Project Presentation

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Automatic Pet Detection With Edge Computing

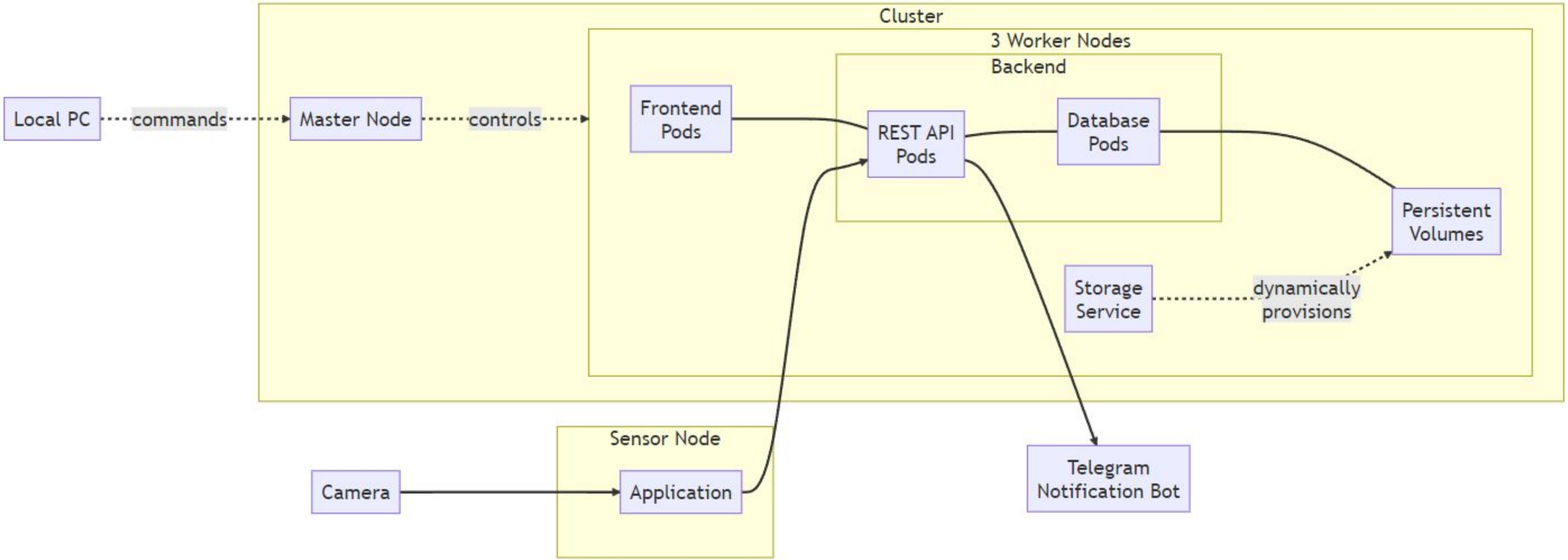
by

Group 2 - Cloud Computing (SS2023)

19.07.2023

Contents

System Architecture

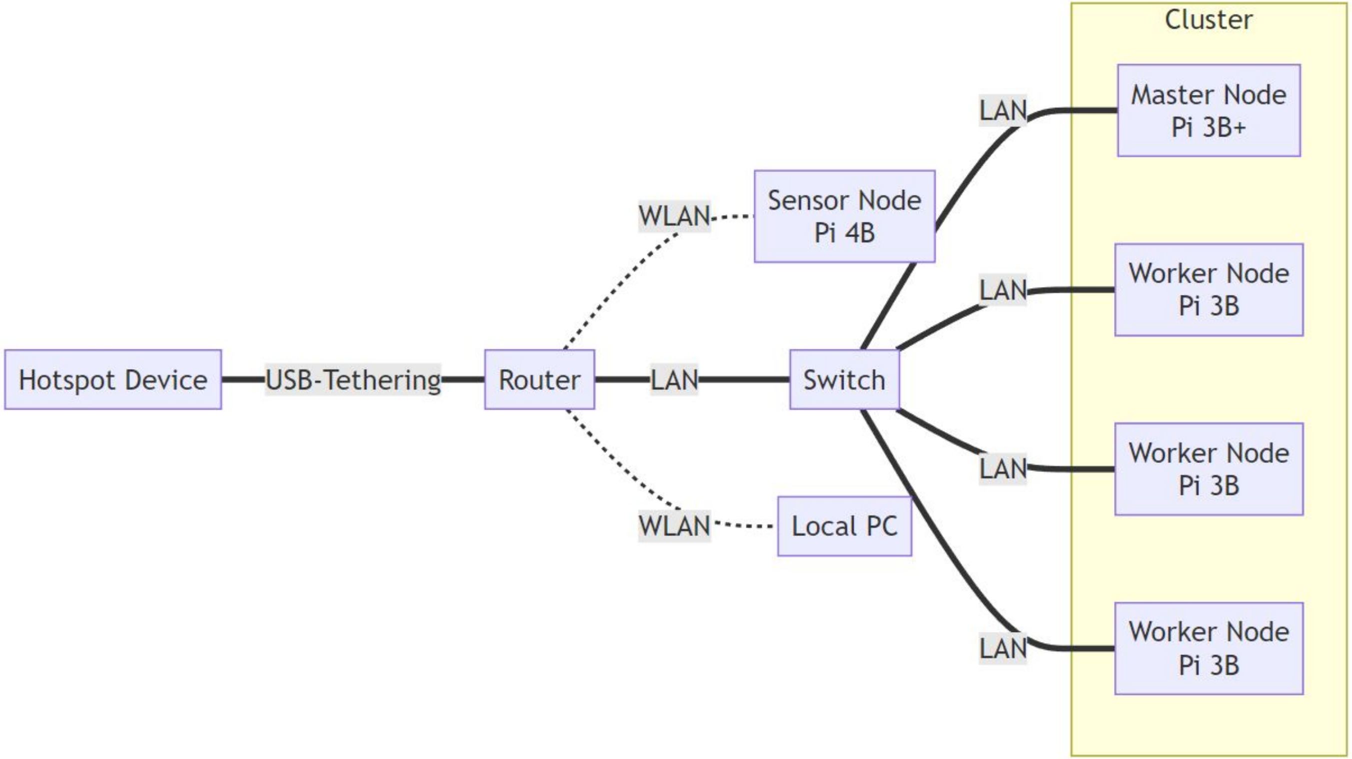


Cluster Design

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Design	Pros	Cons	Decision
1 Master & 3 Workers	- Simple setup - Enables fault tolerance & high availability in worker plane - Enables scalability across worker nodes	No fault tolerance & high availability in control plane	Adopt
2 Masters & 2 Workers	- Enables fault tolerance & high availability in both control & worker planes - Enables scalability across worker nodes	Complex setup	Discard
3 Masters & 1 Worker	Enables fault tolerance & high availability in control plane	- No fault tolerance & high availability in worker plane - Complex setup - No scalability across worker nodes	Discard

Network Architecture



Sensor Node

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Gather Data

- Using Kaggle
- Filtering for Dogs and Cats
- Collected 55817 Images
- Unannotated

kaggle

MegaDetector

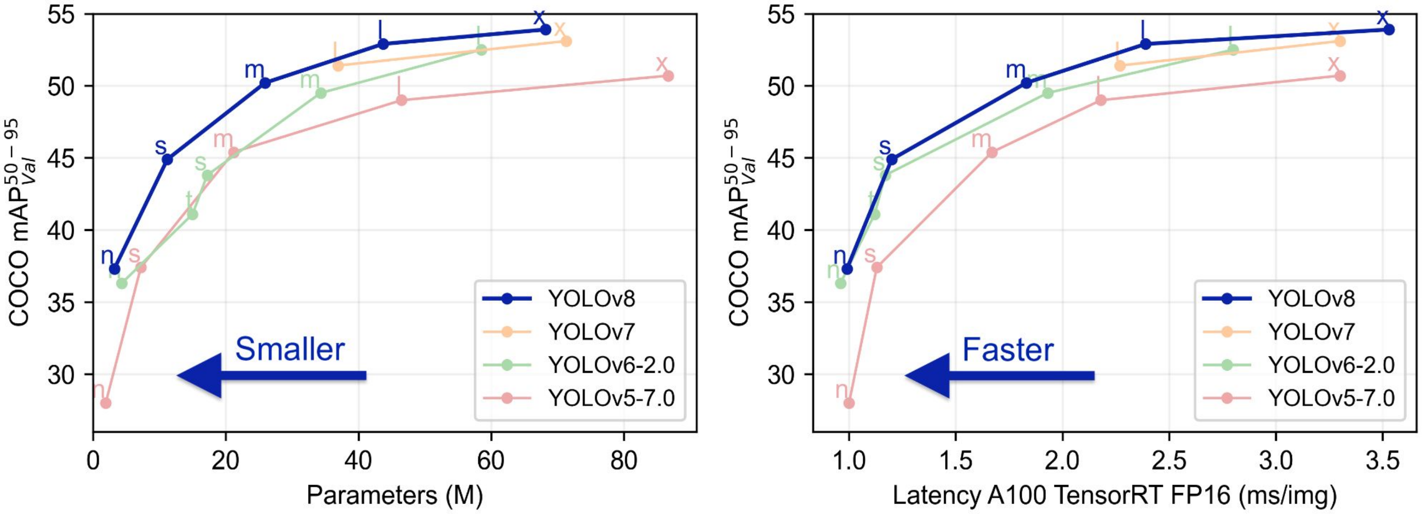
- Automated Annotating
- Can only separate Animal, Human, Vehicles
- Annotated 10 Images per Second
- Trained with 28657 Images

Pet	Training	Validation	Test
Cat	13.875	1.816	1.740
Dog	14.782	1.871	1.848
Total	28.657	3.687	3.588

Sensor Node

YOLOv8

- State of the Art
- Small Model



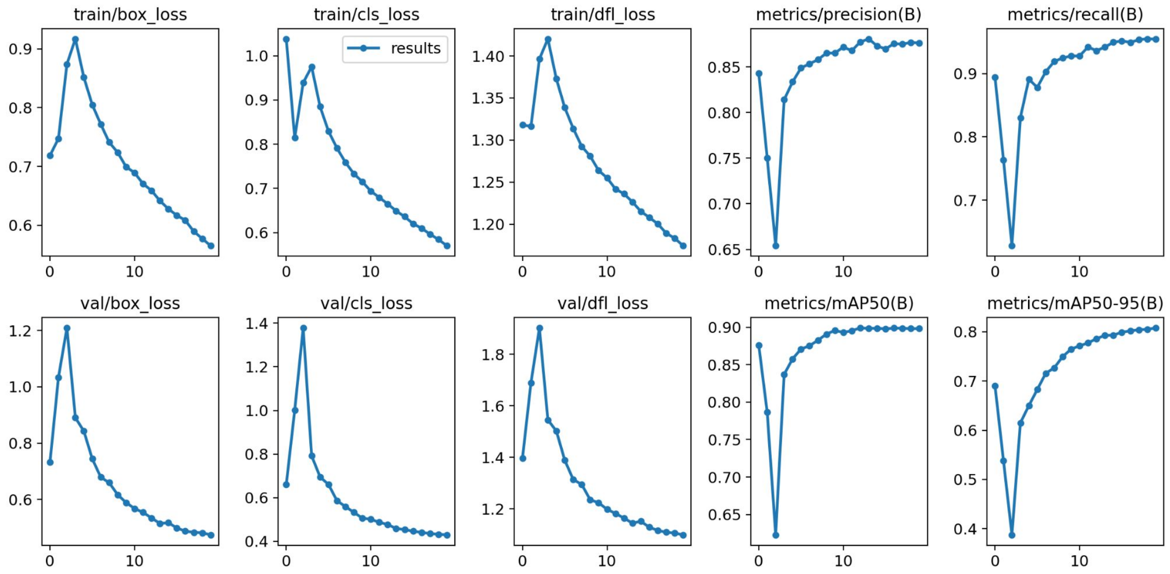
Sensor Node

Annotation Format

- MegaDetector (json): <class> x_top_left, y_top_left, width, height
- YOLOv8 (txt): <class> x_center, y_center, width, height

YOLOv8 Training

- In Google Colab
- Using GPU runtime
- Training in 20 epochs



Sensor Node

Raspberry Pi 4 Model B

- 64-bit OS

Code

- Object-oriented
- Argparse
- Camera, Detection, Network, Package, Compress, SensorNode
- Good Error handling
- dev/shm

```
usage: SensorNode [-h] [--model MODEL] [--url URL] [--conf CONF]
                  [--queue QUEUE] [--debug] [--single]

Detect Cats and Dogs

optional arguments:
  -h, --help       show this help message and exit
  --model MODEL    Path To Model
  --url URL        URL to server
  --conf CONF      Lowest level of detection rate
  --queue QUEUE    Number of Images before it get send
  --debug          Image should be saved
  --single         Only a Single image will be processed
```

Sensor Node

Message

- Image converted to JPG and optimized
- Image to Base64
- Message compressed
- Implemented a Queue


```
{
    “picture”: <Encoded string of image>,
    “date”: “2023-05-28”,
    “time”: “10:15:46”,
    “detections”: [
        {“type”: “Cat”, “accuracy”: 0.912, “bid”: 1},
        {“type”: “Dog”, “accuracy”: 0.728, “bid”: 2}
    ]
}
```


Sample data from Sensor Node

Frontend

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Overview

- Retrieve data from the backend and present them to the user
- Data can be retrieved based on certain filter criteria.

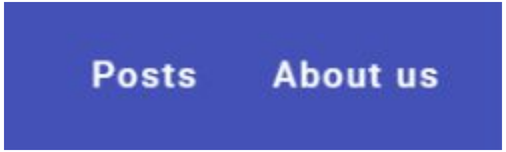
Used Framework

- Angular with TypeScript
- Pros:
  - Component-based Architecture
  - Two-way Data Binding
  - Dependency Injection

Frontend

Components

- Capture (i.e., Post, displaying data retrieved from the backend)
- Navigation bar
- Main page (Posts)
- About Us page



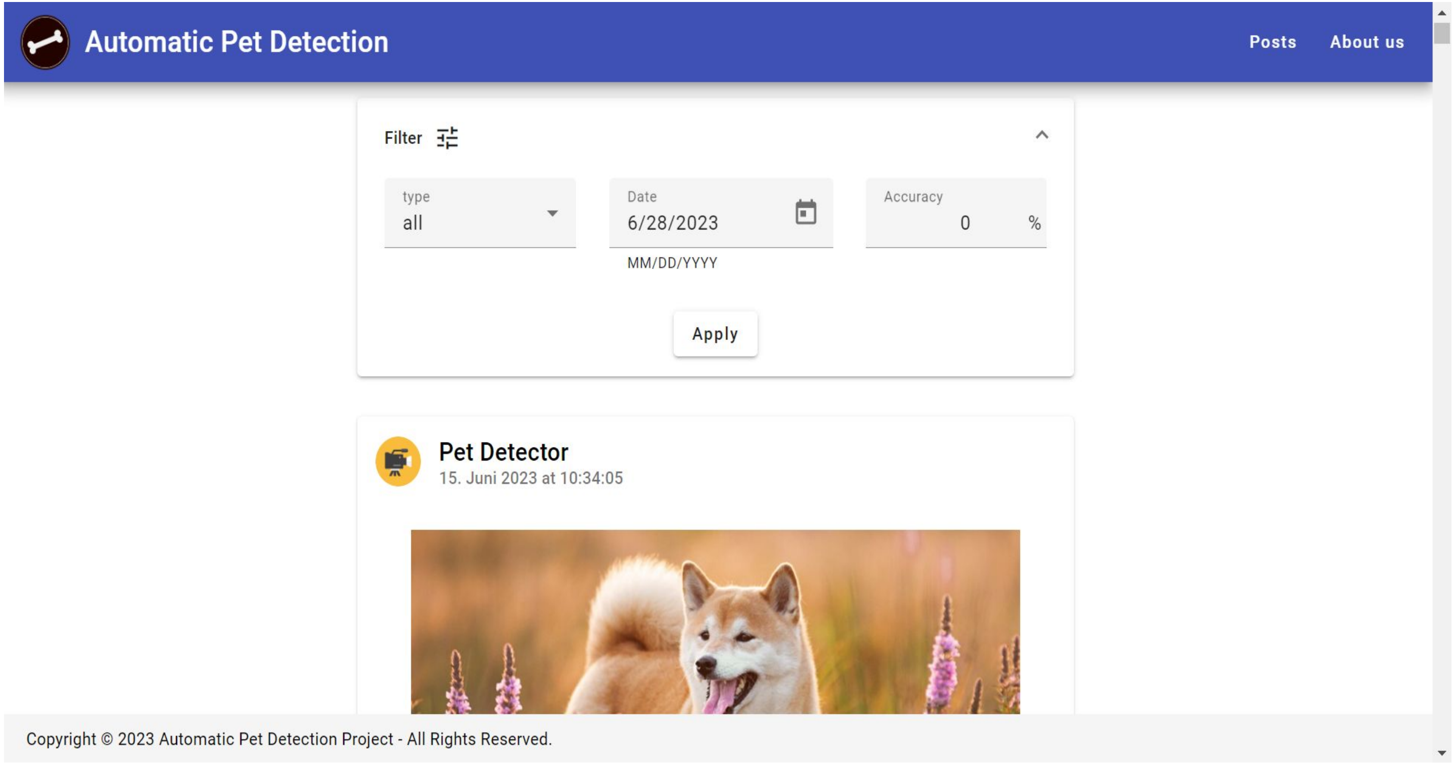
Pet Detector
15. Juni 2023 at 10:34:05

Evaluation:

ID	Type	Accuracy
1	dog	89%

Main Page (Posts)

FRANKFURT UNIVERSITY OF APPLIED SCIENCES



About Us

Automatic Pet Detection Posts About us

Automatic Pet Detection

Overview

The project Automatic Pet Detection With Edge Computing is part of the Cloud Computing SS23 module of Prof. Dr. Christian Baun at the Frankfurt University of Applied Sciences.

Team Members

Vincent Roßknecht, Jonas Hülsmann, Marco Tenderra, Minh Kien Nguyen, Alexander Atanassov

Source Code & Documentation

You can find the source code and documentation in our GitHub repository.

Frontend

Communication With Backend

- Through HTTP requests
- Rest API for retrieving data by a given filter
- At most ten images are retrieved per request
- Filter criteria:
  - Date: Retrieved data must be before the given date
  - Type: Retrieved data must have at least one pet of the given type
  - Accuracy: Minimum accuracy of all pets in the retrieved data

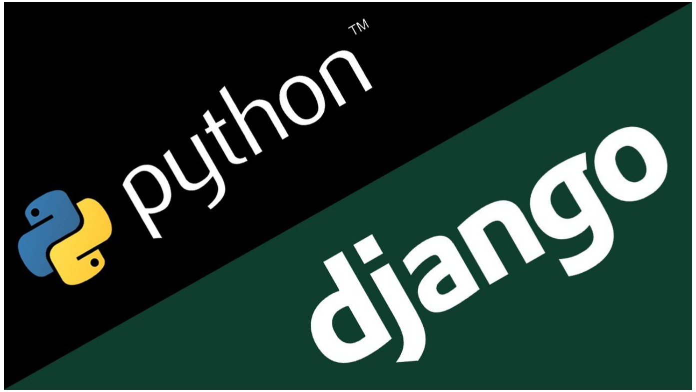
Deployment

- Dockerize the application for **linux/arm64** architecture
- Create a YAML file to specify deployment configuration
- Apply YAML file to deploy frontend on the Kubernetes cluster

API - Django

Overview

- Used Framework: Django
  - Fast setup, easy to use, built-in functions for db-calls and url routing
- Main Task: Providing communication endpoints for the sensor node and frontend for reading/writing to db
- Implementation of:
  - Models, Serializers, Views, URLs
- Set up SQLite 3 DB for Models/Serializers until MySQL DB is ready for deployment



Problems

- MongoDB instead of MySQL makes M/S unnecessary
- Deployment on Kubernetes Cluster not possible

API - Flask

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Overview

- Used Framework: Flask
  - Lightweight framework with built-in dev-server and fast debugger
- Main Task: Take the place of Django-Backend and reduce bloat
- Implementation of:
  - Views, URLs, MongoDB-Connection via PyMongo
- Successful deployment on Kubernetes

Problems

- Round-robin DNS of MongoDB’s Stateful Set not supported by PyMongo
  - Drop of MongoDB instances on Kubernetes from 3 to 1

Solution

- Switch to another language or database (not feasible in time)

Storage Service

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Starting Point

- A storage service that can replicate data on Persistent Volumes (PV) across worker nodes
- provides high availability and fault tolerance for data on cluster

Options

- Longhorn
- OpenEBS with Replicated Volumes

LONGHORN OpenEBS

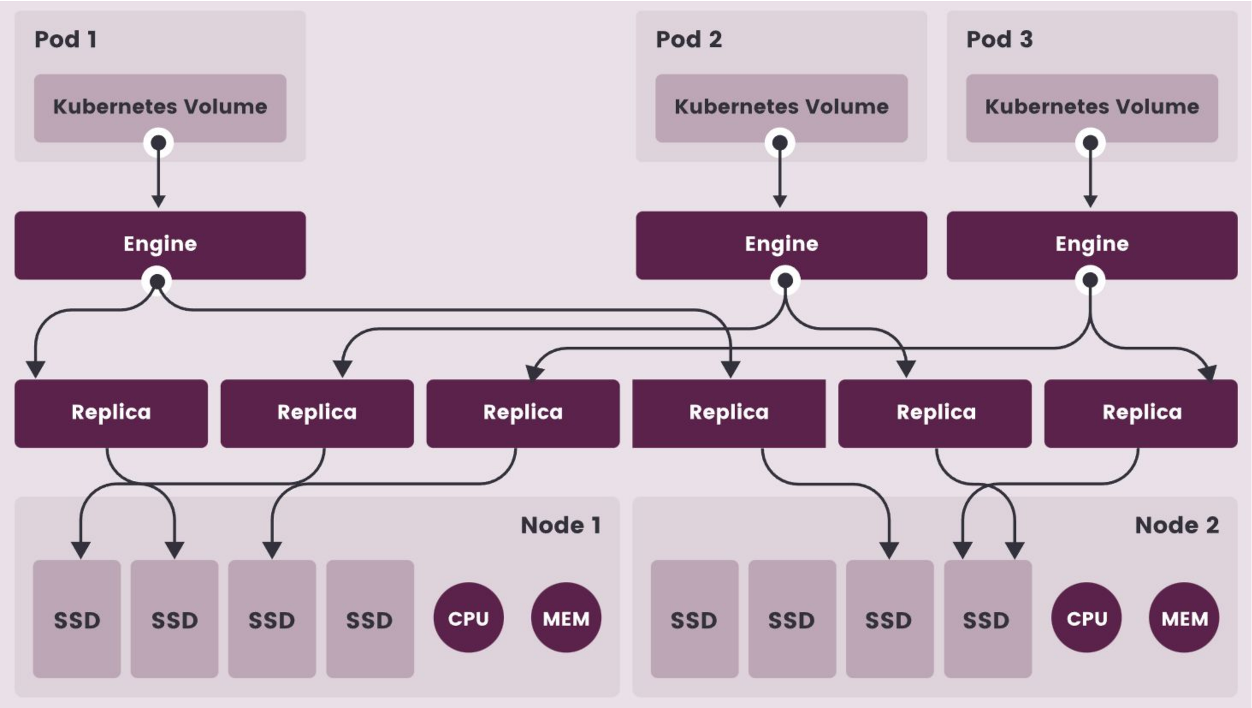
Longhorn

Pros

lightweight

Cons

- CrashLoopBackOff
- Complex Prerequisites



OpenEBS with Replicated Volumes

Pros

easier to set up than Longhorn

Cons

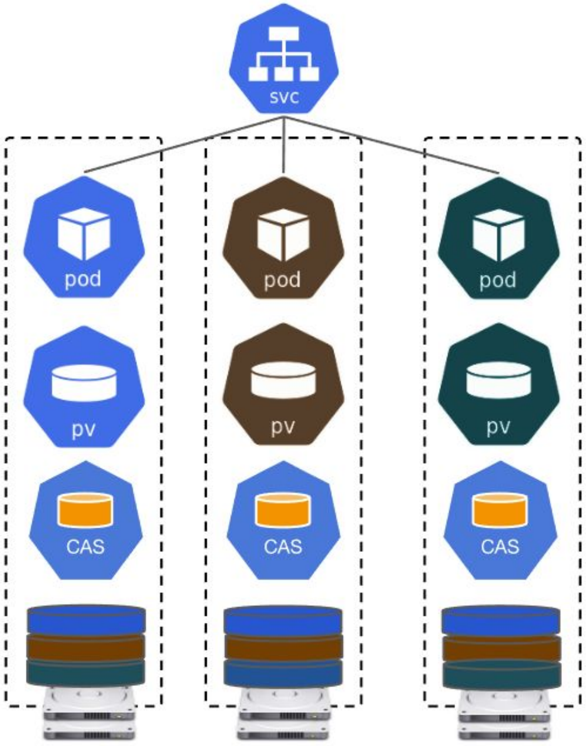
CrashLoopBackOff

Conclusion

Not recommendable to use a storage service for replicating PV data across worker nodes

Reason

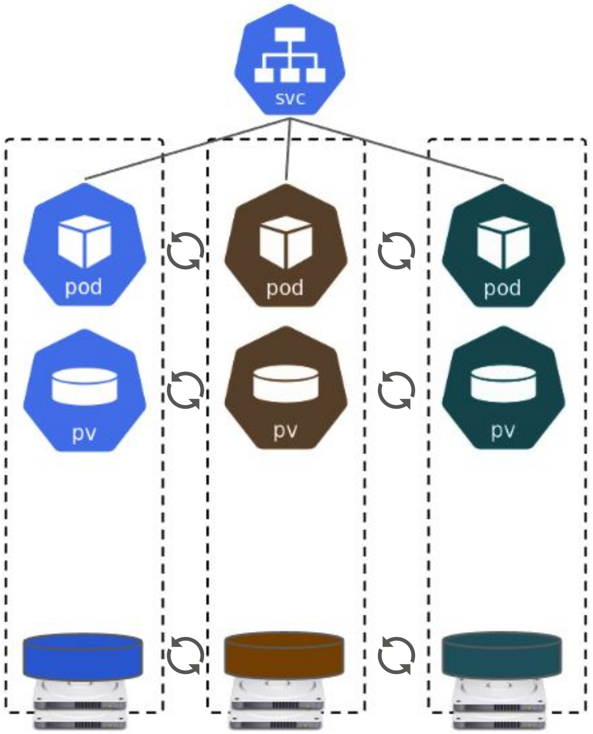
Overhead on cluster, eventually leading to out-of-memory or -resource



Storage Service

Current Design

- Delegate the replication of PV data to DBS pods:
  - Each DBS pod runs on a worker node.
  - When the DBS pods synchronize their data, PV data are also replicated across worker nodes.
- Use **OpenEBS with Local Volumes**: OpenEBS only serves to dynamically provision Local PV for DBS pods.



DBS

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Starting Point

- A DBS that enables data replication across its instances
- must also support **arm64/v8** architecture
- How to store images and detection results for querying later?

Options

- Relational DBS
- NoSQL Document DBS

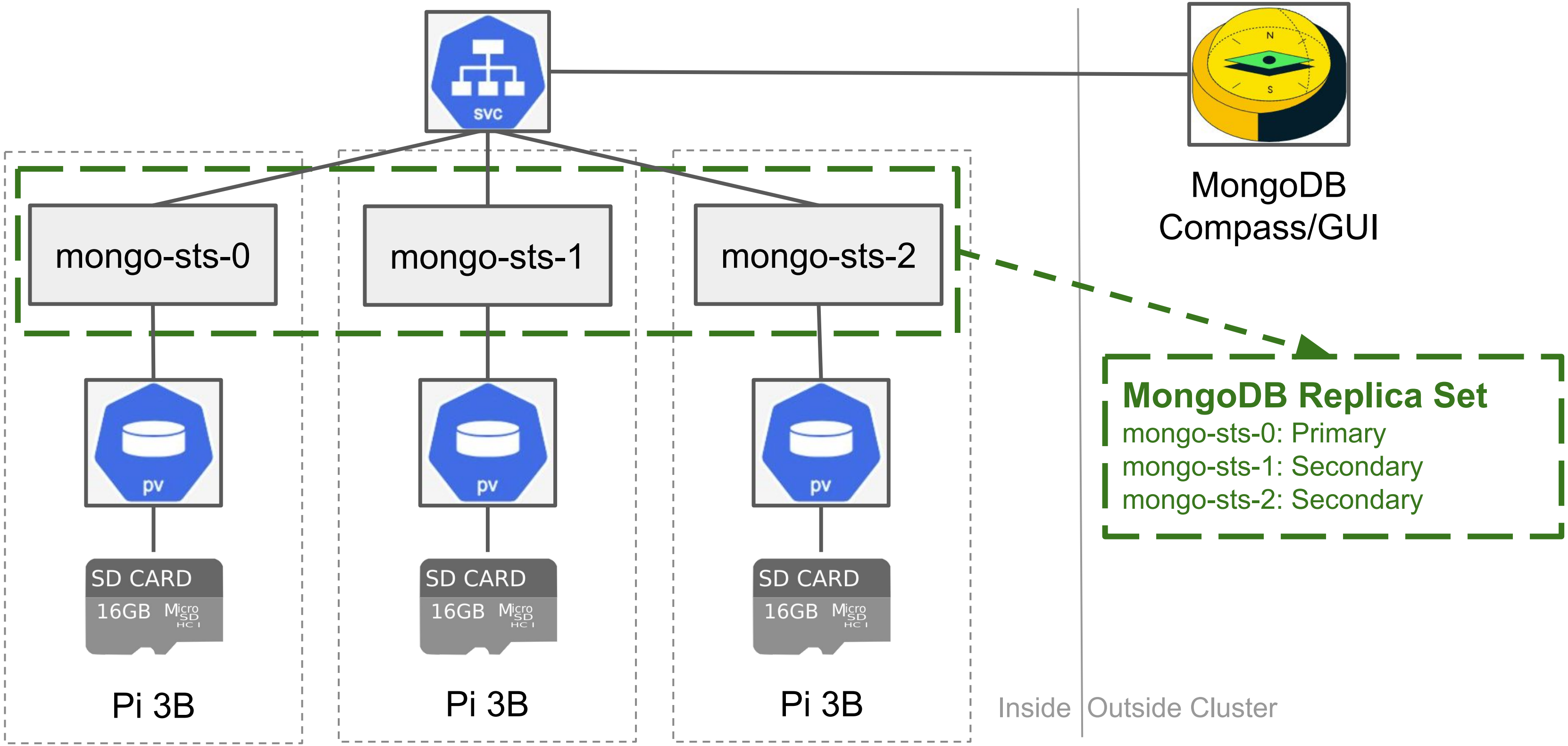

```
{
    “picture”: <Encoded string of image>,
    “date”: “2023-05-28”,
    “time”: “10:15:46”,
    “detections”: [
        {“type”: “Cat”, “accuracy”: 0.912, “bid”: 1},
        {“type”: “Dog”, “accuracy”: 0.728, “bid”: 2}
    ]
}
```


Sample data from Sensor Node

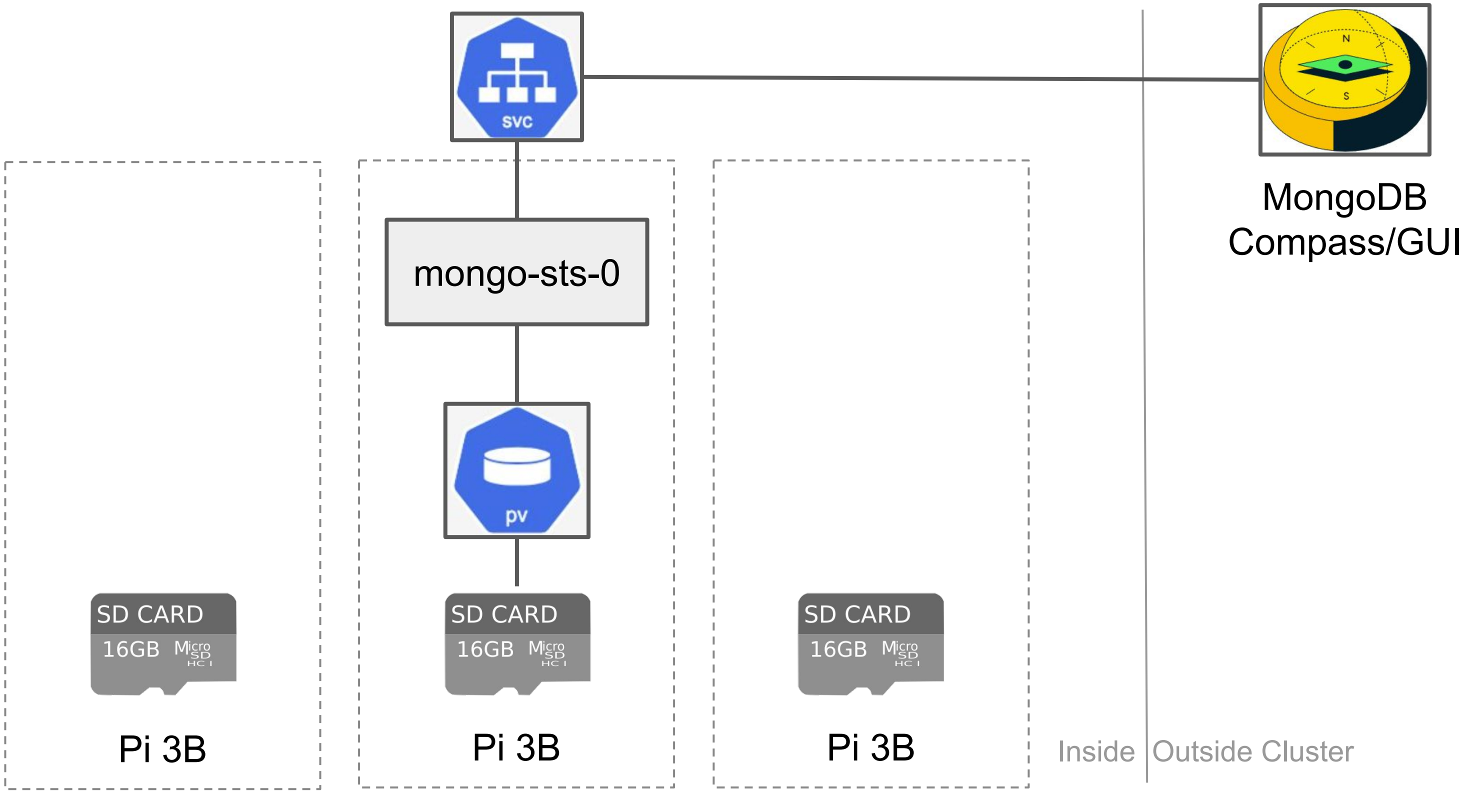
Relational vs. NoSQL Document

`MySQL` (Relational DBS)	`MongoDB` (NoSQL Document DBS)
Complex replication setup	***Simple*** replication setup
Image data stored as BLOB, requiring ***less*** storage space	Image data stored as base64-encoded string, requiring ***more*** storage space
Detection data stored in tables, producing ***possibly quicker*** query results	Detection data stored in JSON documents, producing ***possibly slower*** query results
More work needed in REST API Pods to produce write-queries	***Less*** work needed in REST API Pods to produce write-queries

Initial MongoDB Setup



Current MongoDB Setup



Demo

Scan the QR code above to take part in our Demo!

We will be back shortly after setting up our system.

Thank you for your patience!

Q&A

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

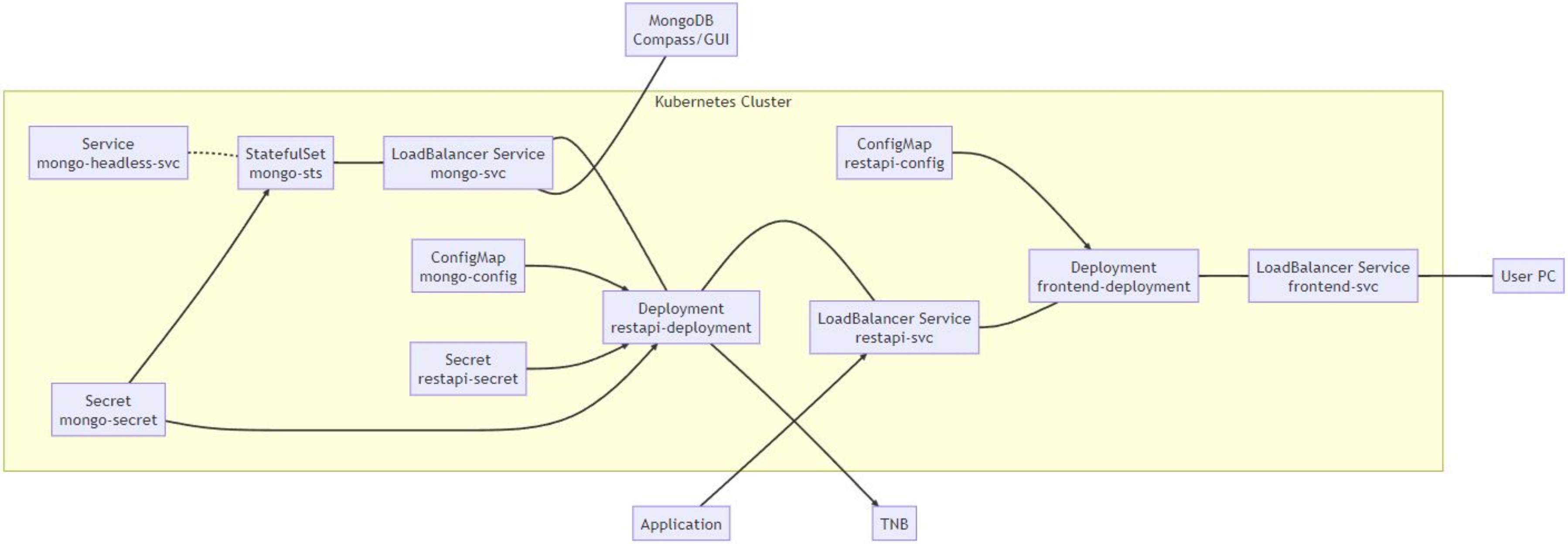
Contact Us!

- Sensor Node:
  - 📧 vincent.rossknecht@stud.fra-uas.de
  - 📧 tenderra@stud.fra-uas.de
- Cluster:
  - 📧 minh.nguyen4@stud.fra-uas.de
  - 📧 jonas.huelsman@stud.fra-uas.de
  - 📧 alexander.atanassov@stud.fra-uas.de

Check Out Our Project Report!

Thank You For Your Attention!

Appendix: Kubernetes Architecture



Appendix: Project Plan