Cloud Based Pet Detection System

Under Guidance of Prof. Dr. Christian Baun

Presented By (Group 4)
Saddam Hossain, A B M Nashrif, Md Motaleb Hossain, Omme Salma, Imrul Kayes Talukdar, Nur Uddin Syeed

Contents

Problem Details

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

1

Image Detection

Training a model that can detect Cats and Dogs

- Used Yolo v5 Model for the training
- Used X images as training data
- Used ….
- Used ….

2

Sensor Node

Sensor Node should be able to detect cats and dogs and should only send image of cat or dog to the backend server

- Device preparation (OS installation, Camera setup
- Load the trained model weight
- Create a script in Python that will detect the image and send the date using RESTful APIs.

2

kubernetes cluster

Create a Cluster and install all required backend and frontend application to get image data and represent the data in User interface (here frontend web page)

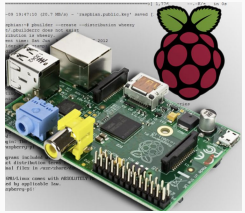
- K3S cluster setup
- Image Storage System
- Data Storage System
- Backend API service
- Frontend Web app

Contents

Hardware & Tools

Hardware:

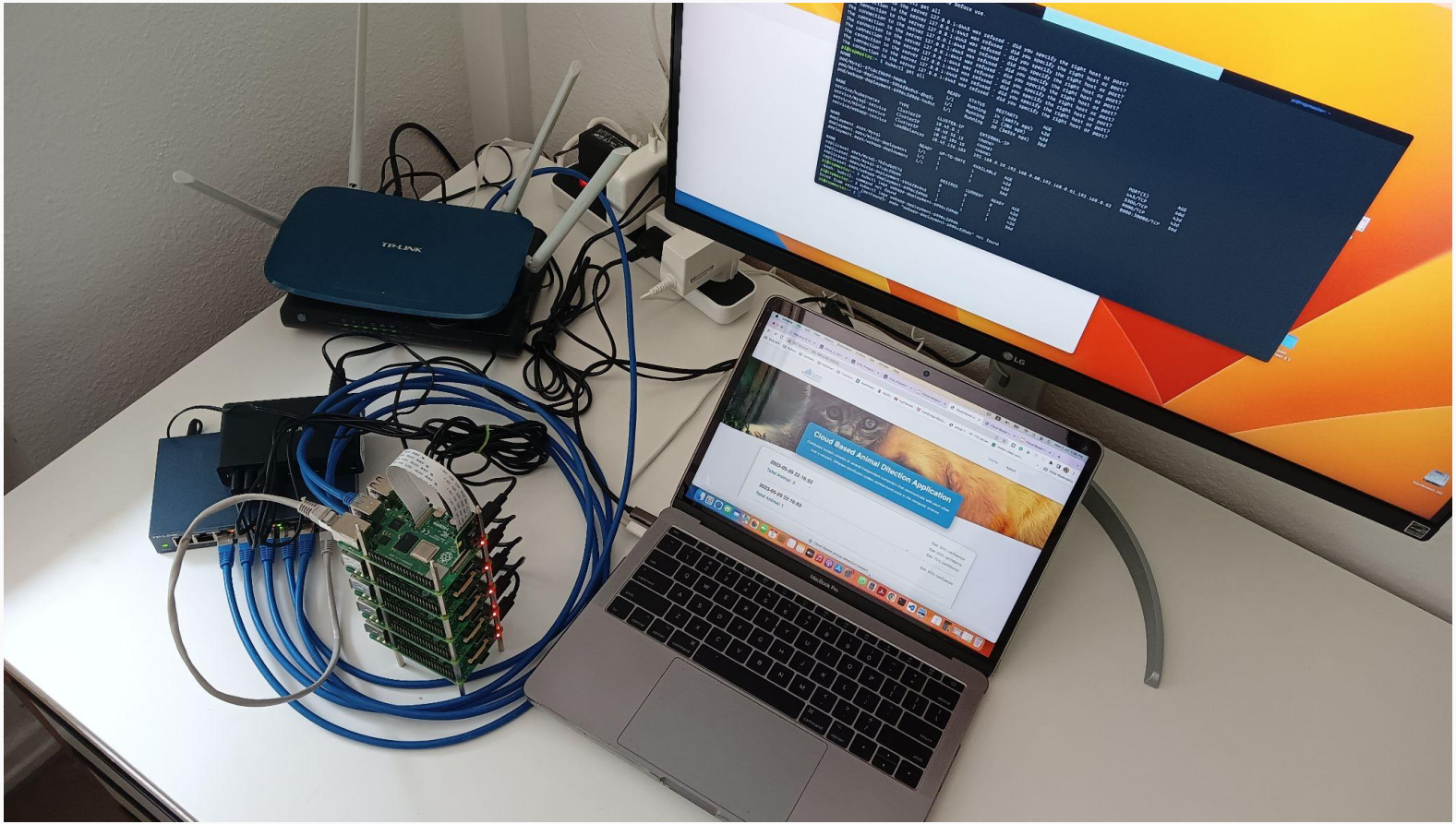
- Raspberry Pi4
- 4 Raspberry pi3
- Raspberry pi Camera Module v2
- TP Link Switch



Tools Used:

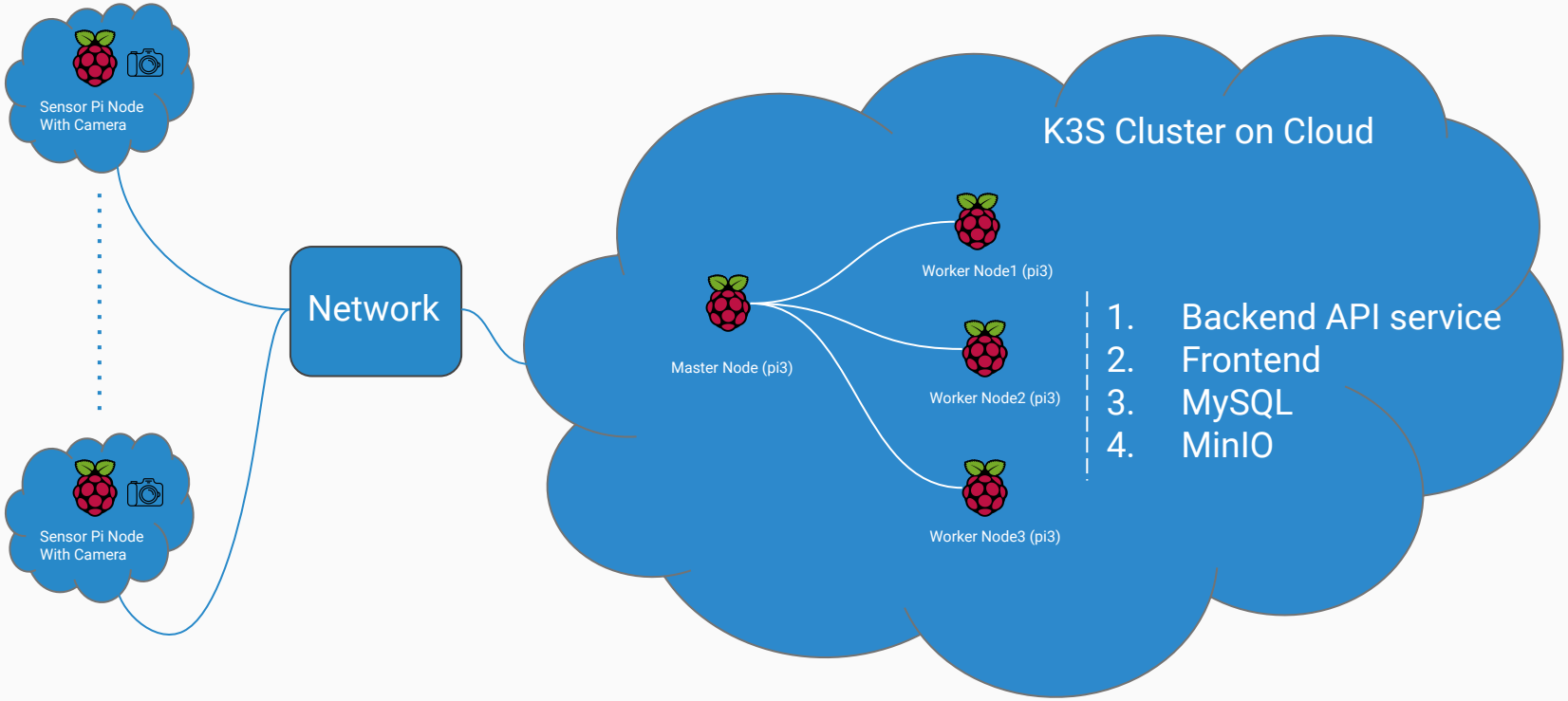
- Python & JavaScript
- FastAPI
- Docker
- Kubernetes (K3s)
- MinIO
- Pytorch
- Yolo v5

Hardware & Tools

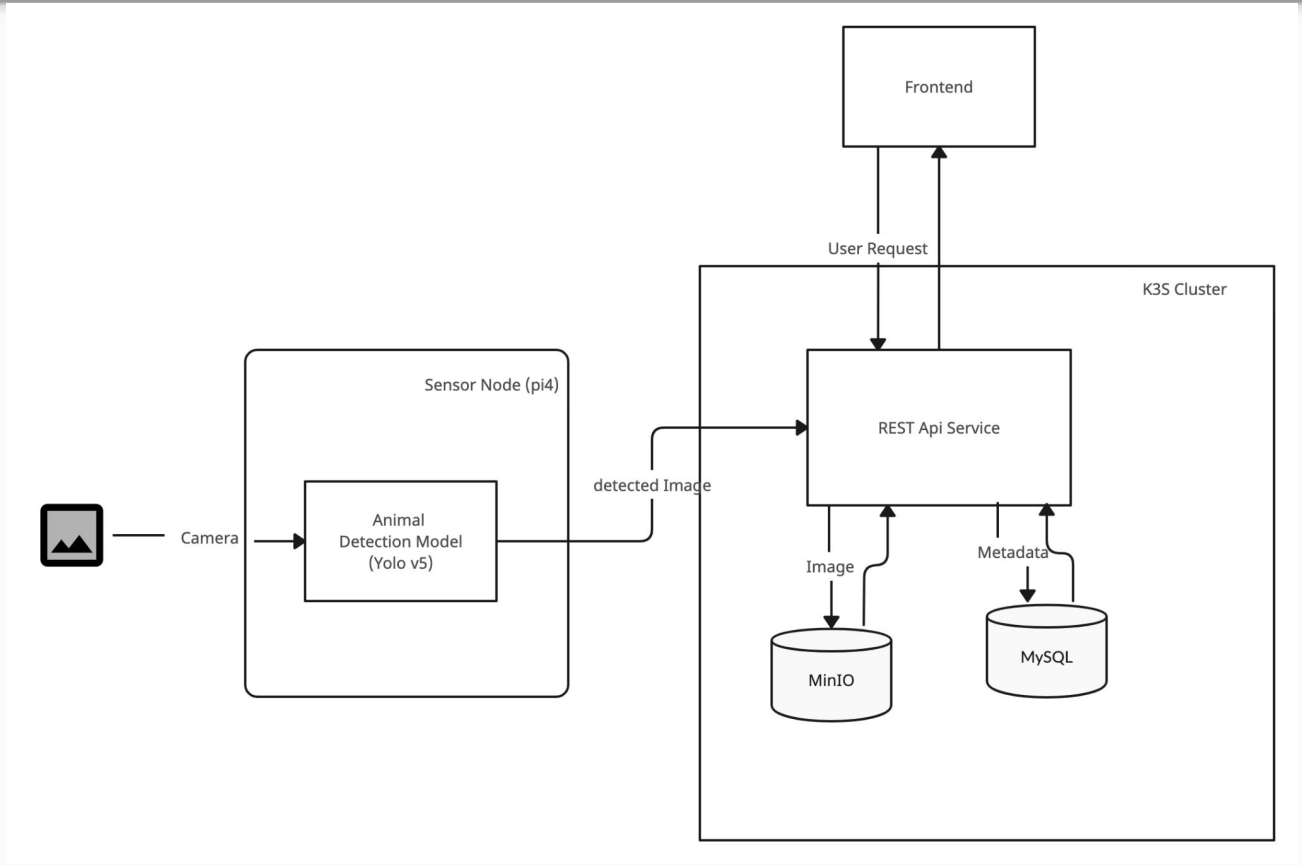


Contents

System Setup

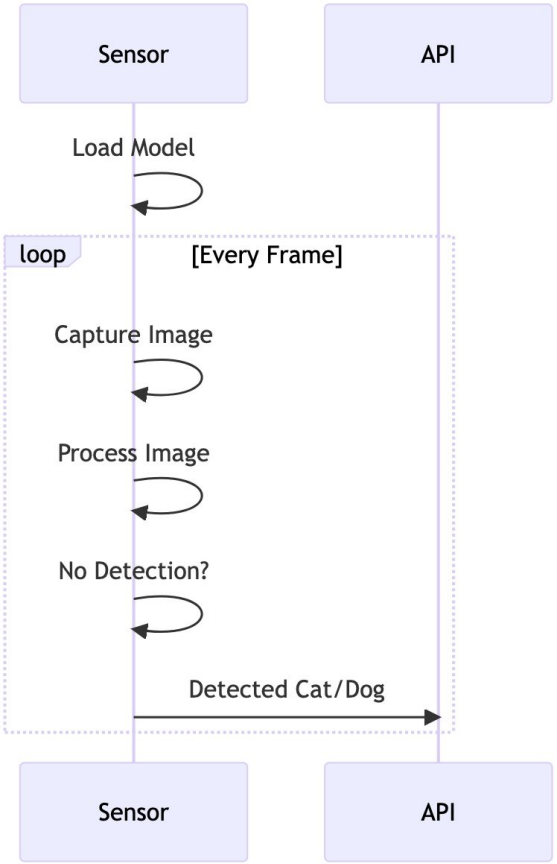


System Architecture

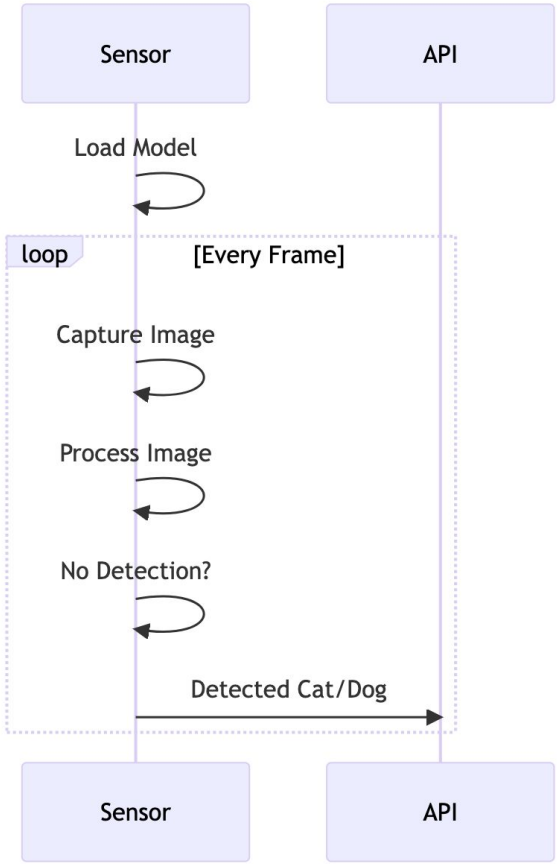


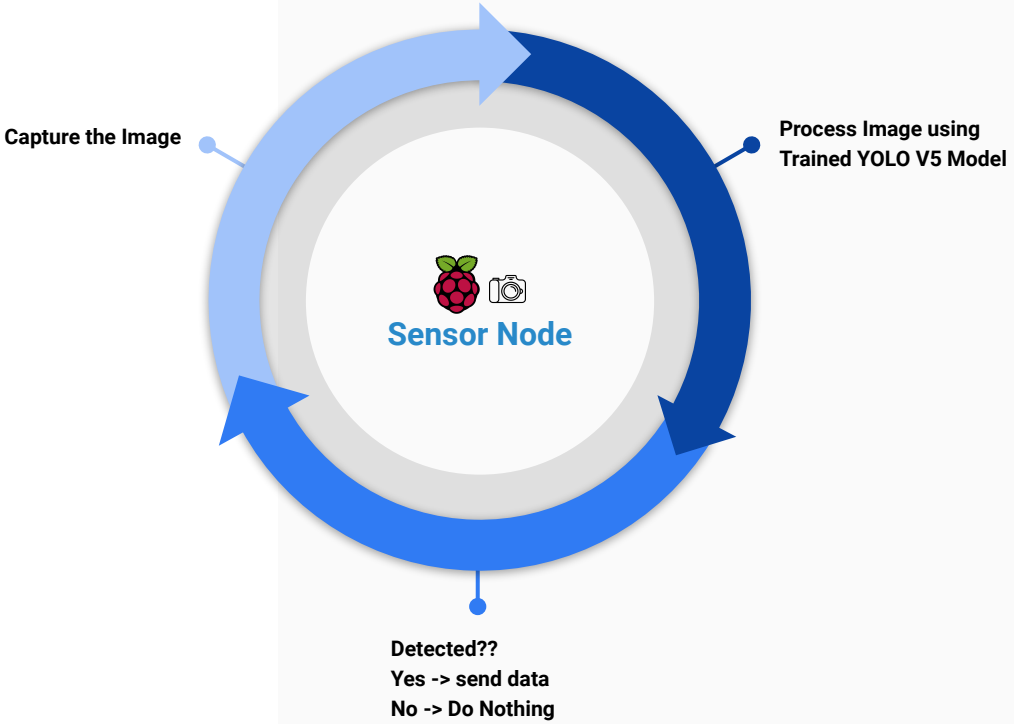
Contents

Sequence Diagram (Sensor Node)

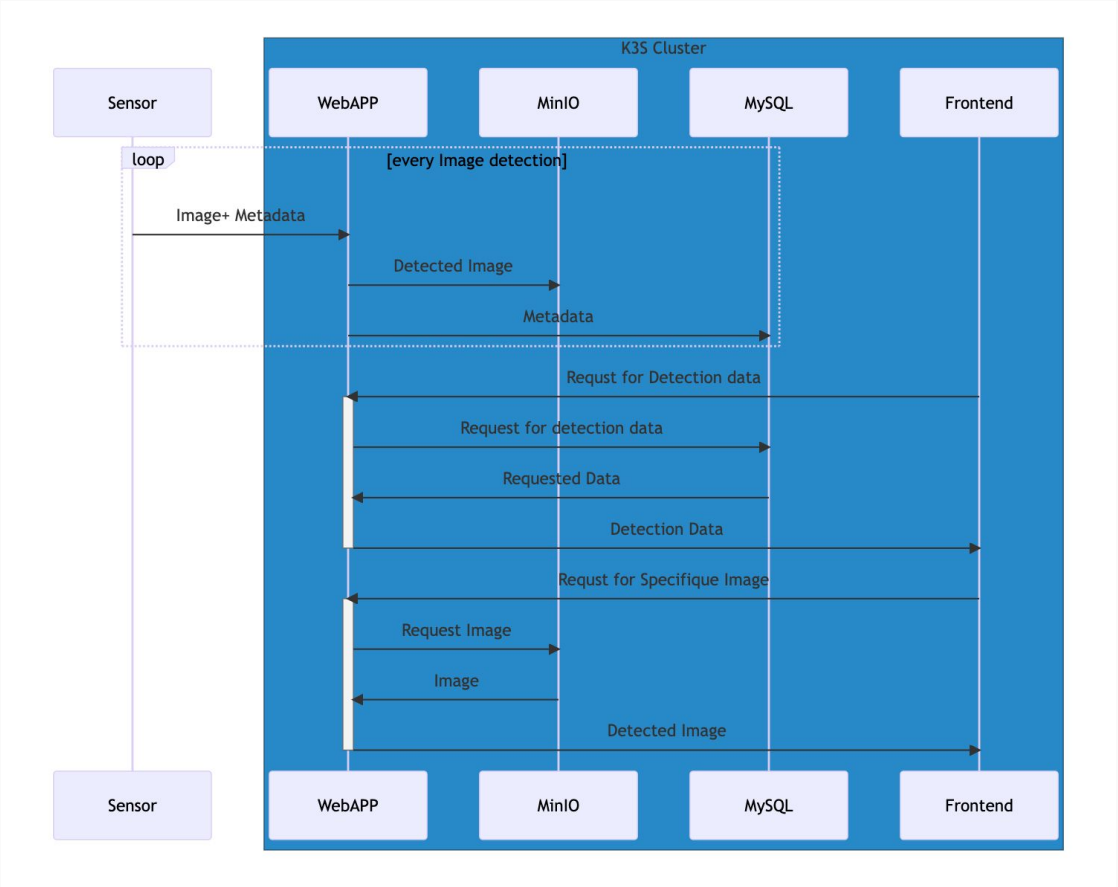


Sequence Diagram (Sensor Node)





Sequence Diagram (K3S)



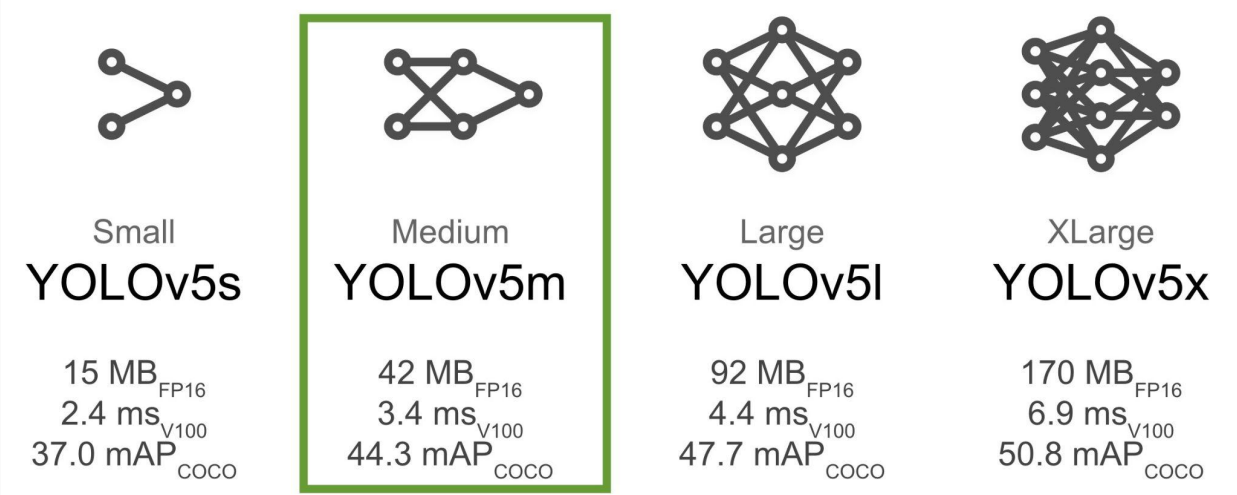
Contents

YoloV5 Model

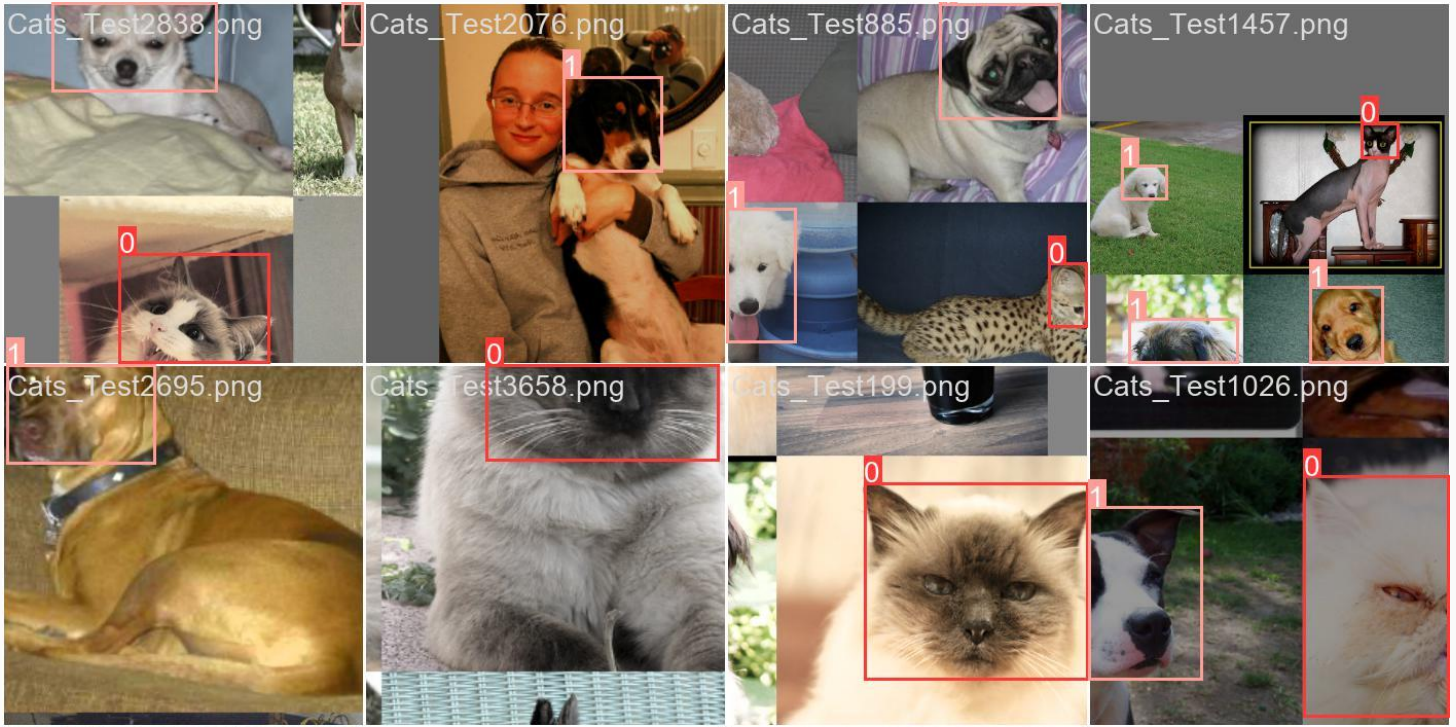
Dataset

https://www.kaggle.com/datasets/andrewmvd/dog-and-cat-detection/code

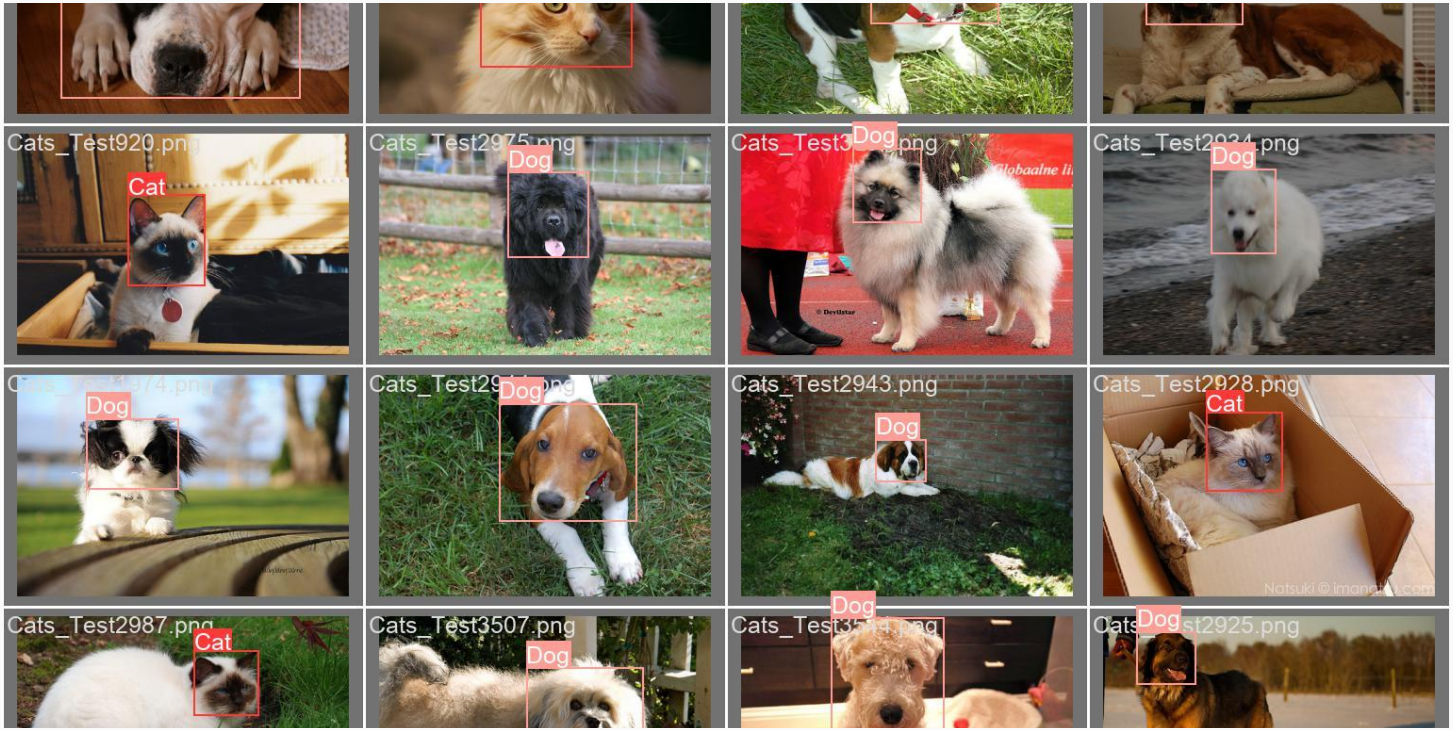
Dataset contains 3686 images of cat and dogs with bounding box.



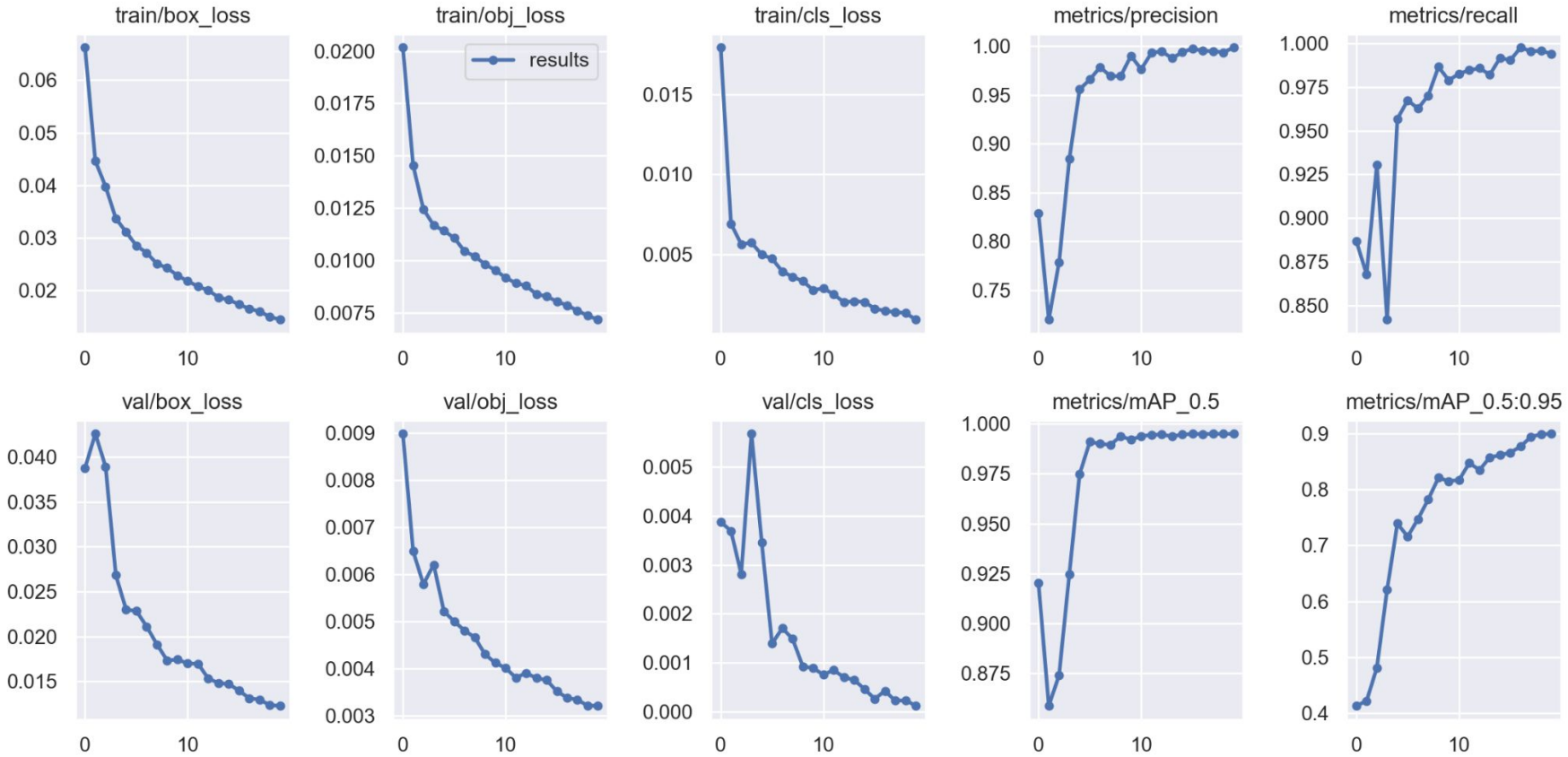
Train Batch image in Model Training



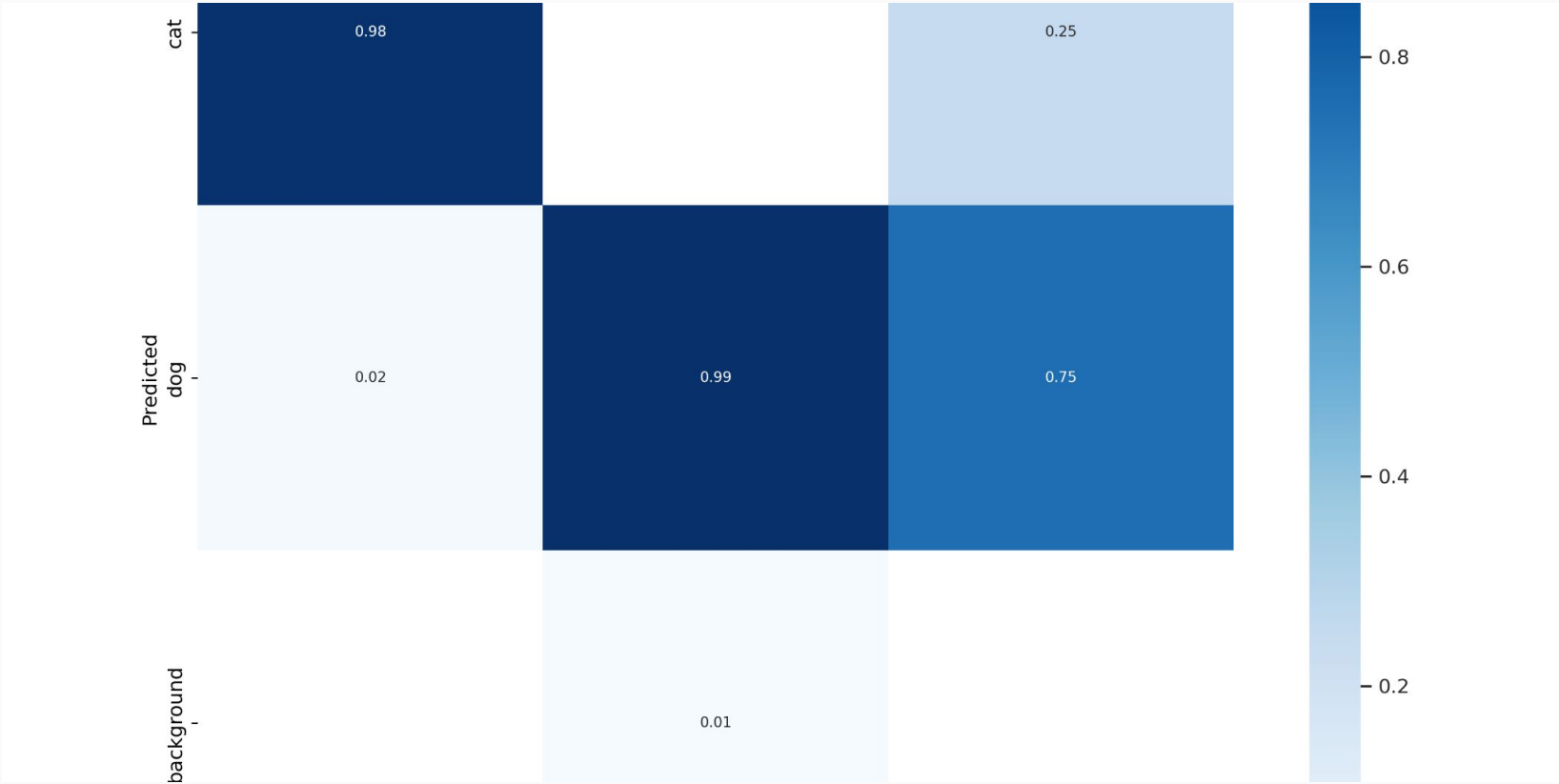
Validation Batch image in Model Training



YoloV5 Model



YoloV5 Model



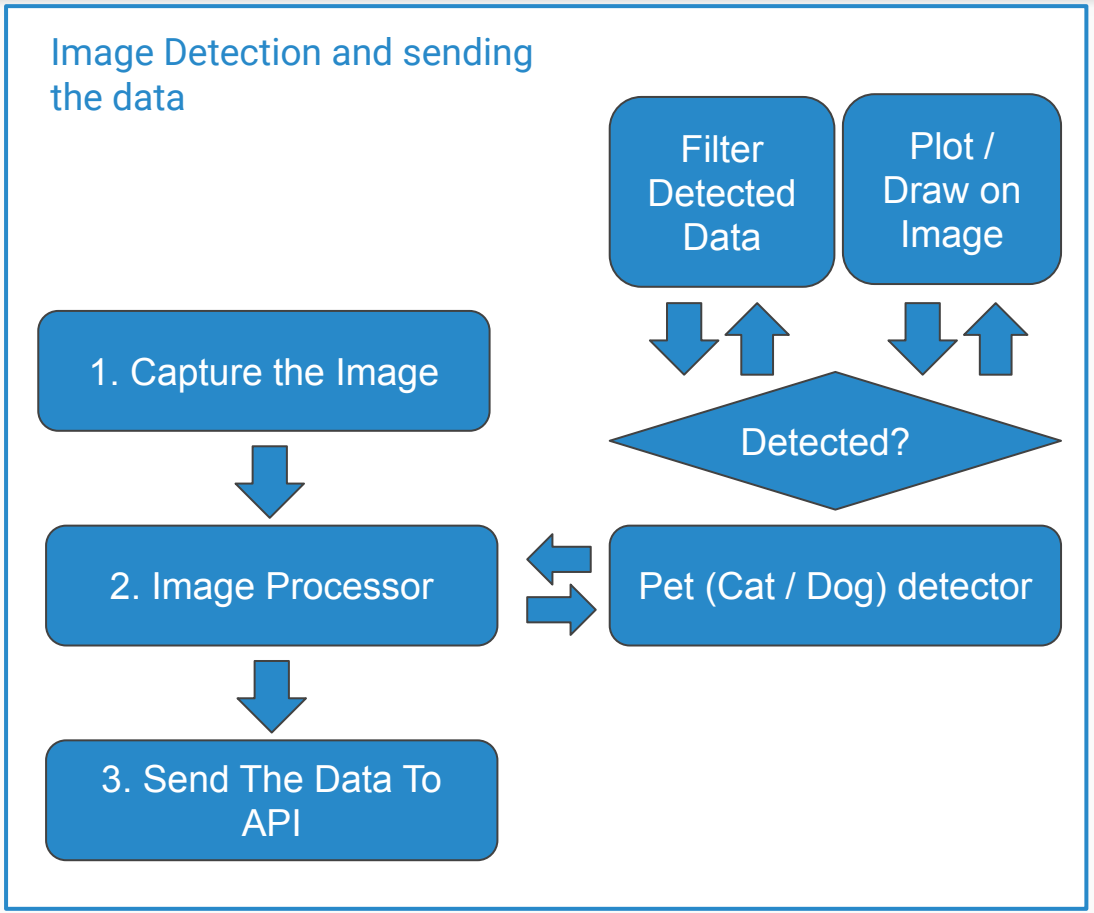
Contents

Sensor Node

Load the trained model

Sensor Node Workflow:

- At first it loads the model we trained to detect cat and dog
- Then its keep capturing images (1)
- Send the image to Image processor (2)
- If any cat or dog is detected, it sends the data to the Backend server through REST api



API

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

1. Introduction
2. Tools and Hardware
3. Architecture
4. Sequence Diagram
5. Detection Model Preparation
6. Sensor Node
7. WebApp (REST API)
8. Data Storage
9. Frontend
10. Kubernetes and Apps Deployment

REST API communication

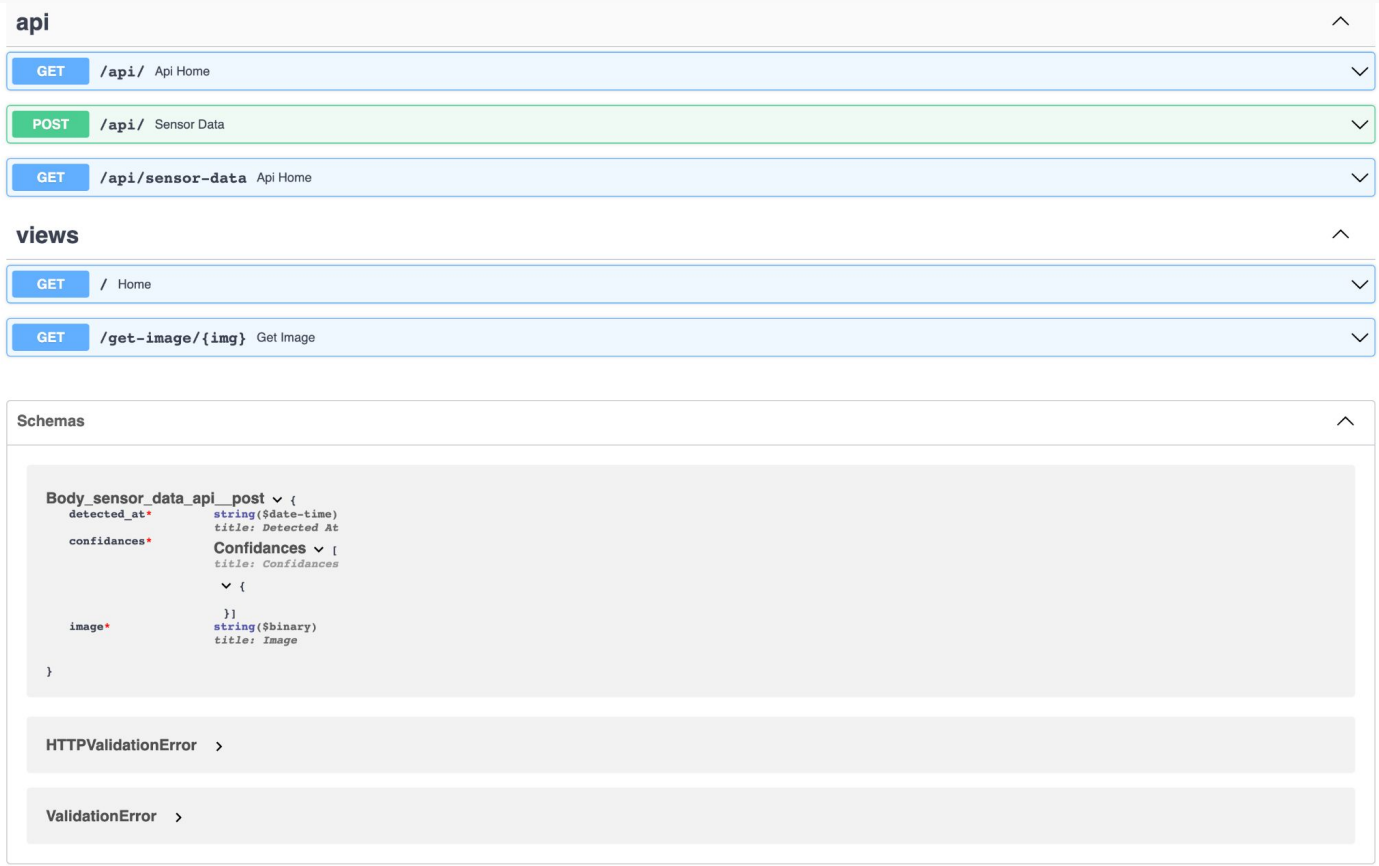
```
//Image
{"image": "base64....."

// Data
{
   "detected_at": "detection time",
   "confidences": [
      "('Animal1', 99)",
      "('Animal 2', 90)",
      :
      :
      "('Animal N', 89)",
   ]
}
```

Detected Image as Base64 encoded

Detection Time

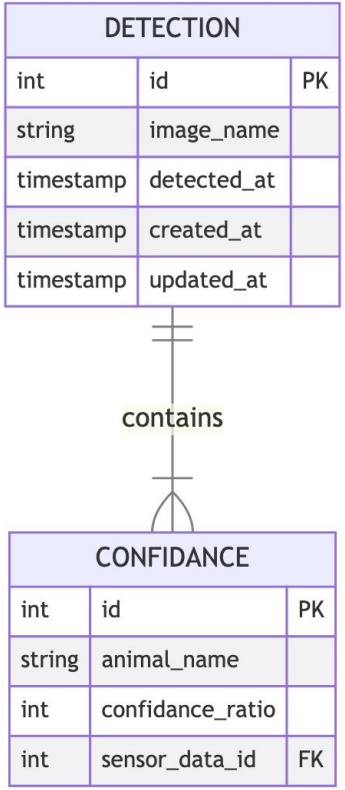
List of detection data

REST API communication



Contents

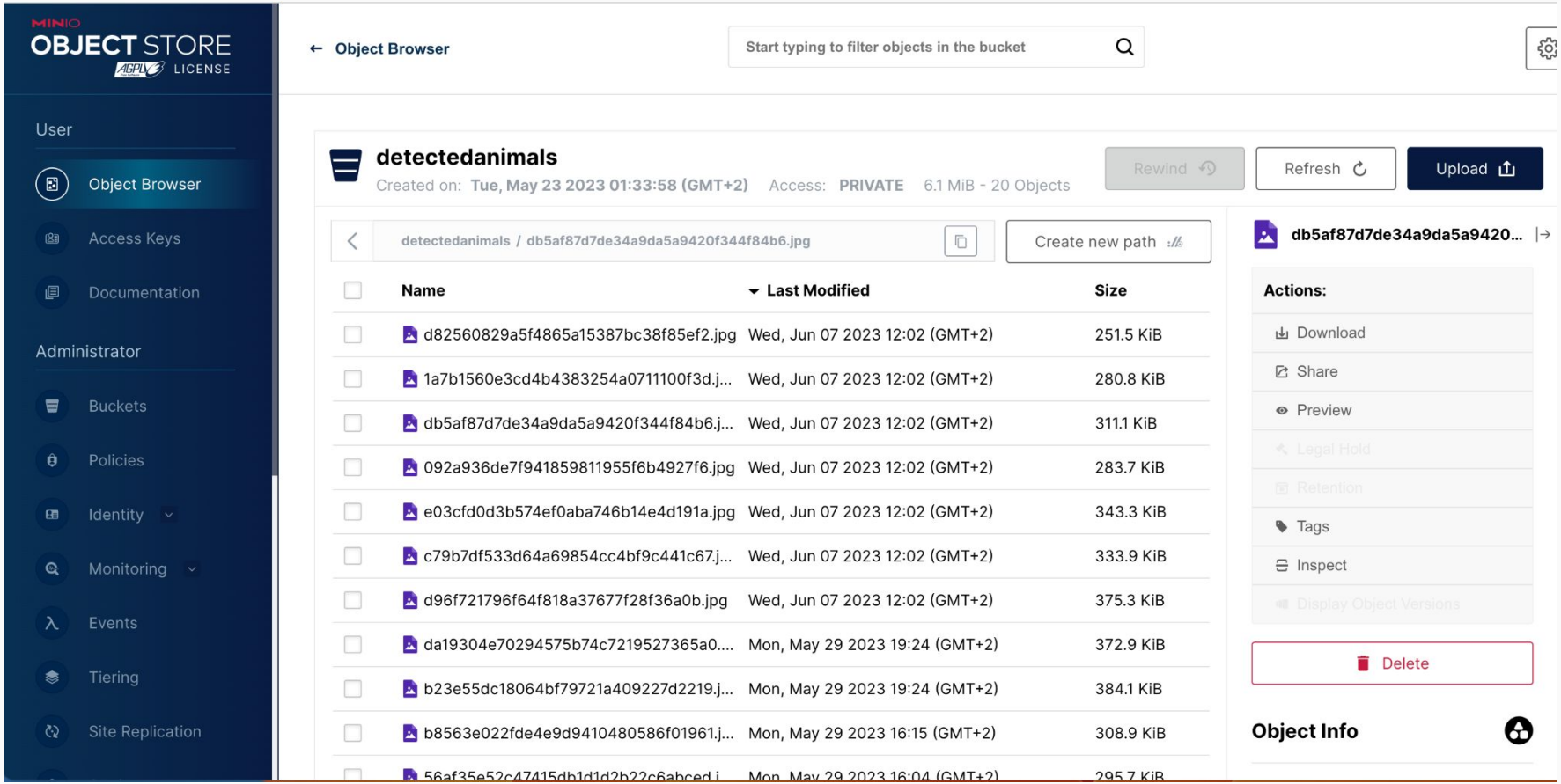
Backend Web_App: Data Entity




```
+-------------+-------------+------+-----+-------------------+-----------------------------------------------+
| Field       | Type        | Null | Key | Default           | Extra                                         |
+-------------+-------------+------+-----+-------------------+-----------------------------------------------+
| id          | int         | NO   | PRI | NULL              | auto_increment                                |
| image_name  | varchar(255) | NO   |     | NULL              |                                               |
| detected_at | timestamp   | NO   |     | NULL              |                                               |
| created_at  | timestamp   | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED                             |
| updated_at  | timestamp   | YES  |     | CURRENT_TIMESTAMP | DEFAULT_GENERATED on update CURRENT_TIMESTAMP |
+-------------+-------------+------+-----+-------------------+-----------------------------------------------+
5 rows in set (0.01 sec)
```

```
mysql> describe confidance;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| id               | int         | NO   | PRI | NULL    | auto_increment |
| animal_name      | varchar(255) | NO   |     | NULL    |                |
| confidance_ratio | int         | NO   |     | NULL    |                |
| sensor_data_id   | int         | NO   | MUL | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
4 rows in set (0.17 sec)
```

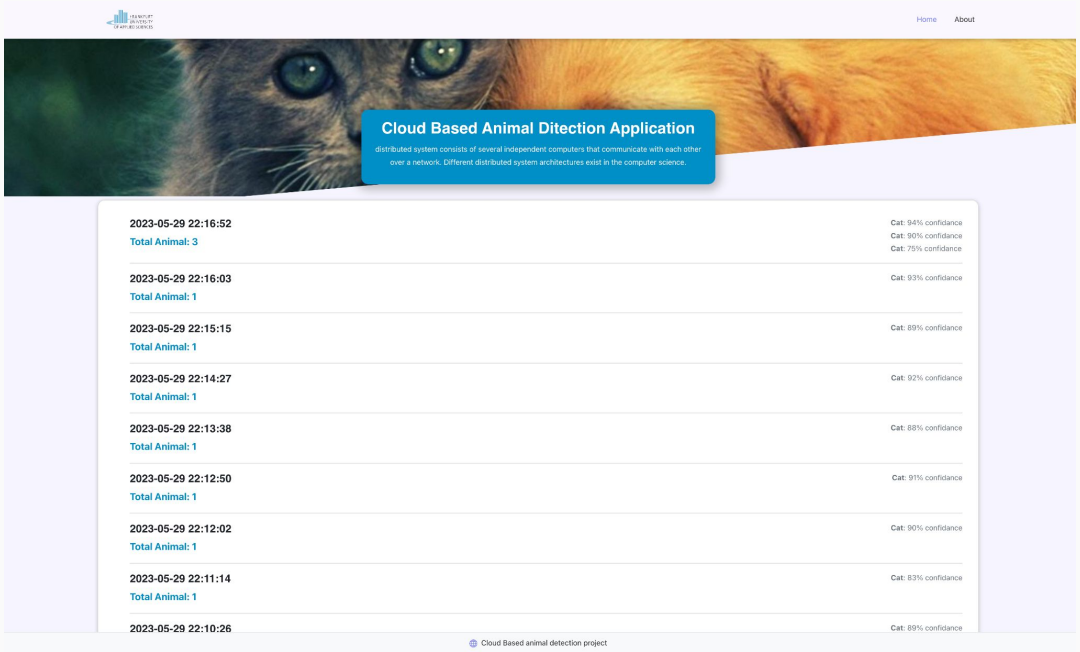
Image Storage:



Contents

Frontend

- ➔ Latest Detection data can be viewed
- ➔ Detected image can be observed in the detection detail page



Contents

Deployment on Kubernetes

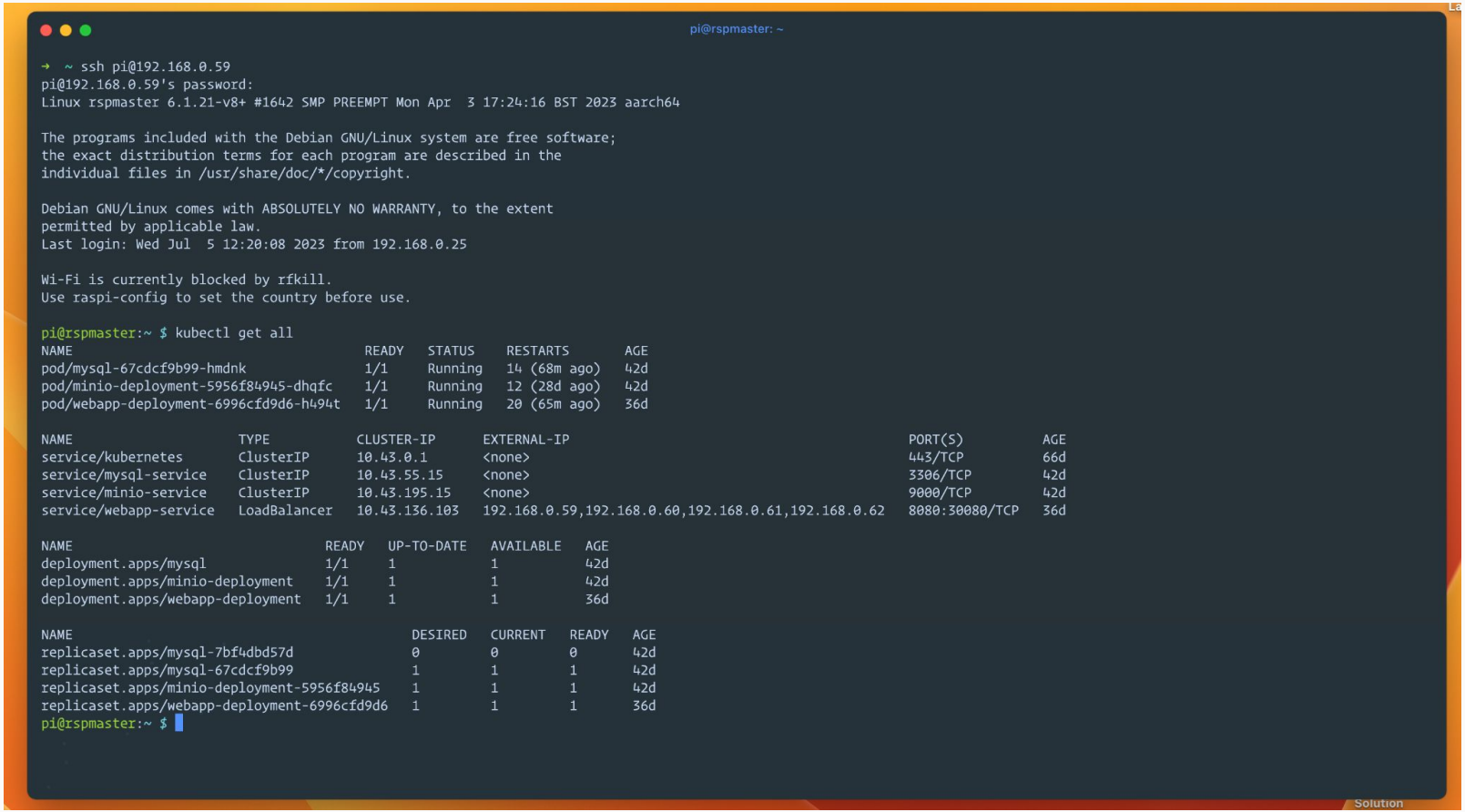
- This project is a combination of several micro services.
- It can be very easily deployed on Kubernetes cluster

This is how we did this:

1. Installed Kubernetes on 4 pi, keeping 1 master and 3 worker node
2. Dockerized WebApp and uploaded the docker image Docker hub.
3. Configured Kubernetes files for MySQL, MinIO, and WebAPP and deployed their docker image to the Kubernetes.

```
pi@rspmaster:~ $ kubectl get nodes
NAME        STATUS   ROLES                  AGE   VERSION
worker3     Ready    <none>                 66d   v1.26.4+k3s1
worker2     Ready    <none>                 66d   v1.26.4+k3s1
worker1     Ready    <none>                 66d   v1.26.4+k3s1
rspmaster   Ready    control-plane,master   66d   v1.26.4+k3s1
pi@rspmaster:~ $
```

kubernetes


pi@rspmaster: ~
→ ~ ssh pi@192.168.0.59
pi@192.168.0.59's password:
Linux rspmaster 6.1.21-v8+ #1642 SMP PREEMPT Mon Apr 3 17:24:16 BST 2023 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: Wed Jul 5 12:20:08 2023 from 192.168.0.25
Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.
pi@rspmaster:~ $ kubectl get all
NAME READY STATUS RESTARTS AGE
pod/mysql-67cdcf9b99-hmdnk 1/1 Running 14 (68m ago) 42d
pod/minio-deployment-5956f84945-dhqfc 1/1 Running 12 (28d ago) 42d
pod/webapp-deployment-6996cfd9d6-h494t 1/1 Running 20 (65m ago) 36d
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
service/kubernetes ClusterIP 10.43.0.1 <none> 443/TCP 66d
service/mysql-service ClusterIP 10.43.55.15 <none> 3306/TCP 42d
service/minio-service ClusterIP 10.43.195.15 <none> 9000/TCP 42d
service/webapp-service LoadBalancer 10.43.136.103 192.168.0.59,192.168.0.60,192.168.0.61,192.168.0.62 8080:30080/TCP 36d
NAME READY UP-TO-DATE AVAILABLE AGE
deployment.apps/mysql 1/1 1 1 42d
deployment.apps/minio-deployment 1/1 1 1 42d
deployment.apps/webapp-deployment 1/1 1 1 36d
NAME DESIRED CURRENT READY AGE
replicaset.apps/mysql-7bf4dbd57d 0 0 0 42d
replicaset.apps/mysql-67cdcf9b99 1 1 1 42d
replicaset.apps/minio-deployment-5956f84945 1 1 1 42d
replicaset.apps/webapp-deployment-6996cfd9d6 1 1 1 36d
pi@rspmaster:~ $

DEMO TIME

Question??

Useful links

- Project GitHub Link: https://github.com/nuruddinsayeed/CloudApp101
- Docker Image of WebApp: https://hub.docker.com/r/nuruddinsayeed/webapp-animal_detector