

# Automatic Pest Detection



Cloud Computing - Group 3 - 09.02.2023

# Contents

- Sensor Node
  - Camera
  - Telegram bot
  - Object detection
- Cluster
  - Virtual Machines
  - Persistent Storage MinIO
  - Persistent Storage NFS
  - WebApp Environment
- Architecture
- Web App
- Live Demonstration

# Sensor Node - Camera Mounting

- Stable mounting enables blur-free images
- Constant camera frames
- Flexible due to pitch and yaw angle
- Easy to adjust
- Safe and comfortable operation since no permanent touching and holding of single-board computer, ribbon cable and camera module needed
- All connectors remain directly accessible

# Sensor Node - Camera Software Setup

- Raspberry Pi OS Bullseye Full 64-bit
- Accessing, configuring and controlling camera by using Python library Picamera2
- Recording and saving pictures repeatedly for YOLO
- Separated from object detection

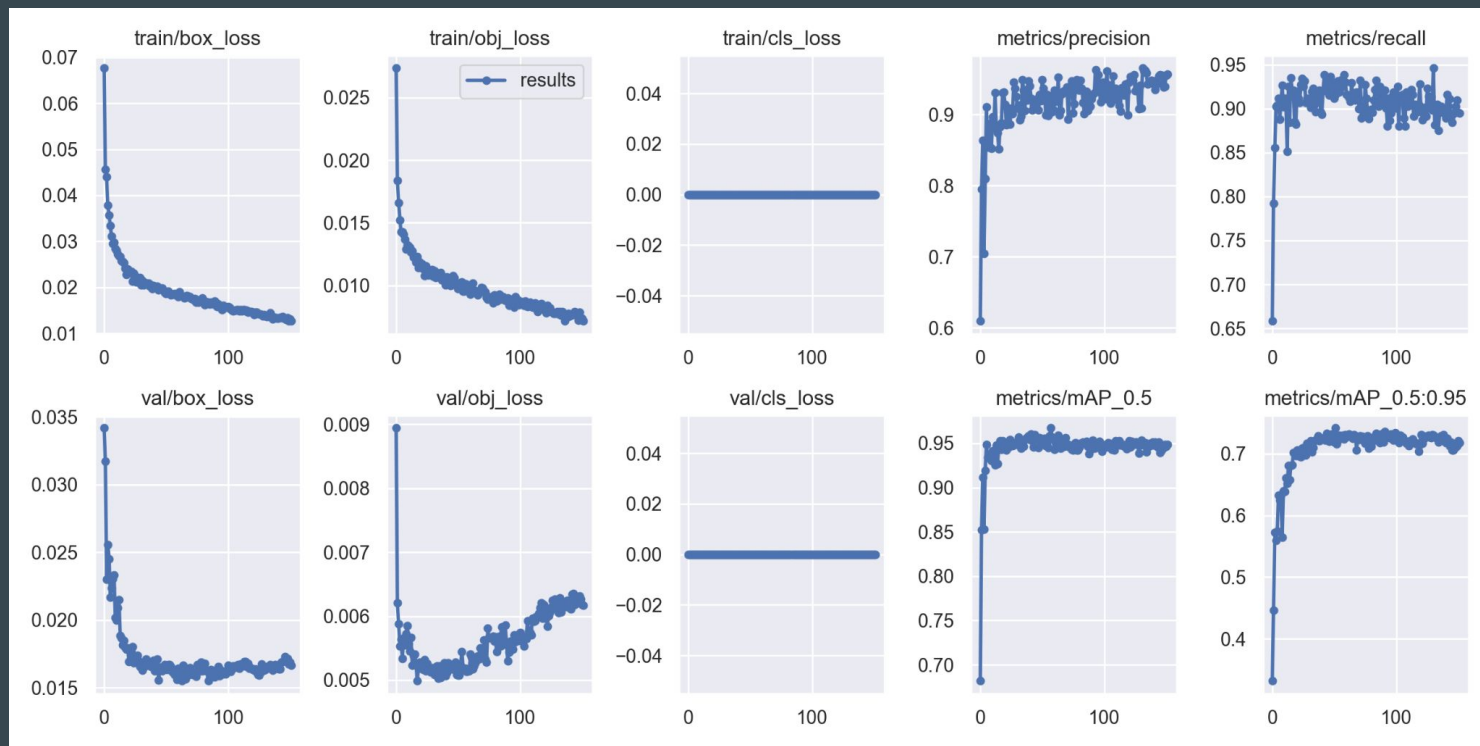
# Sensor Node - Telegram Bot

- active configurable notification system
- recognize and reply to input messages
- add new users automatically
- toggle alert on/off
- input interval with error detection
- sends pictures of rats with counting and confidence

# Sensor Node - Object detection

- YOLOv5
- Data collection/labeling: Roboflow
  - provided dataset
  - Lucas sister rat pictures
  - ~ 1.500 pictures
- Training:
  - CUDA
  - ~ 200 epochs
- Problems:
  - Pytorch needed Python  $\leq 3.8$  -> docker file
  - not compatible with picamera2 (python  $\geq 3.9$ ) -> docker compose with volumes for file exchange

# Sensor Node - Object detection



# Cluster - General

Requirements:

- Store images which were classified as “Rat”
- Show images which were classified as “Rat”

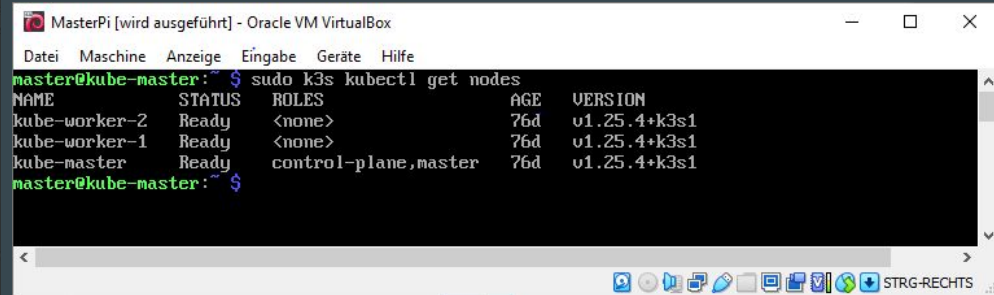
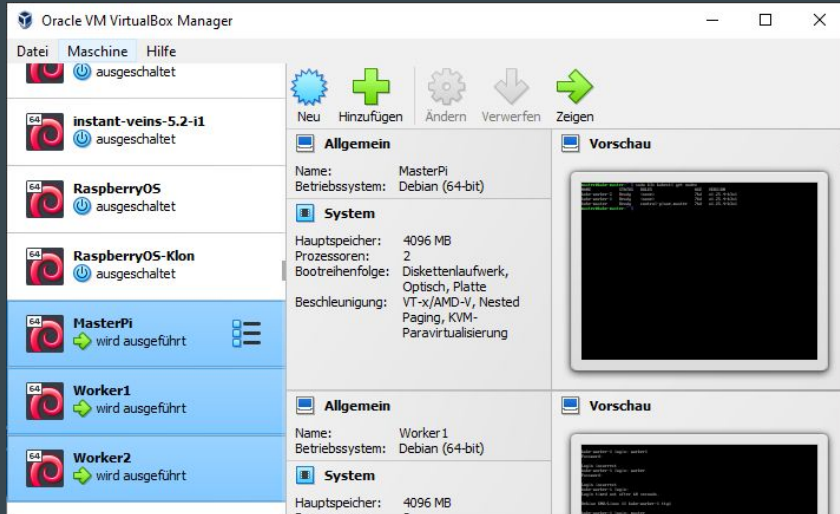
Therefore we did:

- Create persistent storage to save the images
- Create a WebApp to display the images
- Create a suitable environment on the cluster for the WebApp



# Cluster - Virtual Machines

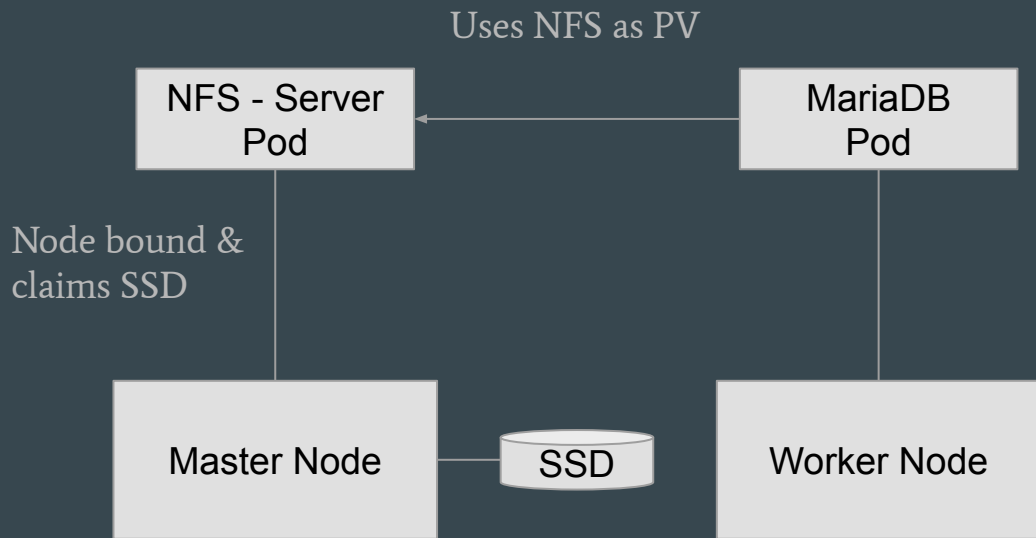
- 3 people working on cluster
- Reduce dependency on hardware
- Simulate Cluster on Raspberries by using Virtual Machines



# Cluster - Persistent Storage MinIO

- Why?
  - Object storage native to Kubernetes
  - Binary less than 100MB
  - Support of K3S
  - Designed for Edge computing
- ..and why not?
  - Pod run. MinIO had repeated crashes
  - Took a long time to come up
  - Worked simultaneously on a diff. solution
    - Problems where hard to fix
    - The oth. solution proved easier to implement

# Cluster - Persistent Storage NFS



```
pi@masternode:~$ sudo kubectl exec -it mariadb-mariadb-5b54ffdf79-gwf2z -n work-sp
root@mariadb-mariadb-5b54ffdf79-gwf2z:~# mariadb -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.3.36-MariaDB-0+deb10u2 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SHOW DATABASES;
+-----+
| Database |
+-----+
| db_pest  |
| information_schema |
| mysql    |
| performance_schema |
+-----+
4 rows in set (0.007 sec)
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
webapp-client-v1-58d5d6fb98-ngxgs	1/1	Running	1 (38m ago)	39h	10.42.0.32	masternode
webapp-server-v1-57f6466959-t4dj9	1/1	Running	4 (38m ago)	40h	10.42.1.14	workernode1
nfs-server-cb5779b4c-5qock	1/1	Running	0	15m	10.42.0.36	masternode
mariadb-5b54ffdf79-zb9cj	1/1	Running	0	40h	10.42.1.18	workernode1

# Cluster - WebApp Environment

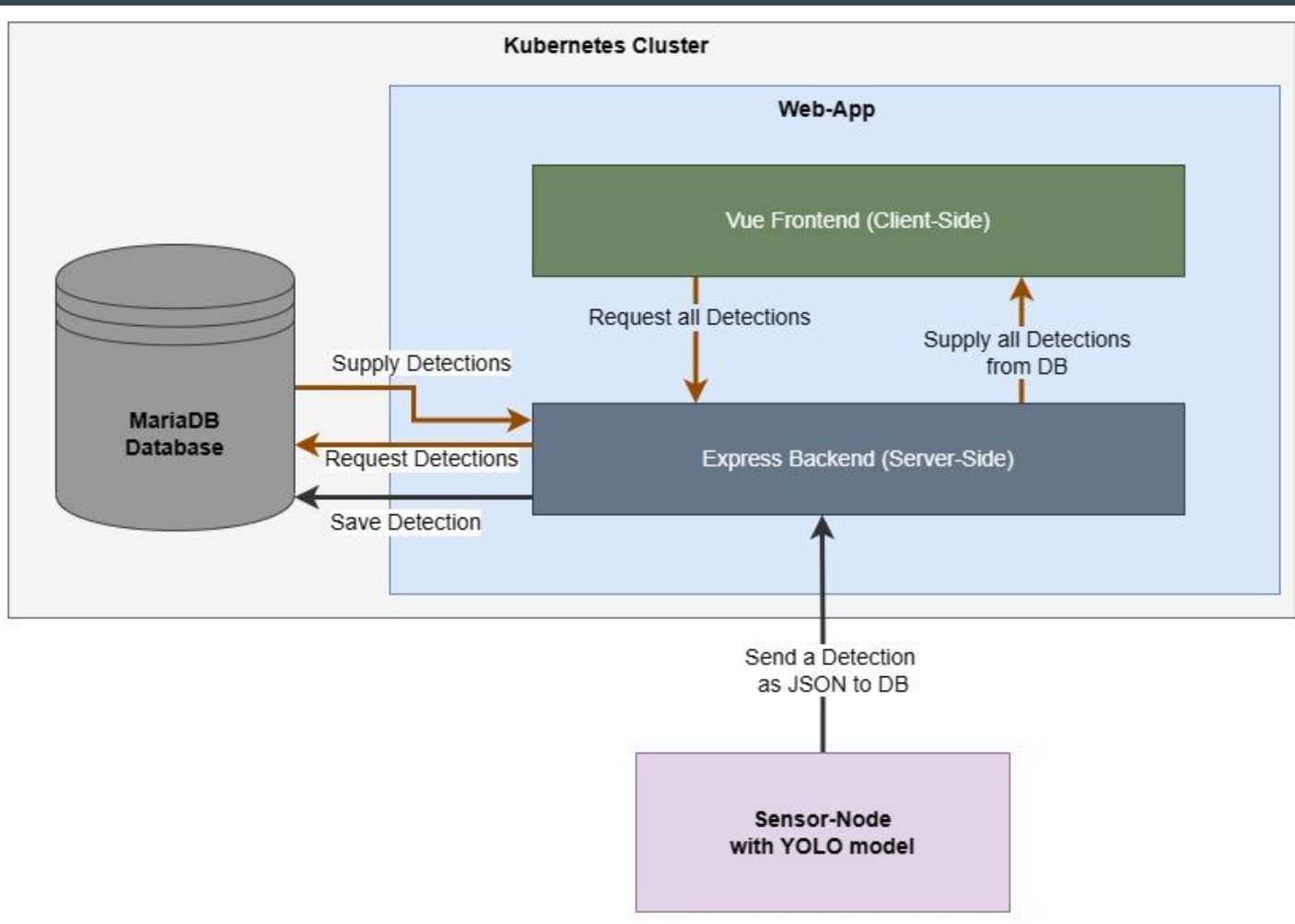
Environment to smoothly run the cluster:

- Store images in Repository
  - Kubernetes needs to pull images frequently
  - DockerHub (Already used in manifest)
- Create network services for the WebApp
  - Service kind: “Loadbalancer”
  - Backend and frontend have to be accessible from outside of the cluster

```
pr@minikube:~$ kubectl get svc -n webapp
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nfs-server	ClusterIP	10.43.184.230	<none>	2049/TCP,20048/TCP,111/TCP	21d
mariadb-mariadb	ClusterIP	10.43.184.232	<none>	3306/TCP	21d
webapp-client	LoadBalancer	10.43.246.61	192.168.188.23,192.168.188.26,192.168.188.29,192.168.188.31	8080:30880/TCP	21d
webapp-server	LoadBalancer	10.43.177.5	192.168.188.23,192.168.188.26,192.168.188.29,192.168.188.31	5000:30500/TCP	21d

# Architecture



# Web-App

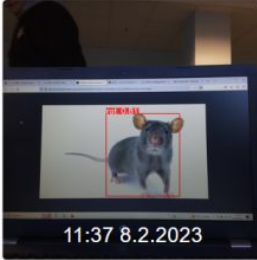
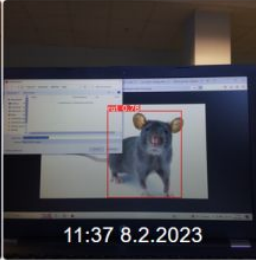
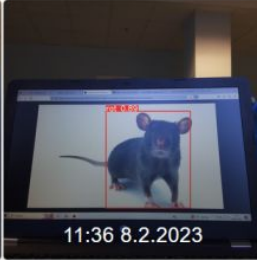









- Buildable and runnable with Docker/ Docker-Compose
- Client-Side:
  - Vue open -source JavaScript framework
  - Used for user interface
  - Basic JS services to backend-API to fetch data

# Web-App

- Server-Side
  - Express open-source JS framework
  - Used for building APIs
  - Connect to database
  - Fetch data from database and send to frontend
  - Save new detections to database
- Constant consultation with team for interface and cluster-setup

# Web App Example

☰ Rat-Detector

 <p>11:37 8.2.2023</p> <p>Confidence: 80.77</p> <p>Number of Rats: 1</p>	 <p>11:37 8.2.2023</p> <p>Confidence: 80.77</p> <p>Number of Rats: 1</p>	 <p>11:36 8.2.2023</p> <p>Confidence: 88.91</p> <p>Number of Rats: 1</p>	 <p>11:35 8.2.2023</p> <p>Confidence: 85,15,78.99</p> <p>Number of Rats: 2</p>	 <p>11:34 8.2.2023</p> <p>Confidence: 89.45,87.28</p> <p>Number of Rats: 2</p>	 <p>11:34 8.2.2023</p> <p>Confidence: 90,85</p> <p>Number of Rats: 2</p>
 <p>10:54 8.2.2023</p> <p>Confidence: 89</p>	 <p>10:53 8.2.2023</p> <p>Confidence: 89</p>	 <p>10:46 8.2.2023</p> <p>Confidence: 83</p>	 <p>10:46 8.2.2023</p> <p>Confidence: 89</p>	 <p>10:46 8.2.2023</p> <p>Confidence: 89</p>	 <p>10:46 8.2.2023</p> <p>Confidence: 83</p>



# Live Demonstration