FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Automatic Cat and Dog Detection Using Edge Computing

Member Name: Sumit Chothani [1457445]

Hardikkumar Suhagiya [1419315]

Rajdeep Kachhadiya [1440737]

Meet Gabani [1442735]

Kavita Vaghasiya [1442706]

Jay Togadiya [1413353]

Under Guidance of: -

Prof. Dr. Christian Baun

Content

- Team Members and Task Distribution
- Introduction
- Architecture
- Sensor/Edge Node Deployment
- Object Detection Model
- Setting up K3S Cluster using Raspberry Pi 3
- REST API
- Kubernetes Cluster Application
- Demo
- Reference

Team Members and Task Distribution

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

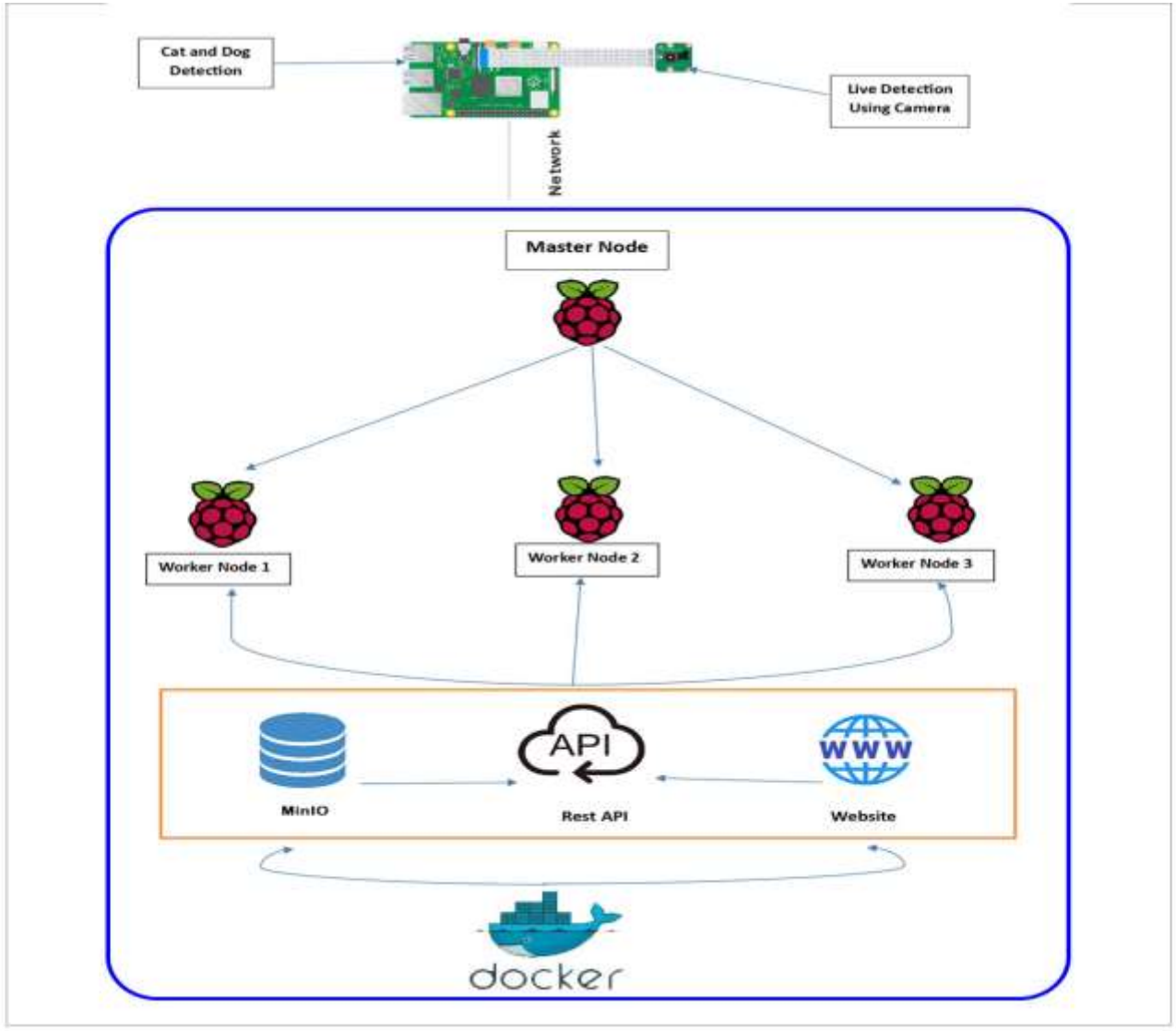
Task Distribution: -	Member Name: -
Initial Hardware Setup, Testing of Hardware	Hardikkumar Suhagiya ,Meet Gabani, Jay Togadiya
Senser/Edge Node Setup, $K3_S$ Cluster Setup	Hardikkumar Suhagiya,Meet Gaban
Object Detection Model, Training Of Model	Rajdeep Kachhadiya, Sumit Chothani
Backend & Frontend API Development	Rajdeep Kachhadiya, Sumit Chothani, Kavita Vaghashiya
User Interface development	Jay Togadiya, Kavita Vaghashiya
Docker & MinIo Setup, Sensor /Edge Node & $K3_s$ Cluster Integration	Hardikkumar Suhagiya, Meet Gabani
Project Integration	Rajdeep Kachhadiya,Sumit Chothani,Jay Togadiya, Kavita Vaghashiya
Documentation	Hardikkumar Suhagiya, Meet Gabani, Rajdeep Kachhadiya, Sumit Chothani, Jay Togadiya,Kavita Vaghashiya

Introduction

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

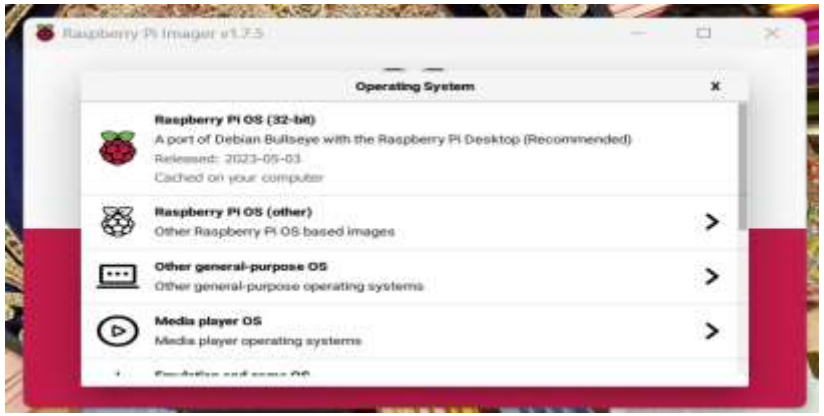
- Data is transmitted to the cloud for processing and storing in cloud computing.
- edge computing processes data at the edge node before sending a little portion to the cloud for storage.
- Compared to conventional cloud computing strategies, edge computing has several advantages.
- Edge computing has lower latency than cloud computing because data is processed at the edge node, which is closer to the source.

Architecture



Sensor/Edge Node Deployment

- Sensor node is a Raspberry Pi 4 single board computer (SBC) with an attached Raspberry Pi camera module 4.
- the sensor node is used as an edge device to detect cat and dog, build relevant data and send it over to the REST API running on K3S cluster.
- To setup the sensor node, we installed Raspberry Pi OS 32 bit using the Raspberry Pi Imager tool.



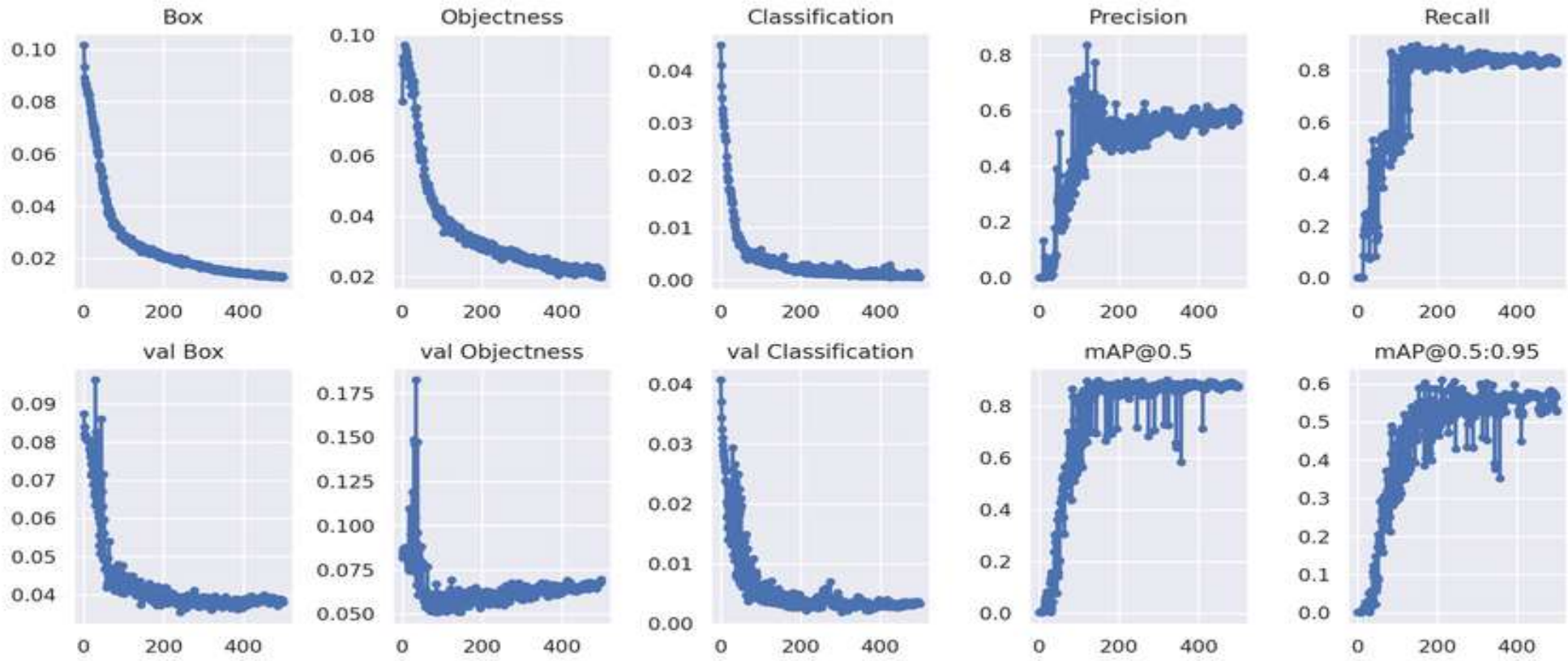
- The program has a Choose OS option where the OS can be chosen. Using the tool's Choose Storage option, the storage, or SD card, was chosen. Once both of these are set, we can use the tool's Write Option to write the OS to the SD card.

Object Detection Model

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- We trained a machine learning model using YOLOv5 framework for automatic cat and dog detection on the sensor node.
- We used Roboflow to train machine learning models as it offers free computing resources
- Used 500+ cat and dog images and their labels to train model
- Splitted whole Dataset in a 70:30% ratio

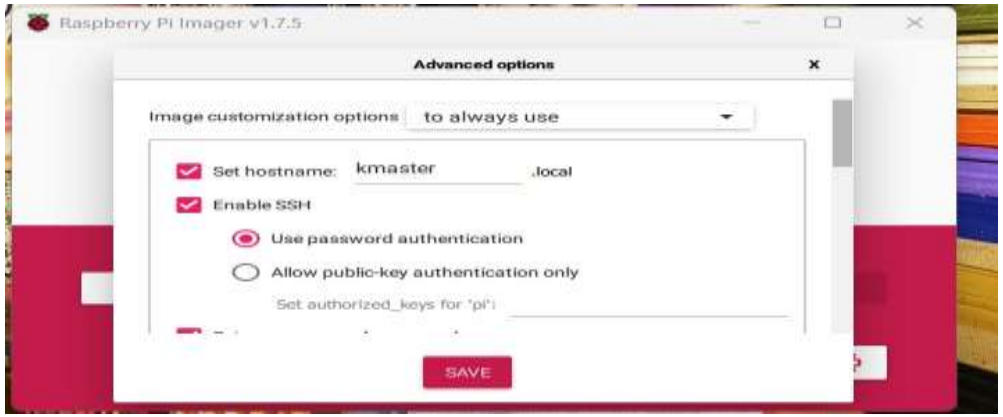
FRANKFURT UNIVERSITY OF APPLIED SCIENCES



Setting up K3S Cluster using Raspberry Pi 3

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- As discussed in the architecture we had used 4 different Raspberry Pi 3 SBC to setup a lightweight Kubernetes cluster or K3S cluster. The cluster was created with 1 master and 3 worker nodes.
- All the Raspberry Pi 3 was equipped with 32GB SD cards, we manually flashed 32- bit Raspberry Pi OS with help of Raspberry Pi Imager v1.7.3.
- In the Pi Imager application, we chose 32-bit Raspberry Pi OS (Debian Bullseye) and configured the hostname, enabled SSH and set password for authentication in the advanced options as shown below image.



- We repeated this process for all 4 SD cards and named our hosts as kmaster, knode1, knode2, knode3 respectively

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Setting up k3s cluster
  - ➢ Install Docker on Master Node and all three worker node using this Command

    sudo apt install docker

    sudo systemctl start docker

    sudo systemctl enable docker

    sudo systemctl status docker

  - ➢ Set up k3s server in master node

    curl -sfL https://get.k3s.io | sh -s - --docker

    sudo kubectl get nodes

```
pi@kmaster:~ $ sudo kubectl get nodes
NAME      STATUS   ROLES                  AGE   VERSION
knode1    Ready    <none>                 36d   v1.26.5+k3s1
knode2    Ready    <none>                 36d   v1.26.5+k3s1
kmaster   Ready    control-plane,master   36d   v1.26.5+k3s1
knode3    Ready    <none>                 36d   v1.26.5+k3s1
```

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Setup k3s agent in worker node

  sudo cat /var/lib/rancher/k3s/server/node-token

```
pi@kmaster:~ $ sudo cat /var/lib/rancher/k3s/server/node-token
K1082dd4ec9ee9aff044b15ae779e8a87c6f1d4807952c2e109e9cc831602959f5b::server:5fc68d1058000b84103c364b947bd447
```

curl -sfL http://get.k3s.io | K3S_URL=http://<master_IP>:6443 K3S_TOKEN=<join_token> sh -s - --docker

sudo kubectl get nodes -o wide

```
pi@kmaster:~ $ sudo kubectl get nodes -o wide
NAME      STATUS   ROLES                  AGE   VERSION        INTERNAL-IP     EXTERNAL-IP   OS-IMAGE                         KERNEL-VERSION   CONTAINER-RUNTIME
knode1    Ready    <none>                 36d   v1.26.5+k3s1   192.168.2.138   <none>        Debian GNU/Linux 11 (bullseye)   6.1.21-v8+       docker://24.0.2
knode2    Ready    <none>                 36d   v1.26.5+k3s1   192.168.2.140   <none>        Debian GNU/Linux 11 (bullseye)   6.1.21-v8+       docker://24.0.2
kmaster   Ready    control-plane,master   36d   v1.26.5+k3s1   192.168.2.139   <none>        Debian GNU/Linux 11 (bullseye)   6.1.21-v8+       docker://24.0.2
knode3    Ready    <none>                 36d   v1.26.5+k3s1   192.168.2.141   <none>        Debian GNU/Linux 11 (bullseye)   6.1.21-v8+       docker://24.0.2
```

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- After successful deployment status of all pods will look similar to below: -

```
pi@kmaster:~/project $ sudo kubectl get pods -A
NAMESPACE      NAME                                      READY   STATUS      RESTARTS          AGE
kube-system    helm-install-traefik-crd-vwlmp            0/1     Completed   0                 36d
kube-system    helm-install-traefik-7h4ph                0/1     Completed   1                 36d
frontend-dev   frontend                                  1/1     Running     1 (3h59m ago)     15h
kube-system    svclb-traefik-cbf739f5-tzzcs              2/2     Running     34 (3h59m ago)    36d
kube-system    svclb-traefik-cbf739f5-j8f86              2/2     Running     30 (3h59m ago)    36d
backend-dev    backend                                   1/1     Running     1 (3h59m ago)     16h
kube-system    svclb-traefik-cbf739f5-wqwkh              2/2     Running     36 (3h58m ago)    36d
kube-system    local-path-provisioner-76d776f6f9-c86qx   1/1     Running     30                36d
kube-system    svclb-traefik-cbf739f5-ppp5z              2/2     Running     60 (3h58m ago)    36d
kube-system    coredns-59b4f5bbd5-vtcx4                  1/1     Running     73                36d
kube-system    metrics-server-7b67f64457-cvgnp           1/1     Running     117               36d
minio-dev      minio                                     1/1     Running     1 (3h58m ago)     17h
kube-system    traefik-57c84cf78d-6l2gm                  1/1     Running     134 (3h18m ago)   36d
```

REST API

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

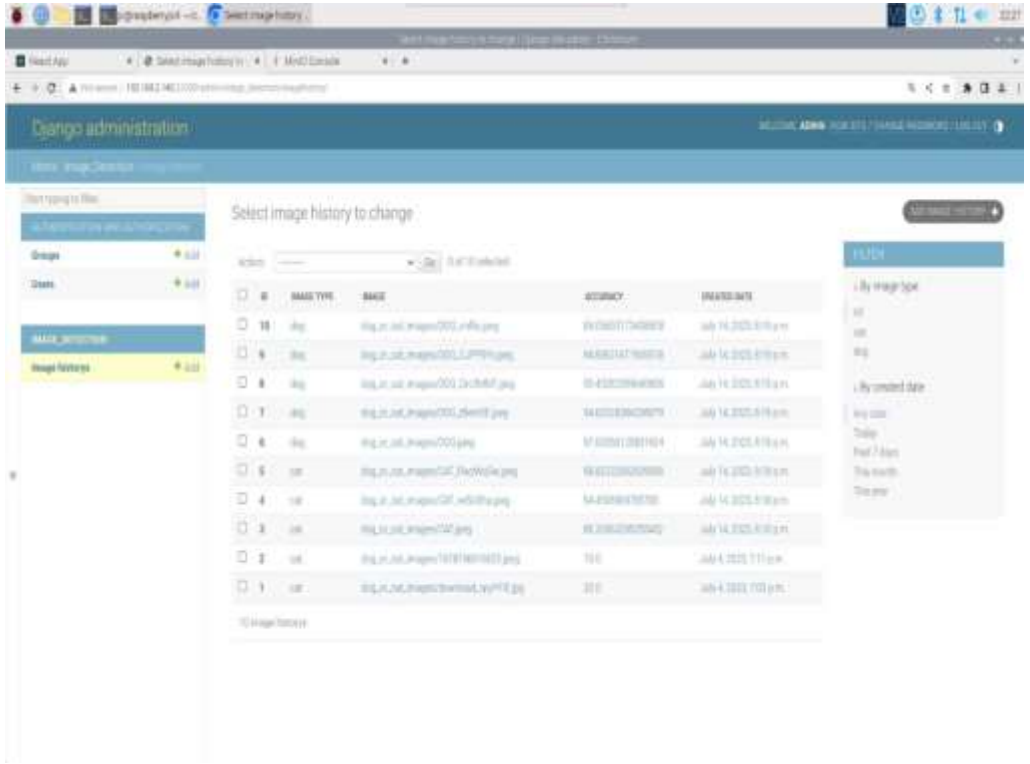
- Technology we have used:
  - Django
  - Django REST Framework
  - Django Filter
  - REST API
  - Docker:

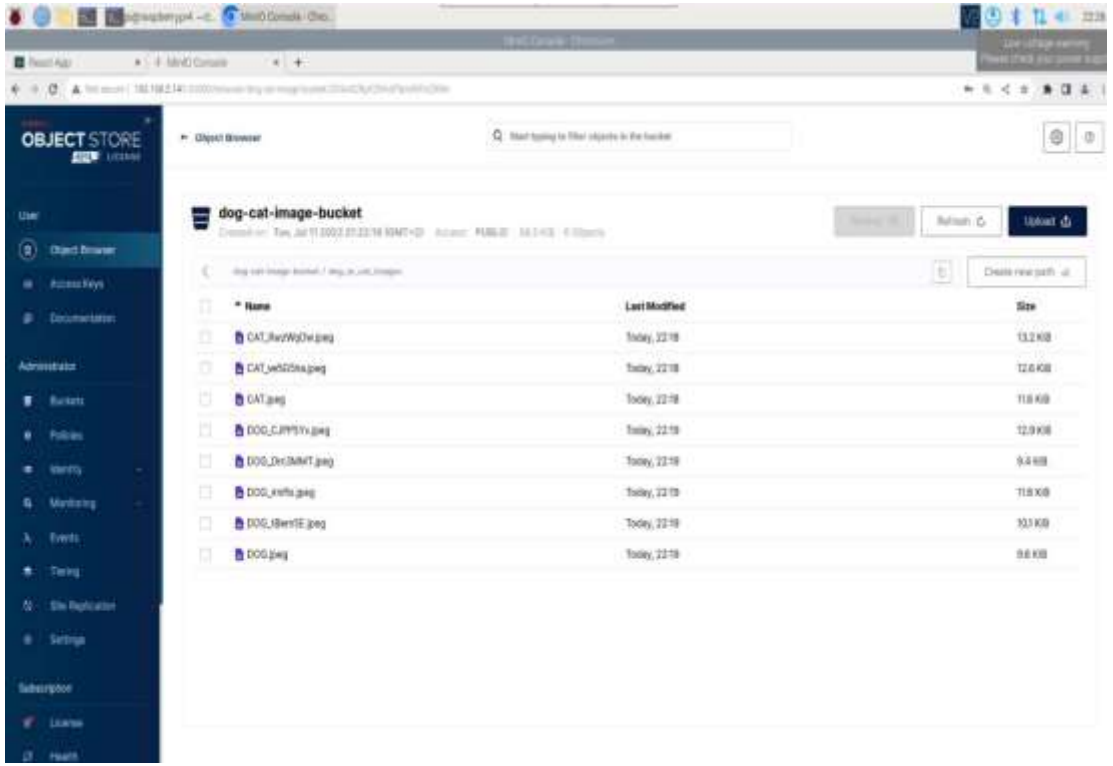
  - Django REST Framework converts the objects into data types that are understandable by javascript and front-end frameworks.
  - **A REST APIis a popular way for systems to expose useful functions and data.**
  - **REST, which stands for representational state transfer, can be made up of one or more resources that can be accessed at a given URL and returned in various formats, like JSON.**

API Working

```
pi@raspberrypi4:~/cat_dog_detector $ ls
api_thread.py  cat_or_dog_detection.py  MobileNetSSD_deploy.caffemodel  MobileNetSSD_deploy.prototxt.txt  __pycache__  requirements.txt
pi@raspberrypi4:~/cat_dog_detector $ python3 cat_or_dog_detection.py
[INFO] loading model...
[INFO] starting video stream...
{"id":3,"image_type":"CAT","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/CAT.jpeg","accuracy":86.20663285255432,"created_date":"2023-07-14T20:18:41.781598Z"}
{"id":4,"image_type":"CAT","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/CAT_ve5G5ha.jpeg","accuracy":94.4508969783783,"created_date":"2023-07-14T20:18:44.264848Z"}
{"id":5,"image_type":"CAT","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/CAT_RwzWqOw.jpeg","accuracy":98.82332682609558,"created_date":"2023-07-14T20:18:47.275988Z"}
{"id":6,"image_type":"DOG","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/DOG.jpeg","accuracy":97.00356125831604,"created_date":"2023-07-14T20:19:07.799155Z"}
{"id":7,"image_type":"DOG","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/DOG_tBern5E.jpeg","accuracy":54.62028384208679,"created_date":"2023-07-14T20:19:11.393645Z"}
{"id":8,"image_type":"DOG","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/DOG_Drc3MMT.jpeg","accuracy":95.43282389640808,"created_date":"2023-07-14T20:19:14.424944Z"}
{"id":9,"image_type":"DOG","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/DOG_CJPP5Yv.jpeg","accuracy":84.69631671905518,"created_date":"2023-07-14T20:19:36.783333Z"}
{"id":10,"image_type":"DOG","image":"http://192.168.2.141:30001/dog-cat-image-bucket/dog_or_cat_images/DOG_irnlfis.jpeg","accuracy":84.05693173408508,"created_date":"2023-07-14T20:19:40.741065Z"}
[INFO] elapsed time: 121.92
[INFO] approx. FPS: 0.61
pi@raspberrypi4:~/cat_dog_detector $
```

Minlo and Django Database





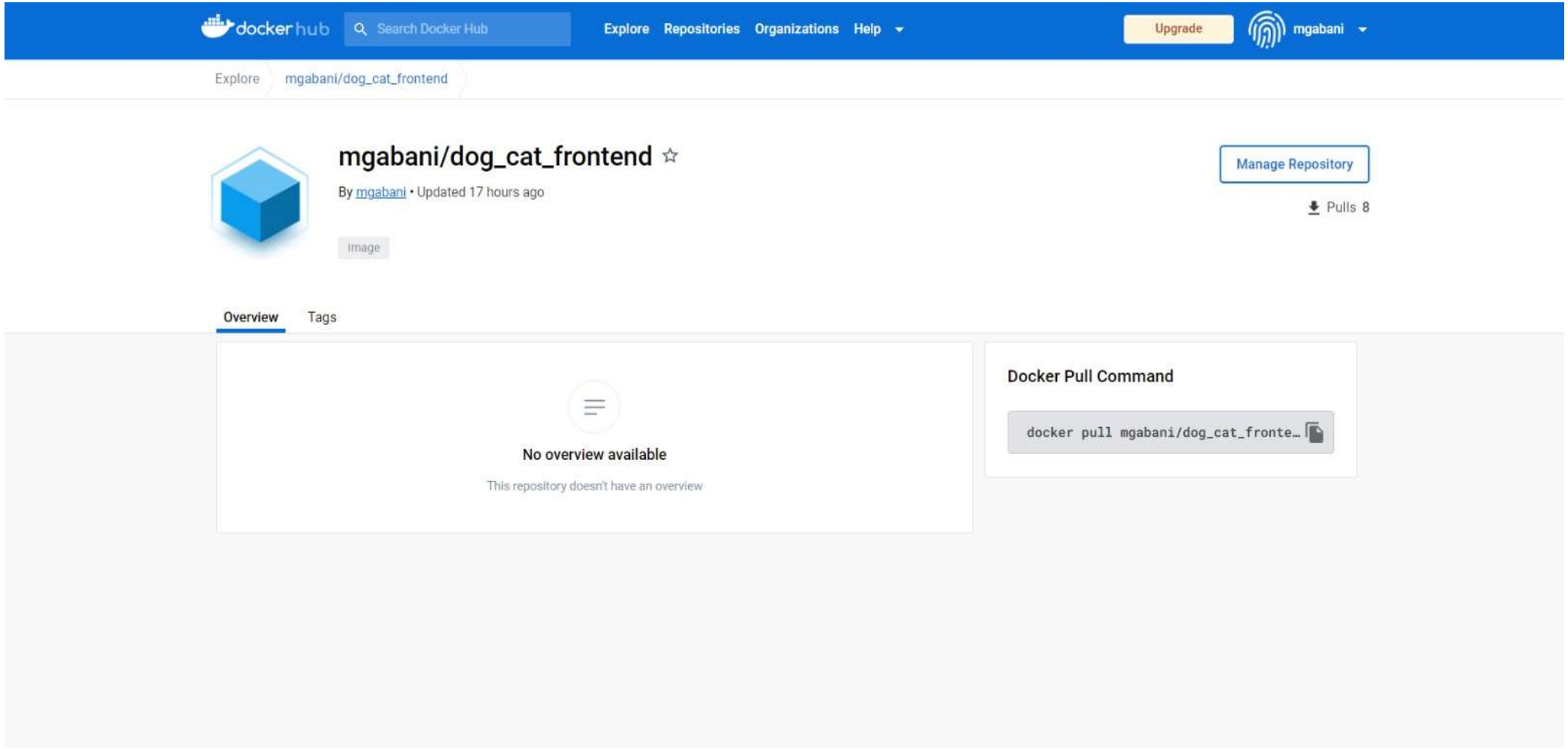
Kubernetes Cluster Application

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- Docker
  - On the K3s cluster, we have setup Docker.
  - So, we can use all docker command In Master Node.
  - In local computer, we must build Docker image for Backend & Frontend rest API and User interface using Dockerfile.
  - We must push this Docker image from local computer to Docker hub.

- Kubernetes Cluster Application's docker Image
  - ➢ Web App Frontend:-

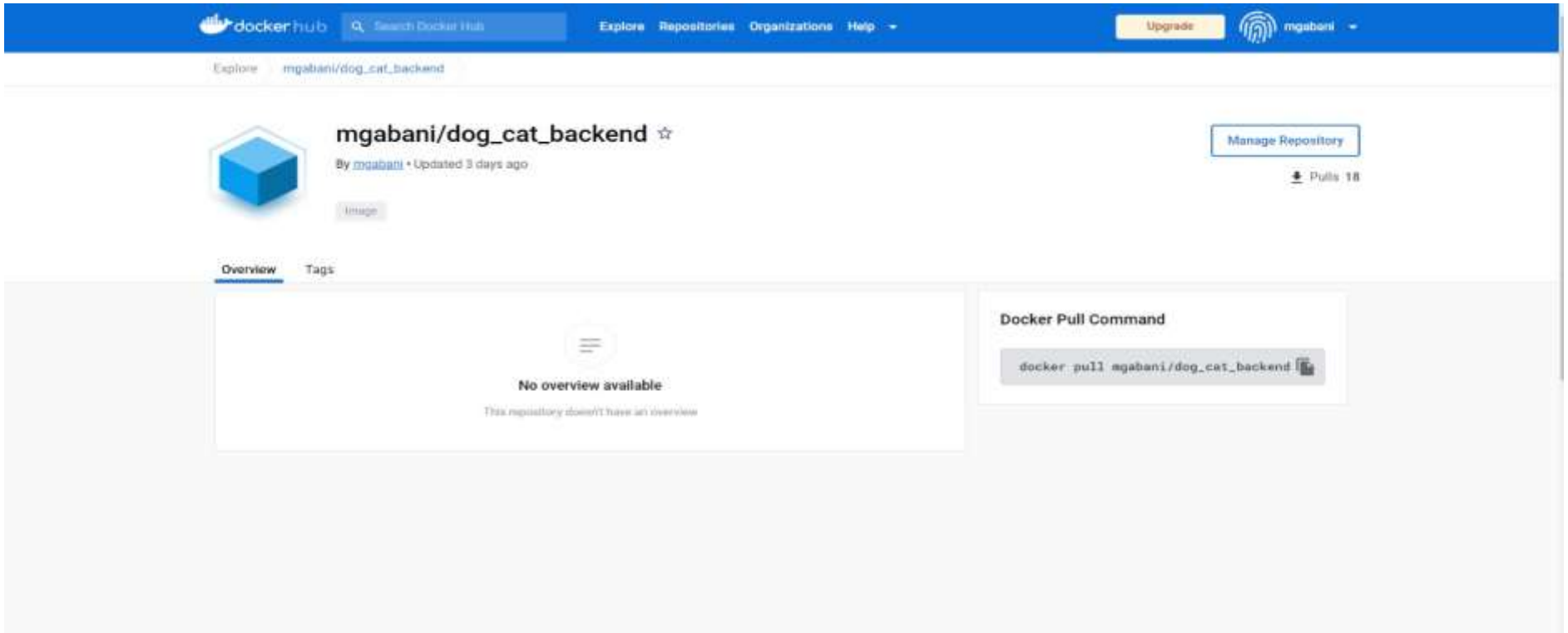
    Web Application is Fronted user interface which is showing the detected images of pet either cat or dog on the web page. The images are retrieving from the MinIo to web browser.


docker hub
Search Docker Hub
Explore
Repositories
Organizations
Help
Upgrade
mgabani
Explore
mgabani/dog_cat_frontend
mgabani/dog_cat_frontend
By mgabani • Updated 17 hours ago
Image
Manage Repository
Pulls 8
Overview
Tags
No overview available
This repository doesn't have an overview
Docker Pull Command
docker pull mgabani/dog_cat_fronte…

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

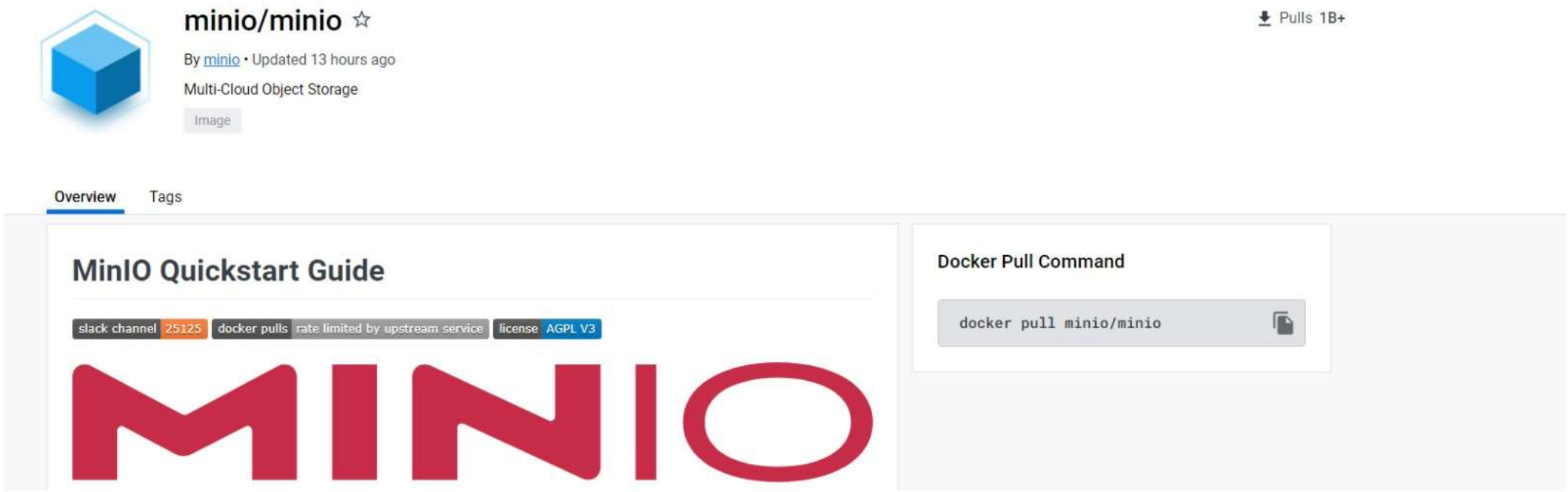
- Web App Backend: -

  Web Application backend is used to store images of cat and dog to later display in the frontend.



MinIo Object Storage

- This is a third-party open-source application which is used in the system to store and access objects received from sensors. MinIO is a high-performance object storage solution that provides an Amazon Web Services S3-compatible API and supports all core S3 features.



FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- MinIO Object Storage Deployment:

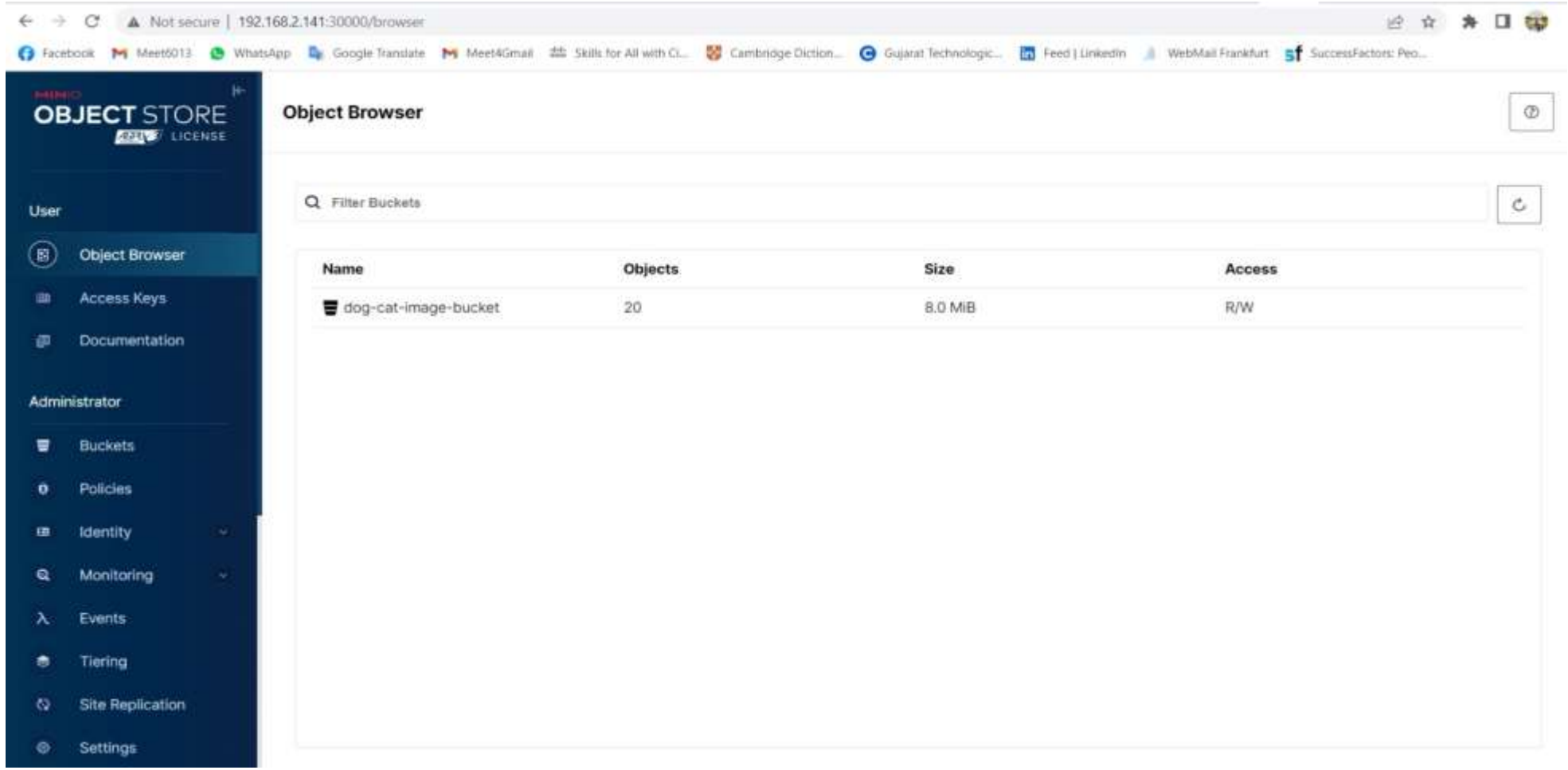
  Go to directory: project /minio_k8s

  Alternatively, you can execute below mentioned commands.

  sudo kubectl apply -f minio-dev.yml
  sudo kubectl apply -f minio.yml
  sudo kubectl apply -f minio-service.yml

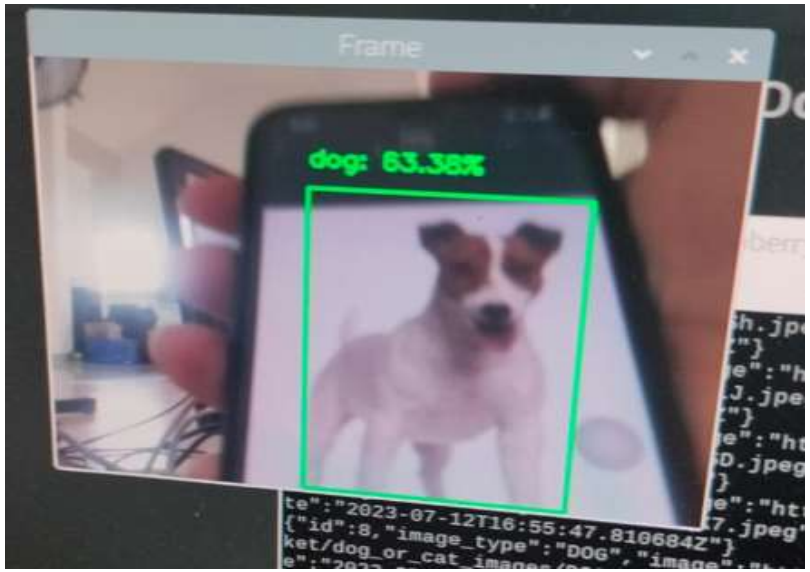
FRANKFURT UNIVERSITY OF APPLIED SCIENCES

- MinIo Console

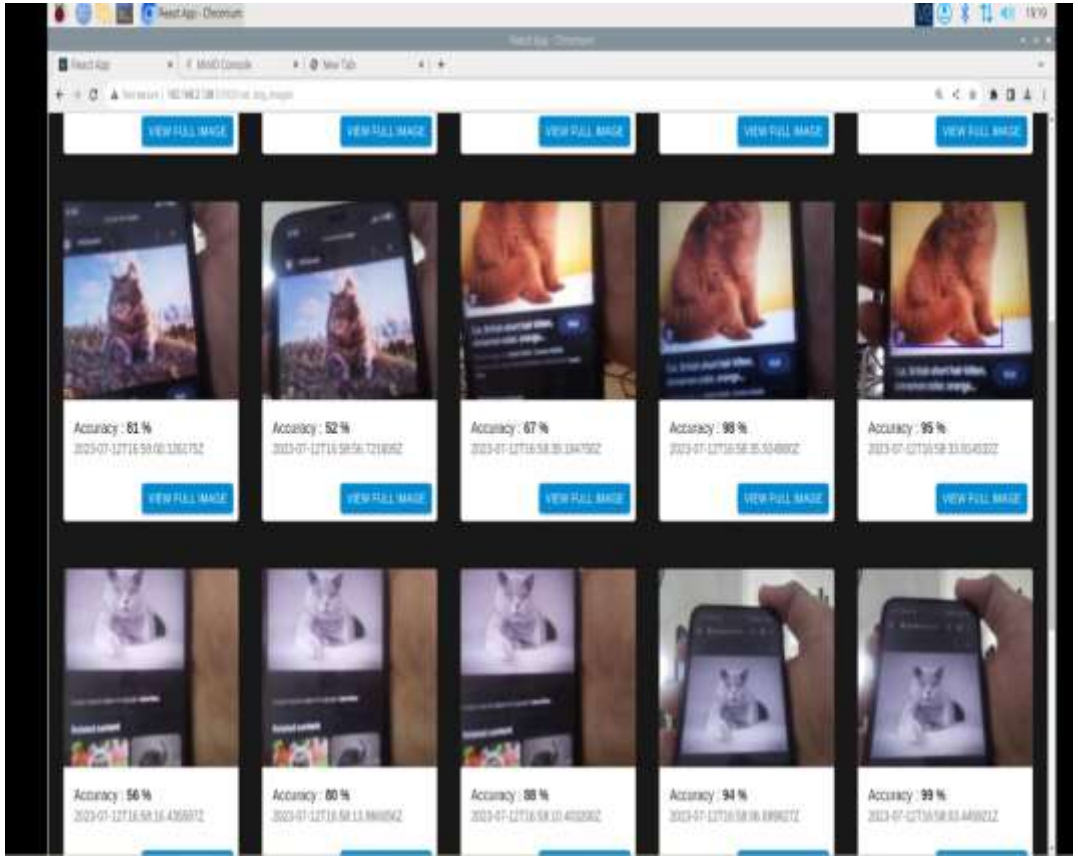


Demo

- After all setup and check all pods are running and services are active as described in all above section.
- Go to Directory: cat_dog_detector and then write this following command in Raspberry Pi 4:
    - Python3 cat_or_dog_detection.py
- After this command camera frame will open and it look like this:



- Frontend: -


VIEW FULL IMAGE
Accuracy : 81 %
Accuracy : 52 %
Accuracy : 67 %
Accuracy : 98 %
Accuracy : 95 %
Accuracy : 50 %
Accuracy : 80 %
Accuracy : 88 %
Accuracy : 94 %
Accuracy : 99 %

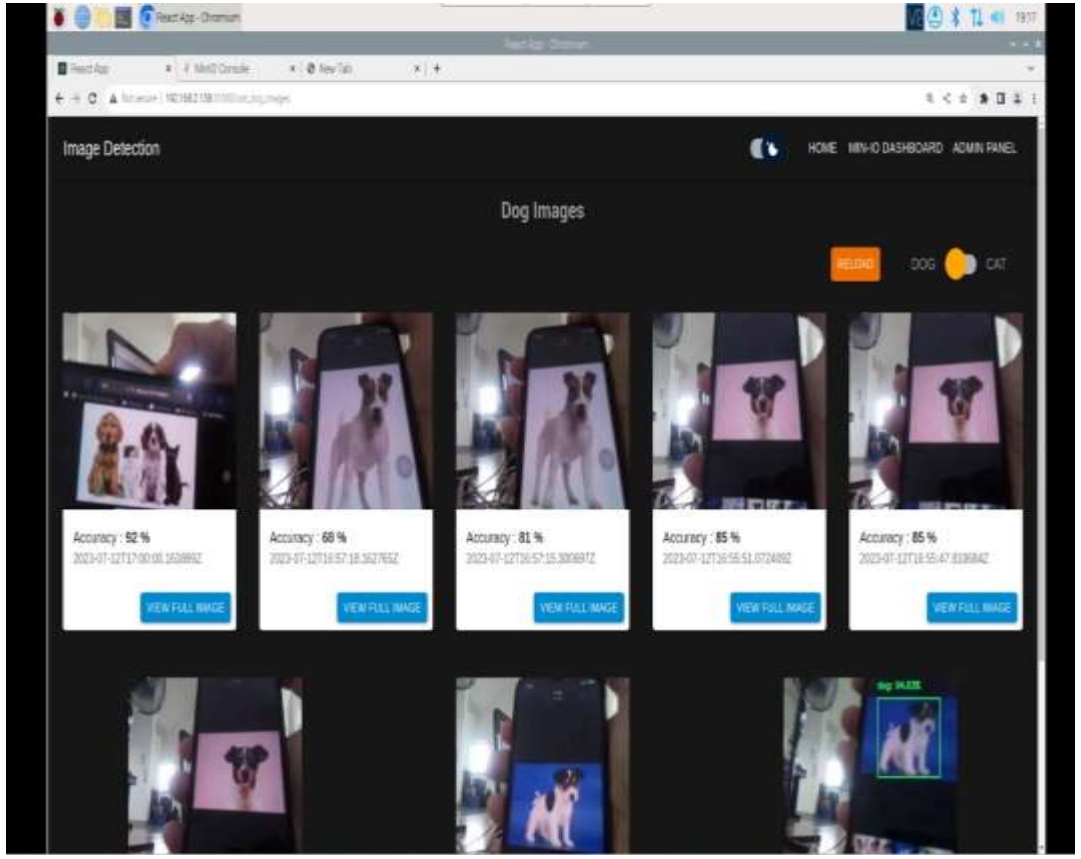


Image Detection
HOME
MINIO DASHBOARD
ADMIN PANEL
Dog Images
RELOAD
DOG
CAT
Accuracy : 92 %
Accuracy : 68 %
Accuracy : 81 %
Accuracy : 85 %
Accuracy : 85 %
VIEW FULL IMAGE

References

- https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
- https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/
- https://medium.com/thinkport/how-to-build-a-raspberry-pi-kubernetes-cluster-with-k3s-76224788576c
- https://www.analyticsvidhya.com/blog/2022/08/how-to-train-a-custom-object-detection-model-with-yolov7/
- https://www.raspberrypi.com/documentation/computers/configuration.html
- https://docs.docker.com/docker-hub/
- https://kubernetes.io/

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Thank you