single token entry for "film-zomdu's Token". The entry includes details: Created at: 2024-04-02 21:23:09, Owner: film-zomdu, Last Modified: 3 minutes ago. To the right of the entry are two small icons.

Column 1	Column 2
Created at: 2024-04-02 21:23:09	Owner: film-zomdu
Last Modified: 3 minutes ago	

# Rancher – Deploy InfluxDB

Return to Rancher and store the credentials for InfluxDB in the "secrets" feature for secure access management.

The screenshot shows the Rancher web interface. At the top, there is a dark header bar with the Rancher logo, the cluster name "iotcloudserve Film-ZOMDU", and navigation links for Resources, Apps, Namespaces, Members, and Tools. On the far right of the header is a save icon. Below the header is a main content area with a title "Workloads" and tabs for Secrets, Certificates, and Pipelines. The "Secrets" tab is currently selected and highlighted in blue. The main table has columns for State, Name, Secrets, Namespace, Keys, and Created. A search bar is located at the bottom right of the table area. A message at the bottom center states "There are no secrets defined".

# Rancher – Deploy InfluxDB

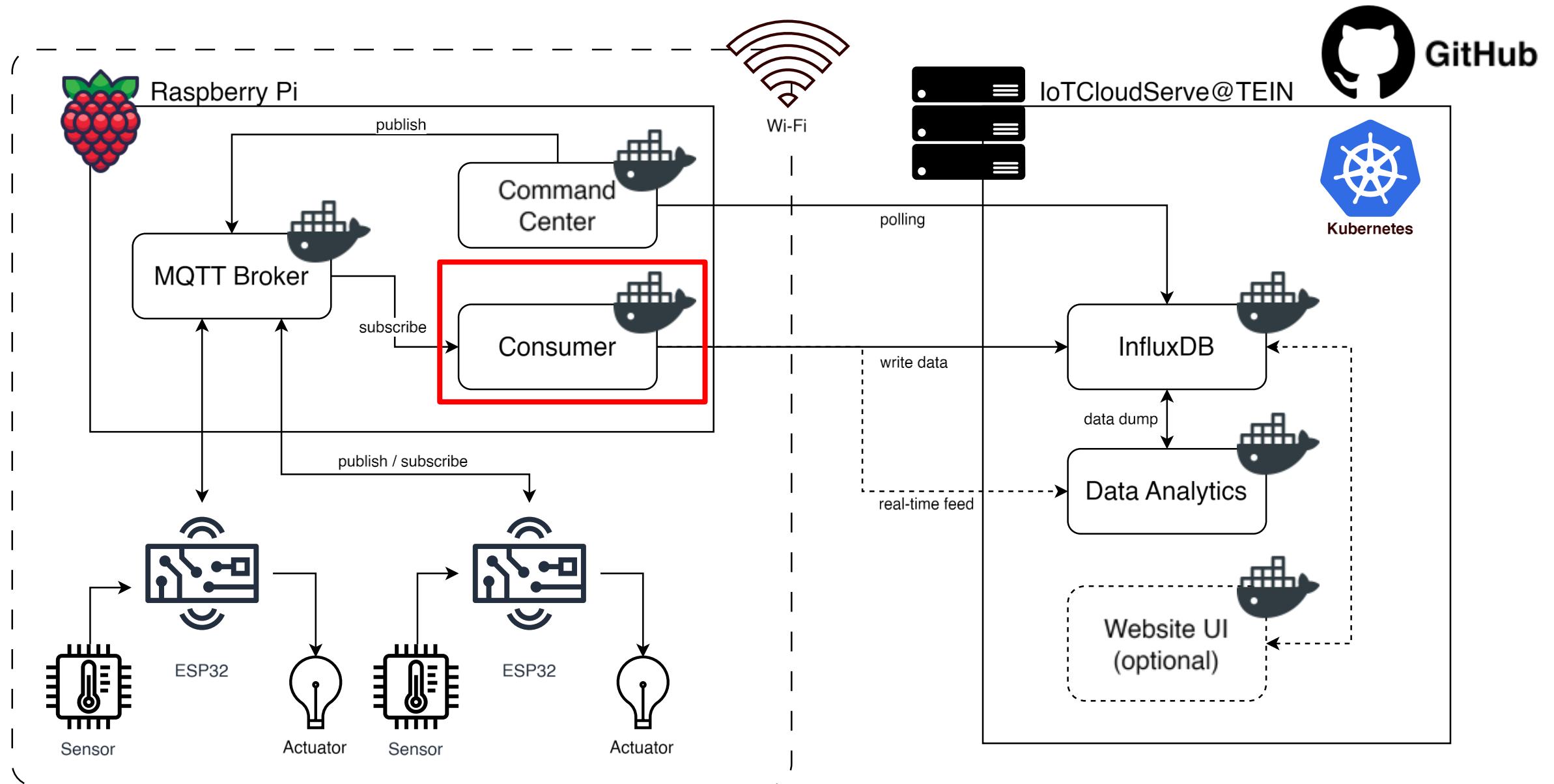
Secrets Values

Key *	Value	
INFLUXDB_USERNAME	= ..... 	
INFLUXDB_PASSWORD	= ..... 	
INFLUXDB_TOKEN	= ..... 	
INFLUXDB_ORG	= ..... 	
INFLUXDB_BUCKET	= ..... 	
INFLUXDB_URL	= <a href="https://film-zomdu-influxdb.iotcloudserve.net/">https://film-zomdu-influxdb.iotcloudserve.net/</a> 	

*ProTip: Paste lines of key=value pairs into any key field for easy bulk entry.*

 [Add Secret Value](#)

Your InfluxDB credentials are now stored in the Rancher.



# Run Consumer on Raspberry Pi

The **consumer** receives the published data from the subscribed topic and puts the data into InfluxDB

## Inside Raspberry Pi

git clone <https://github.com/PatchapongKul/IoT-Final-Project.git>

**Look for Reference\_Code > RaspberryPi > Consumer**

## Build your docker image

sudo docker build -t [image-name]:[tag] .

# Run Consumer on Raspberry Pi

If you want to test the Python script inside Raspberry Pi, you can use the following commands to activate a virtual environment

```
> cd ~/IoT-Final-Project/Training-code/RaspberryPi  
> source pienv/bin/activate  
  
> python [file name].py
```

# Run Consumer on Raspberry Pi

Make sure you change the “.env” file before running the image

```
> sudo nano .env
```

Kindly include all your credentials there

To run the docker image

```
> docker images – to see list of images
```

```
> docker run --env-file=.env [image name]
```

# Run Consumer on Raspberry Pi

To find your IP address

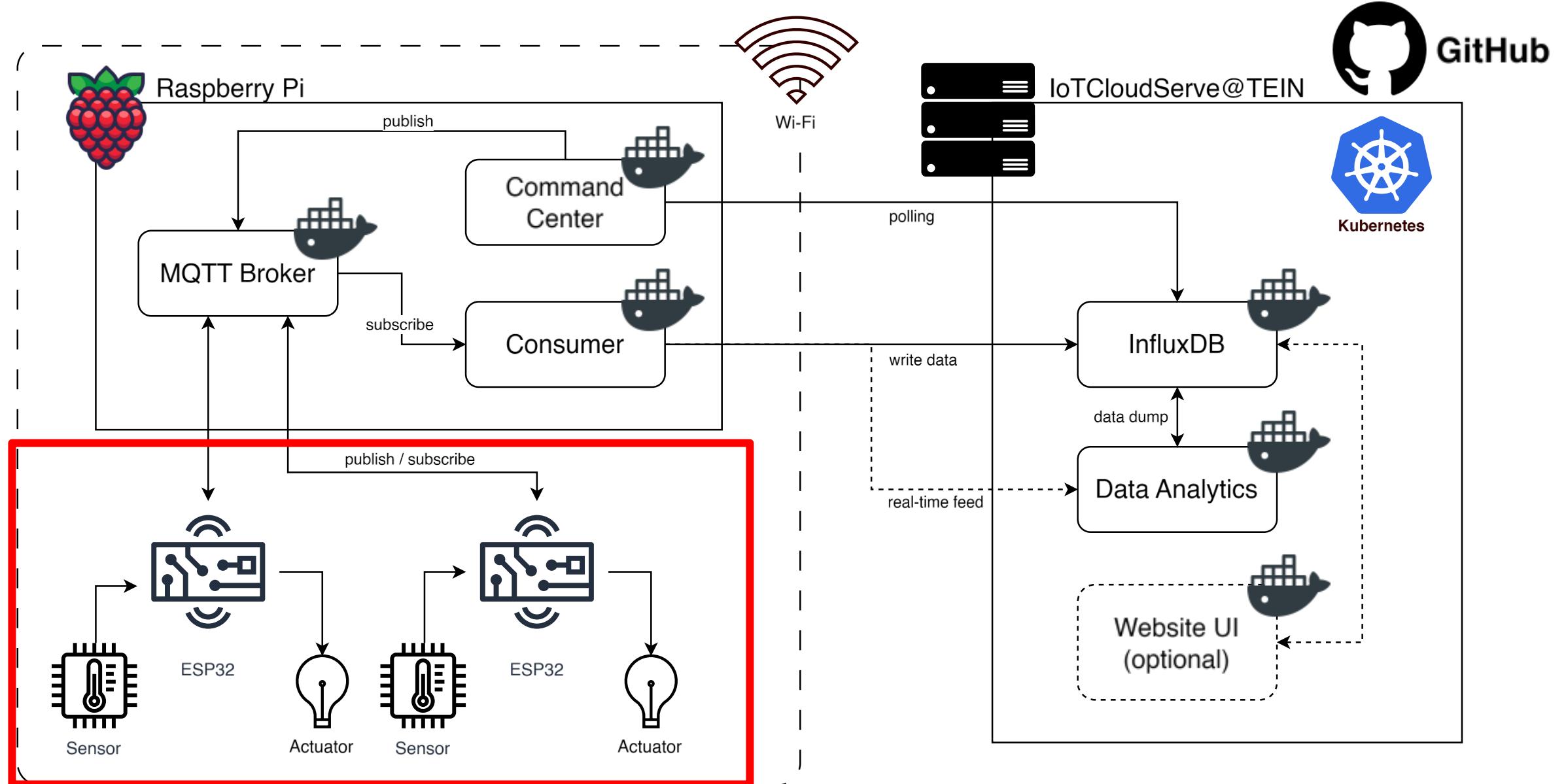
> ip address

or

> ifconfig

Find network interface  
eth: Ethernet  
wlan: Wireless LAN

```
pi-group-0@raspberrypi0: ~
File Edit Tabs Help
pi-group-0@raspberrypi0:~ $ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
            inet6 ::1/128 scope host noprefixroute
                valid_lft forever preferred_lft forever
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default
    qlen 1000
        link/ether dc:a6:32:2c:e4:5d brd ff:ff:ff:ff:ff:ff
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
        link/ether dc:a6:32:2c:e4:5e brd ff:ff:ff:ff:ff:ff
        inet 192.168.141.34/24 brd 192.168.141.255 scope global dynamic noprefixroute wlan0
            valid_lft 3568sec preferred_lft 3568sec
            inet6 2001:44c8:4531:ddaa:36cb:8a8d:4ca1:1c2b/64 scope global dynamic noprefixroute
                valid_lft 7171sec preferred_lft 7171sec
                inet6 fe80::3d4c:8693:fe6:991/64 scope link noprefixroute
                    valid_lft forever preferred_lft forever
4: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN
    group default
        link/ether 02:42:ed:d6:bb:8f brd ff:ff:ff:ff:ff:ff
        inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
            valid_lft forever preferred_lft forever
pi-group-0@raspberrypi0:~ $
```



# Run Producer on LAPTOP

The **producer** publishes the data from the dataset available at  
<https://www.kaggle.com/datasets/ananthr1/room-occupancy-estimation-data-set/data>

The dataset contains ambient room data, including temperature, sound, light, and PIR sensors. We aim to predict the number of people inside the room (0 to 3 people) using this data.

## On your Laptop

git clone <https://github.com/PatchapongKul/IoT-Final-Project.git>

# Run Producer on LAPTOP

The **producer** publishes the data from the dataset available at

<https://www.kaggle.com/datasets/ananthr1/room-occupancy-estimation-data-set/data>

Activate virtual environment

```
> cd Training-code/  
> .\venv\Script\activate
```

Run Python Script

```
> cd [your directory]  
> python Producer.py
```

# Check the result in InfluxDB

**Data Explorer**

Graph CUSTOMIZE Local SAVE AS

Looks like you don't have any queries. Be a lot cooler if you did!

Query 1 +

View Raw Data CSV Past 5m SCRIPT EDITOR SUBMIT

FROM

Search buckets

**film-zomdu**

\_monitoring

\_tasks

+ Create Bucket

Filter

\_measurement

1

Search \_measurement tag values

query\_influxdb\_source...

**sensor\_data**

service\_bucket\_new\_ca...

service\_bucket\_new\_du...

service\_onboard\_new\_c...

Filter

\_field

2

Search \_field tag values

S1\_Temp

S2\_Light

S2\_Sound

S2\_Temp

S3\_Light

No tag keys found in the current time range

WINDOW PERIOD

CUSTOM AUTO

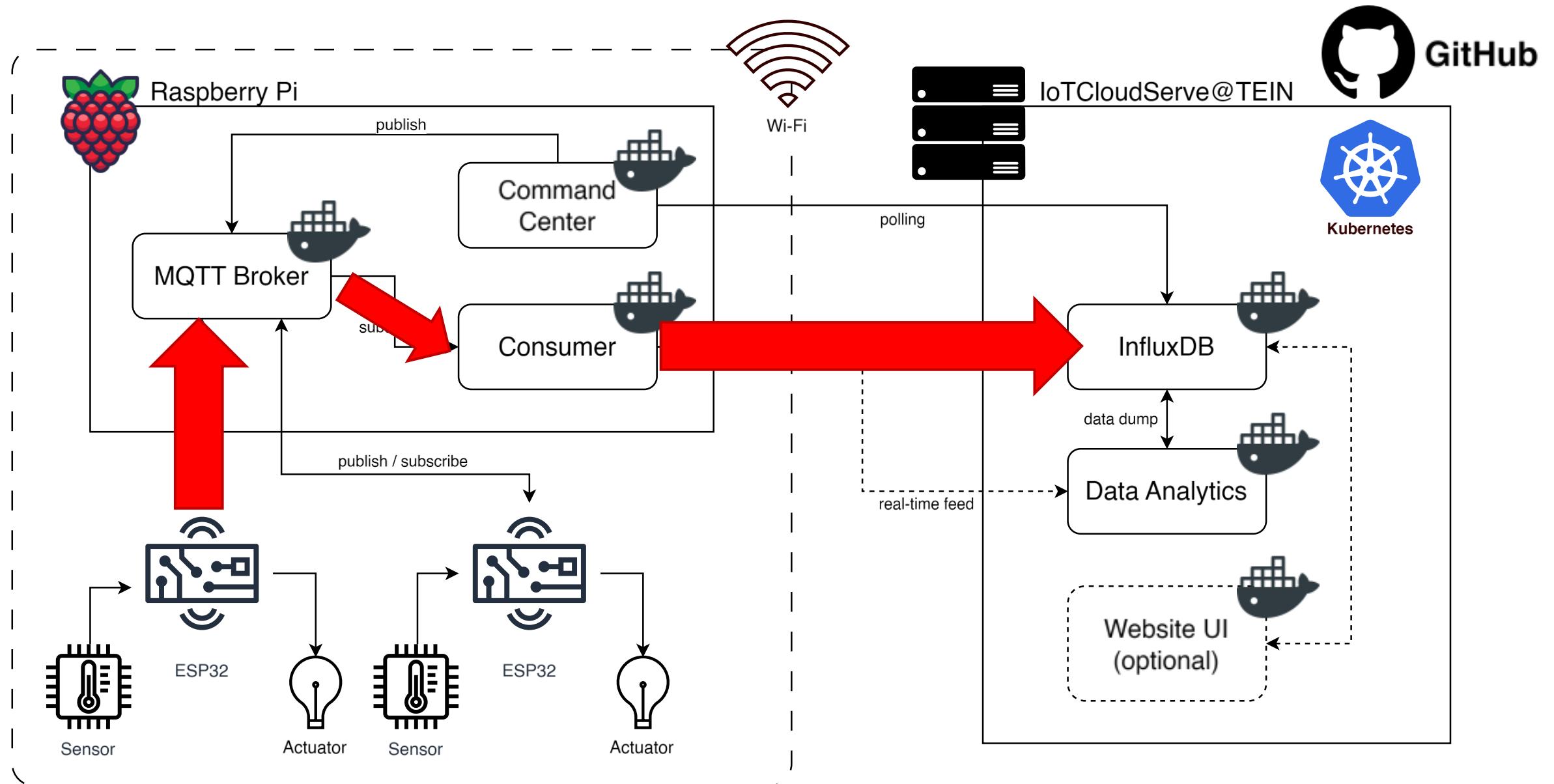
auto (10s)

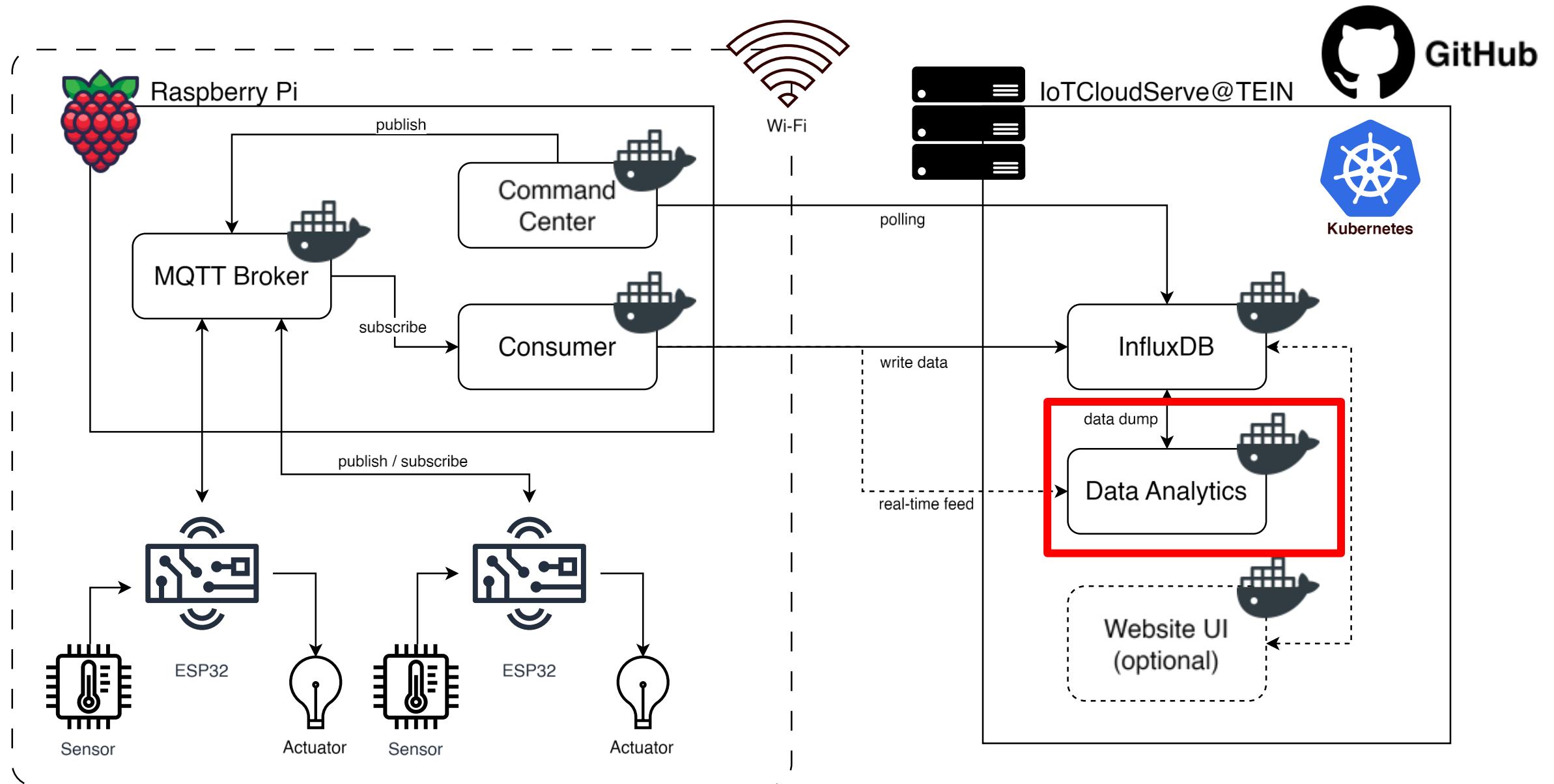
Fill missing values ?

AGGREGATE FUNCTION

CUSTOM AUTO

The screenshot shows the InfluxDB Data Explorer interface. On the left is a sidebar with various icons. The main area has a dark background with a central graphic of a 3D bar chart and a line graph. A message says "Looks like you don't have any queries. Be a lot cooler if you did!". Below this is a search bar labeled "Query 1" with a plus sign. To its right are buttons for "View Raw Data" (disabled), "CSV", "Past 5m", "SCRIPT EDITOR", and "SUBMIT". The "FROM" section shows a dropdown menu with "Search buckets" and a list of buckets: "film-zomdu" (highlighted in blue), "\_monitoring", "\_tasks", and "+ Create Bucket". The "Filter" section has two dropdown menus: one for "\_measurement" (with "sensor\_data" selected) and one for "\_field" (with "S1\_Temp", "S2\_Light", "S2\_Sound", "S2\_Temp", and "S3\_Light" listed). A note below says "No tag keys found in the current time range". On the right, there are sections for "WINDOW PERIOD" (set to "auto (10s)" with a "Fill missing values" checkbox) and "AGGREGATE FUNCTION" (set to "auto").





# Train KNN Model on LAPTOP

Within the limited time, we will apply the analysis from the Kaggle to our work.

Ref: <https://www.kaggle.com/code/vivekaryan/room-occupancy-estimation-with-variable-selection>

**Look inside Training-code> IoTCloudServe@TEIN > ModelTraining**

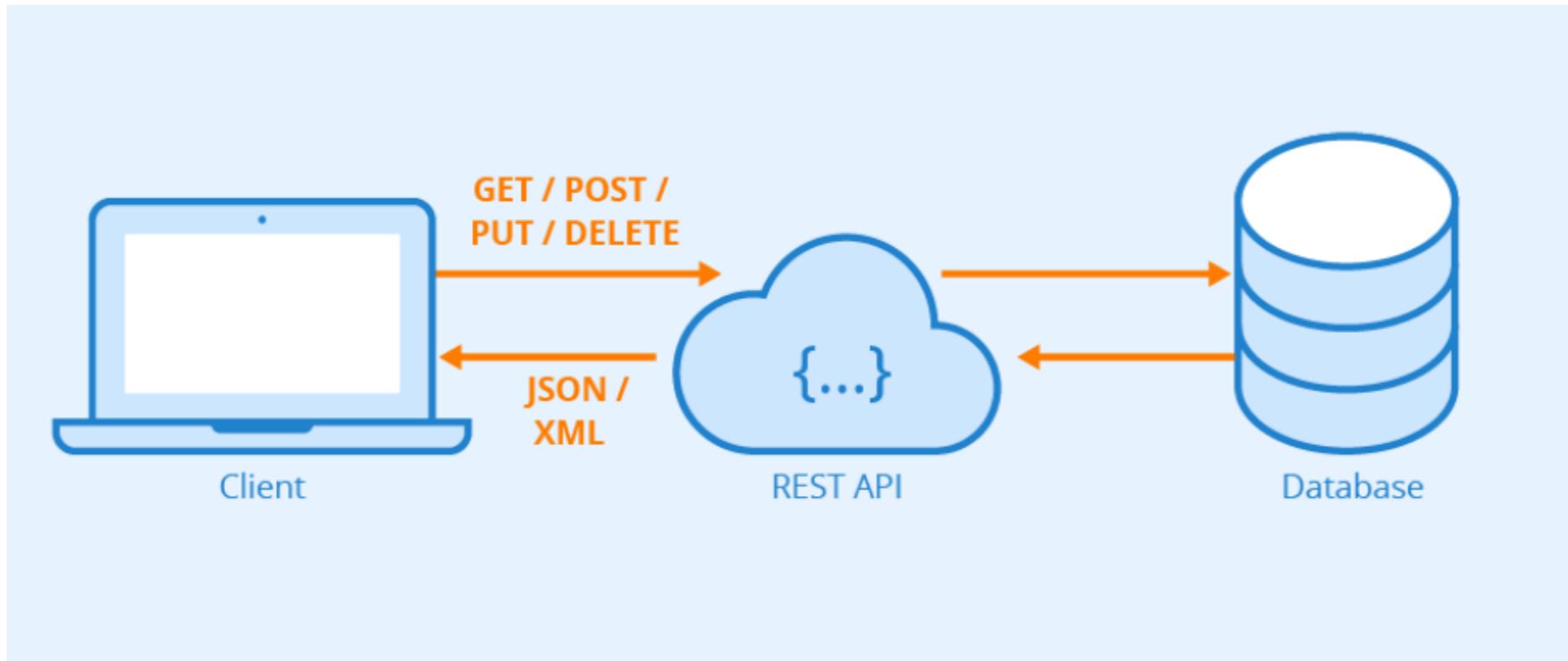
# Deploy Model on IoTCloudServe@TEIN

We will develop a REST API server to serve as the endpoint for consumers to send data, enabling our server to predict the occupancy of a room based on the received information.

**Look inside Training-code > IoTCloudServe@TEIN >  
RealTimePrediction**

# Develop REST API Server using Flask

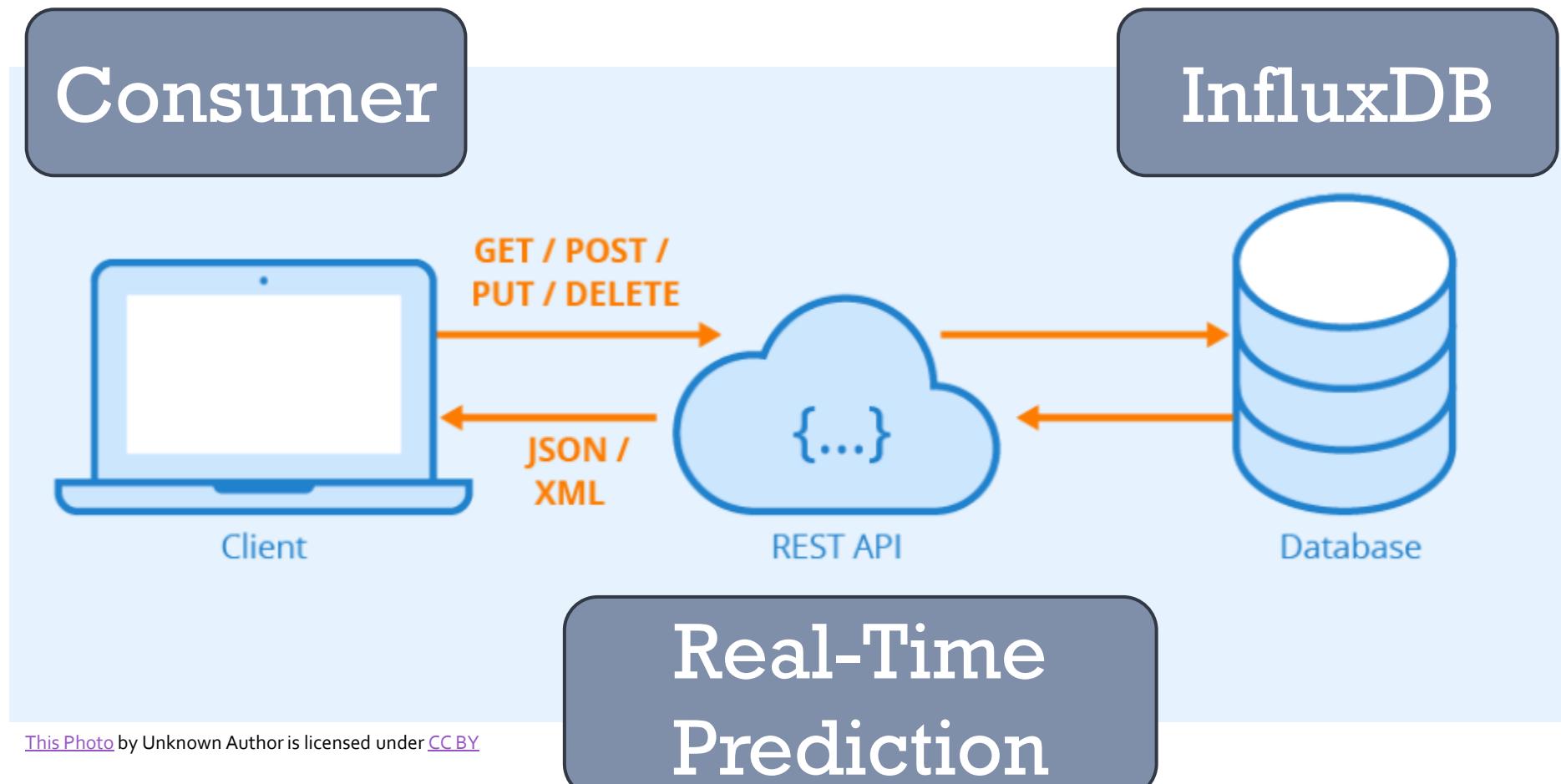
<https://flask.palletsprojects.com/en/3.0.x/>



This Photo by Unknown Author is licensed under [CC BY](#)

# Develop REST API Server using Flask

<https://flask.palletsprojects.com/en/3.0.x/>



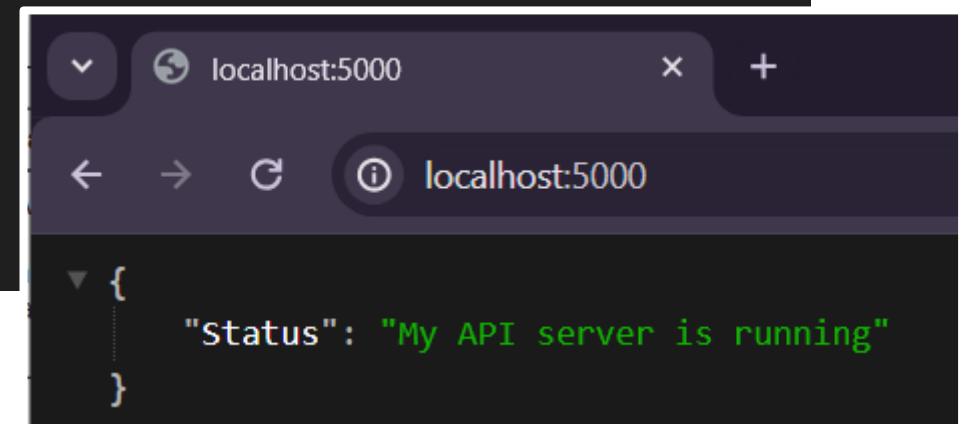
# Develop REST API Server using Flask

```
from flask import Flask, request, jsonify

app = Flask(__name__)

# Default Route
@app.route('/')
def hello():
    return jsonify({"Status": "My API server is running"})

if __name__ == '__main__':
    app.run()
```



# Develop REST API Server using Flask

Look inside Training-code > IoTCloudServer@TEIN > SimpleAPI

```
# Route for handling GET requests
@app.route('/get_data', methods=['GET'])
def get_data():
    return jsonify(data)

# Route for handling POST requests
@app.route('/add_data', methods=['POST'])
def add_data():
    req_data = request.get_json()
    if req_data:
        data.append(req_data)
        return jsonify({"message": "Data added successfully"})
    else:
        return jsonify({"error": "No data provided"}), 400
```

# Develop REST API Server using Flask

```
> cd [your directory]
```

```
> python app.py
```

By default, Flask will run on port 5000

You can test the API server using **Postman**

# Test REST API Server using Postman

Create three new HTTP request

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Demo' and 'iot-cloud-workshop'. A 'HTTP' request is selected in the 'iot-cloud-workshop' collection. The main workspace shows various protocol icons: HTTP, GraphQL, gRPC, WebSocket, and Socket.IO. Below these are MQTT, Collection, Environment, Flow, and Workspace icons. A tooltip for the HTTP icon provides a brief description: 'Hypertext Transfer Protocol (HTTP) is an application-layer protocol often used to build REST APIs. Test your HTTP API with an HTTP request.' The right side of the screen shows the request details panel with fields for 'Save', 'Send', 'Cookies', and 'Description'. At the bottom, there are status indicators for the request (200 OK, 7 ms, 191 B) and options to 'Save as example'.

# Test REST API Server using Postman

## Default Route shows the Status

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Demo' and 'iot-cloud-workshop', environments like 'Magel', 'MHE Project', and 'MQTT-NETPIE', and a history section. The main workspace has three tabs at the top: 'GET Get Data', 'POST Post Data', and 'GET Status'. The 'GET Status' tab is selected and highlighted with a red box around its URL input field. The URL is set to 'http://localhost:5000/'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is selected and shows a table for 'Query Params' with columns for 'Key', 'Value', 'Description', and 'Bulk Edit'. The 'Body' section also includes tabs for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The 'JSON' tab is selected and highlighted with a red box, showing the response body: a single-line JSON object '{ "Status": "My API server is running" }'. At the bottom of the workspace, there are buttons for 'Save as example', '200 OK', '11 ms', '207 B', and 'Save as example'. The bottom navigation bar includes links for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Trash', and help icons.

# Test REST API Server using Postman

## Get Data Route shows list of all data

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Demo', 'iot-cloud-workshop' (which has 'GET Status' and 'GET Get Data' requests), 'Magel', 'MHE Project', and 'MQTT-NETPIE'. The main area shows a 'GET Get Data' request under the 'iot-cloud-workshop' environment. The request URL is 'http://localhost:5000/get\_data'. The response body is displayed in JSON format, showing a single item with an empty array: '1 [ ]'. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings, and Cookies. At the bottom, there are buttons for Pretty, Raw, Preview, Visualize, and JSON, along with status information (200 OK, 6 ms, 166 B) and a 'Save as example' button.

# Test REST API Server using Postman

Add Data Route shows the response of adding JSON data

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Demo', 'iot-cloud-workshop' (which has 'GET Status' and 'GET Get Data' under it), and 'POST Post Data'. Below that are environments like 'Magel', 'MHE Project', and 'MQTT-NETPIE'. The main area shows a 'POST Post Data' collection. A red box highlights the 'POST' method and the URL 'http://localhost:5000/add\_data'. Another red box highlights the 'Body' tab where raw JSON data is sent: `{ "temp": 25 }`. A third red box highlights the response body, which is a JSON object: `{ "message": "Data added successfully" }`. The status bar at the bottom shows '200 OK'.

# Test REST API Server using Postman

## Get Data Route shows list of all data

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Demo', 'iot-cloud-workshop' (which has 'GET Status' and 'GET Get Data' requests), 'Magel', 'MHE Project', and 'MQTT-NETPIE'. The main area shows a 'GET Get Data' request under the 'iot-cloud-workshop' environment. The request URL is highlighted with a red box: `http://localhost:5000/get_data`. The response body, also highlighted with a red box, is a JSON array with one item:

```
1 [  
2 {  
3   "temp": 25  
4 }  
5 ]
```

The status bar at the bottom indicates a 200 OK response with 7 ms latency and 191 B size.

# Deploy Model on IoTCloudServe@TEIN

We will develop a REST API server to serve as the endpoint for consumers to send data, enabling our server to predict the occupancy of a room based on the received information.

**Look inside Training-code > IoTCloudServe@TEIN > RealTimePrediction**

# Deploy Model on IoTCloudServe@TEIN

**Your Turn!!!**

- > Add your ML model to the folder
- > Build image
- > Push to Docker Hub
- > Deploy on Rancher (map port, add secrets)
- > Assign Domain Name

# Deploy Model on IoTCloudServe@TEIN

You will need to include your InfluxDB secrets as a “.env”

[Expand All](#)

▶ Environment Variables  
Set the environment that will be visible to the container, including injecting values from other resources like Secrets.

▶ Node Scheduling  
Configure what nodes the pods can be deployed to.

▶ Health Check  
Periodically make a request to the container to see if it is alive and responding correctly.

▼ Volumes  
Persist and share data separate from the lifecycle of an individual container.

**Add Volume...** ▾  
Add an ephemeral volume  
Add a new persistent volume (claim)  
Use an existing persistent volume (claim)  
Bind-mount a directory from the node  
Use a secret  
Use a config map  
Help & Support  
Use a certificate

an upgrade.

Show advanced options

**Save**   **Cancel**

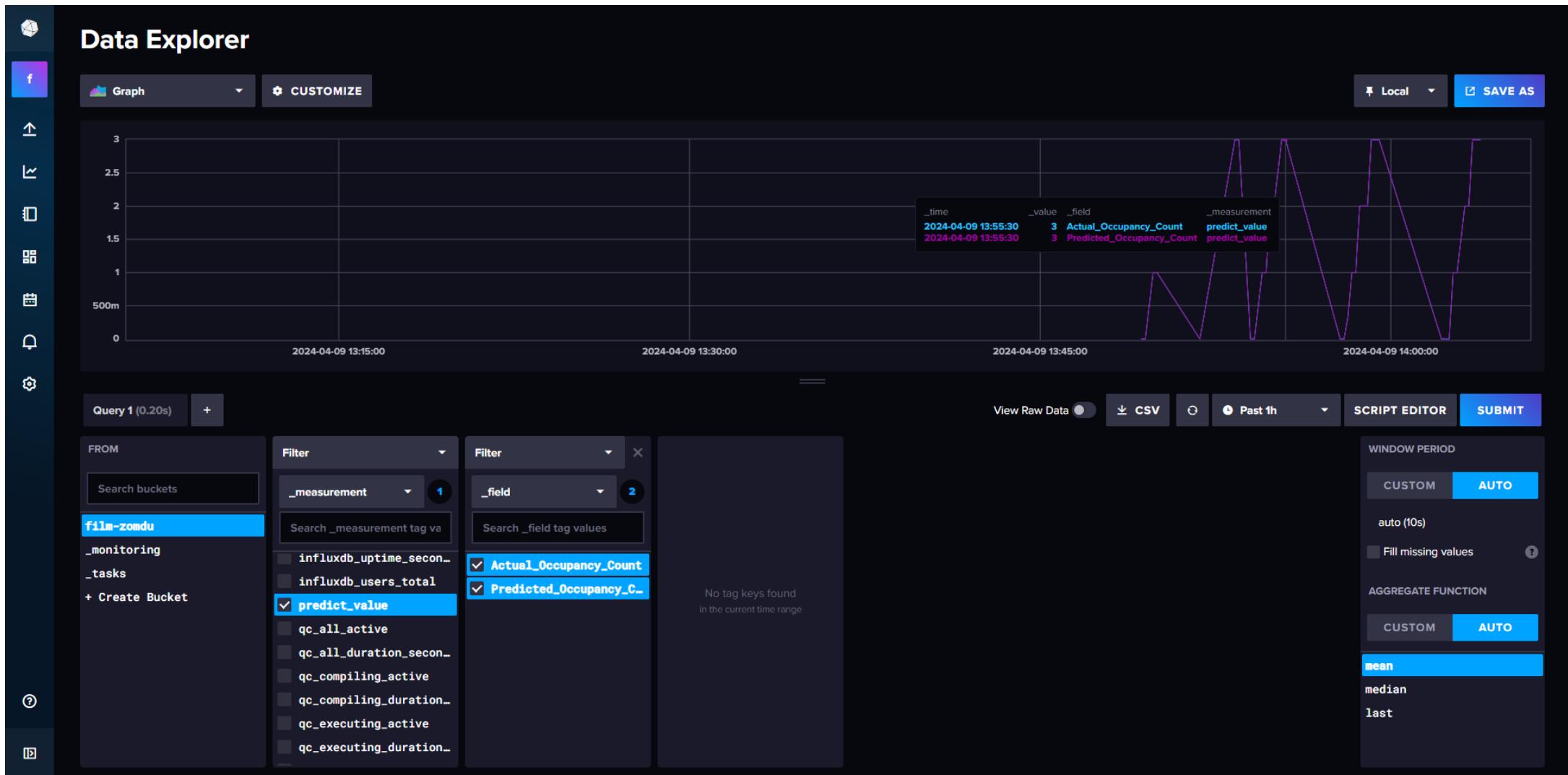
English ▾   Do

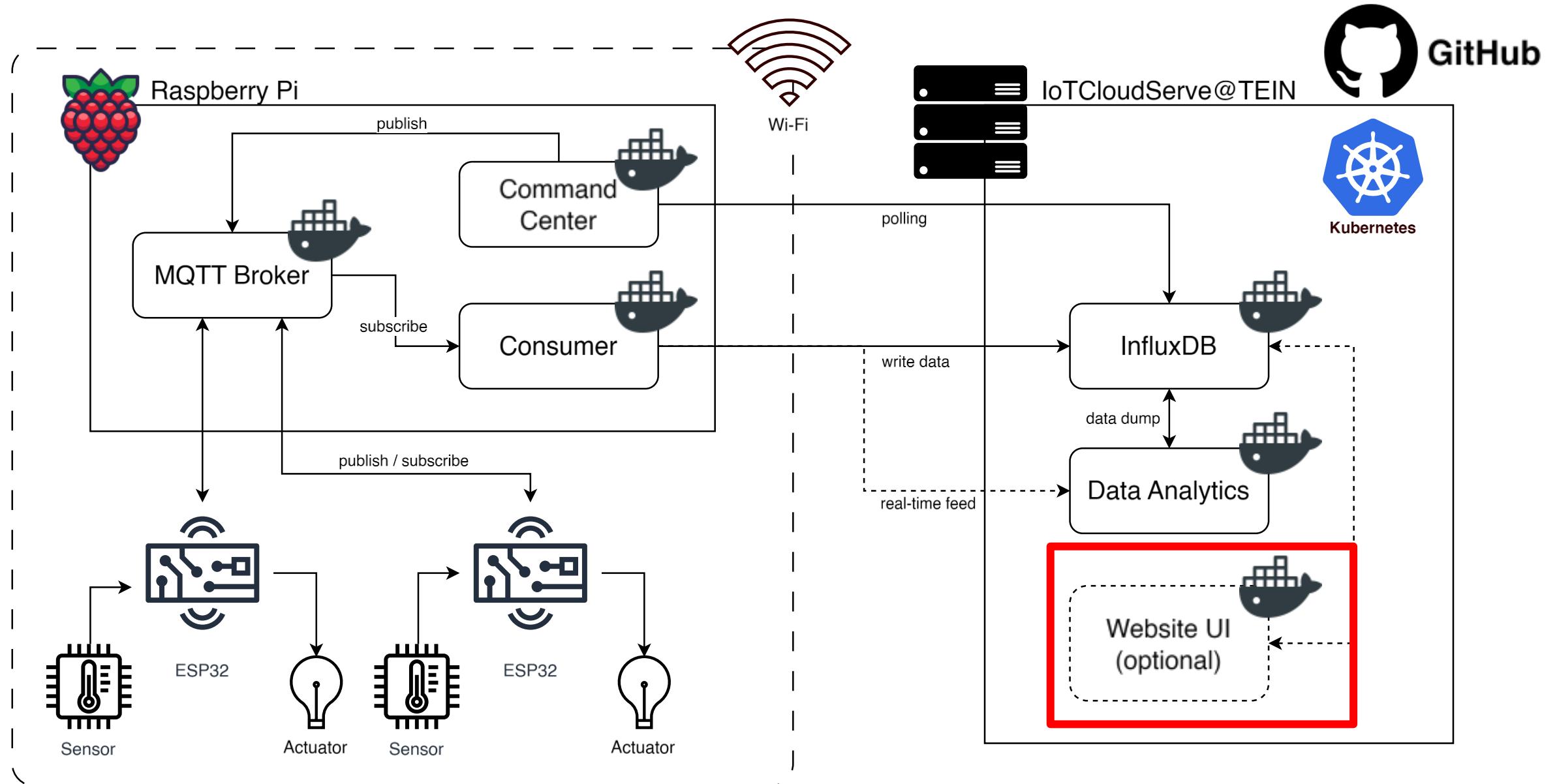
# Deploy Model on IoTCloudServe@TEIN

**Your Turn!!!**

- > Go back to Consumer in Raspberry Pi
- > Change the PREDICT\_URL inside “.env”
- > Run the Consumer again with the new “.env”
- > See the result in InfluxDB
  - measurement: “predict\_value”

# Check the result in InfluxDB

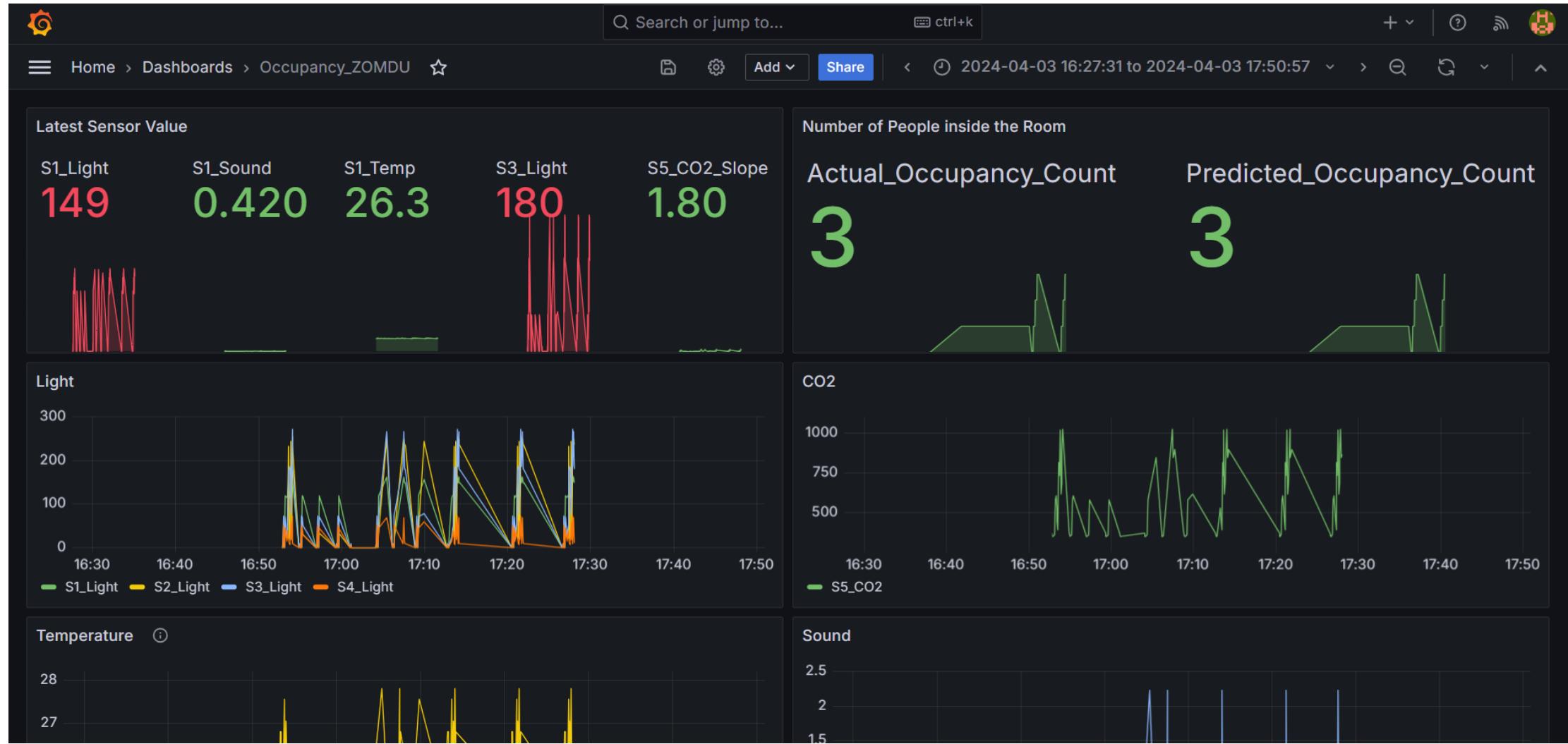




# Website UI - InfluxDB Dashboard



# Deploy Grafana on IoTCloudServe@TEIN

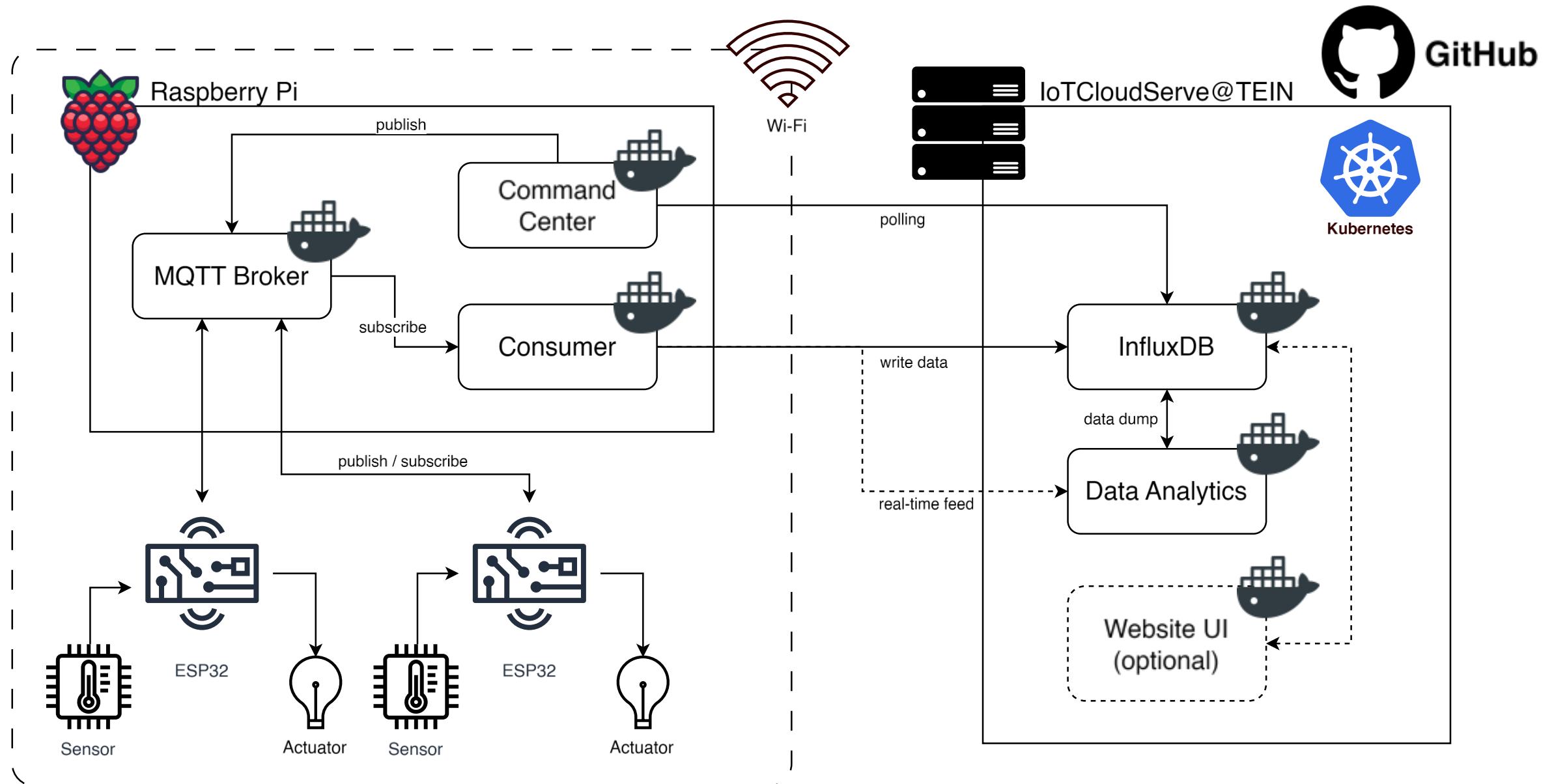


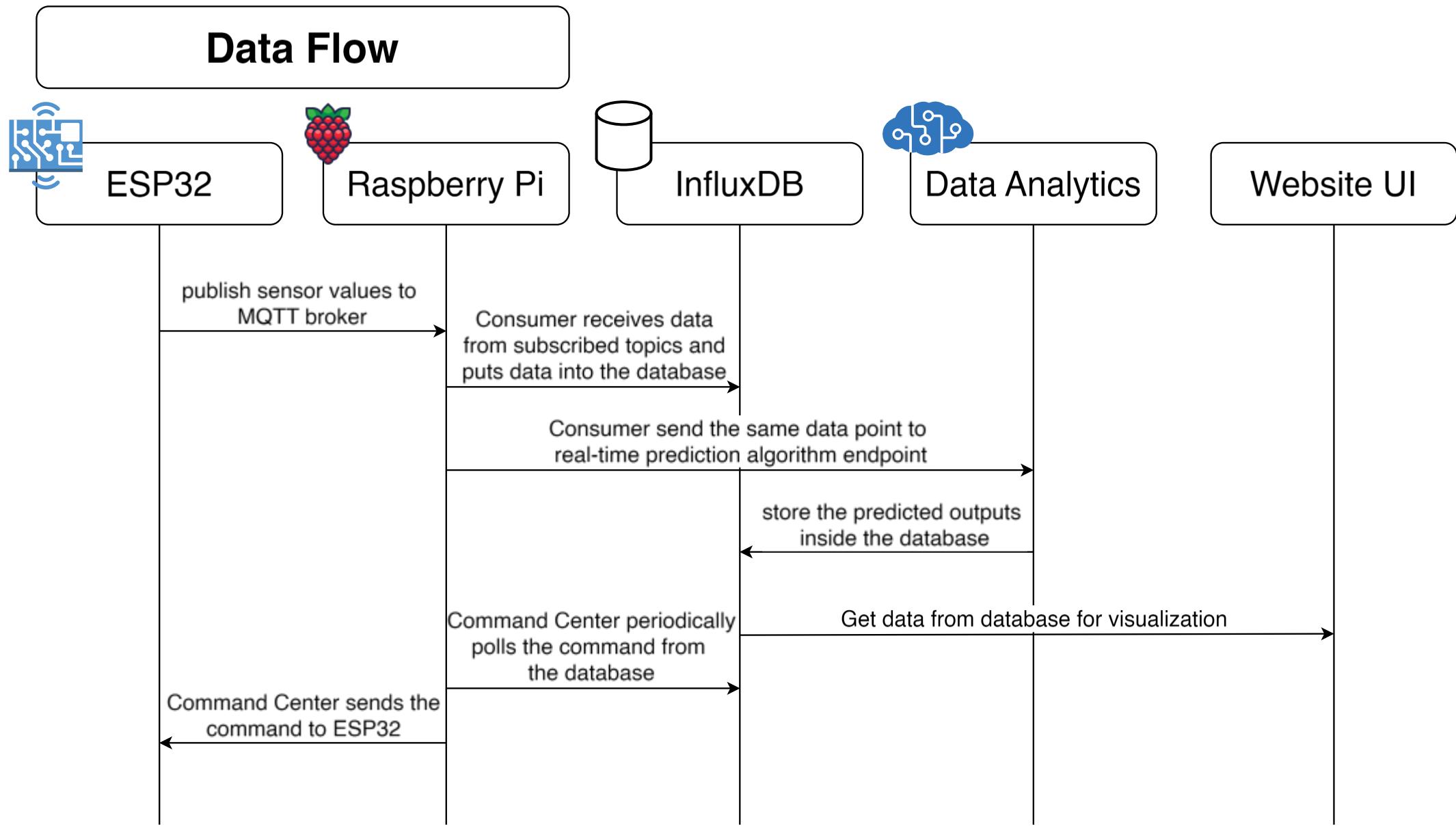
# Deploy Grafana on IoTCloudServer@TEIN

Docker Image:

<https://hub.docker.com/r/grafana/grafana-oss>

Grafana is running on Port 3000





**HOPE YOU ALL ENJOY  
THE WORKSHOP!**

---