

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.

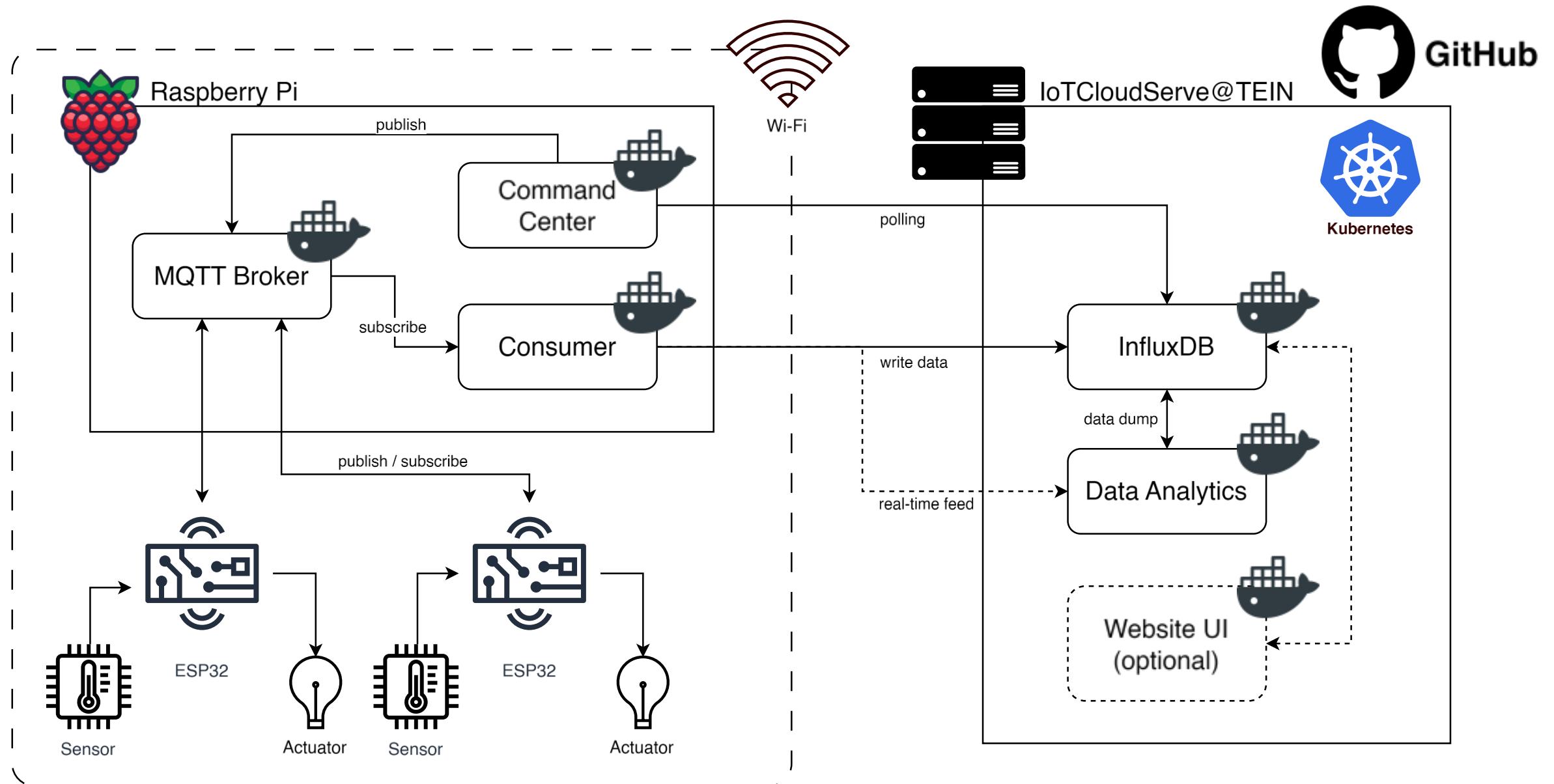
Before the workshop!

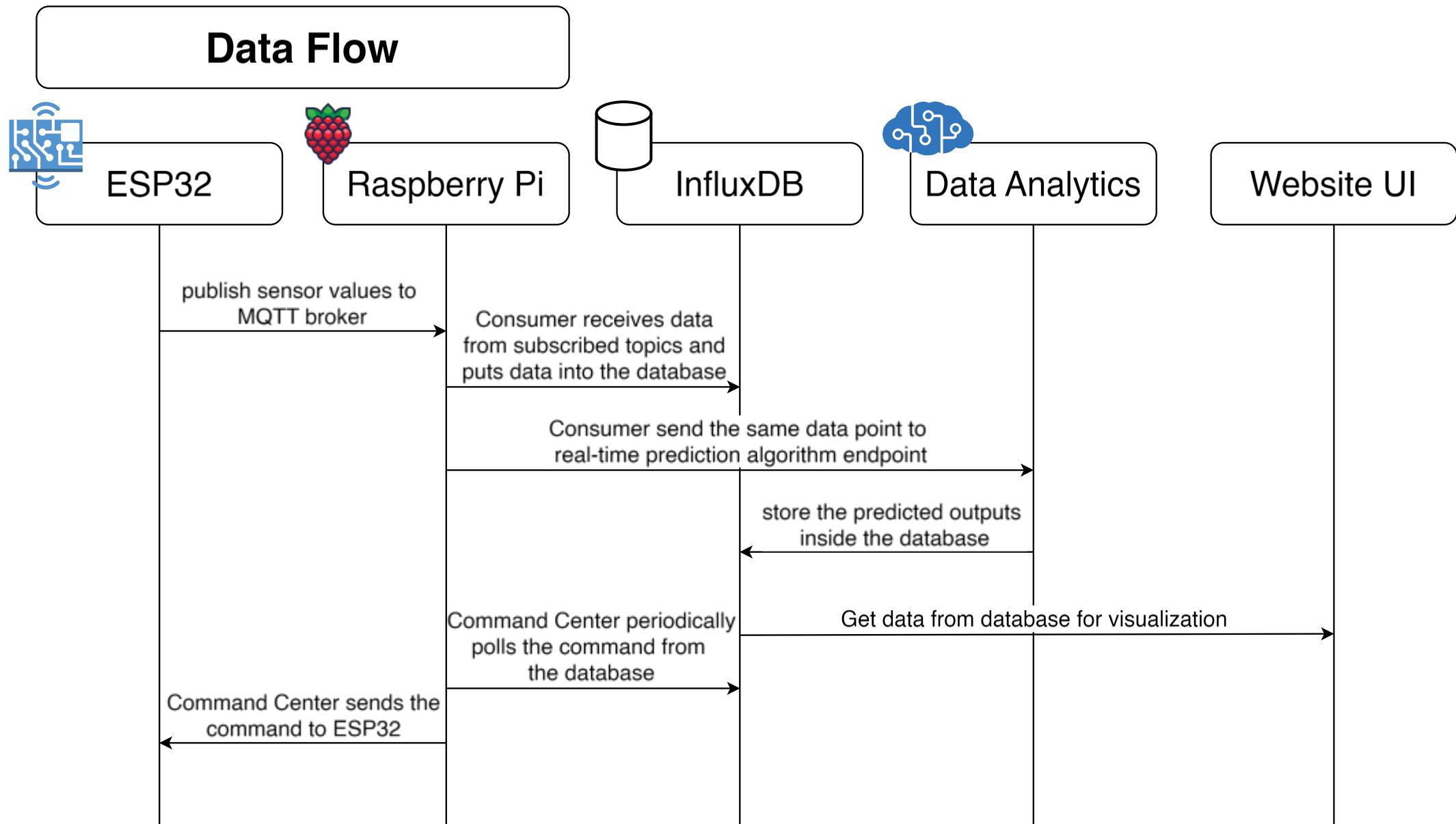
Please prepare the following:

- 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/>

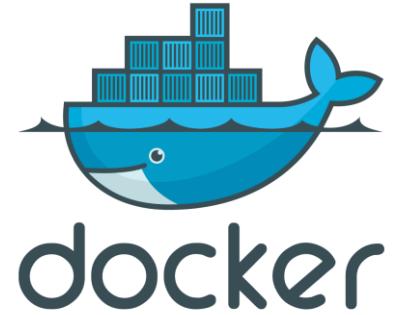
You're free to use other software variants if you prefer, but please note that during the workshop, I'll be demonstrating based on the software list provided.

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 Networks. 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 Signed in. A notification bar at the very bottom indicates a 'New version available' with 3 notifications.

Learning center

What is a container?
Estimated time: 5 mins

Containers on Docker Desktop

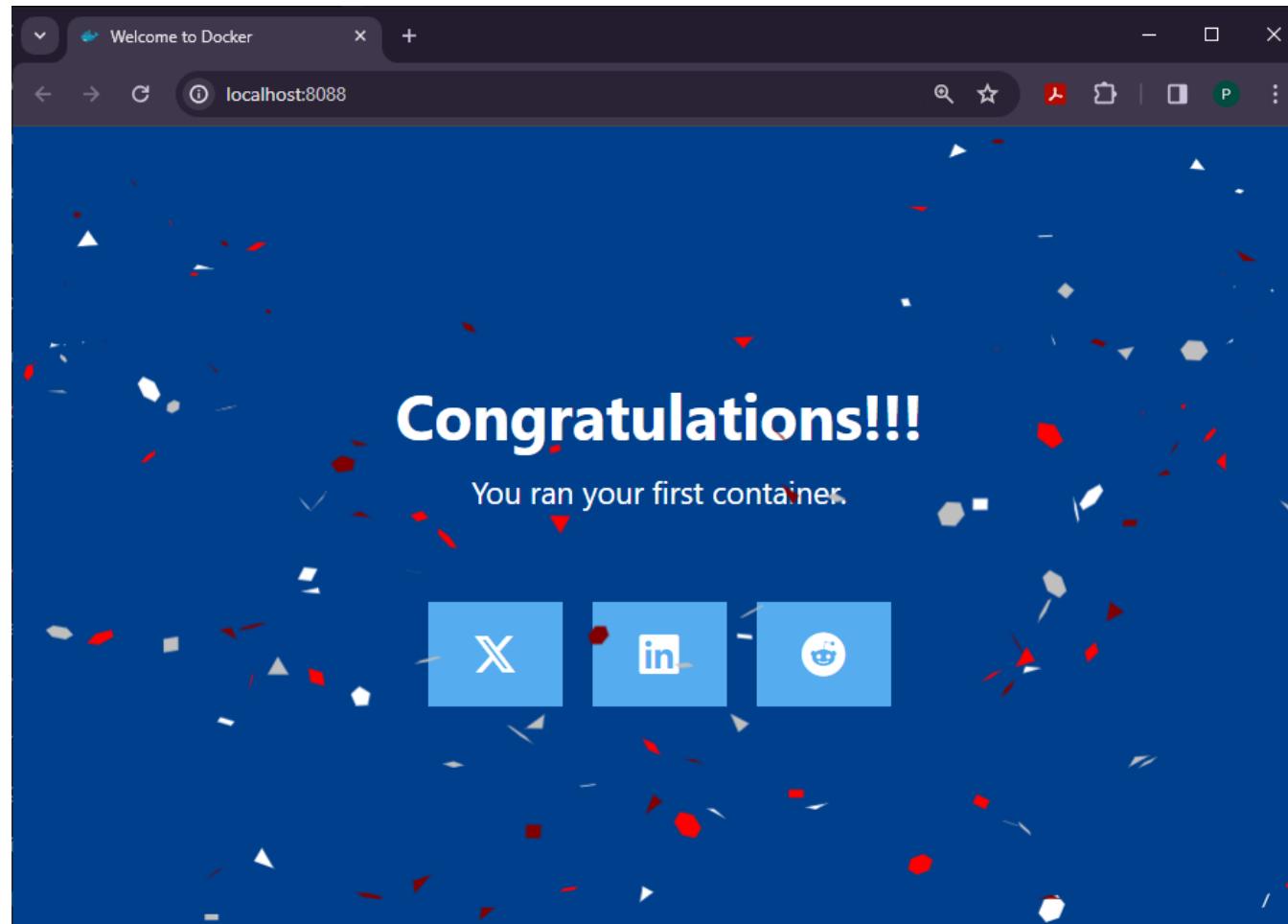
The best way to learn about containers is to first see it in action. We have created a welcome container for you.

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

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

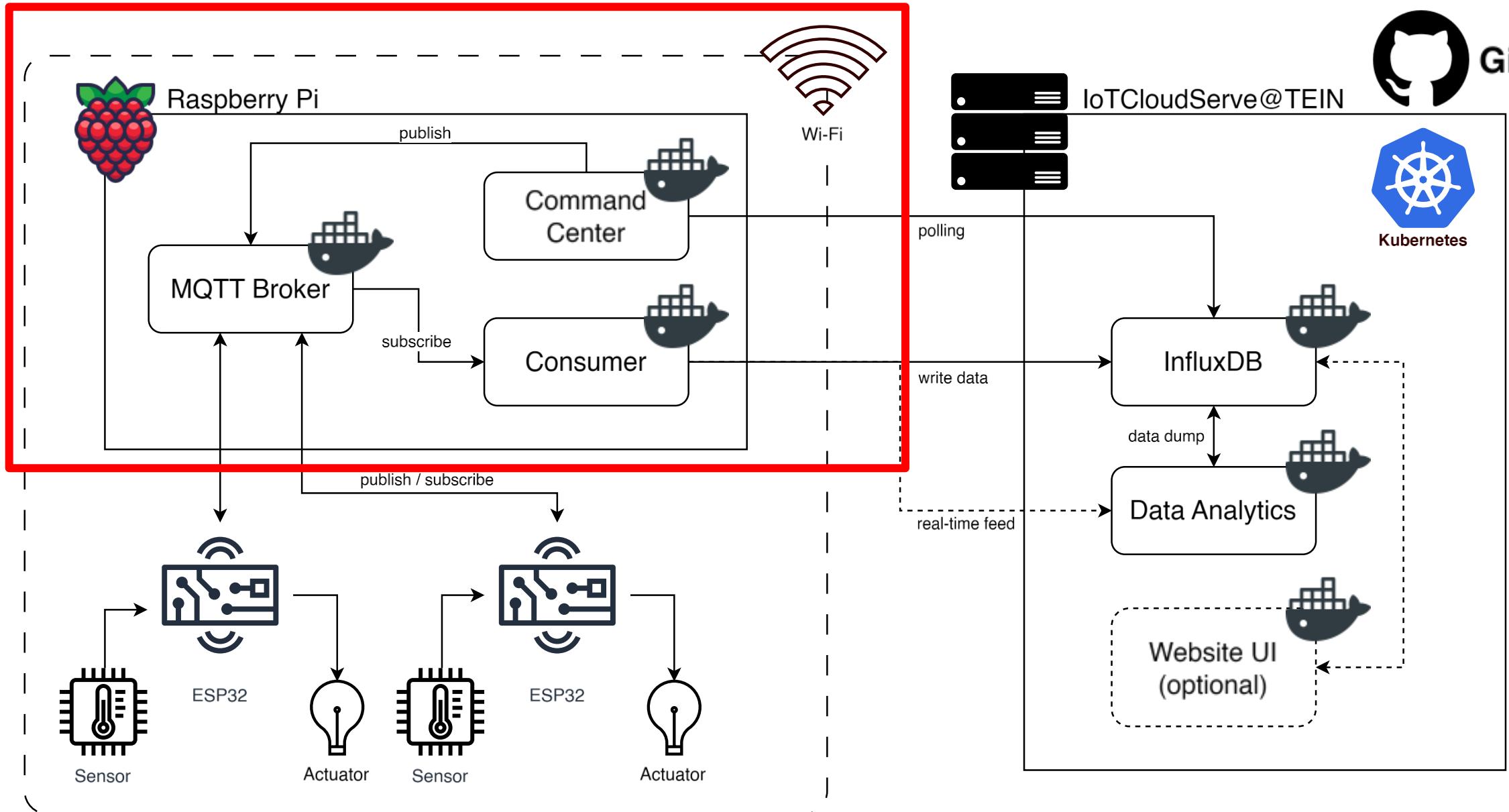
`docker login -u <username>`

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

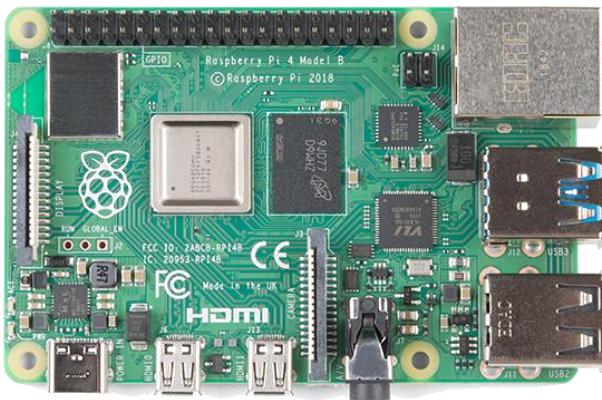
You'll have several options for connecting network for Raspberry Pi

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

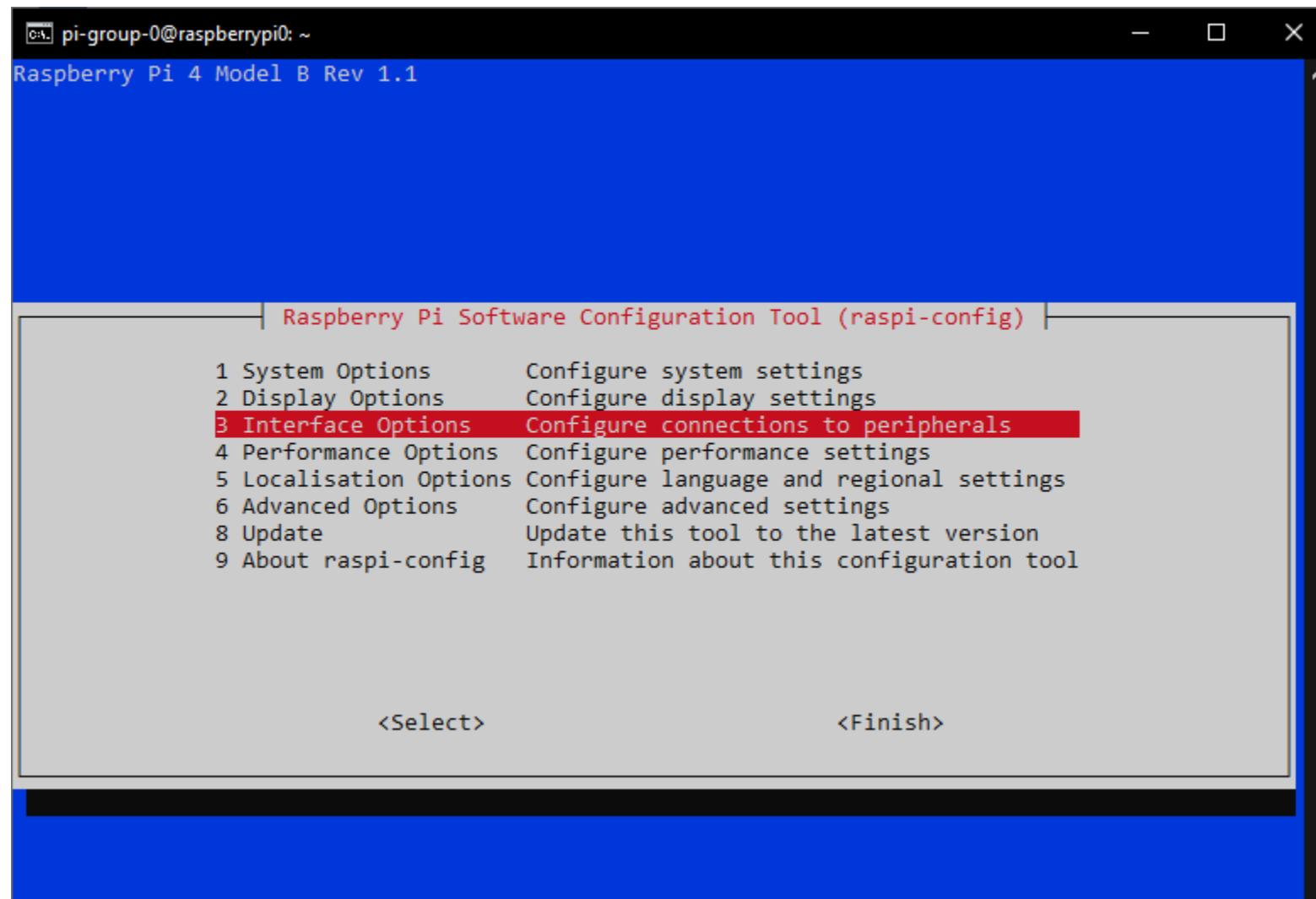
```
pi-group-0@raspberrypi0: ~
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

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

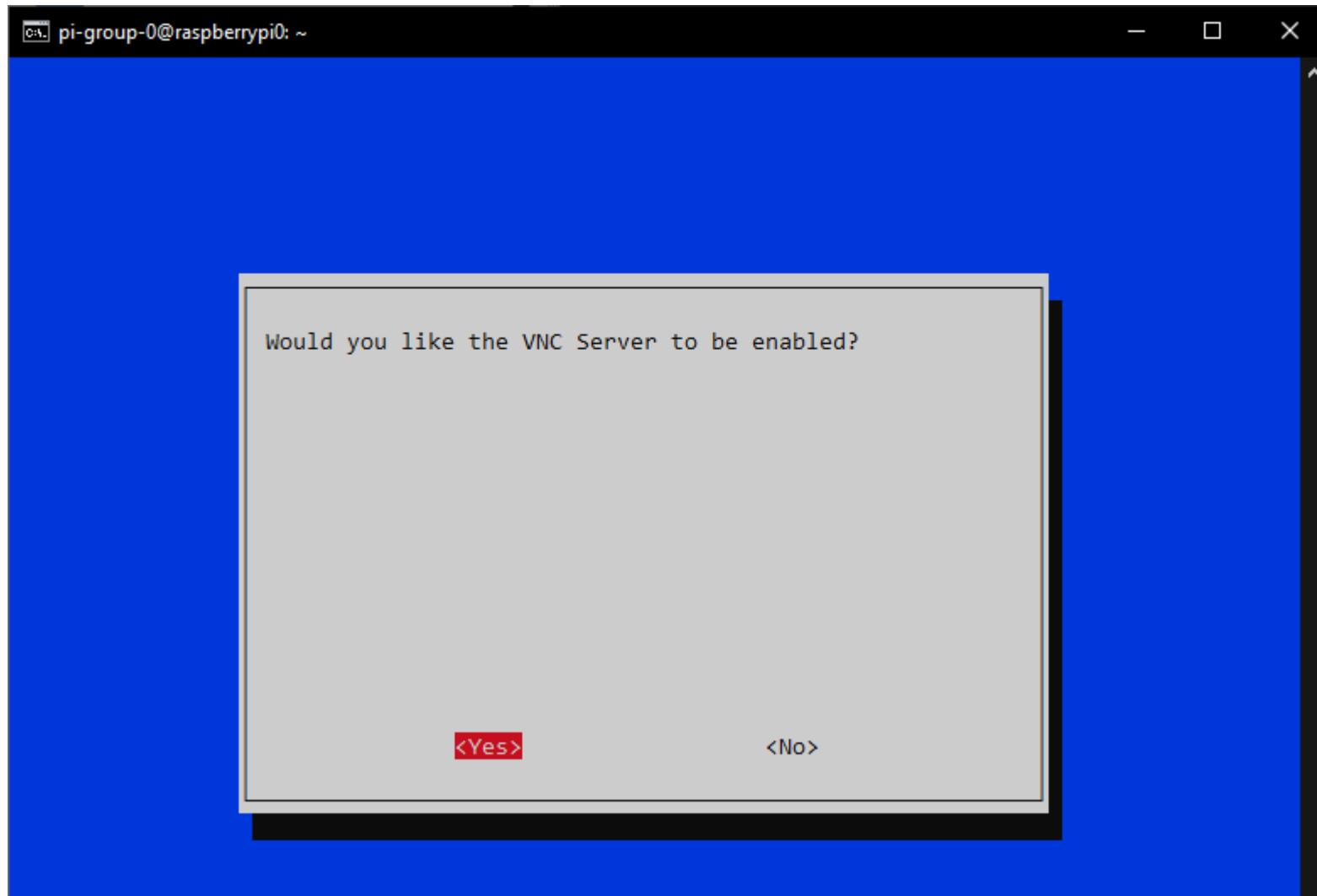
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

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:~ $
```

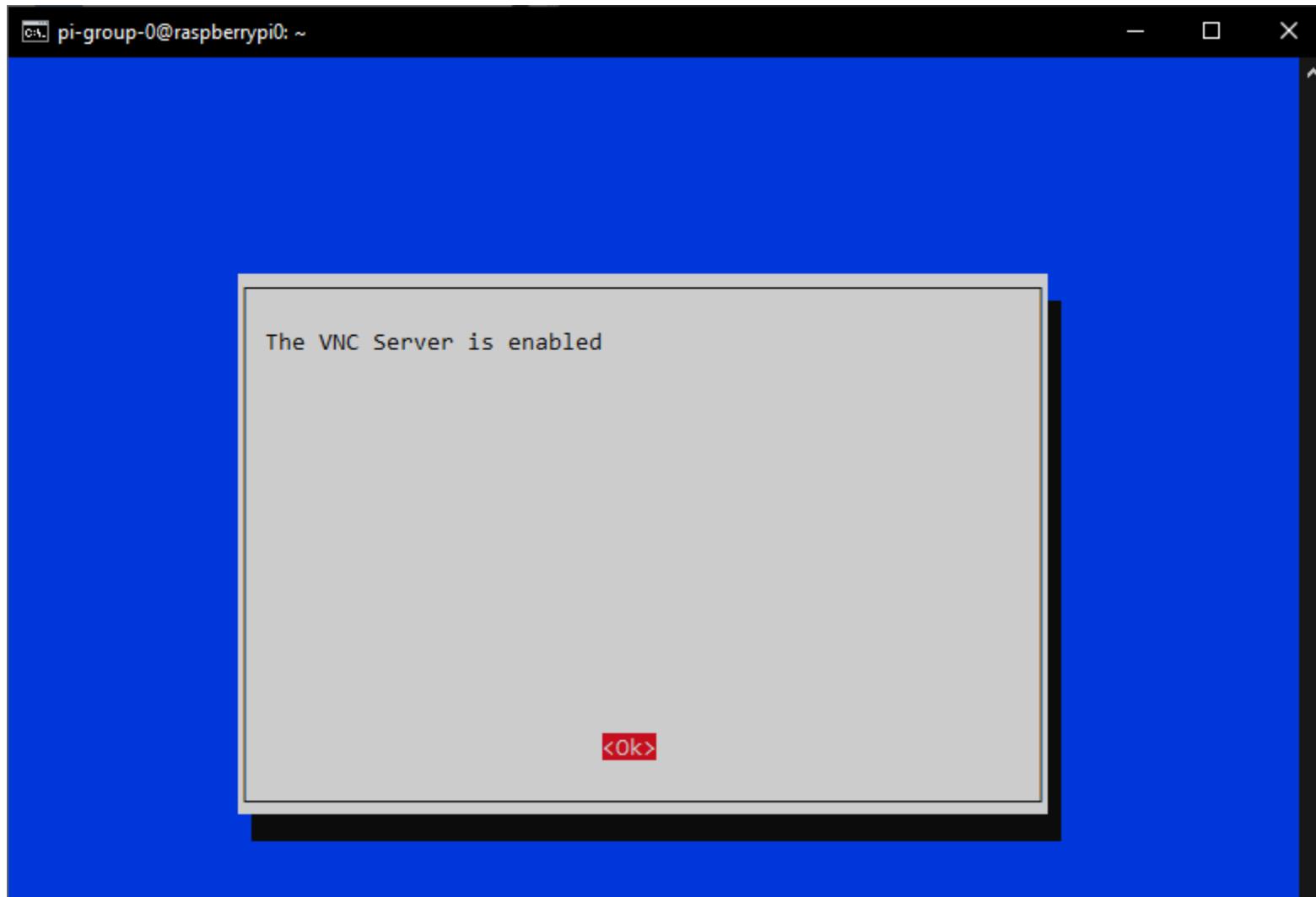
Connect Raspberry Pi via SSH



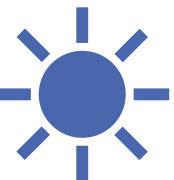
Connect Raspberry Pi via SSH



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

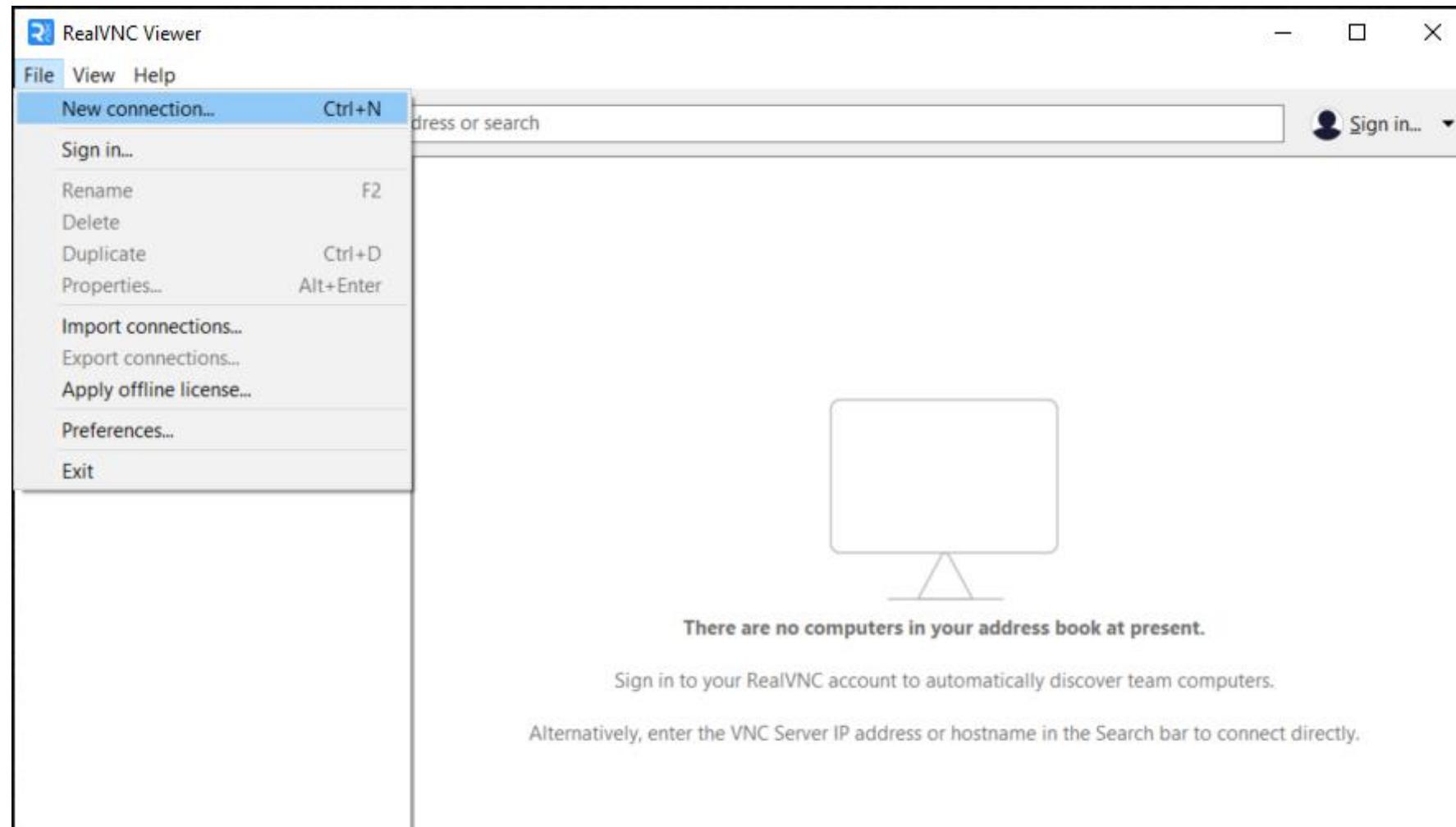


Raspberry Pi

*We need LAN for using VNC Viewer

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

VNC Viewer

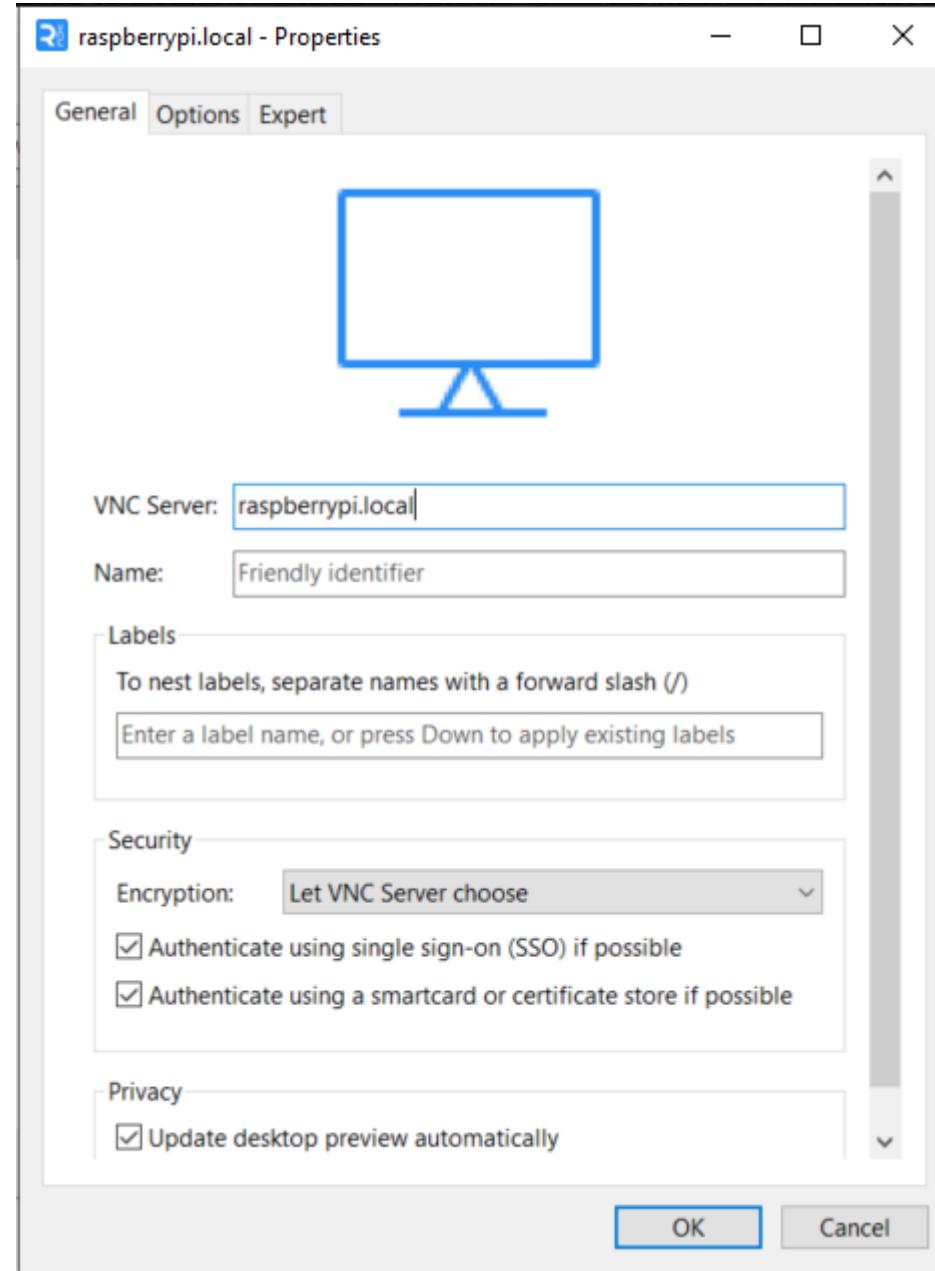


VNC Viewer

VNC Server:

- [Hostname].local
- or IP Address

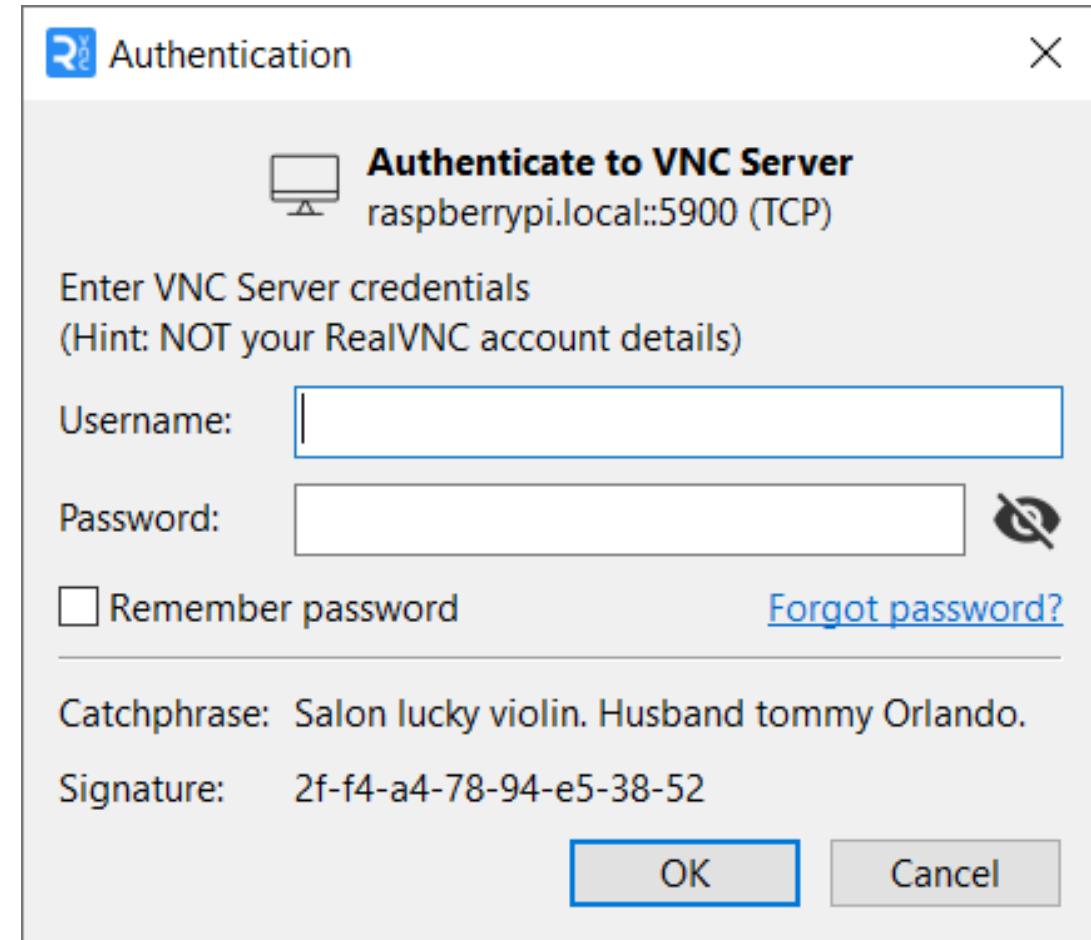
*Note: pi-group-1 is your **username** and raspberrypi1 is your **hostname***



VNC Viewer

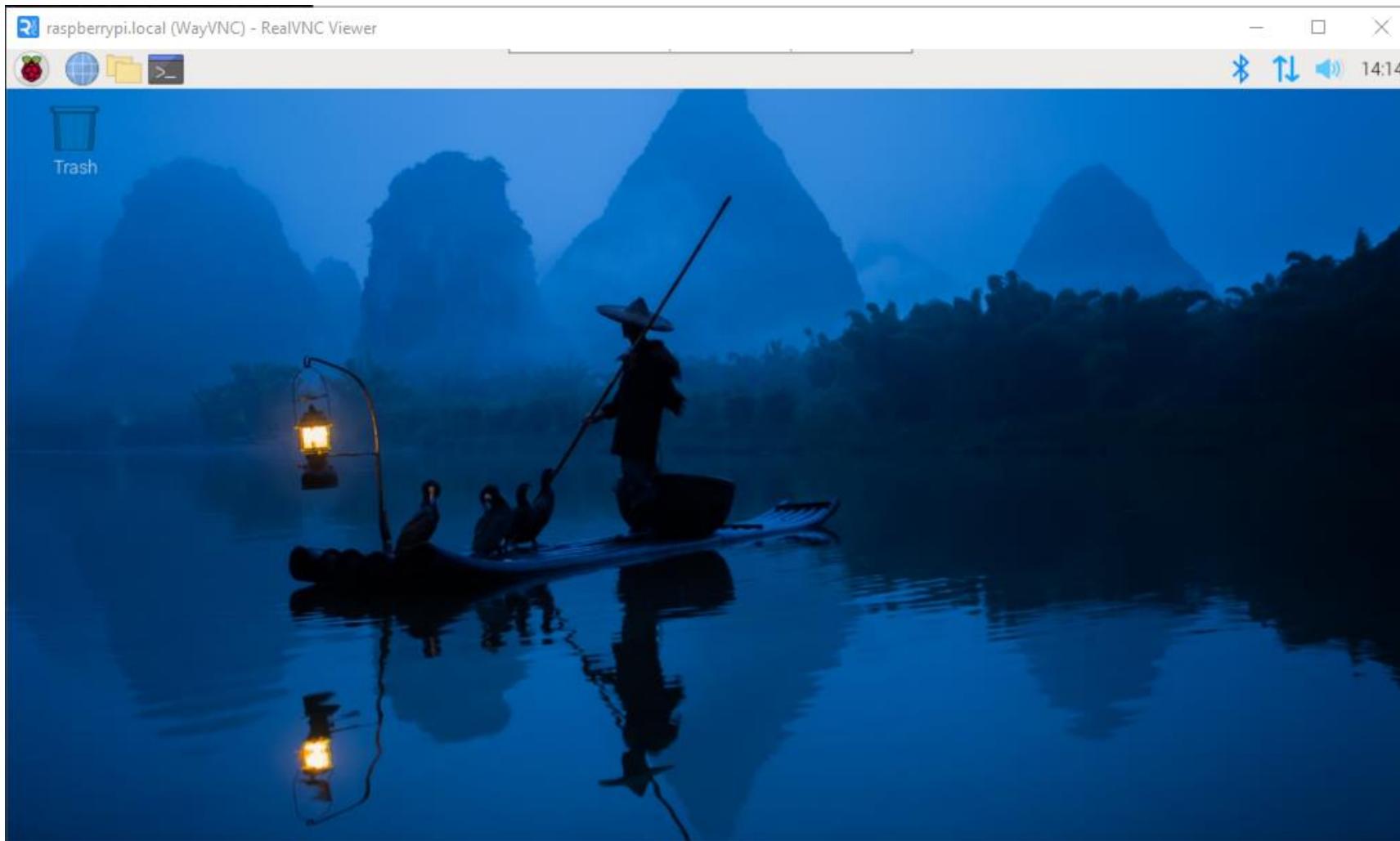
Username: [Pi username]
Password: iotfun2023

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

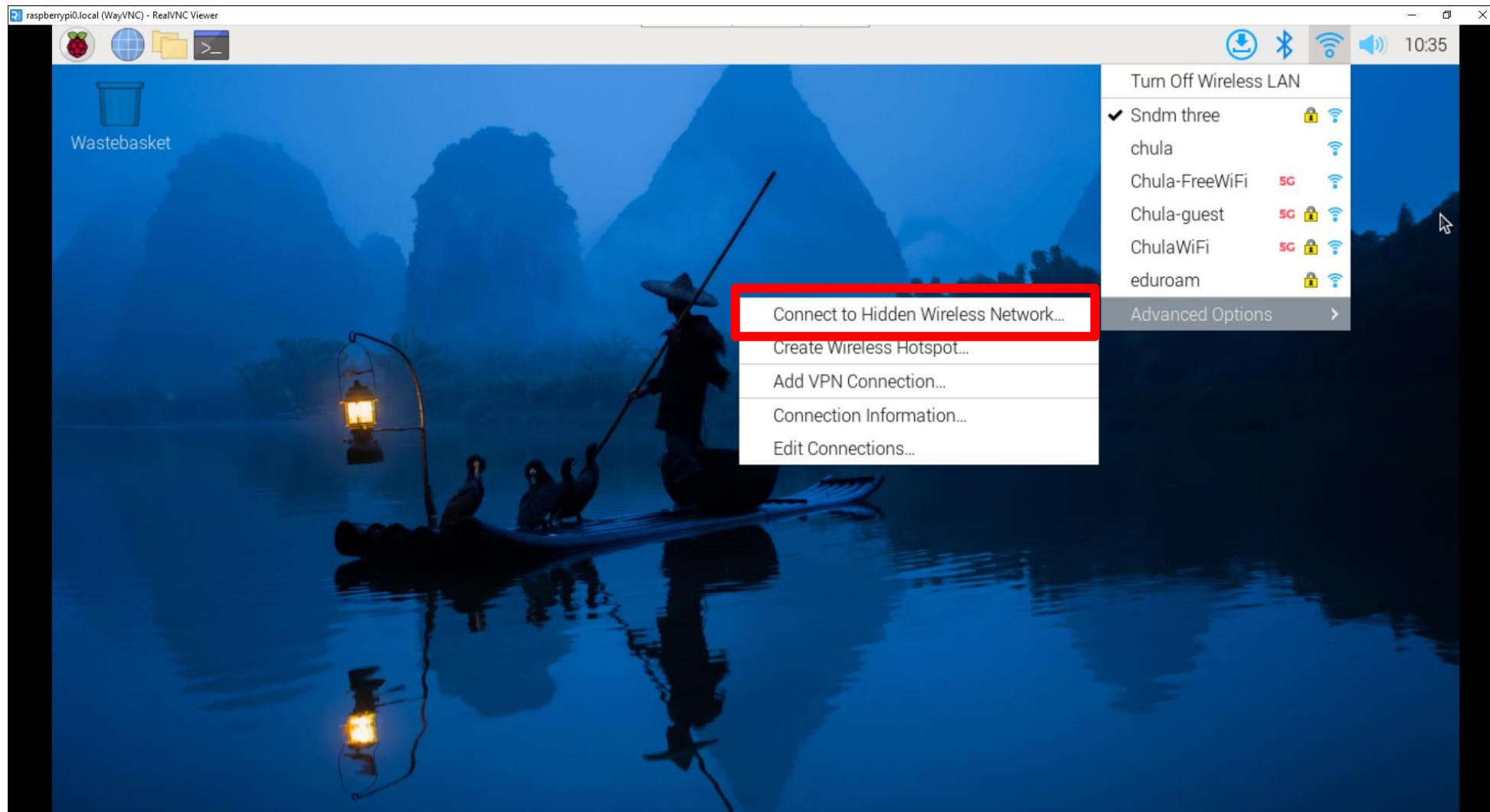


VNC Viewer

You can now access to Pi screen



Connect to your Mobile Hotspot



Connect to your Mobile Hotspot



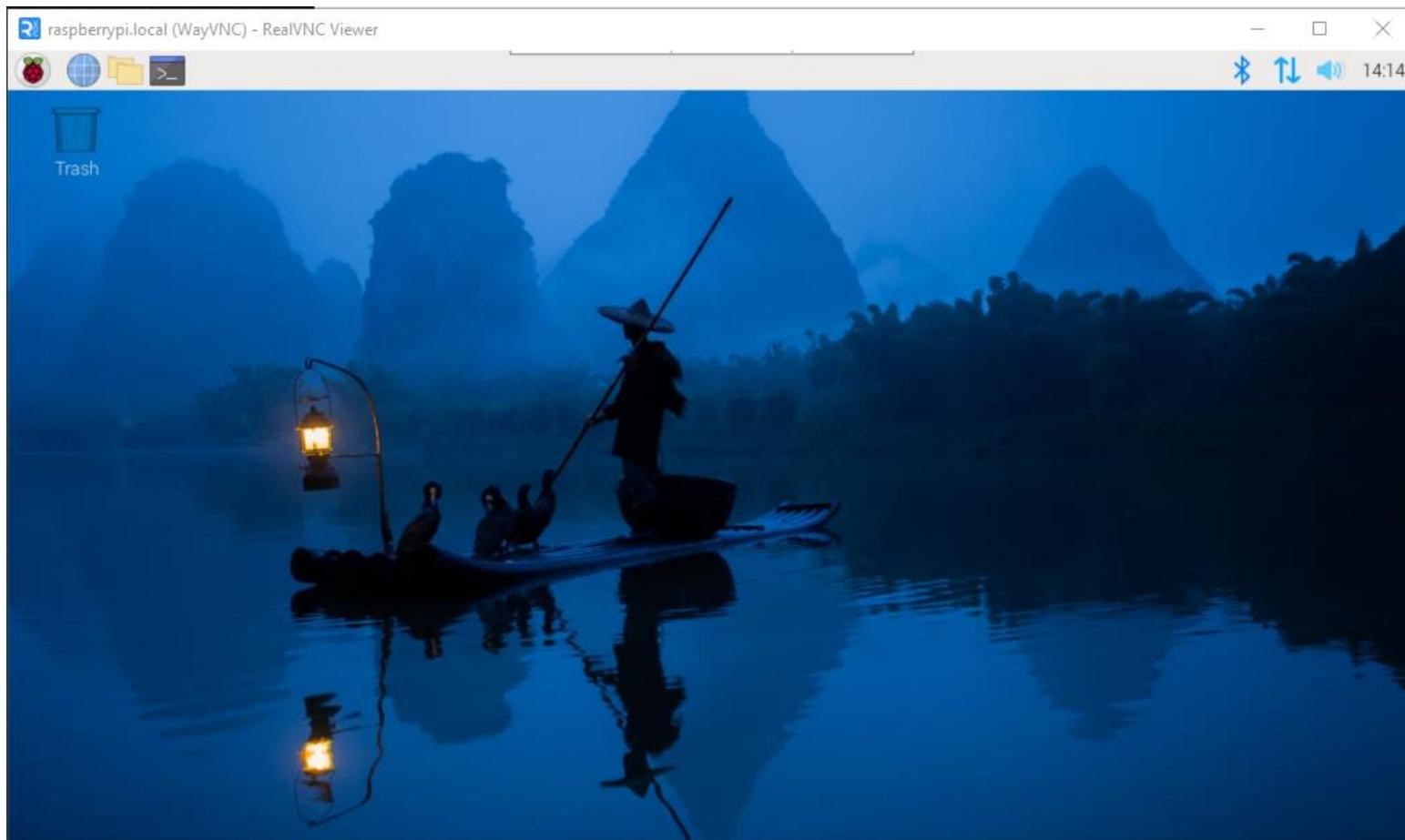
This Photo by Unknown Author is
licensed under [CC BY-SA-NC](#)

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 title '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 WiFi signal icon above it, and a caption: '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.

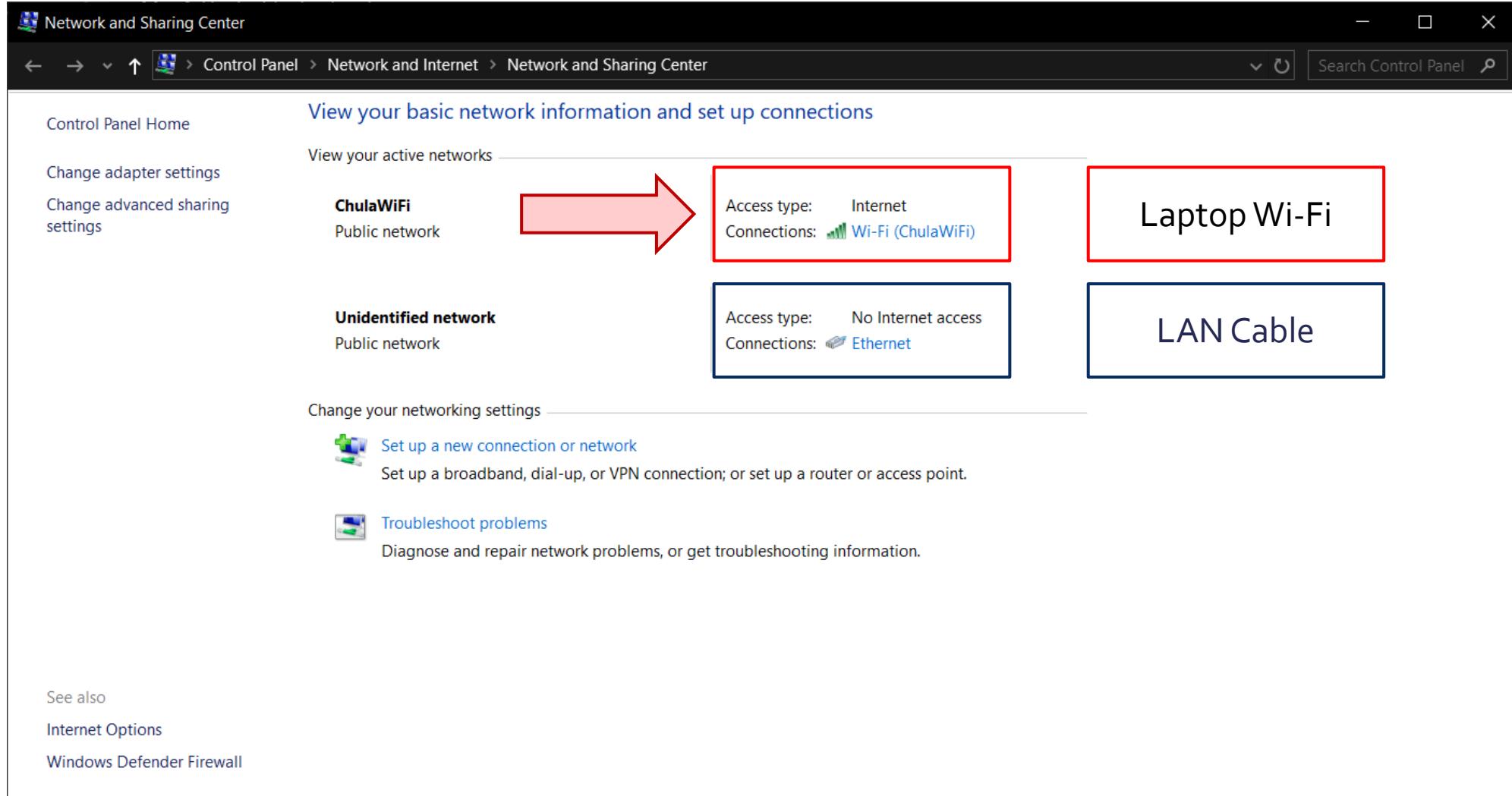
Group 1-3
TP-LINK_E0BE

Group 4-6
TP-Link_735E

Group 7-9
TP-Link_8A28

(Optional) Setup Network for Raspberry Pi

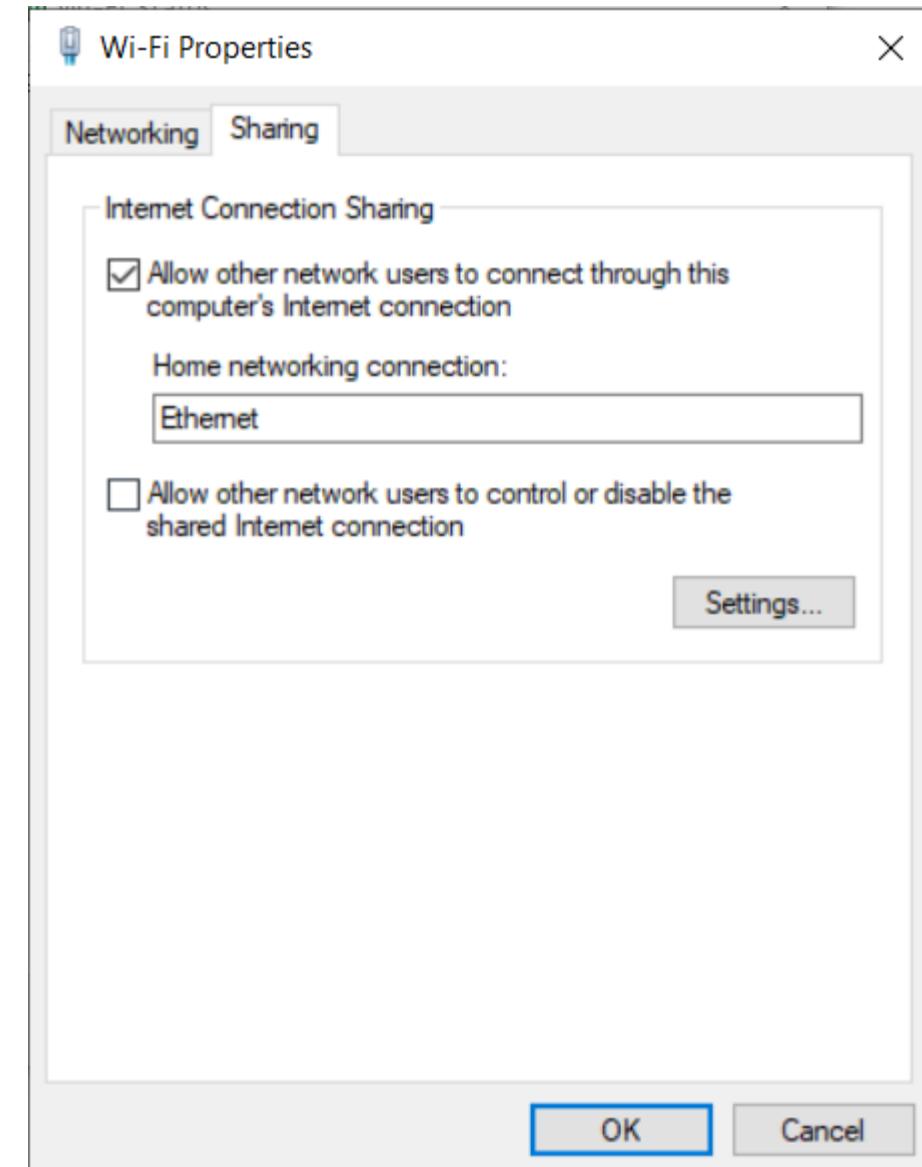
For those who connect LAN using LAN Cable only!



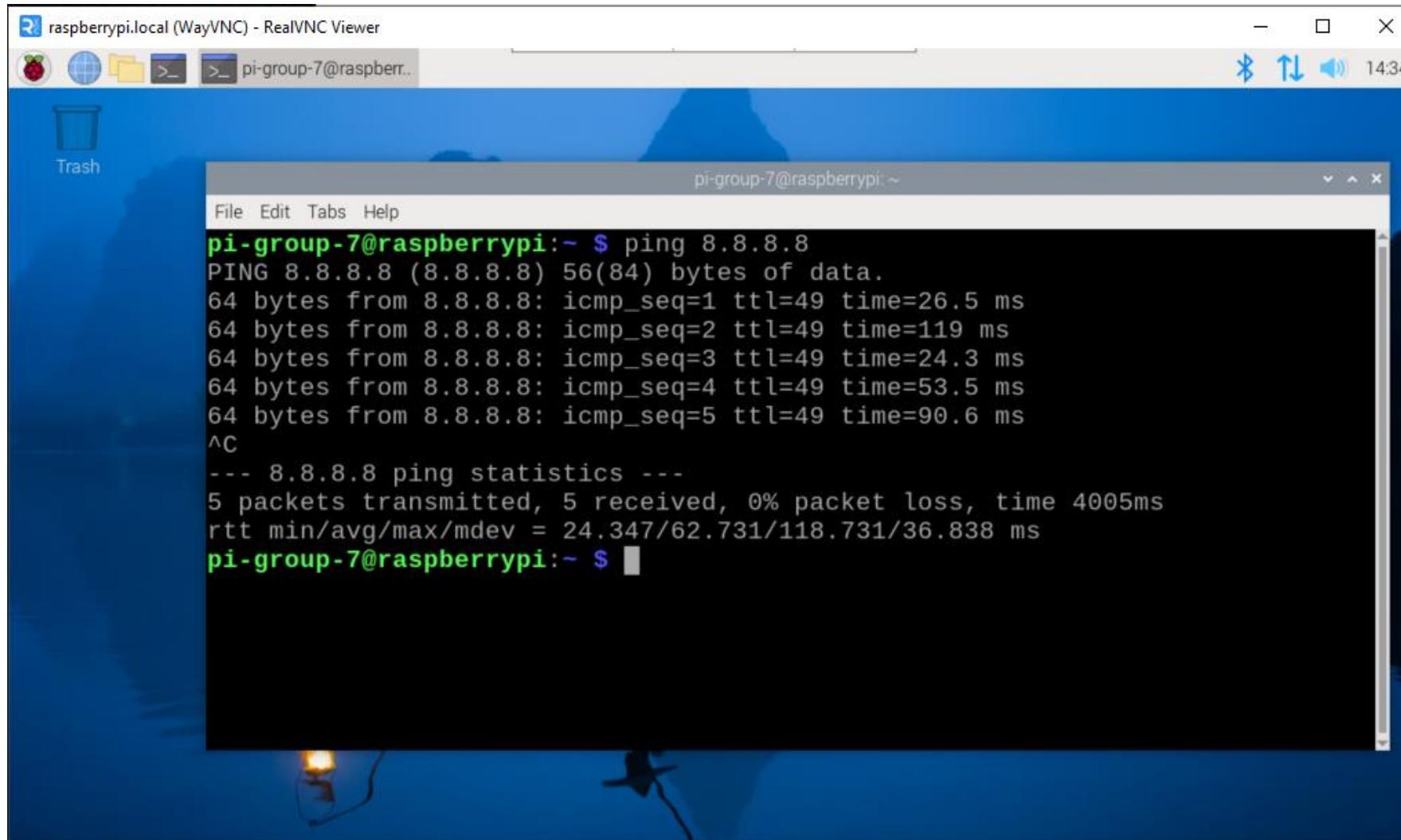
(Optional) Setup Network for Raspberry Pi

For those who connect LAN using LAN Cable only!

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



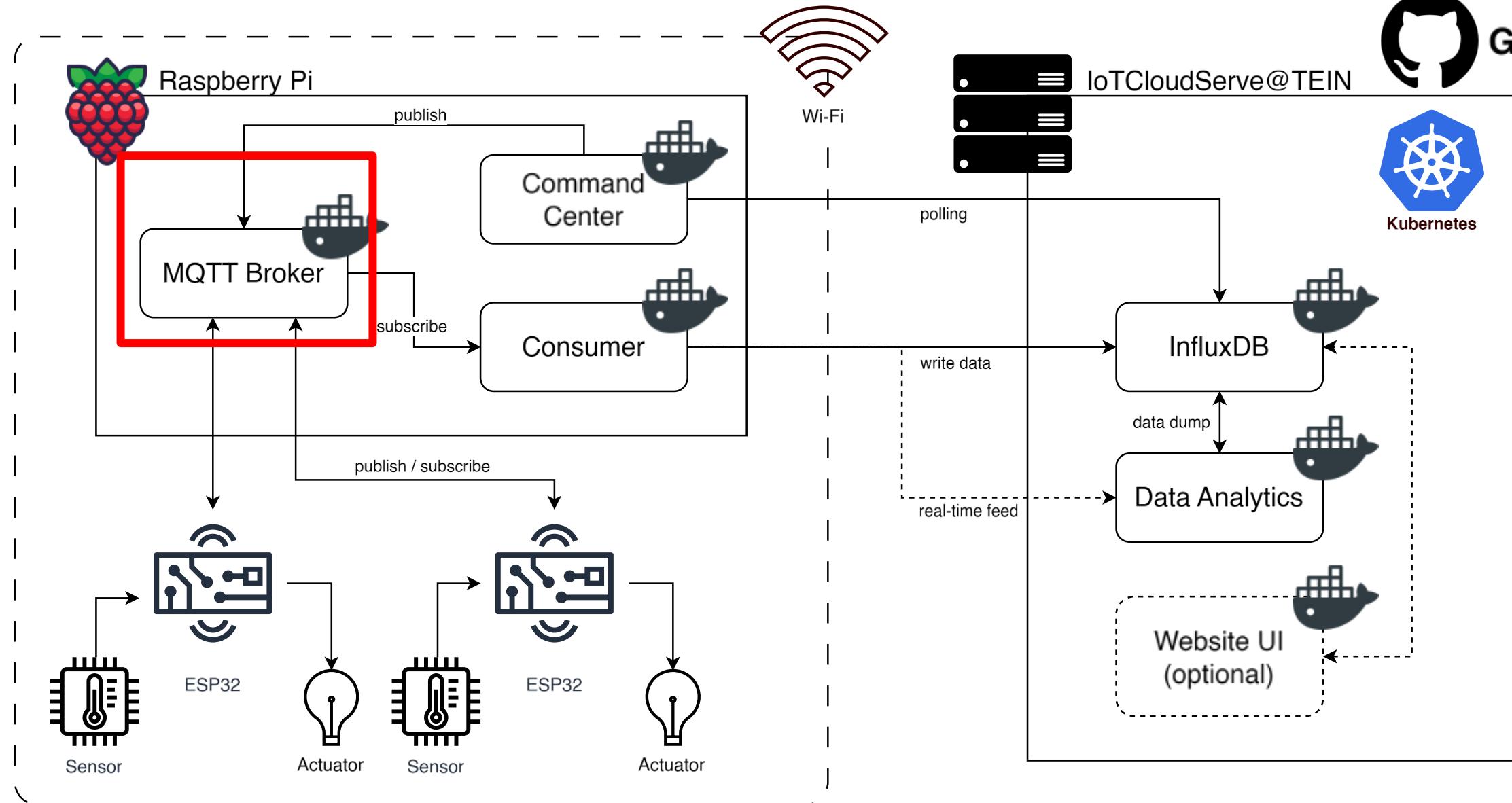
Setup Network for Raspberry Pi



Pi now can
connect to
the internet



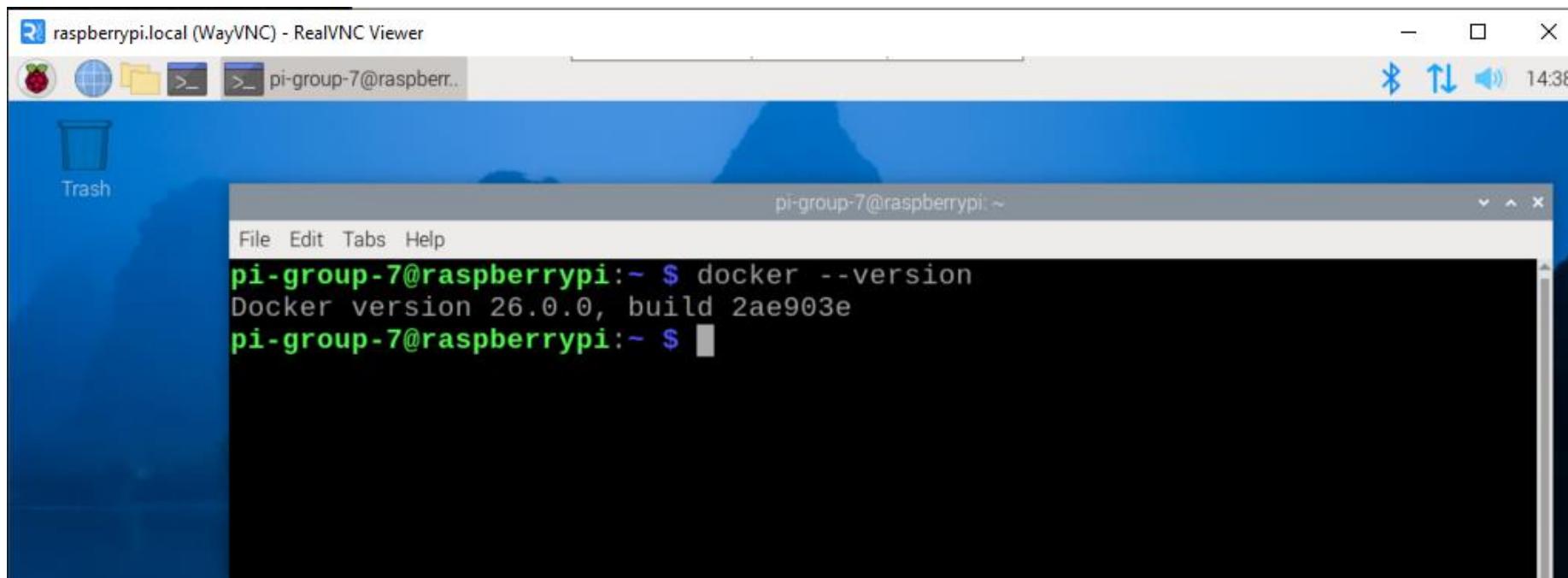
GitHub



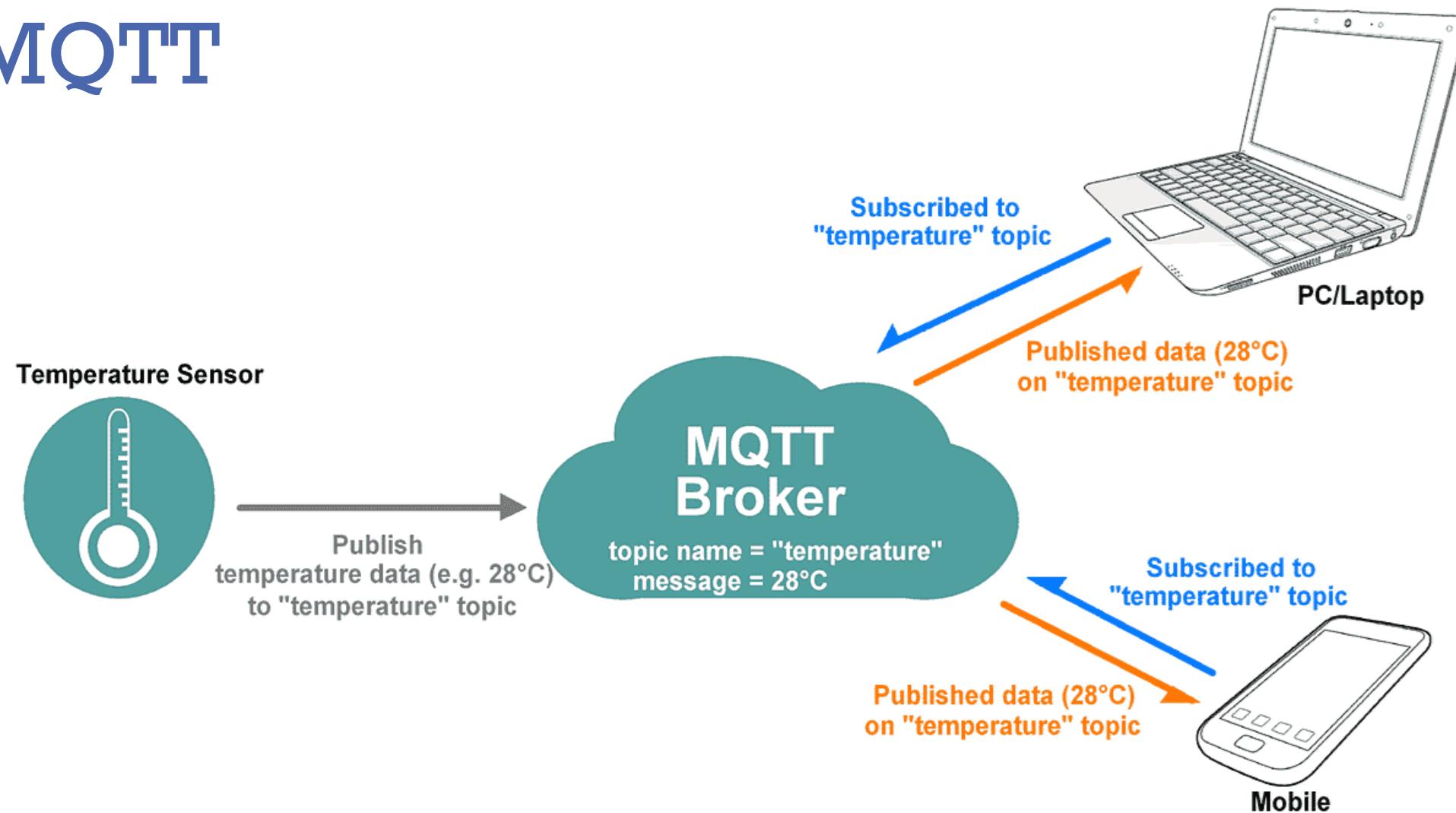
Install Docker on Raspberry Pi

<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
```

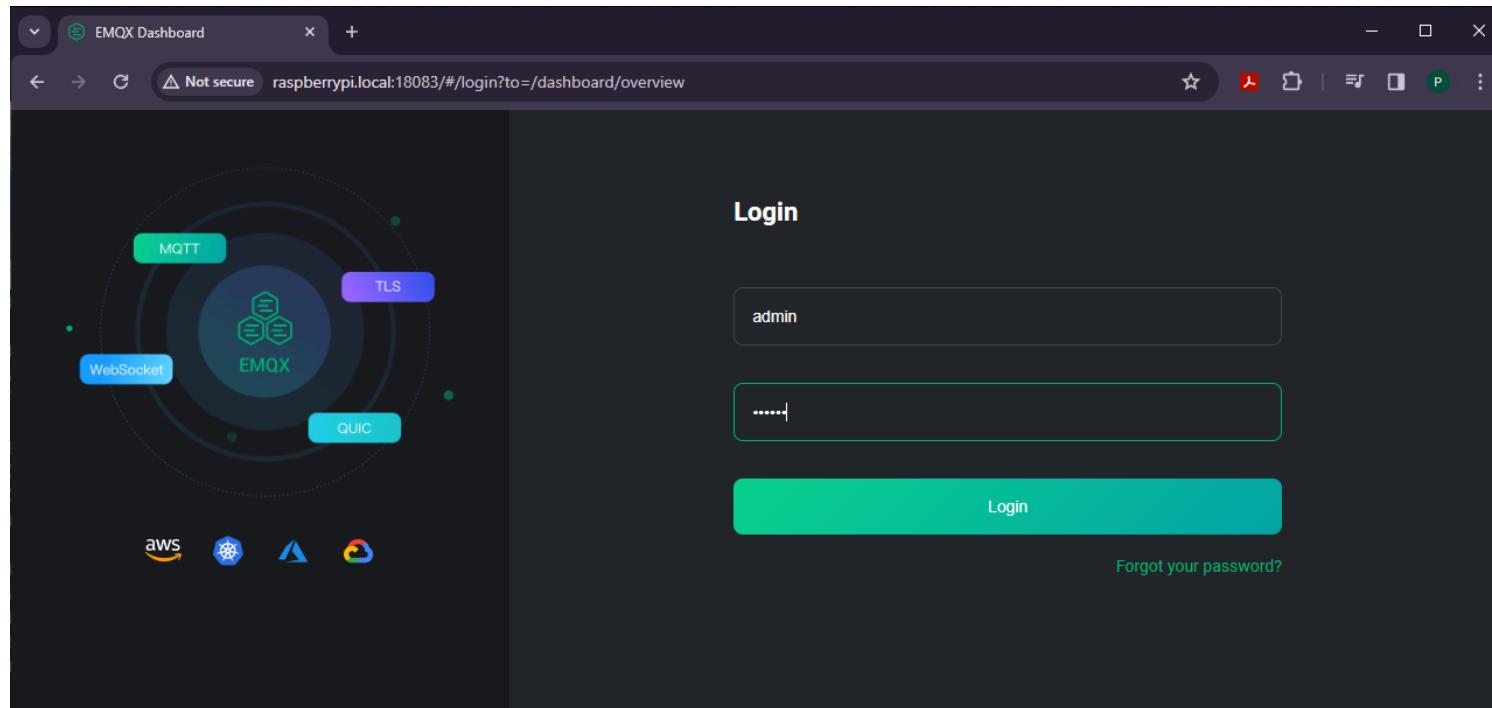
We will use port 1883 for the MQTT connection and port 18083 for the MQTT Broker management

Deploy MQTT Broker

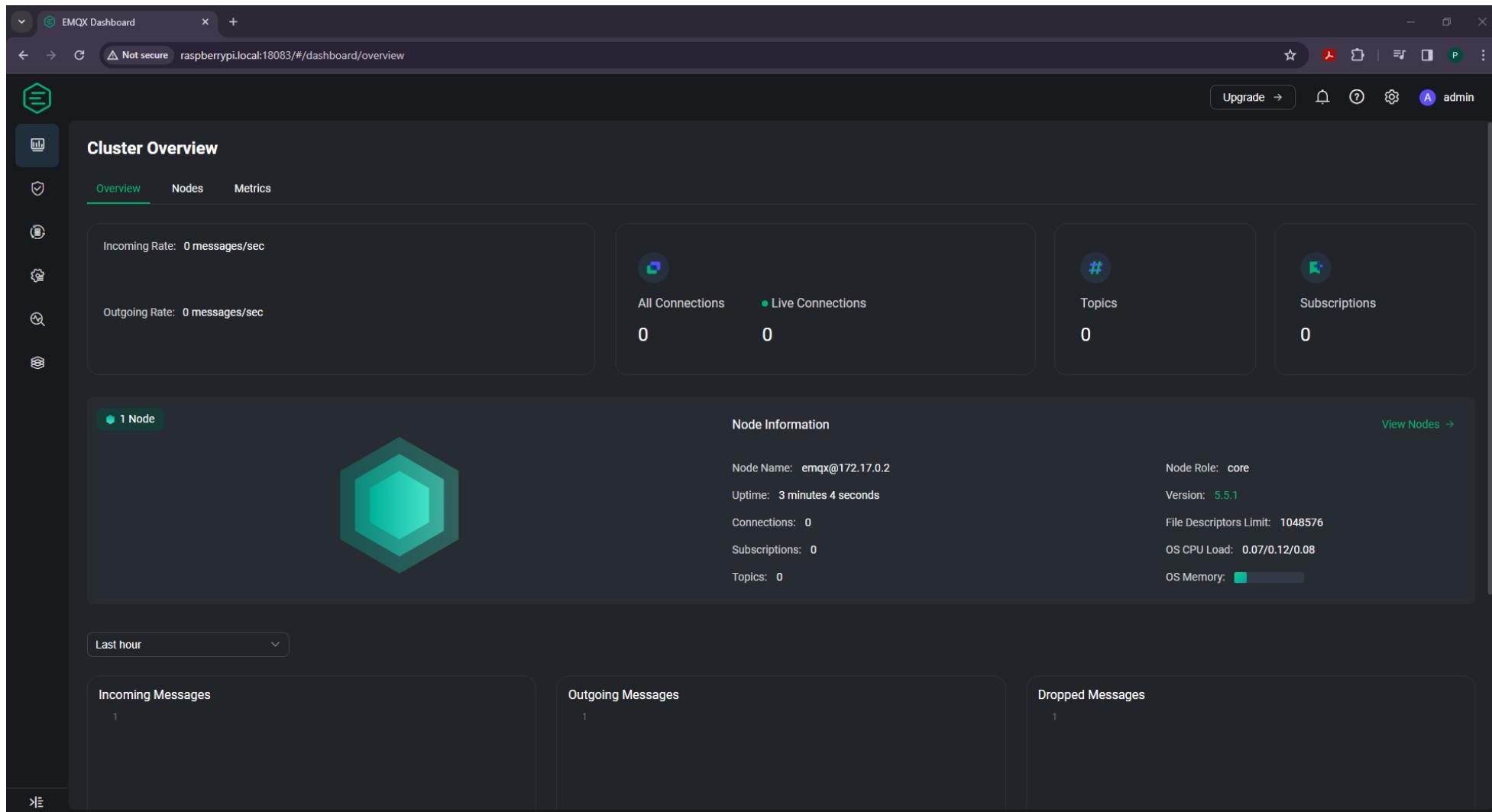
Go to “raspberrypi**1**.local:18083” and login

Username: admin

Password: public



Deploy MQTT Broker



The screenshot shows the EMQX Dashboard interface, specifically the 'Overview' page. The top navigation bar includes tabs for 'Overview', 'Nodes', and 'Metrics'. On the left, there's a sidebar with icons for 'Overview', 'Nodes', 'Metrics', 'Logs', 'Metrics', 'Topics', and 'Subscriptions'. The main content area features several key metrics and a central node summary.

Cluster Overview

- Incoming Rate:** 0 messages/sec
- Outgoing Rate:** 0 messages/sec
- All Connections:** 0 (● Live Connections: 0)
- Topics:** 0
- Subscriptions:** 0

Node Information (1 Node)

Node Name	Node Role
emqx@172.17.0.2	core

Uptime: 3 minutes 4 seconds
Connections: 0
Subscriptions: 0
Topics: 0
Version: 5.5.1
File Descriptors Limit: 1048576
OS CPU Load: 0.07/0.12/0.08
OS Memory:

Metrics (Last hour)

- Incoming Messages:** 1
- Outgoing Messages:** 1
- Dropped Messages:** 1

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

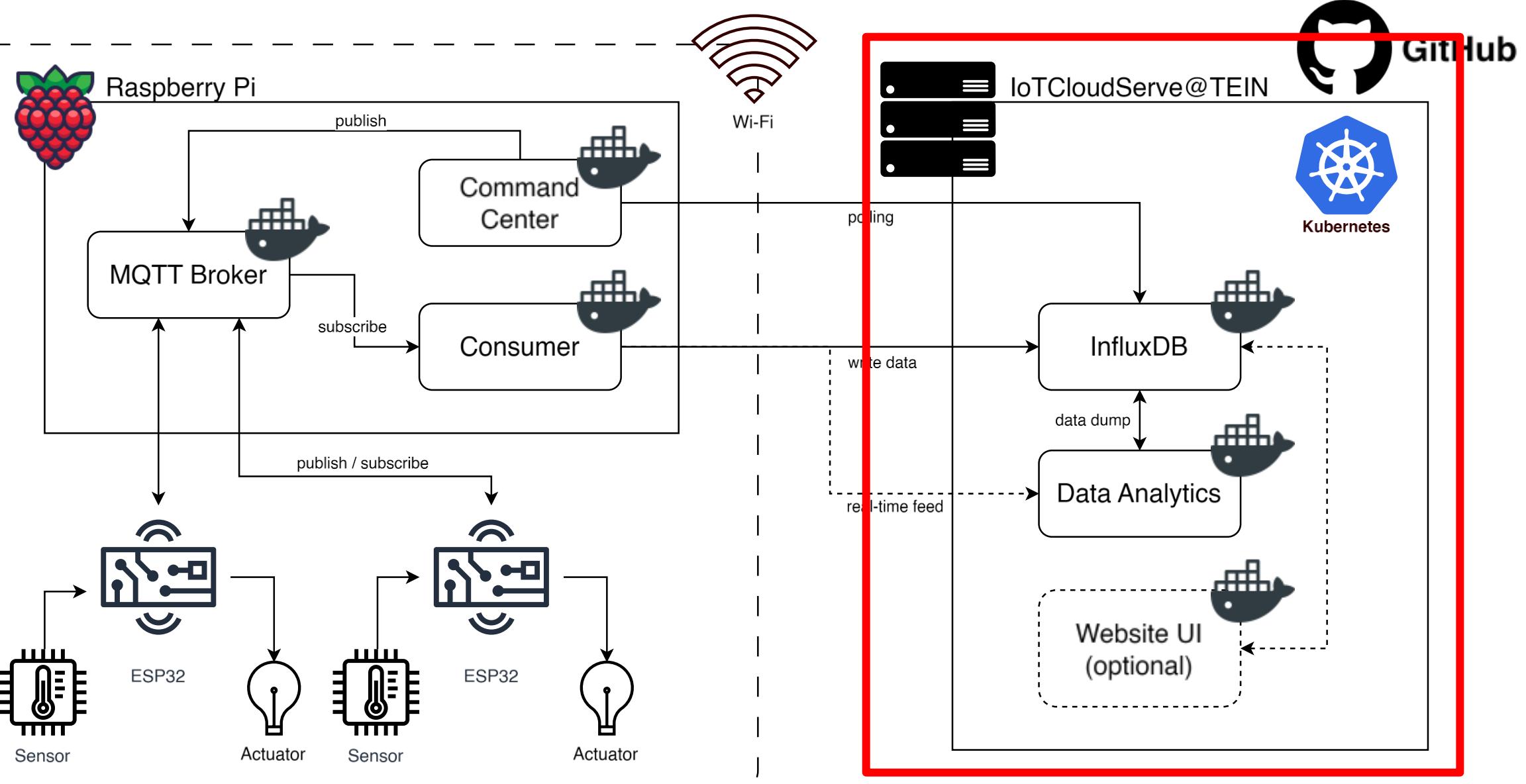
Test MQTT Broker using Postman

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 (with "iot-cloud-workshop" expanded), Environments (empty), History (empty), and a main area for API requests. The "iot-cloud-workshop" collection contains several items: "GET Status", "GET Get Data", "POST Post Data", and "iot-cloud-workshop-mqtt". The "iot-cloud-workshop-mqtt" item is selected, showing its details. The "Request" tab is active, displaying the topic "raspberrypi0.local" and version V3. 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. A status indicator on the right says "Subscribed to 1 topic" and "Connected". The bottom navigation bar includes Online, Find and replace, Console, and various utility buttons.

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), RaspberryPi.
 - Environments: No environment selected.
 - History: No history items.
- Request Tab:** Shows a connection to "iot-cloud-workshop-mqtt / RaspberryPi" at "raspberrypi0.local" (V3). Status: Connected. Subscribed to 1 topic.
- Response Tab:** Shows the following messages:
 - Downward arrow icon: @msg/data {"data":42} 15:31:45
 - Upward arrow icon: @msg/data {"data":42} 15:31:45
 - Information 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.



Kubernetes

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



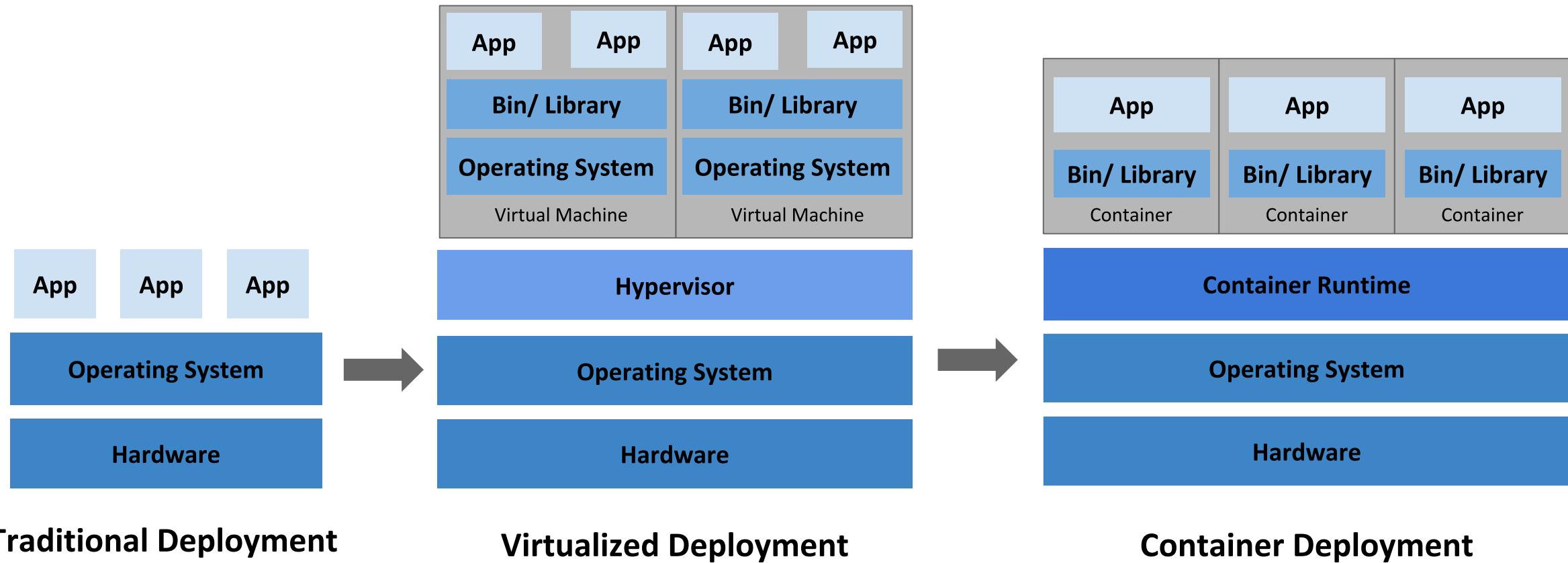
kubernetes

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](#)

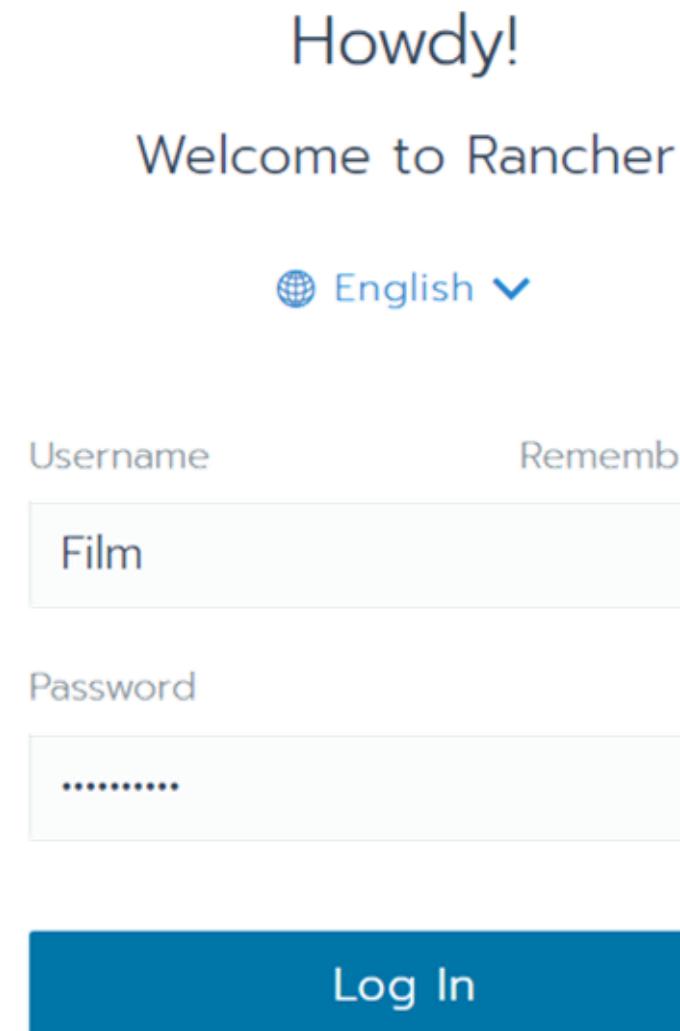
Rancher is a container management platform built for organizations that deploy containers in production. Rancher makes it easy to run Kubernetes everywhere

Rancher

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

Username: **iot-group-1**

Password: [Student ID of
Edge & Cloud operators]
e.g. 6230361921



The image shows 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 form has fields for "Username" (containing "Film") and "Password" (containing "....."). There is also a "Remember" checkbox. A large blue "Log In" button is at the bottom.

Howdy!

Welcome to Rancher

English ▾

Username

Remember

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 navigation menu (Global, Clusters, Apps, Users, Settings, Security, Tools), and a cluster selection dropdown. Below the header is a sub-header with a blue bull icon and a purple square icon.

The main content area is titled "Clusters" and features a table listing a single cluster named "iotcloudserve". The table columns are: State (checkbox, Active), Cluster Name (dropdown, currently set to "iotcloudserve"), Provider (dropdown, Imported v1.20.15+k3s1), Nodes (0), CPU (0.3/24 Cores, 1%), and RAM (0.1/93.7 GiB, 0%). There are "Delete" and "Add Cluster" buttons at the top of the table, and a "Search" input field.

At the bottom of the page, there is a footer bar with links: v2.3.11-rc2, Help & Docs, Forums, Slack, File an Issue, English (dropdown), and Download CLI (dropdown). The page number 51 is also visible in the bottom right corner.

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%

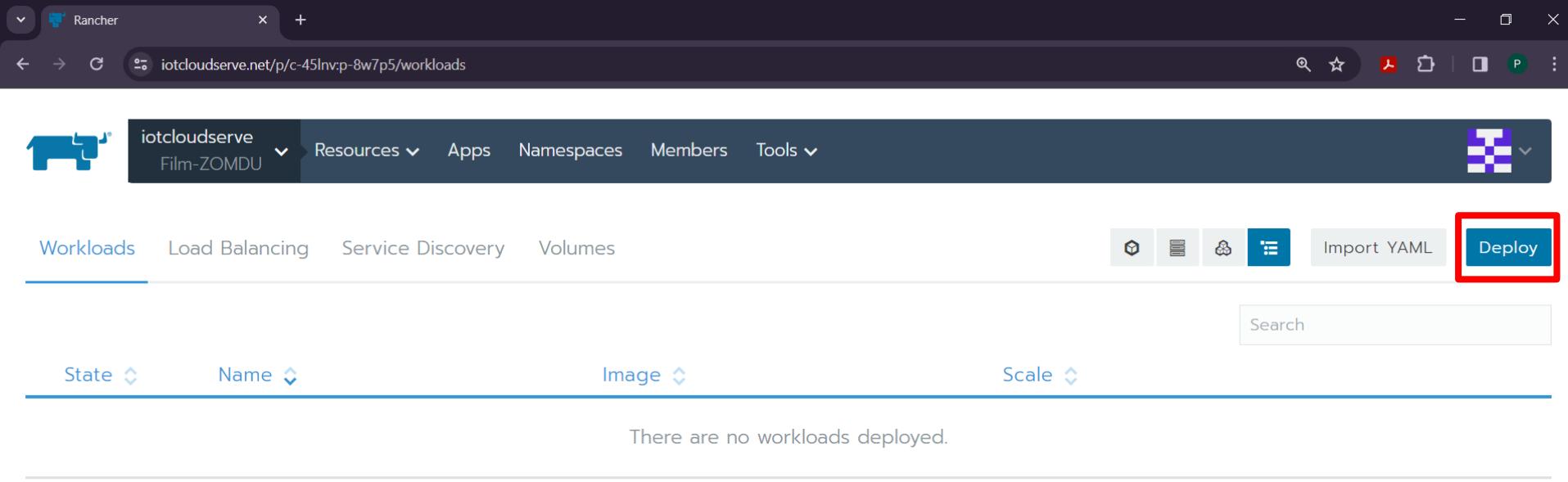
Rancher

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, and Download CLI.

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 the Rancher logo, the namespace "iotcloudserve Film-ZOMDU", and links for Resources, Apps, Namespaces, Members, and Tools. On the right side of the top bar are icons for search, star, refresh, and more. Below the top bar, there are tabs for Workloads, Load Balancing, Service Discovery, and Volumes, with "Workloads" being the active tab. To the right of these tabs are icons for creating a new workload, importing YAML, and a prominent blue "Deploy" button, which is highlighted with a red box. A "Search" input field is also present. The main content area displays the message "There are no workloads deployed." 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 language selection, and a Download CLI link.

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

English Download CLI

Rancher – Deploy InfluxDB

Docker image: https://hub.docker.com/_/influxdb

The screenshot shows the Docker Hub page for the `influxdb` image. At the top, there's a logo for `influxdata`, the name `influxdb`, and a badge indicating it's a Docker Official Image with over 1B+ pulls and 1.9K stars. Below this, a brief description states: "InfluxDB is the open source time series database built for real-time analytic workloads." To the right, there's a button to "docker pull influxdb" with a "Copy" link next to it. The page has two tabs: "Overview" (which is selected) and "Tags". The "Overview" section contains a note about the description being trimmed, links to GitHub and Docker feedback pages, and a "Quick reference" section with links to maintainers and help resources. The "Tags" section lists recent tags like 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, and 1.8-alpine. The "About Official Images" section explains that Docker Official Images are curated sets of Docker open source and drop-in solution repositories.

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](#)

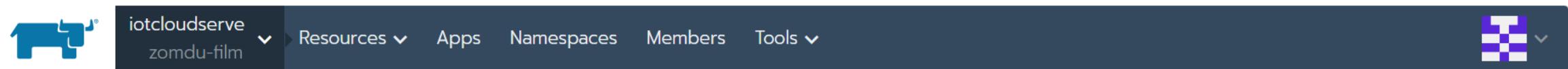
Recent Tags

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)

+ Add Port

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

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

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. The top 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' in the 'film-zomdu' namespace. Key details shown include:

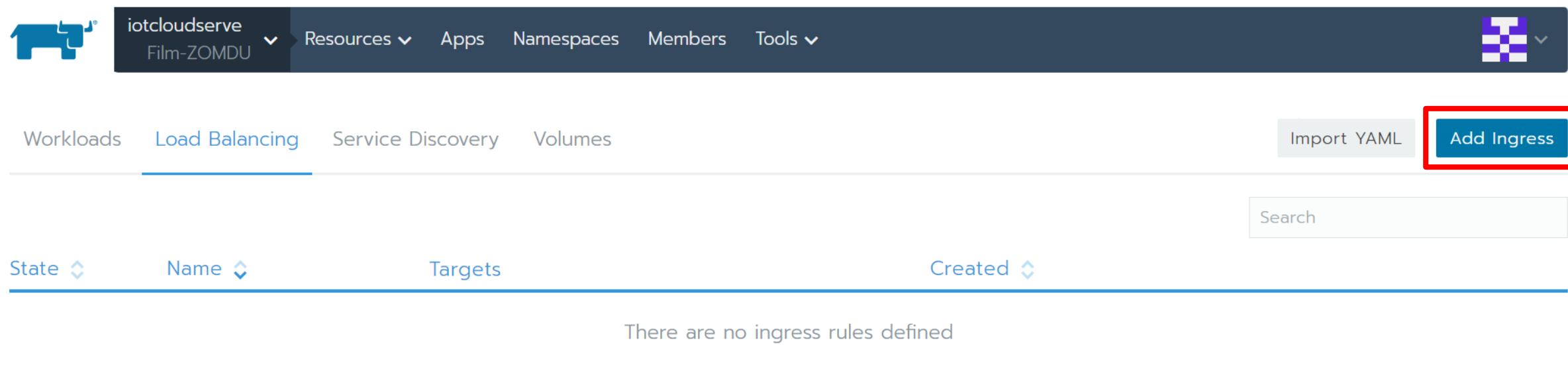
- Namespace:** film-zomdu
- Image:** influxdb:2.7.5
- Workload Type:** Deployment
- Endpoints:** n/a
- Config Scale:** 1 (Ready Scale: 1)
- Created:** 8:40 PM
- Pod Restarts:** 0

An 'Expand All' button is located at the bottom right of the workload summary. Below this, a 'Pods' section lists the single pod associated with the workload. The pod details are as follows:

Action	Value
Download YAML	Download YAML
Delete	Delete
State	Running
Name	film-influxdb-59cc8db8b5-g56j4
Image	influxdb:2.7.5 10.42.2.111 / Created a few seconds ago / Restart...
Node	[...]

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's 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 gear 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 are filter options for "State" and "Name", and columns for "Targets" and "Created". A message at the bottom states "There are no ingress rules defined".

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

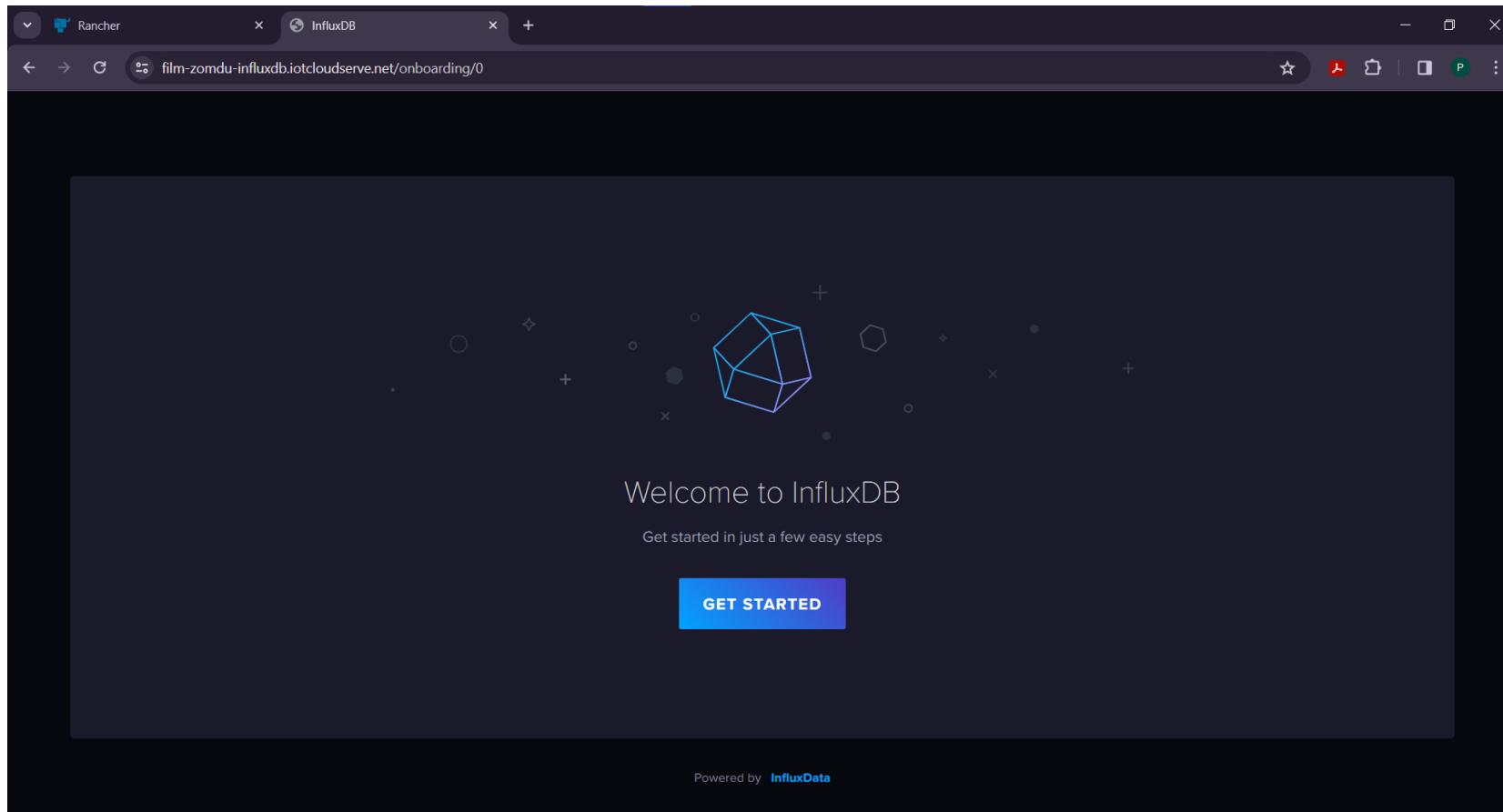
Port *

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 Home, Sources, Buckets, Telegraf, Scrapers, API Tokens (selected), and Notifications. The main content area has a dark header with "Load Data" and tabs for SOURCES, BUCKETS, TELEGRAF, SCRAPERS, and API TOKENS. Below the header are search and sort filters: "Filter Tokens..." and "Sort by Description (A → Z)". A prominent blue button on the right says "+ GENERATE API TOKEN". A single token entry is listed: "film-zomdu's Token" (Owner: film-zomdu), created at 2024-04-02 21:23:09, last modified 3 minutes ago. There are edit and delete icons next to the token name.

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 namespace '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 Reconcilers. The 'Secrets' tab is selected and highlighted in blue. To the right of these tabs are three small icons: a key, a list, and a plus sign. Further to the right is a large blue button labeled 'Add Secret'. Below the tabs, there are filter options for State, Name, Namespace, Keys, and Created, each with a dropdown arrow. A search bar is located on the right side of the filters. At the bottom of the content area, a message states 'There are no secrets defined'.

Rancher – Deploy InfluxDB

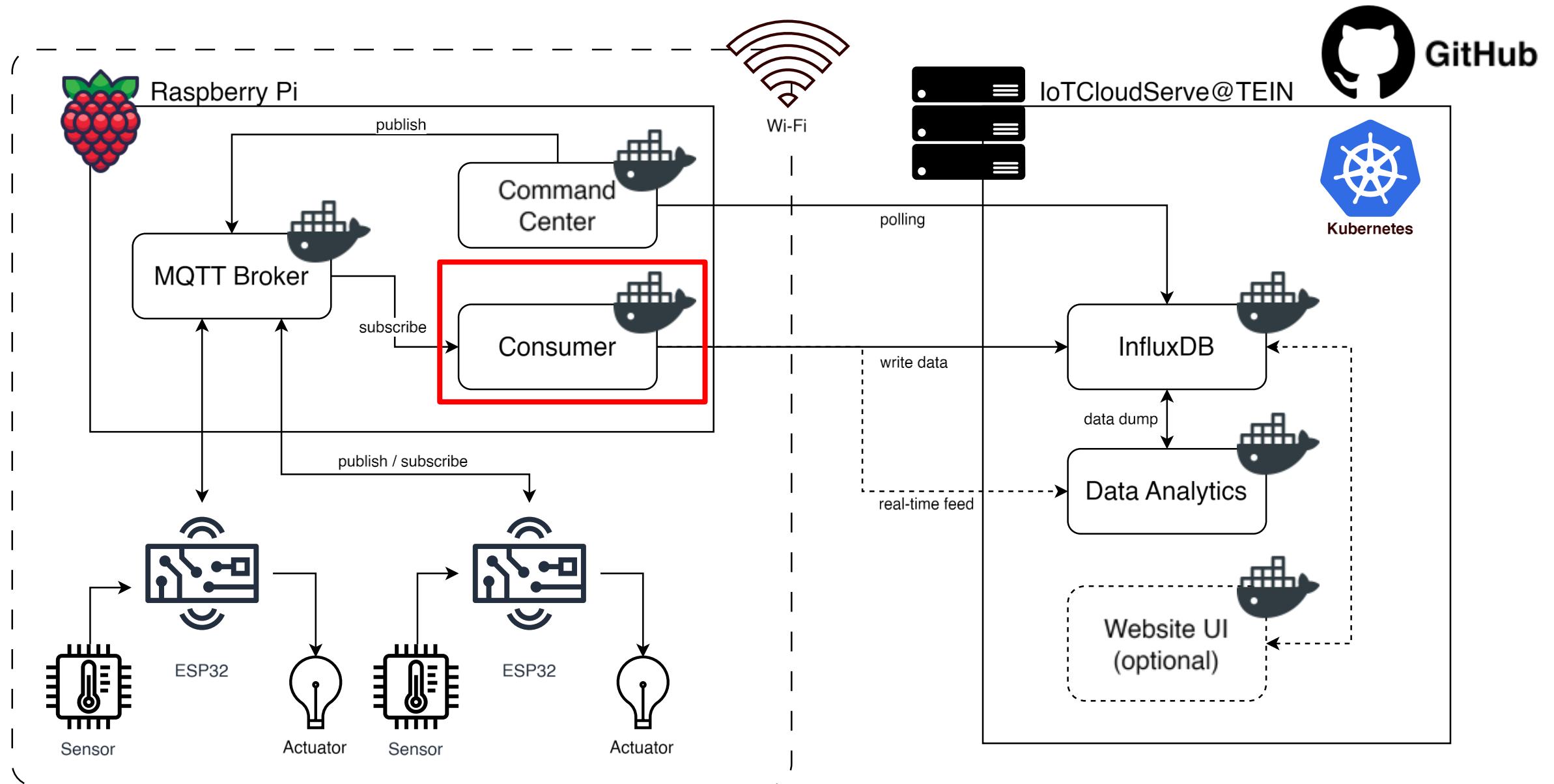
Secrets Values

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

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

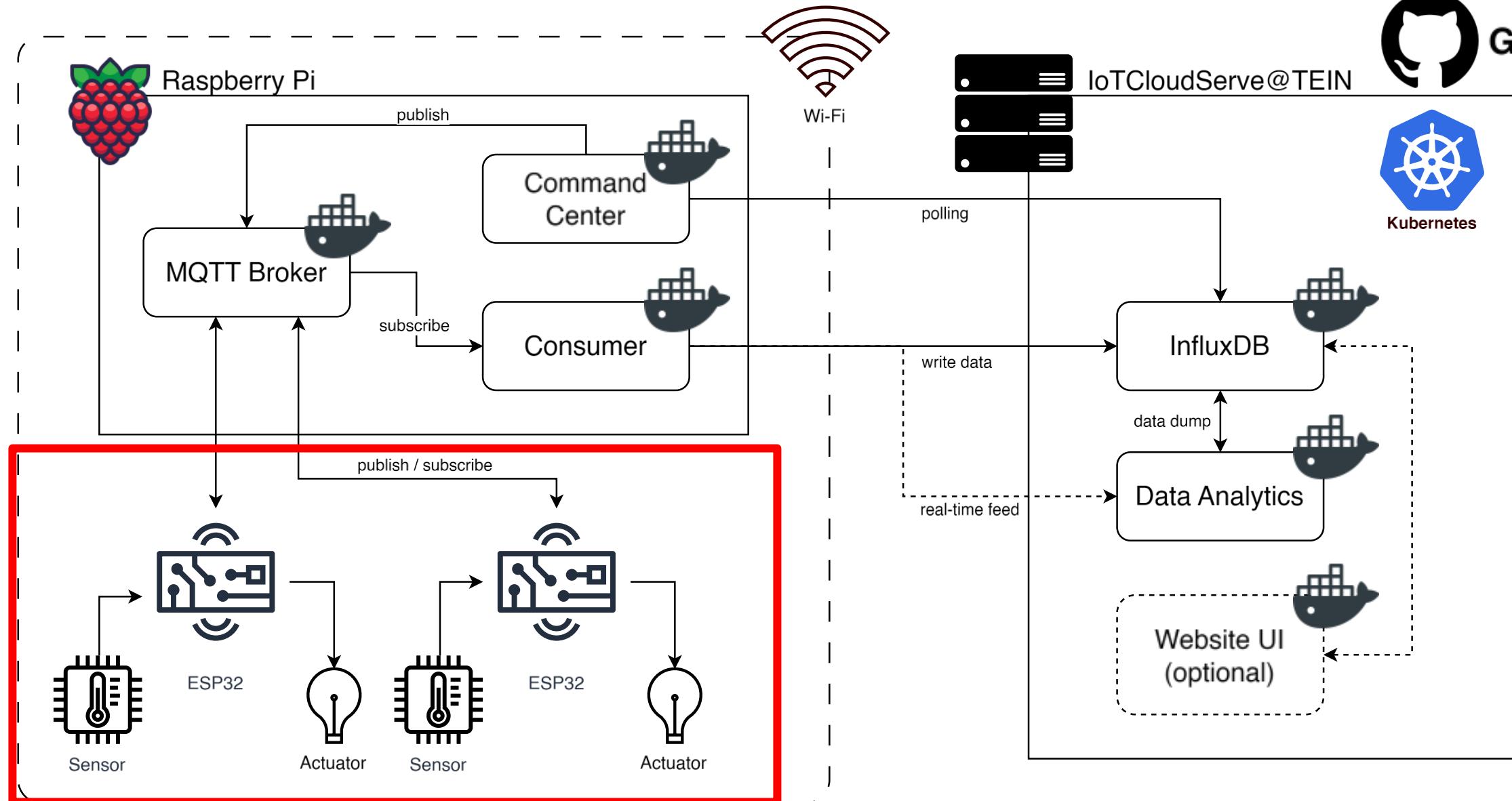
> ifconfig

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:~ $
```



GitHub



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

S1_Temp

S2_Temp

_field

No tag keys found in the current time range

WINDOW PERIOD

CUSTOM AUTO

auto (10s)

Fill missing values

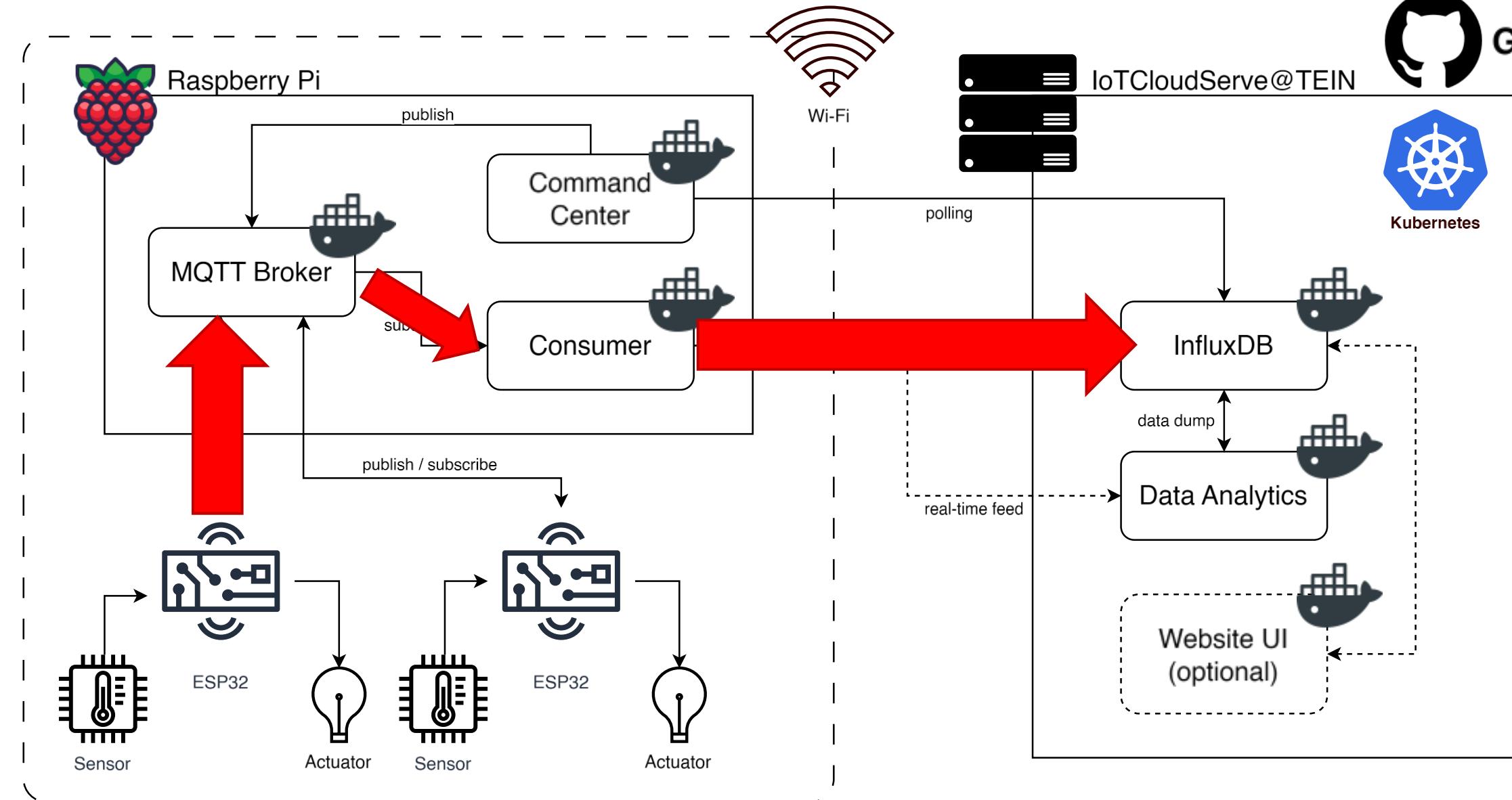
AGGREGATE FUNCTION

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 circular graph. A message says "Looks like you don't have any queries. Be a lot cooler if you did!". Below this is a query editor with fields for "Query 1" and a "+" button. To the right are buttons for "View Raw Data", "CSV", "Past 5m", "SCRIPT EDITOR", and "SUBMIT". The "FROM" section shows a dropdown menu with "Search buckets" and a list including "film-zomdu" (highlighted in blue), "_monitoring", "_tasks", and "+ Create Bucket". The "Filter" section has two dropdown menus: one for "_measurement" (with "S1_Temp" and "S2_Temp" selected) and one for "_field" (with "S2_Light", "S2_Sound", and "S3_Light" listed). A message in the center says "No tag keys found in the current time range". On the right, there are sections for "WINDOW PERIOD" (set to "auto (10s)" and "Fill missing values"), and "AGGREGATE FUNCTION" (set to "CUSTOM" and "AUTO").

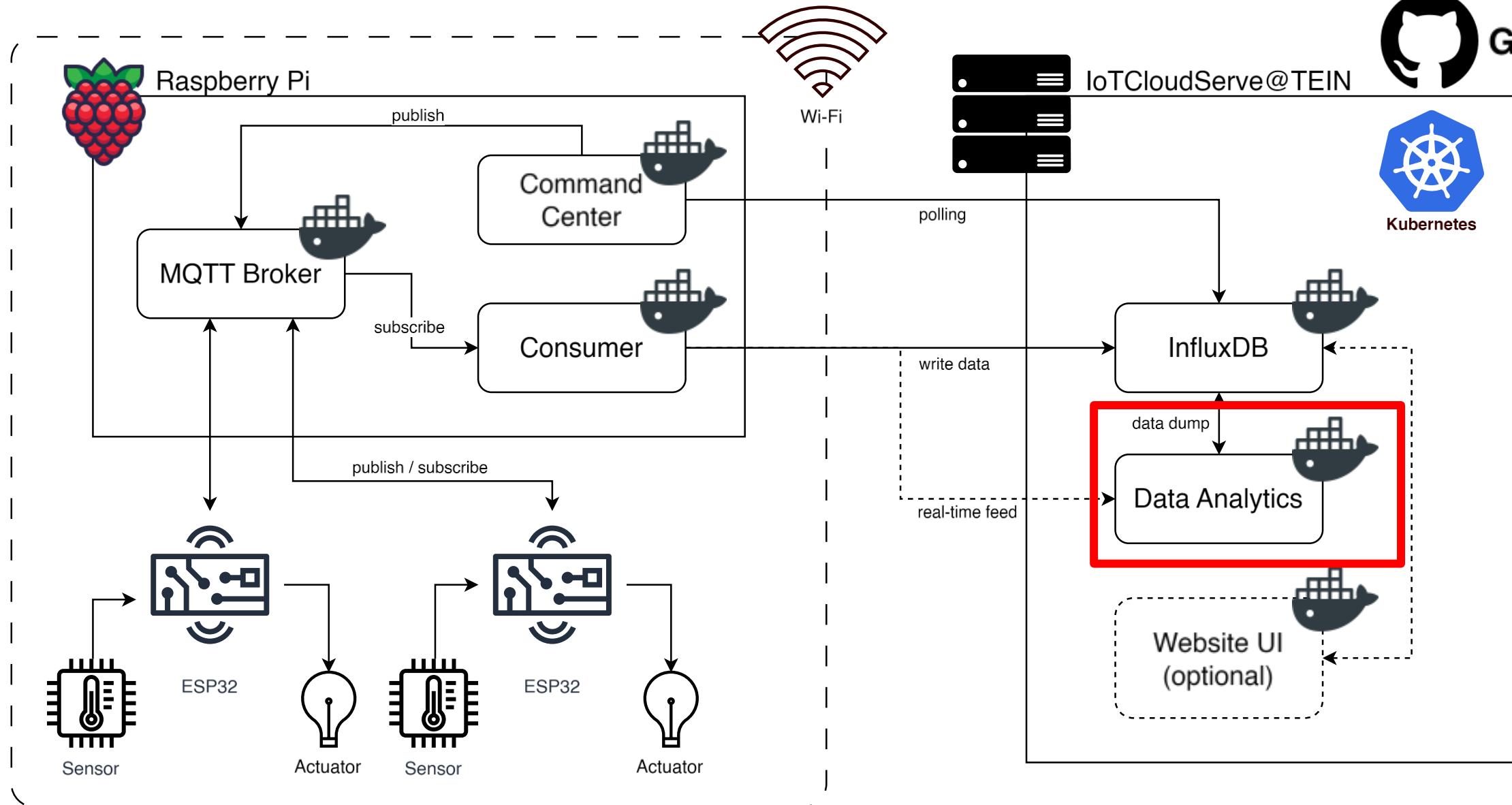


GitHub





GitHub



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

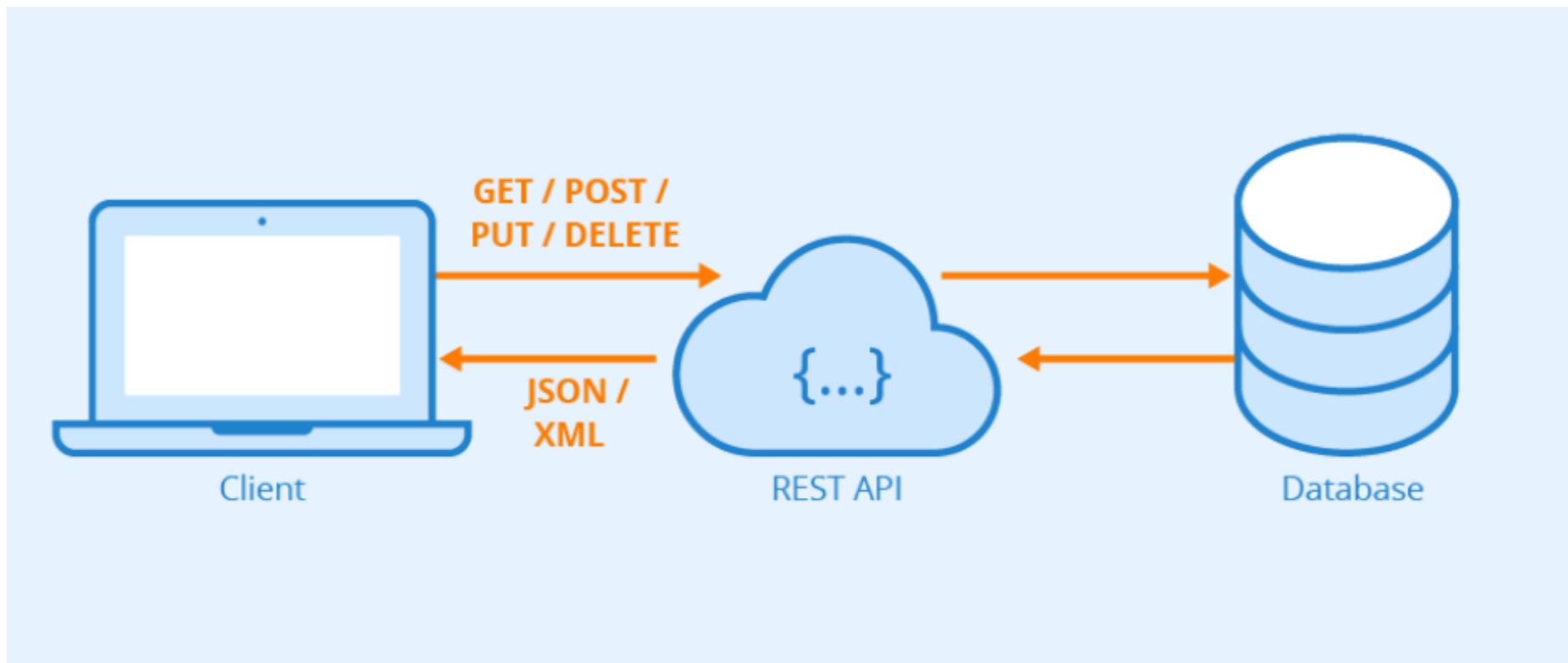
Deploy Model on IoTCloudServe@TEIN

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

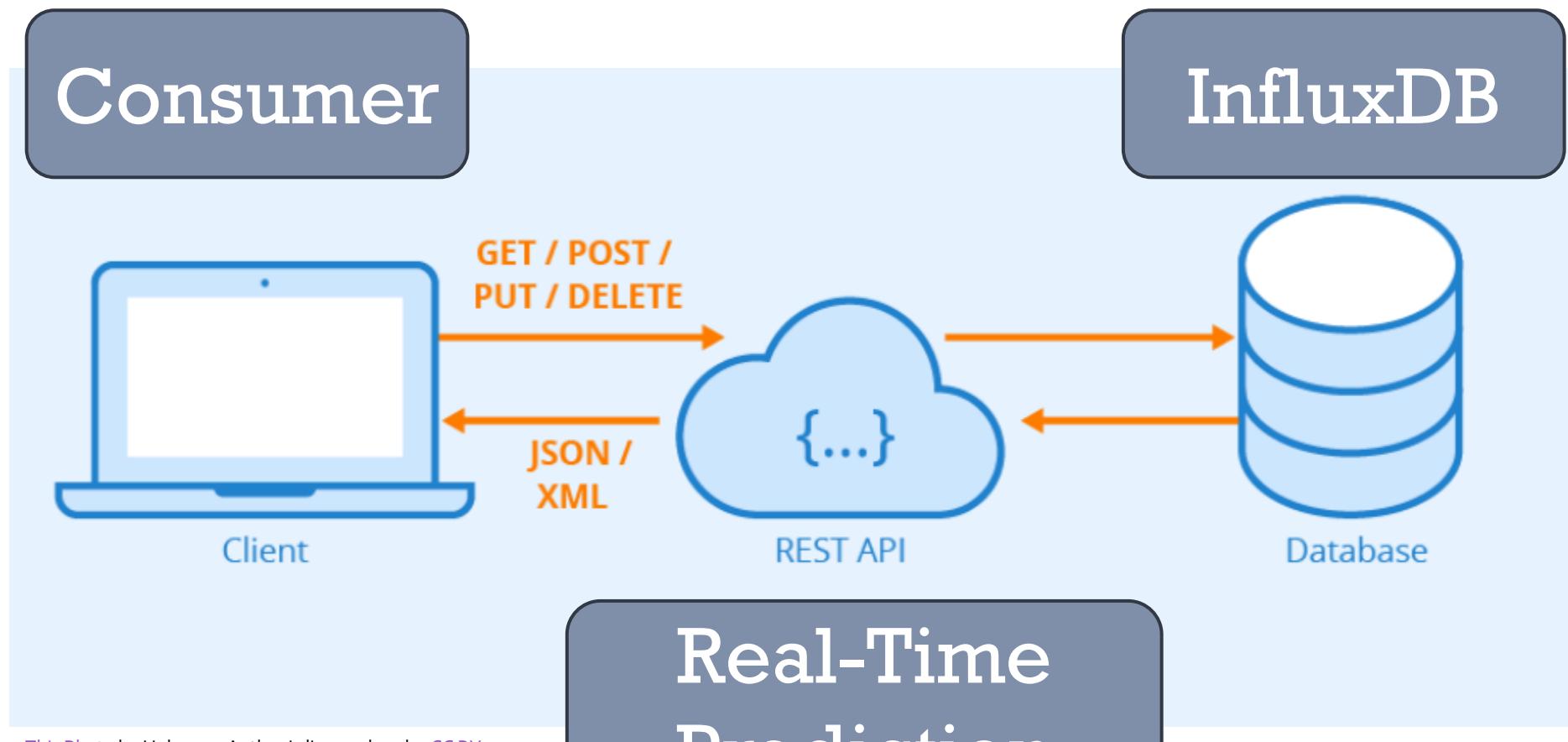
<https://flask.palletsprojects.com/en/3.0.x/>



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

Develop REST API Server using Flask

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



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

Real-Time
Prediction

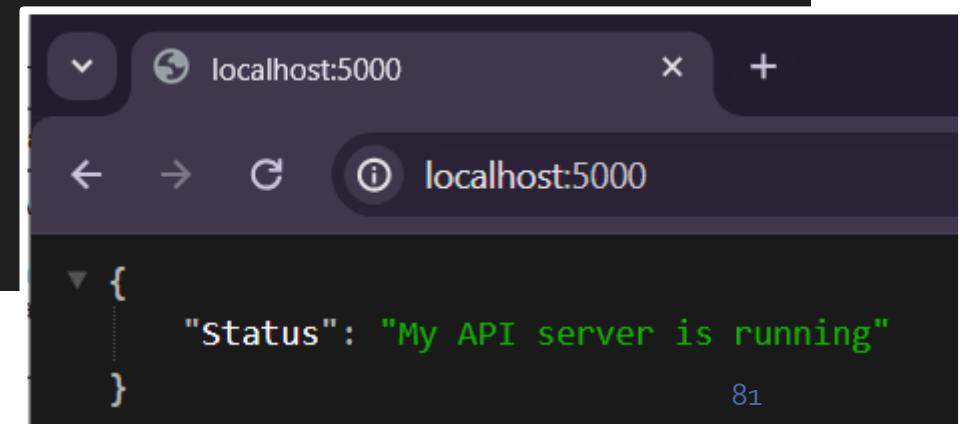
Develop REST API Server using Flask

```
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', 'iot-cloud-workshop' (which contains 'Get Data' and 'Post Data' requests), 'Magel', 'MHE Project', and 'MQTT-NETPIE'. A central modal window titled 'HTTP' provides a brief description of the protocol: 'Hypertext Transfer Protocol (HTTP) is an application-layer protocol often used to build REST APIs. Test your HTTP API with an HTTP request.' Below the modal are icons for GraphQL, gRPC, WebSocket, and Socket.IO. At the bottom of the interface, there are buttons for 'Save', 'Send', and 'Cookies', along with a status bar showing '200 OK 7 ms 191 B' and a 'Save as example' button.

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 tabs for 'GET Get Data', 'POST Post Data', and 'GET Status'. The 'GET Status' tab is active, showing a red box around the URL input field which contains 'http://localhost:5000/'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. Under 'Body', there's a 'Query Params' table with one row: 'Key' (Kav) and 'Value' (Value). The 'Pretty' tab in the results panel is selected, displaying a JSON response:

```
1 {  
2   "Status": "My API server is running"  
3 }
```

The results panel also shows a status of '200 OK' with a response time of '11 ms' and a size of '207 B'. There are buttons for 'Save as example' and other options. At the bottom, there are links for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Trash', and a page number '85'.

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 status bar at the bottom indicates a 200 OK response with 6 ms and 166 B.

GET Get Data

HTTP iot-cloud-workshop / Get Data

GET http://localhost:5000/get_data

1 []

200 OK 6 ms 166 B

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' and '9 ms'.

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`. Below the URL, the 'Body' tab is selected, showing a JSON response:

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

The status bar at the bottom right 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**

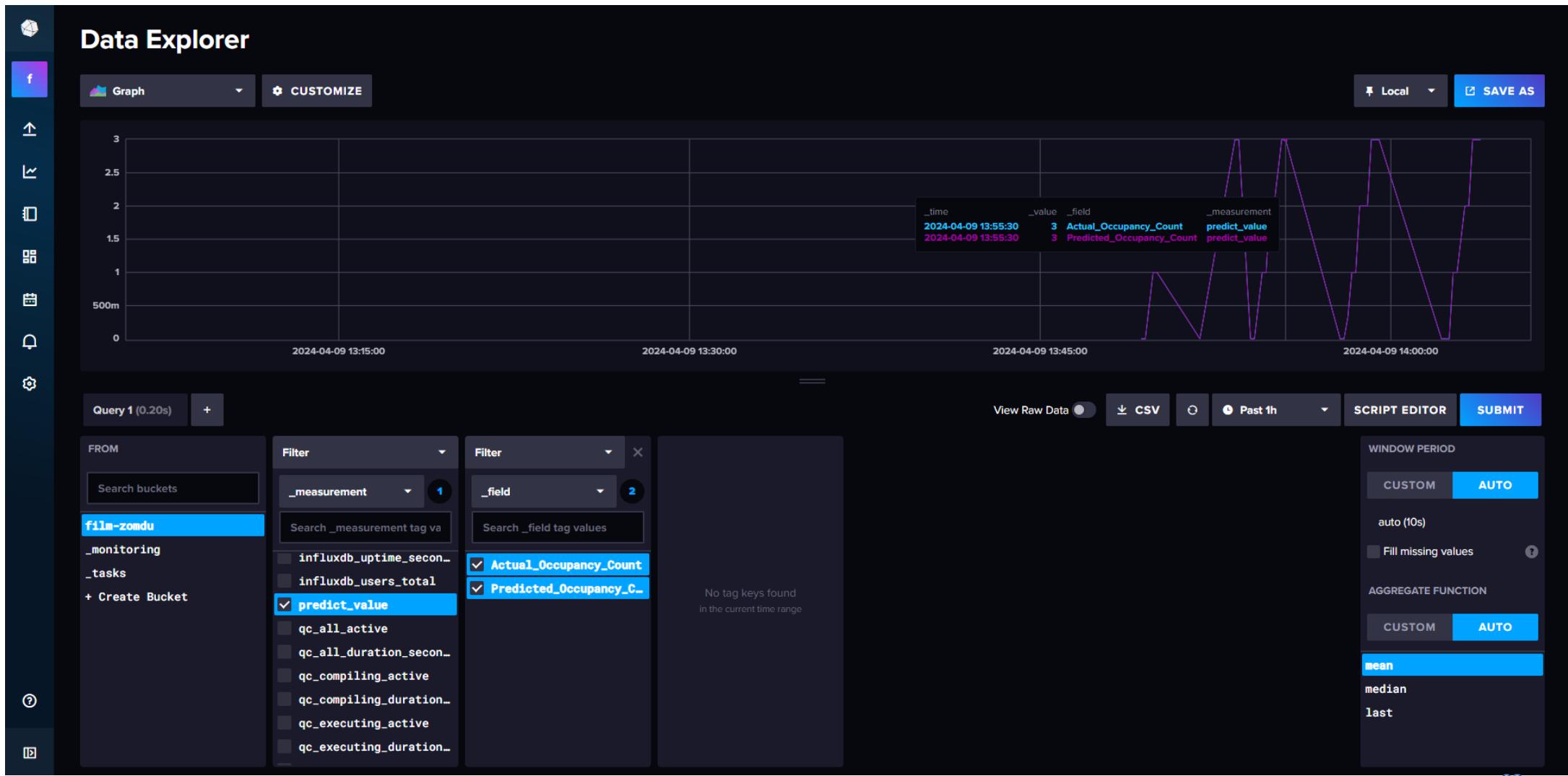
English ▾ 91 Do

Deploy Model on IoTCloudServe@TEIN

Your Turn!!!

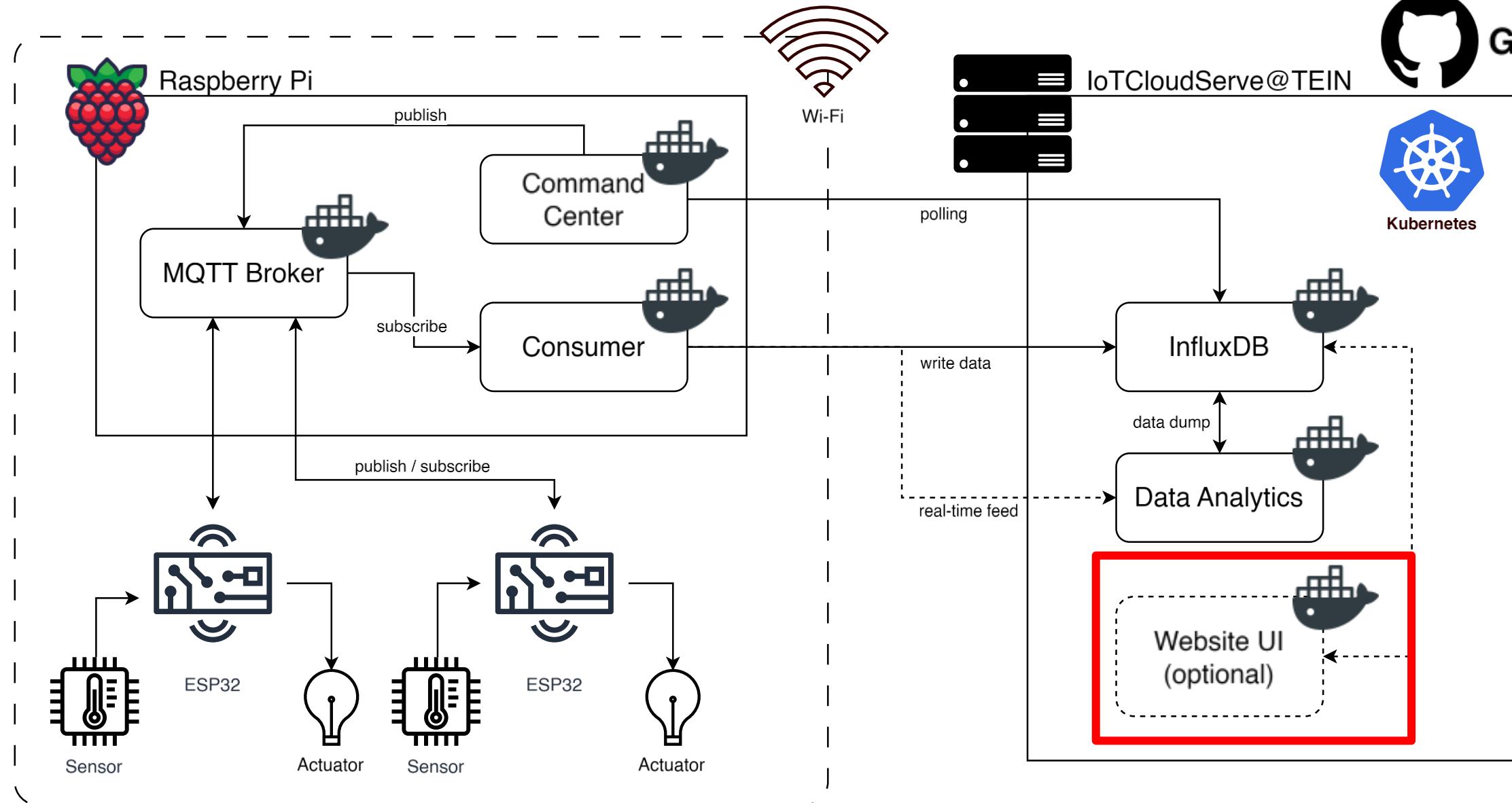
- > Go back to Consumer in Raspberry Pi
- > Change the PREDICT_URL inside “.env”
- > Run the Consumer again with the new “.env”
- > See the result in InfluxDB
 - measurement: “predict_value”

Check the result in InfluxDB





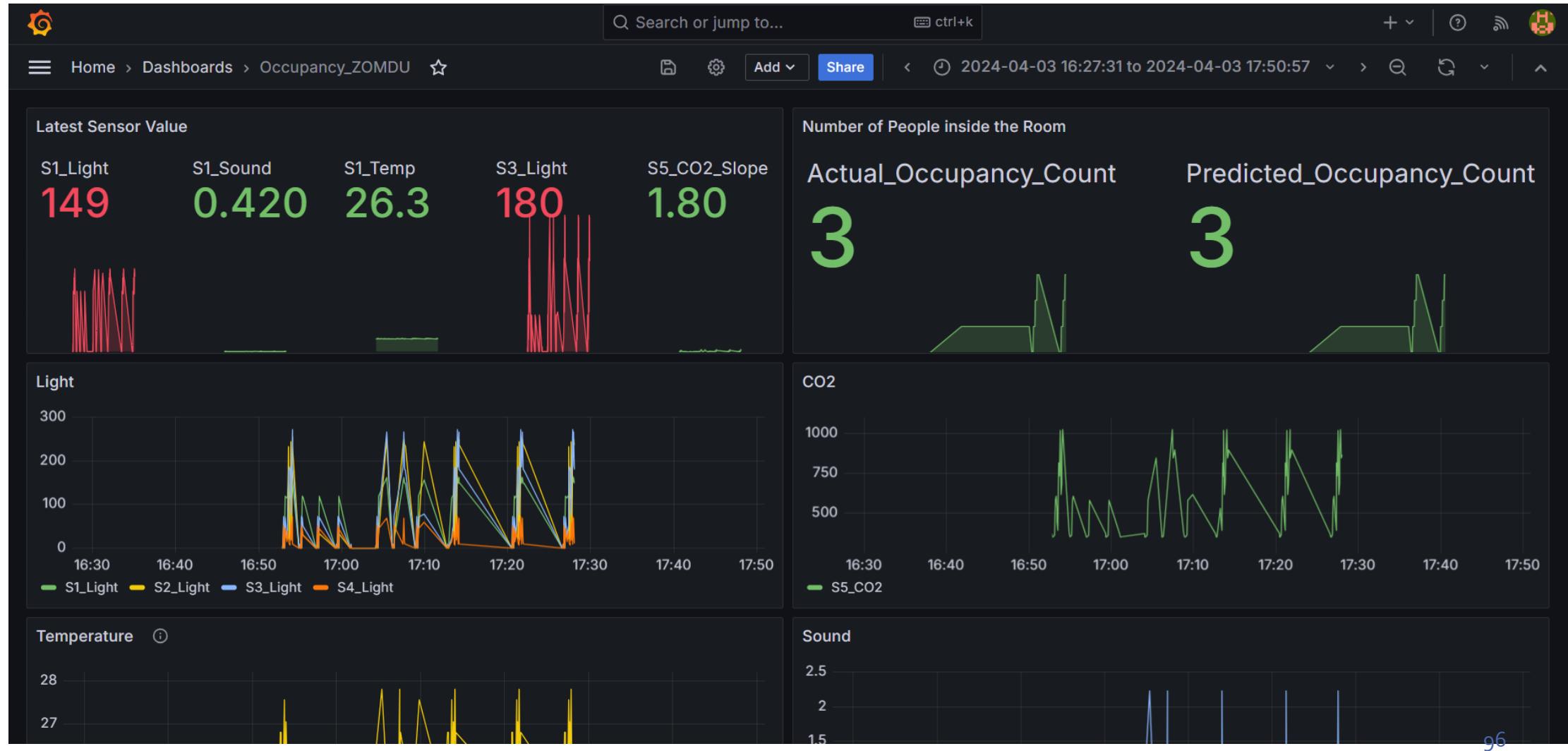
GitHub



Website UI - InfluxDB Dashboard



Deploy Grafana on IoTCloudServer@TEIN

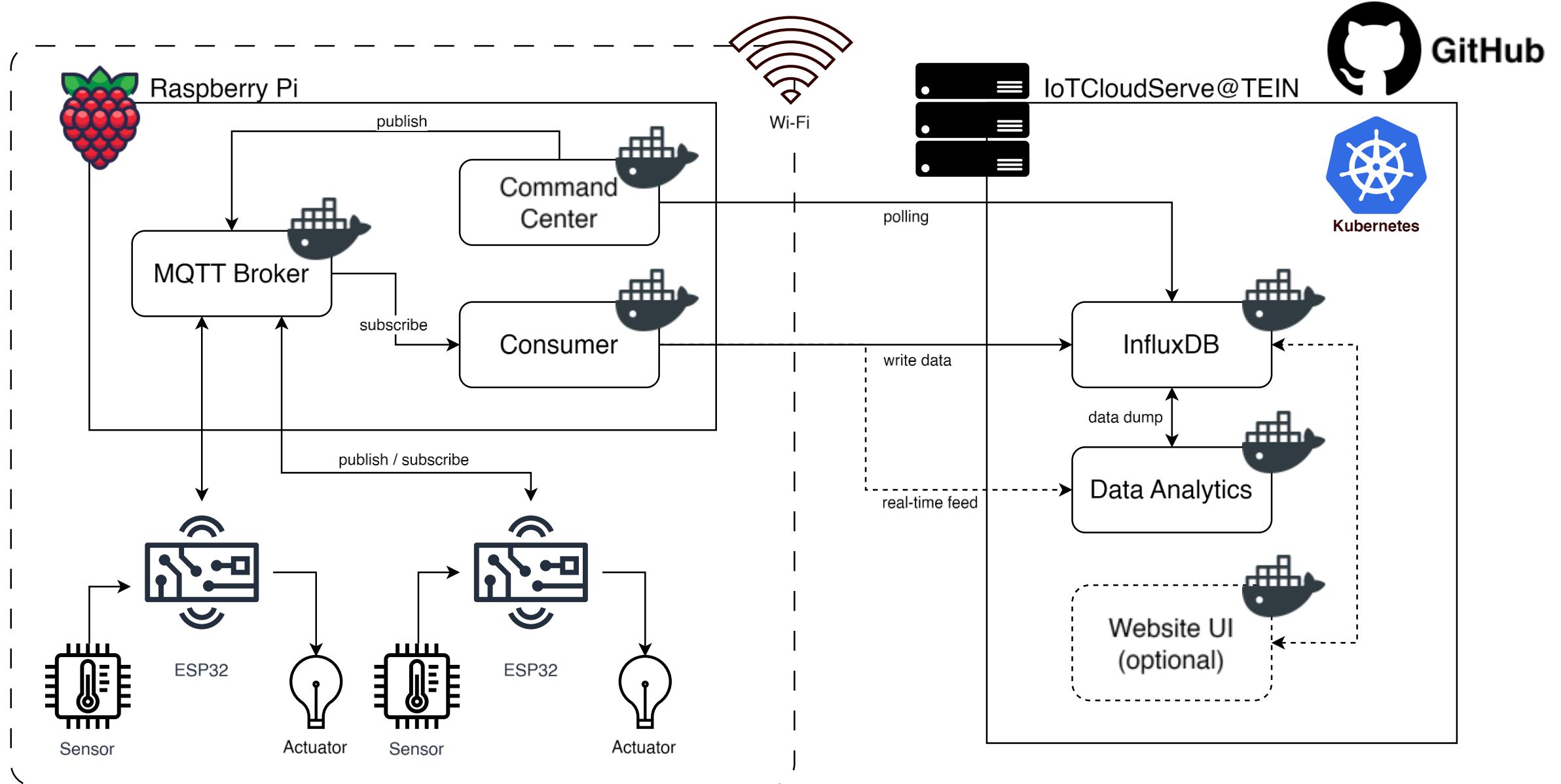


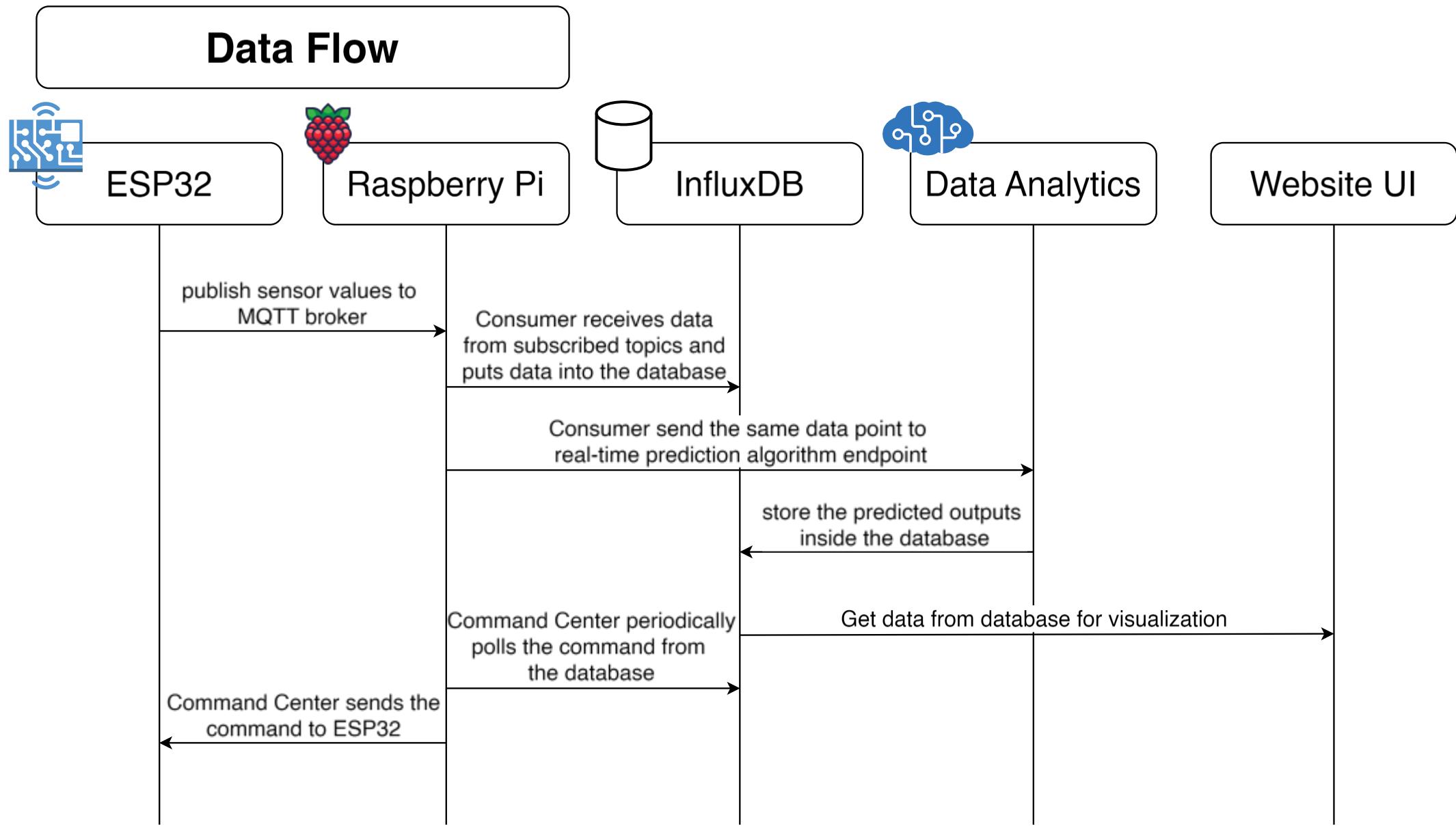
Deploy Grafana on IoTCloudServer@TEIN

Docker Image:

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

Grafana is running on Port 3000





HOPE YOU ALL ENJOY
THE WORKSHOP!
