Cloud Computing SS23

Automatic Cat and Dog Detection Using Edge Computing

Under Guidance of

Prof. Dr. Christian Baun

Member Name: Sumit Chothani [1457445]

Meet Gabani [1442735]

Hardikkumar Suhagiya [1419315]

Rajdeep Kachhadiya [1440737]

Kavita Vaghasiya [1442706]

Jay Togadiya [1413353]

Submission Date: 14th July,2023

Index

- Team Members and Task Distribution

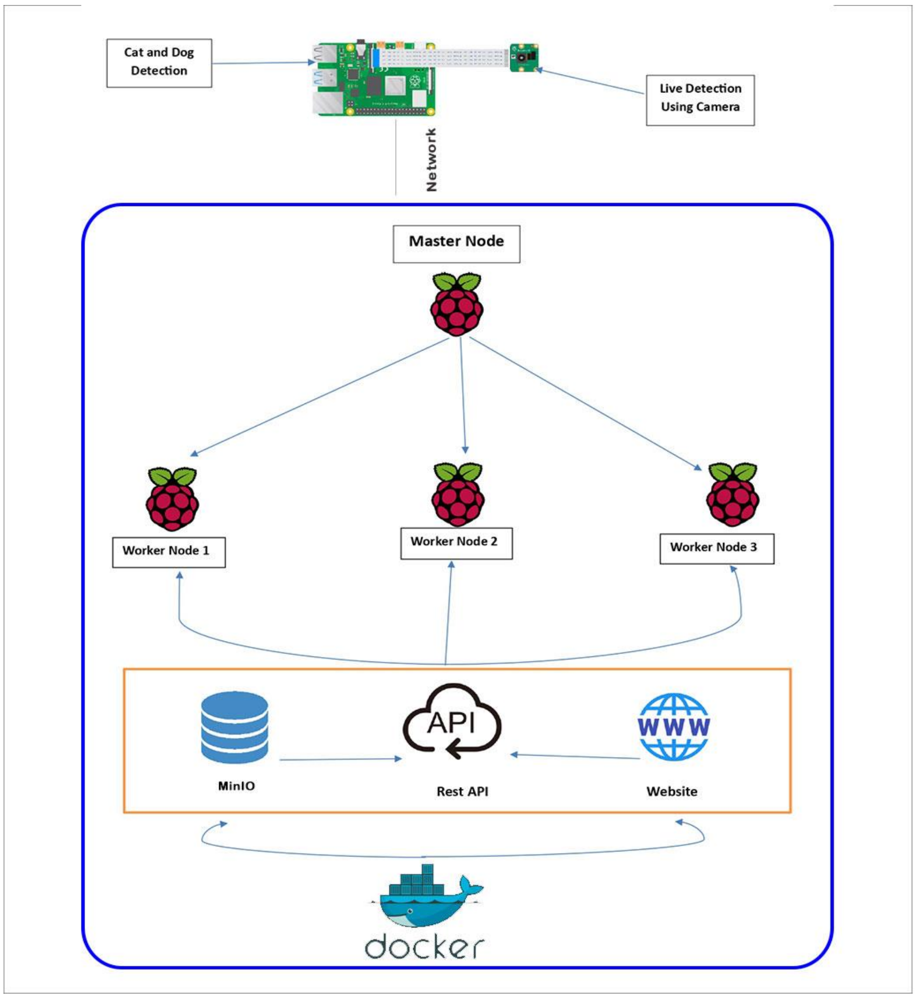
Task Distribution: -	Member Name: -
Initial Hardware Setup, Testing of Hardware	Hardikkumar Suhagiya, Meet Gabani, Jay Togadiya
Senser/Edge Node Setup, $K3_s$ Cluster Setup	Hardikkumar Suhagiya, Meet Gabani
Object Detection Model, Training Of Model	Rajdeep Kachhadiya, Sumit Chothani
Backend & Frontend 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

1. Introduction

Our intention was to create an edge computing solution to identify Cat and Dog at the sensor node and save the data in the cloud service (MinIo).

Data is transmitted to the cloud for processing and storing in cloud computing. Contrarily, 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. Less bandwidth is needed for edge computing solutions because the data processing takes place at the edge node. Solutions for edge computing are scalable, dependable, economical, and provide.

2. Architecture



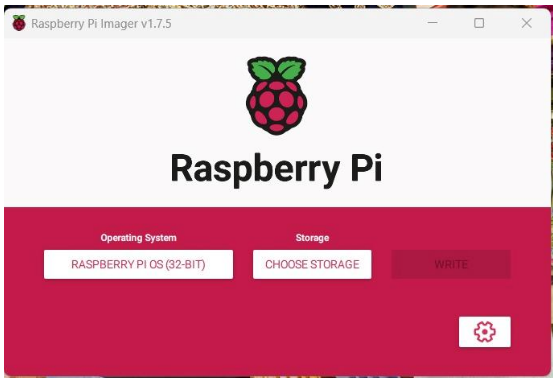
Our Cat and Dog detection project is built as an edge computing solution. We used a Raspberry Pi 4 single board computer (SBC) as the edge/sensor node on which our machine learning model trained to detect Cat and Dog was deployed. The SBC is also equipped with a Raspberry Pi 4 camera module. We also have a K3S cluster built using 4 Raspberry Pi 3 SBC's where 1 SBC behaves as the master node and the rest 3 behave as the worker nodes. Live detection is run using the camera module at the edge node. Whenever a cat and Dog is detected, the frame is captured and is sent to the K3S cluster. Master Node in K3s Work As s Control Pannal. Master node distribute a work load in three worker nodes (knode 1, knode 2, knode 3). 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. Now, we can use docker image in k3s cluster form docker hub. We On the K3S cluster, we have setup a MinIO database which is used to store data relevant to the detection such as confidence level, timestamp of the capture and the image frame itself. We also have setup a React web application on the K3S cluster, which is used to view the detected images. To send the image from the edge node to the K3S cluster, we have used REST API's. The detected images along with the relevant data are sent to the REST API. Upon receiving data, the REST API pushes the data to the MinIO database. from where we are show live data at localhost.

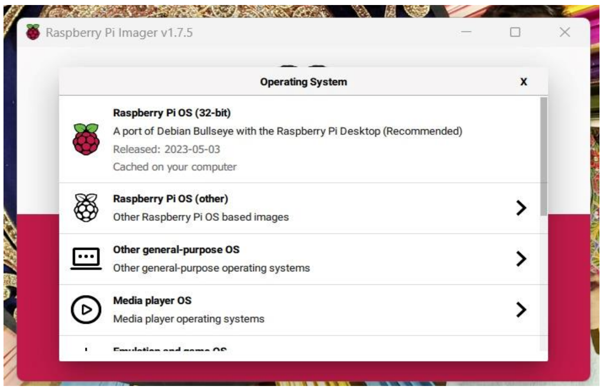
3. Sensor Node

Sensor node is a Raspberry Pi 4 single board computer (SBC) with an attached Raspberry Pi camera module 4. In our project, 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.

3.1. Sensor Node Setup

To setup the sensor node, we installed Raspberry Pi OS 32 bit using the Raspberry Pi Imager tool.





Raspberry Pi OS (32-bit)
A port of Debian Bullseye with the Raspberry Pi Desktop (Recommended)
Released: 2023-05-03
Cached on your computer

Raspberry Pi OS (other)
Other Raspberry Pi OS based images

Other general-purpose OS
Other general-purpose operating systems

Media player OS
Media player operating systems

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.

4. Object Detection Model

- YOLOv5
- Python Library:
  OpenCV
  certifi==2023.5.7
  charset-normalizer==3.1.0
  idna==3.4
  imutils==0.5.4
  numpy==1.24.3
  opencv-python==4.7.0.72
  Pillow==9.5.0
  requests==2.31.0
  urllib3==2.0.3
  python-dotenv==1.0.0
- Roboflow:
  Here, we used it's for train model.

5. Setting up K3S Cluster using Raspberry Pi 3

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.

5.1. Setting up all Raspberry Pi 3

- 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.
- Download Raspberry Pi Imager for your computer and insert an empty SD card to your PC.
- 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.

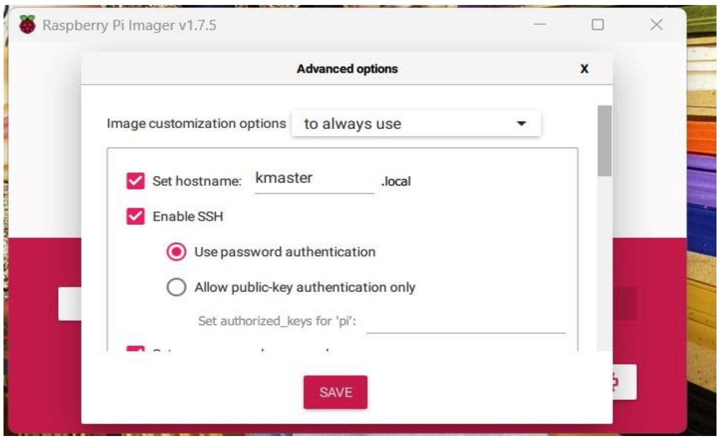


Image customization options: to always use

Set hostname: kmaster .local
Enable SSH
Use password authentication
Allow public-key authentication only
Set authorized_keys for 'pi':

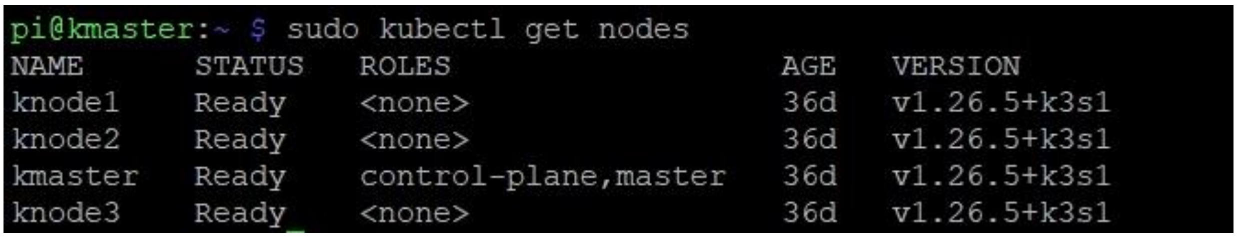
- Once values are configured and saved, select storage option as the SD card and click on 'write' button that fill flash the SD card with chosen OS.
- We repeated this process for all 4 SD cards and named our hosts as kmaster, knode1, knode2, knode3 respectively.
- Insert all the SD cards back to the Raspberry Pi and power up and connect them to your network via LAN switch.

5.2. Setting up k3s cluster

- Install Docker on Master Node:
  sudo apt install docker
  sudo systemctl start docker
  sudo systemctl enable docker
  sudo systemctl status docker

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

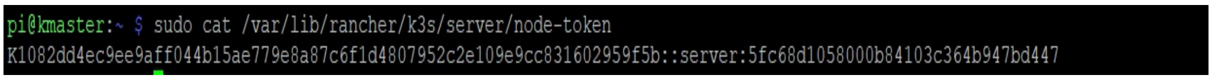
- After successful run, this will start K3S service, and it will automatically create few configurations file as well.
- Now check if the master node is ready using the command:
  sudo kubectl get nodes



```
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
```

- To add agent or worker nodes, first we will generate token from master, this token will then be used by all agent nodes while creating the K3S cluster.

- Run below command and save the generated in a text editor:
  sudo cat /var/lib/rancher/k3s/server/node-token

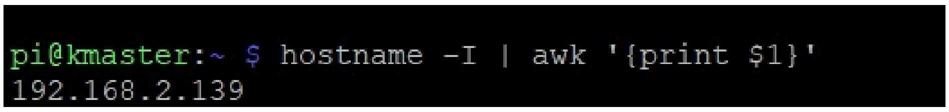


```
pi@kmaster:~ $ sudo cat /var/lib/rancher/k3s/server/node-token
K1082dd4ec9ee9aff044b15ae779e8a87c6f1d4807952c2e109e9cc831602959f5b::server:5fc68d1058000b84103c364b947bd447
```

- Also check the Master IP using the command

  hostname -I | awk '{print $1}'

s



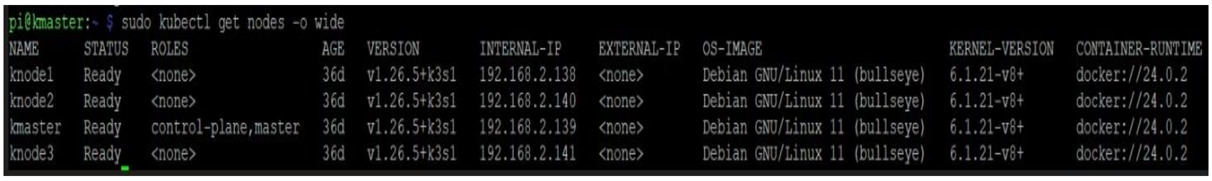
```
pi@kmaster:~ $ hostname -I | awk '{print $1}'
192.168.2.139
```

- Now SSH each worker nodes and following command:
  Install docker In all three worker node using docker installation command.

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

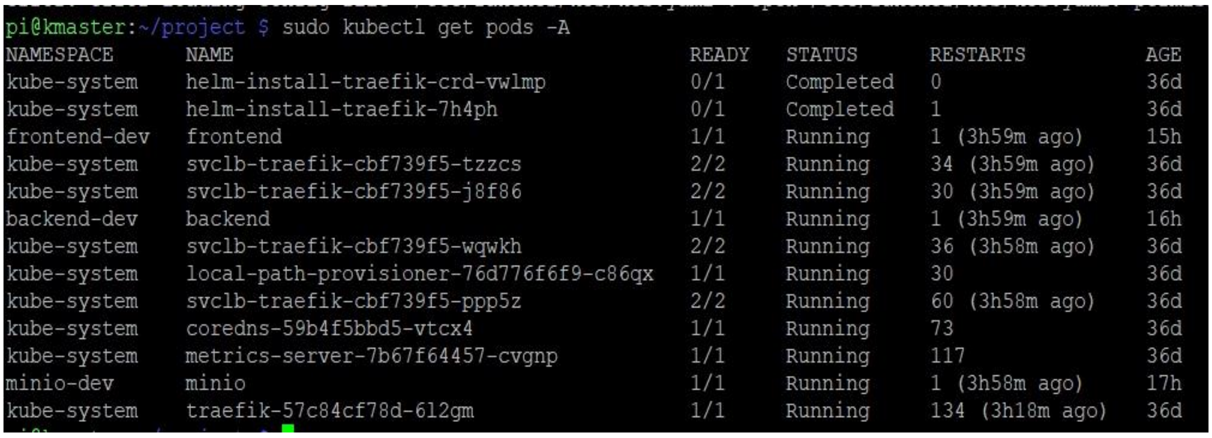
- Now repeat this for all 3 worker nodes.
- Once done our K3S cluster is ready and we can check again in master node by using the command:

  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
```

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

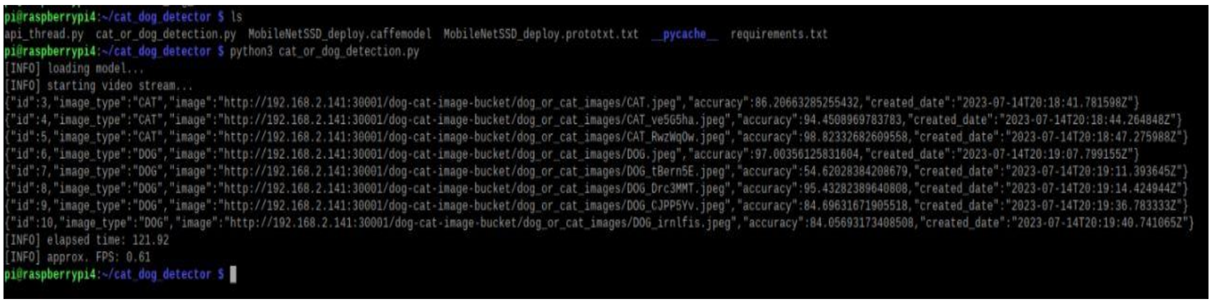


```
pi@kmaster:~/project $ sudo kubectl get pods -A
NAMESPACE     NAME                                      READY   STATUS      RESTARTS        AGE
kube-system   helm-install-traefik-crd-vwlmp            0/1     Completed   0               36d
kube-system   helm-install-traefik-7h4ph                0/1     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-612gm                  1/1     Running     134 (3h18m ago) 36d
```

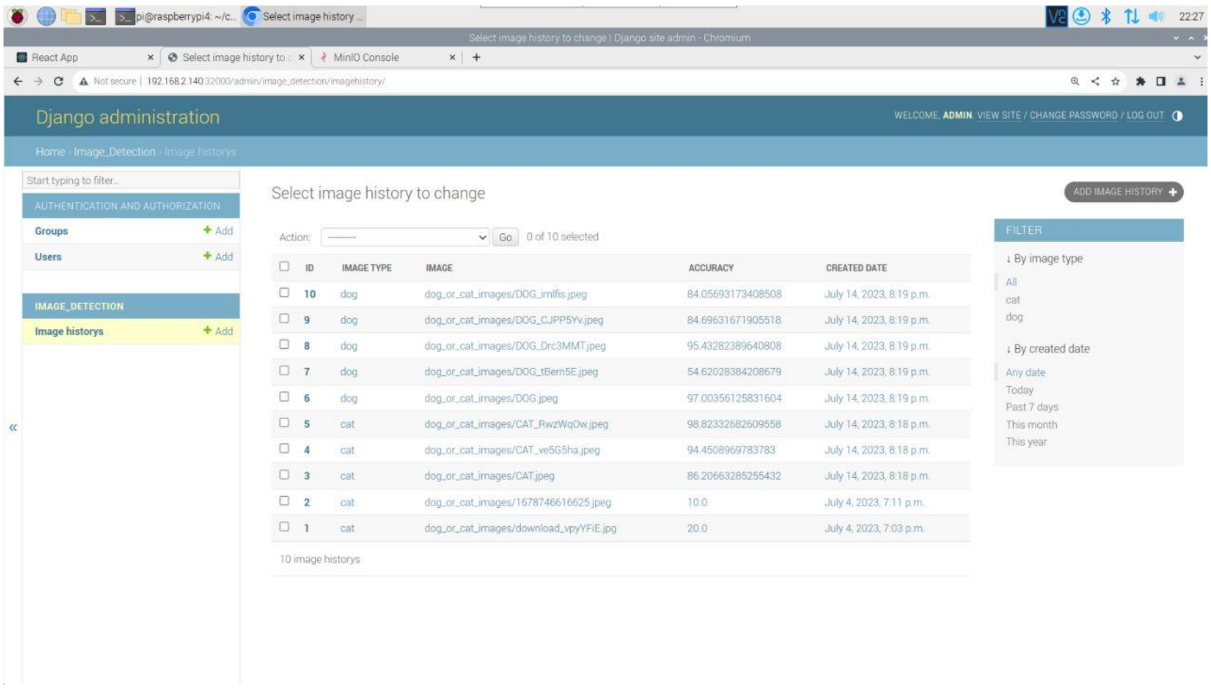
6. REST API:

Technology we have used:

- Django: Django is a popular Python web framework that provides a robust set of tools and libraries for building web applications, including APIs (Application Programming Interfaces). It follows the model-view-controller (MVC) architectural pattern and emphasizes code reusability, simplicity, and rapid development.
- Django REST Framework: Django REST Framework simplifies API creation in Django by providing serialization, authentication, and permission handling, along with view sets and routers for CRUD operations. It also offers request/response handling, pagination, filtering, and built-in API documentation, making API development faster and more efficient.
- Django Filter: Django Filter is a robust package that simplifies API development in Django by enabling effortless filtering of query sets based on user-defined parameters. With its declarative syntax for defining filters, users can easily apply complex filtering operations, making API creation more straightforward and efficient.
- REST API: REST API uses Https requests to access and use data. There are some methods which are Get, Post, Put, Delete used to read, delete, update, and create data as needed.
- Docker: Docker simplifies API development with Django REST Framework by providing a containerized environment that ensures consistent and reproducible deployments. It streamlines the setup process, isolates dependencies, and allows easy scaling and deployment across different environments, enhancing the development and deployment experience.



```
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 $
```

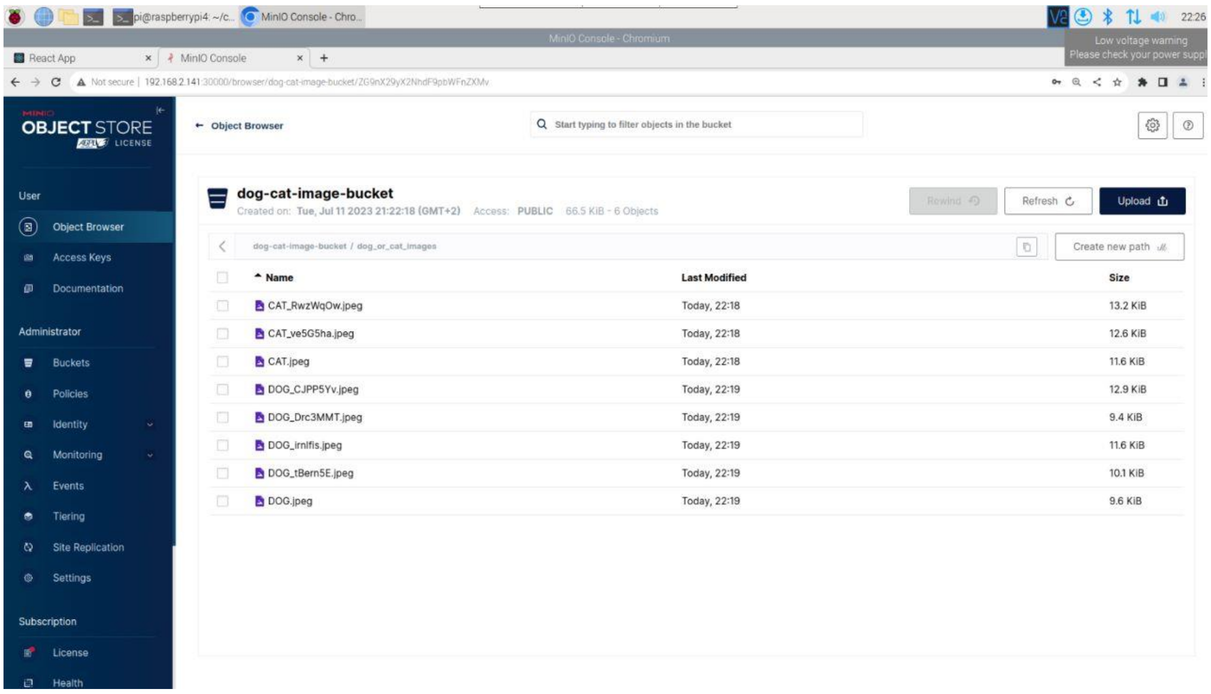

Select image history to change | Django site admin - Chromium
Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Image_Detection › Image historys
AUTHENTICATION AND AUTHORIZATION
Groups
Users
IMAGE_DETECTION
Image historys
ADD IMAGE HISTORY
FILTER
By image type
All
cat
dog
By created date
Any date
Today
Past 7 days
This month
This year

Select image history to change

Action: --------- Go 0 of 10 selected

ID	IMAGE TYPE	IMAGE	ACCURACY	CREATED DATE
10	dog	dog_or_cat_images/DOG_irnlfis.jpeg	84.05693173408508	July 14, 2023, 8:19 p.m.
9	dog	dog_or_cat_images/DOG_CJPP5Yv.jpeg	84.69631671905518	July 14, 2023, 8:19 p.m.
8	dog	dog_or_cat_images/DOG_Drc3MMT.jpeg	95.43282389640808	July 14, 2023, 8:19 p.m.
7	dog	dog_or_cat_images/DOG_tBern5E.jpeg	54.62028384208679	July 14, 2023, 8:19 p.m.
6	dog	dog_or_cat_images/DOG.jpeg	97.00356125831604	July 14, 2023, 8:19 p.m.
5	cat	dog_or_cat_images/CAT_RwzWqOw.jpeg	98.82332682609558	July 14, 2023, 8:18 p.m.
4	cat	dog_or_cat_images/CAT_ve5G5ha.jpeg	94.4508969783783	July 14, 2023, 8:18 p.m.
3	cat	dog_or_cat_images/CAT.jpeg	86.20663285255432	July 14, 2023, 8:18 p.m.
2	cat	dog_or_cat_images/1678746616625.jpeg	10.0	July 4, 2023, 7:11 p.m.
1	cat	dog_or_cat_images/download_vpyYFiE.jpg	20.0	July 4, 2023, 7:03 p.m.

10 image historys



dog-cat-image-bucket
Created on: Tue, Jul 11 2023 21:22:18 (GMT+2) Access: PUBLIC 66.5 KiB - 6 Objects

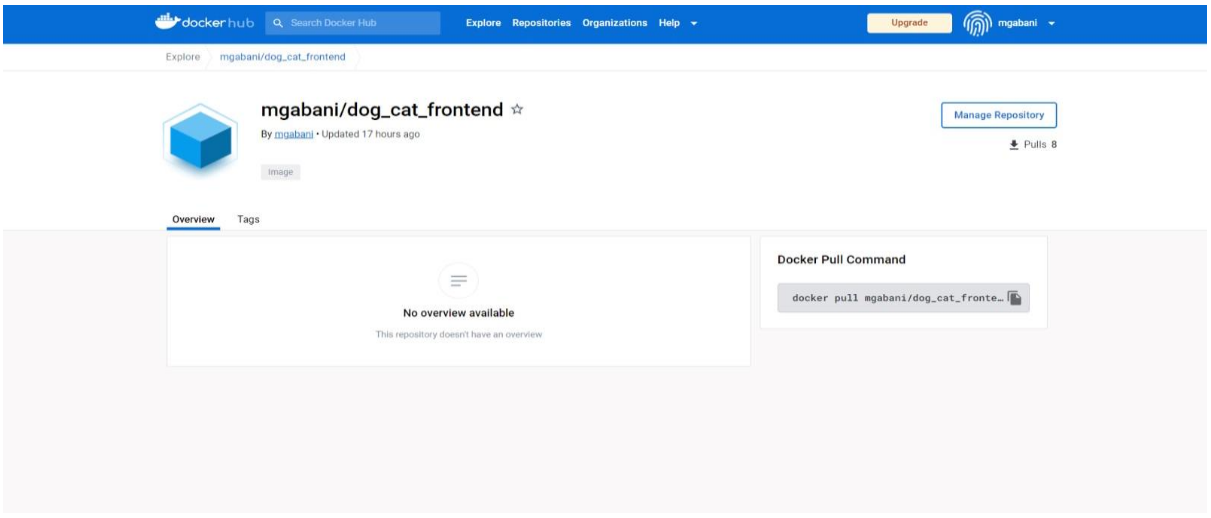
dog-cat-image-bucket / dog_or_cat_images

Name	Last Modified	Size
CAT_RwzWqOw.jpeg	Today, 22:18	13.2 KiB
CAT_ve5G5ha.jpeg	Today, 22:18	12.6 KiB
CAT.jpeg	Today, 22:18	11.6 KiB
DOG_CJPP5Yv.jpeg	Today, 22:19	12.9 KiB
DOG_Drc3MMT.jpeg	Today, 22:19	9.4 KiB
DOG_irnlfis.jpeg	Today, 22:19	11.6 KiB
DOG_tBern5E.jpeg	Today, 22:19	10.1 KiB
DOG.jpeg	Today, 22:19	9.6 KiB

7. Kubernetes Cluster Application

7.1. 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.



mgabani/dog_cat_frontend

By mgabani • Updated 17 hours ago

Pulls 8

No overview available

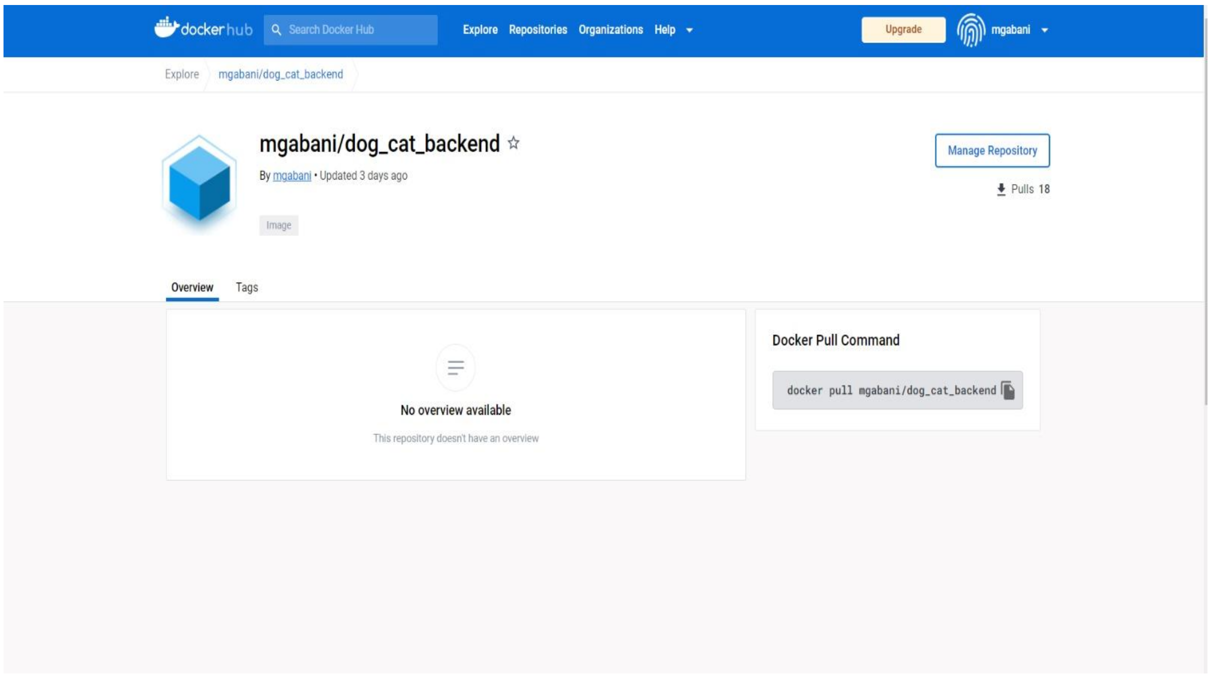
This repository doesn't have an overview

Docker Pull Command

```
docker pull mgabani/dog_cat_fronte…
```

7.2. Web App Backend:

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



mgabani/dog_cat_backend

By mgabani • Updated 3 days ago

Pulls 18

No overview available

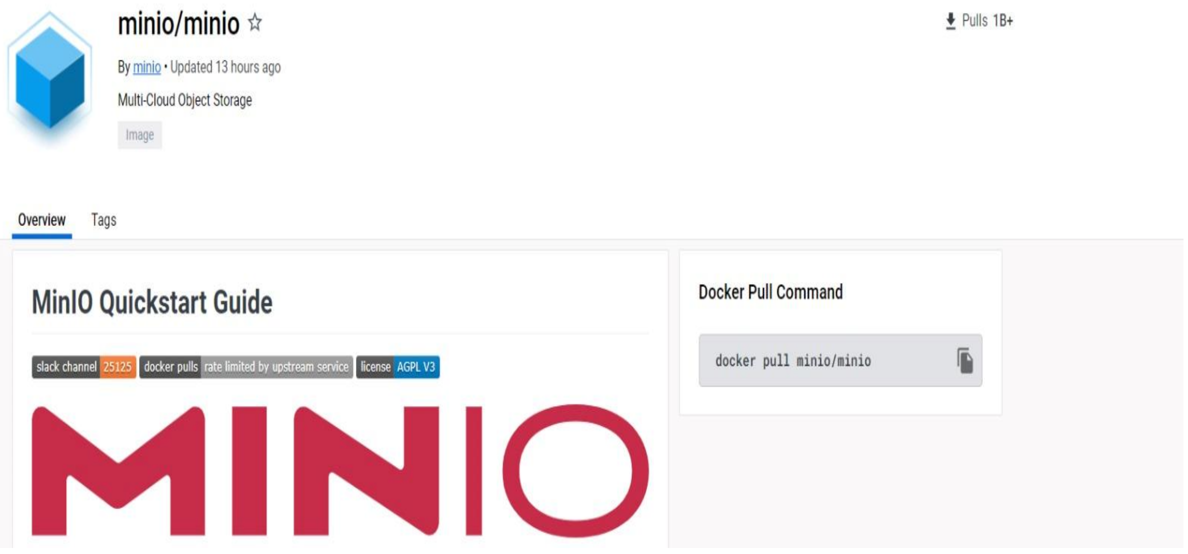
This repository doesn't have an overview

Docker Pull Command

```
docker pull mgabani/dog_cat_backend
```

7.3. 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.


minio/minio ☆

By minio • Updated 13 hours ago

Multi-Cloud Object Storage

Image

Pulls 1B+

Overview Tags

MinIO Quickstart Guide

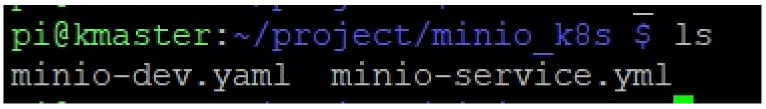
slack channel 25125 | docker pulls rate limited by upstream service | license AGPL V3

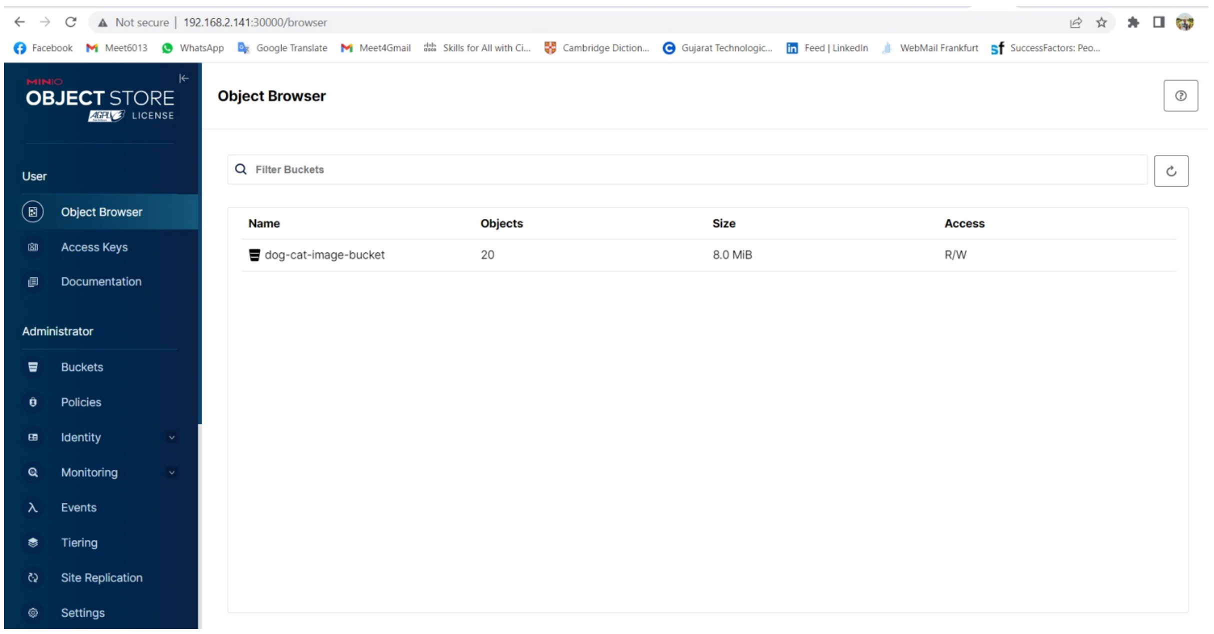
Docker Pull Command

```
docker pull minio/minio
```

7.4. 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


```
pi@kmaster:~/project/minio_k8s $ ls
minio-dev.yaml  minio-service.yml
```



Not secure | 192.168.2.141:30000/browser

MINIO OBJECT STORE AGPLV3 LICENSE

Object Browser

User
Object Browser
Access Keys
Documentation

Administrator
Buckets
Policies
Identity
Monitoring
Events
Tiering
Site Replication
Settings

Filter Buckets

Name	Objects	Size	Access
dog-cat-image-bucket	20	8.0 MiB	R/W

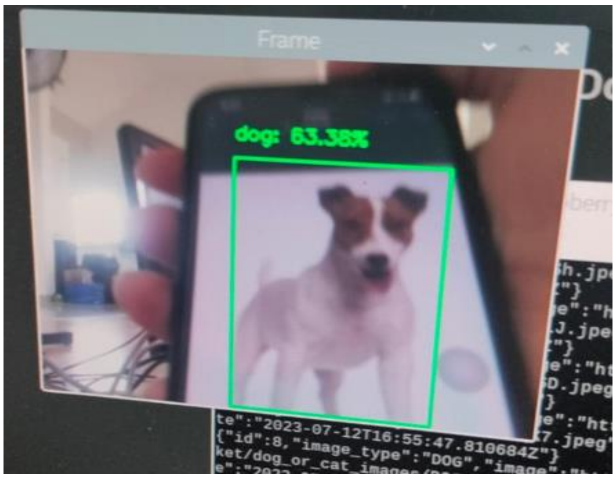
8. Testing of Cat and Dog Detection: -

After all setup and check all pods are running and services are active as described in all above section.

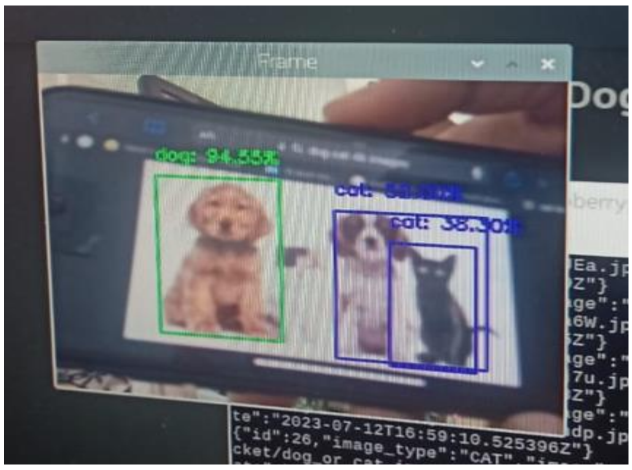
Go to Directory: cat_dog_detector and then write this following command in Raspberry Pi 4:

```
Python3 cat_or_dog_detection.py
```

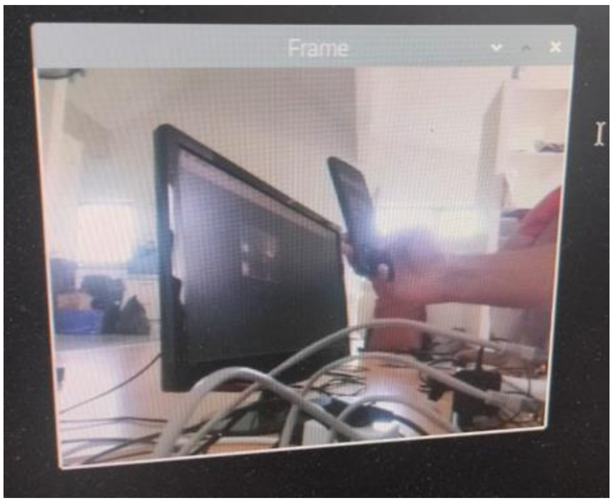
After this command camera frame will open and it look like this:

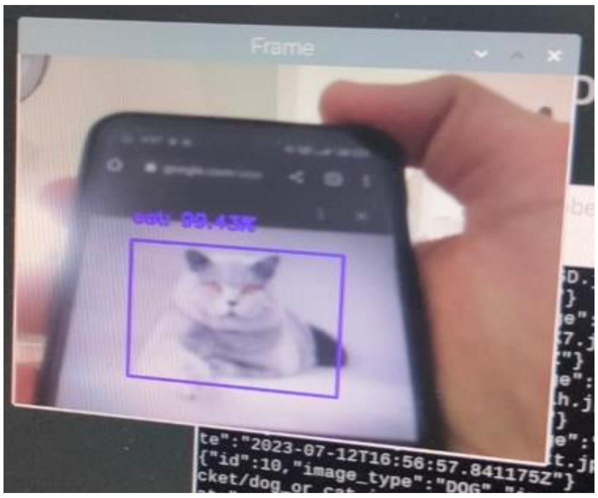


```
te":"2023-07-12T16:55:47.810684Z"}
{"id":8,"image_type":"DOG","image"
ket/dog_or_cat_images/
```



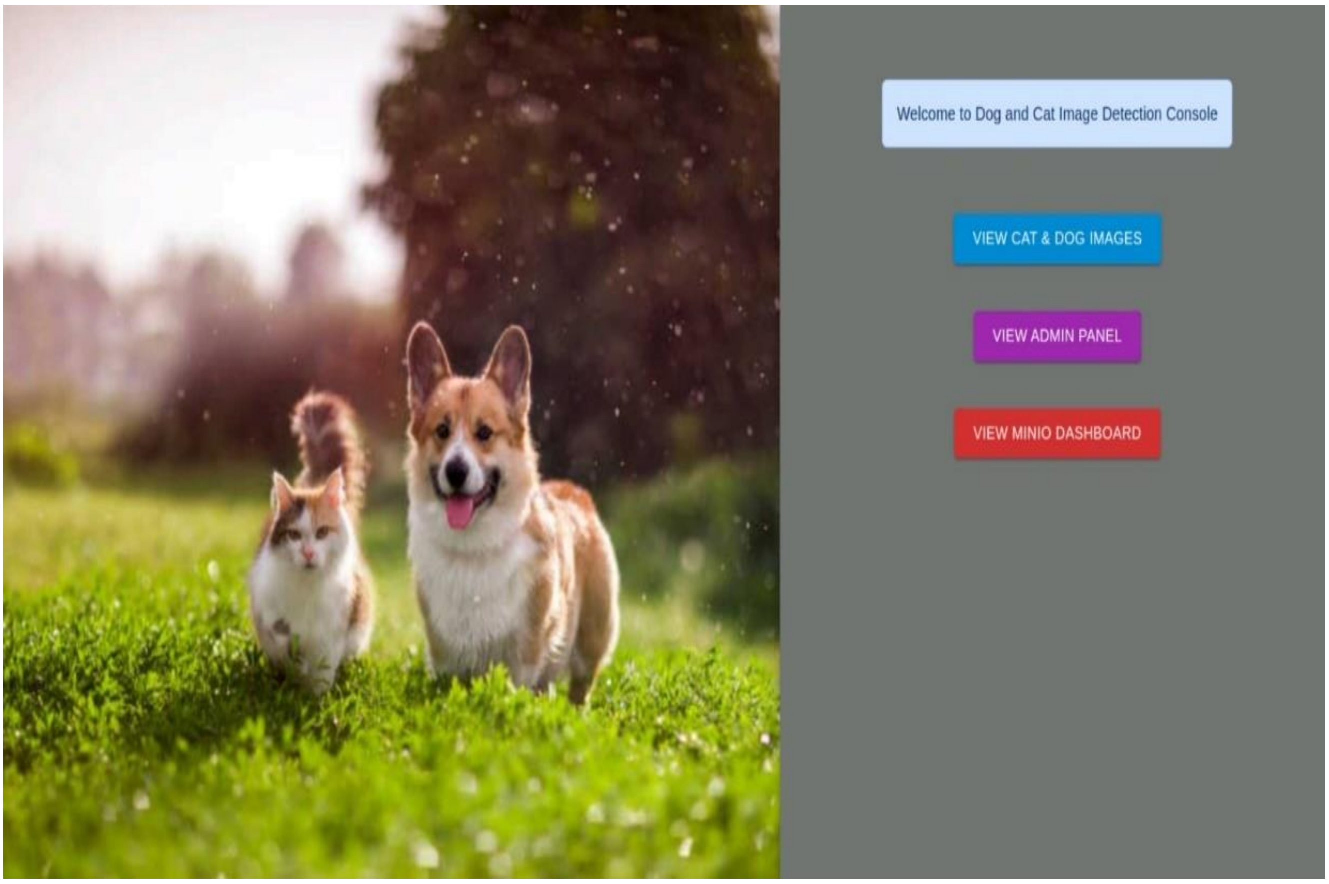
```
te":"2023-07-12T16:59:10.525396Z"}
{"id":26,"image_type":"CAT",
cket/dog_or_cat
```

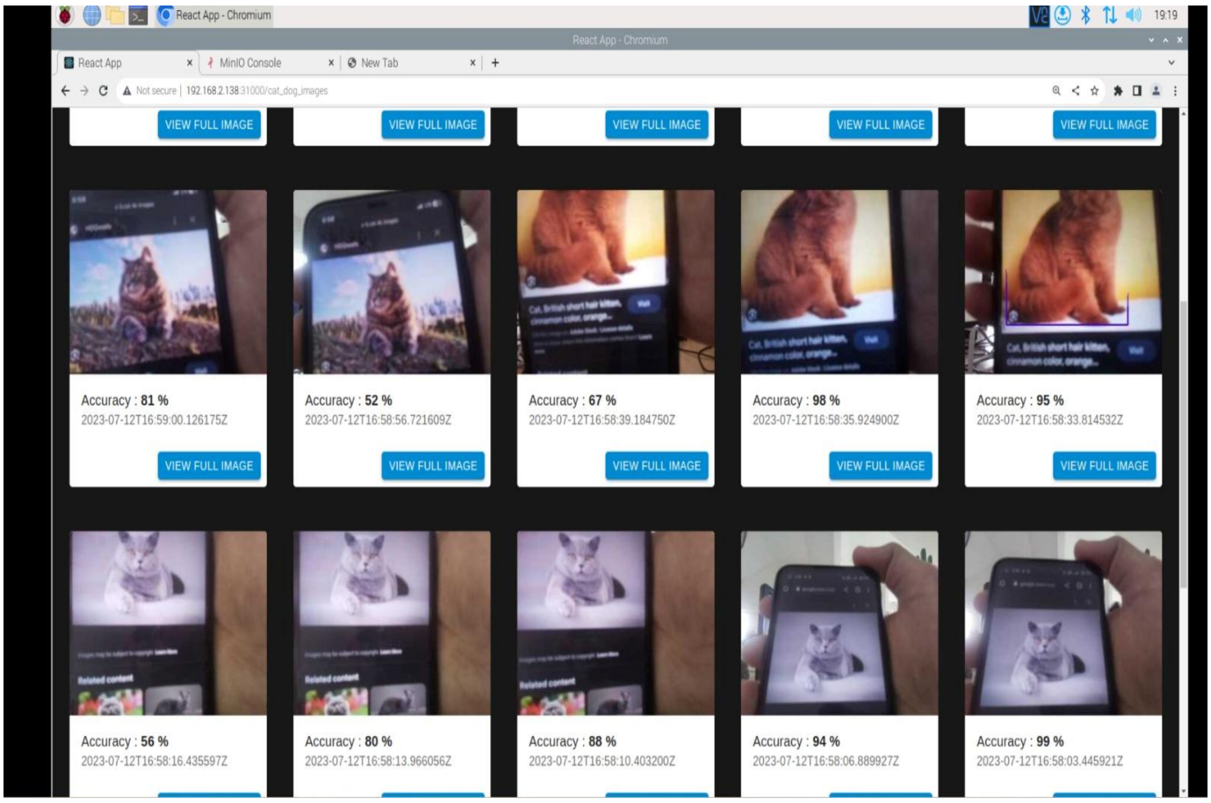




```
te":"2023-07-12T16:56:57.841175Z"}
{"id":10,"image_type":"DOG",
cket/dog_or_cat
```

Frontend: -





Accuracy : 81 %
2023-07-12T16:59:00.126175Z

Accuracy : 52 %
2023-07-12T16:58:56.721609Z

Accuracy : 67 %
2023-07-12T16:58:39.184750Z

Accuracy : 98 %
2023-07-12T16:58:35.924900Z

Accuracy : 95 %
2023-07-12T16:58:33.814532Z

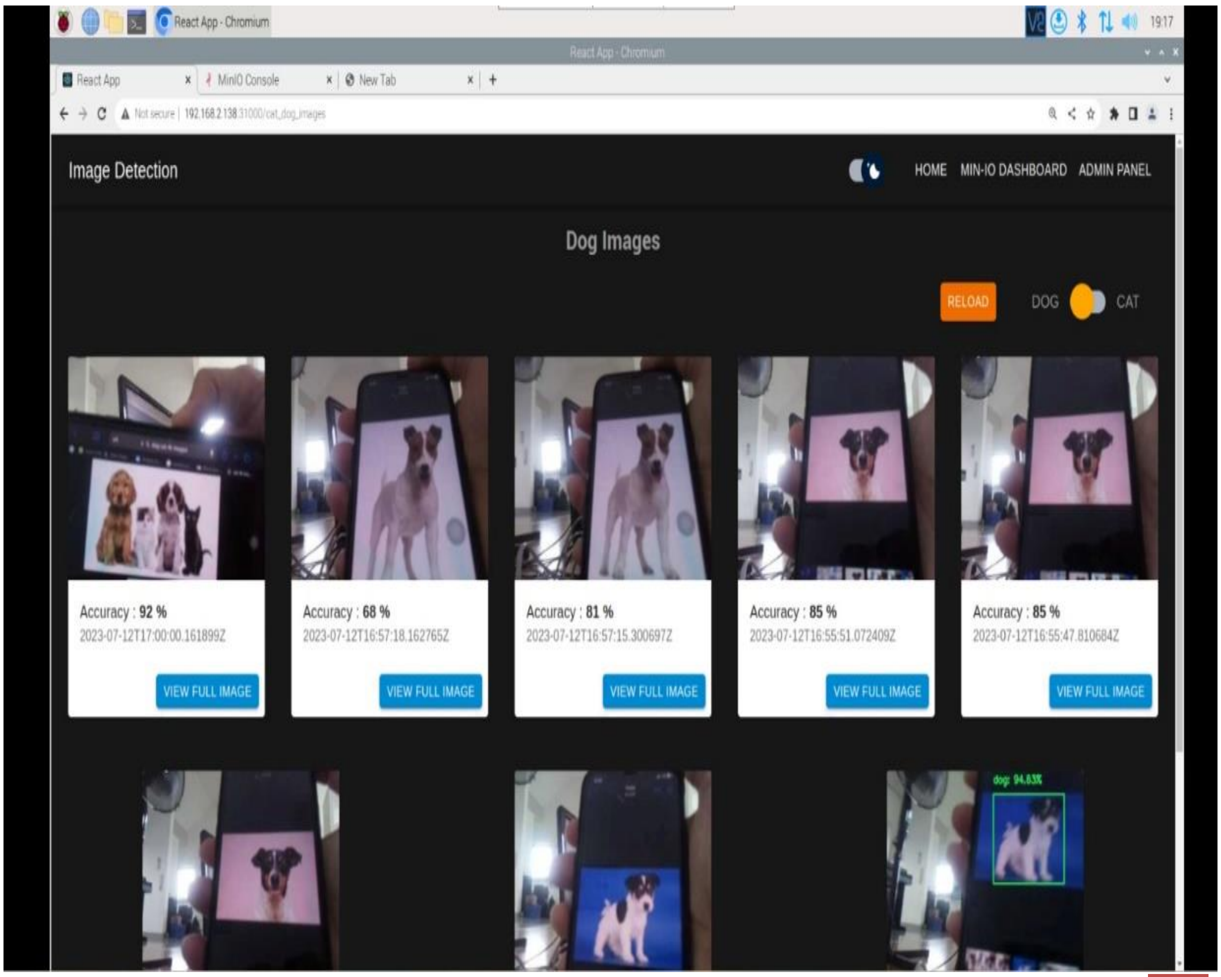
Accuracy : 56 %
2023-07-12T16:58:16.435597Z

Accuracy : 80 %
2023-07-12T16:58:13.966056Z

Accuracy : 88 %
2023-07-12T16:58:10.403200Z

Accuracy : 94 %
2023-07-12T16:58:06.889927Z

Accuracy : 99 %
2023-07-12T16:58:03.445921Z



Accuracy : 92 %
2023-07-12T17:00:00.161899Z

Accuracy : 68 %
2023-07-12T16:57:18.162765Z

Accuracy : 81 %
2023-07-12T16:57:15.300697Z

Accuracy : 85 %
2023-07-12T16:55:51.072409Z

Accuracy : 85 %
2023-07-12T16:55:47.810684Z

9. GitHub Links

GitHub Link for object detection model: -
https://github.com/rajdeep9603/cat_dog_detector.git

GitHub Link for Backend & Frontend API: -
https://github.com/rajdeep9603/cat_dog_backend.git

GitHub Link for User Interface (Frontend): -
https://github.com/rajdeep9603/cat_dog_backend.git

10. References

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