Wireless, low latency edge cluster setup

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, DESIGN AND MANUFACTURING, KURNOOL

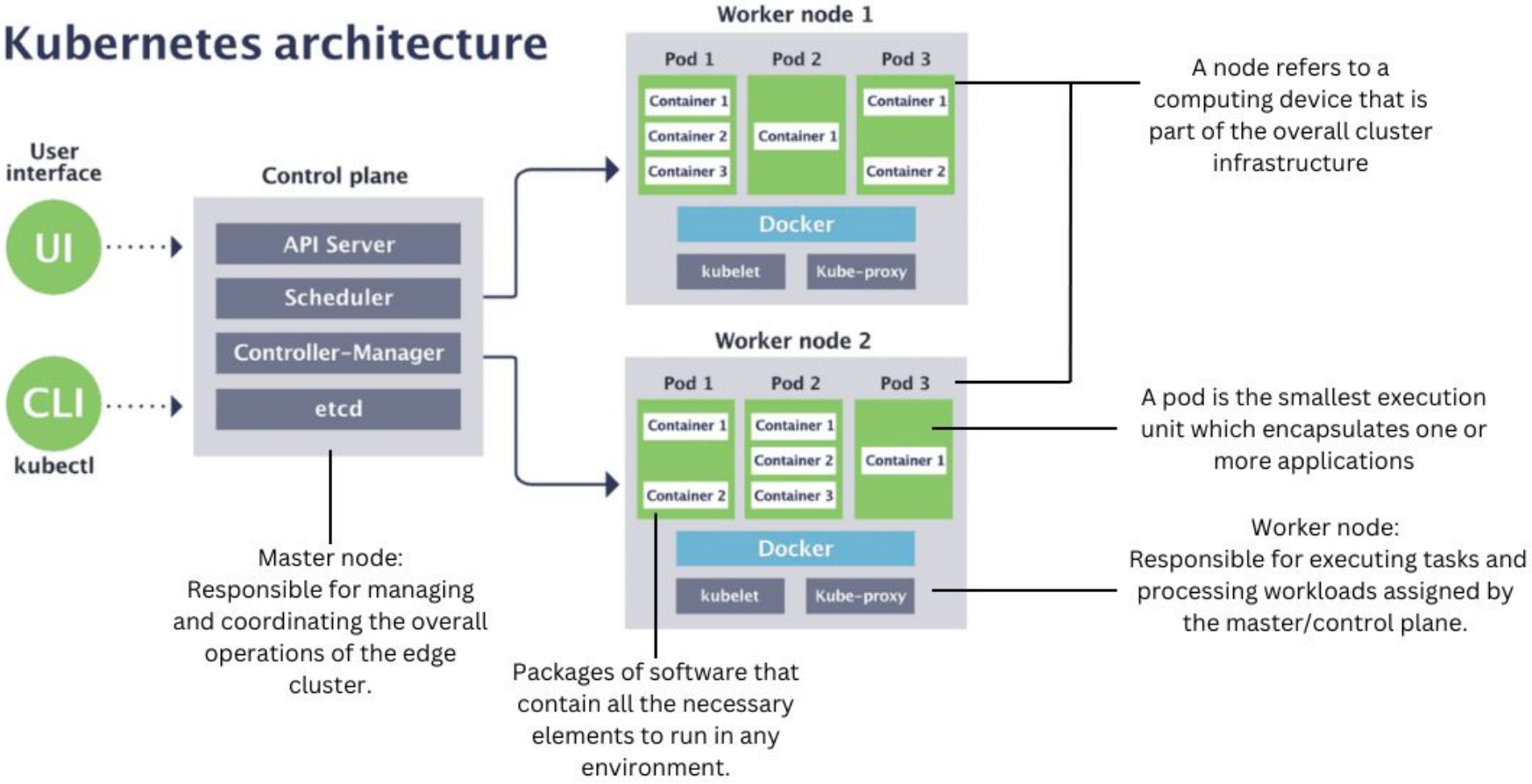
MENTOR: Dr. R. ANIL KUMAR

Team members:
1. P. Lakshmi Sujitha: 120CS0017
2. Y. Bhavishya: 120CS0025
3. K. R. Ramya Shri Shakthi: 120CS0046
4. S. Satakshi: 120CS0047

PROBLEM STATEMENT

Deploying a wireless, low latency edge cluster that addresses notable deficiencies in an existing Kubernetes-based cluster.

Kubernetes architecture



Our kubernetes edge cluster

We deployed our edge cluster using a Kubernetes tool called kubeadm.

Our kubernetes cluster consists of four nodes (our laptops):

- A master node or a control plane
- Three workers

```
root@ramyashri:/home/ramya# kubectl get nodes
NAME                     STATUS   ROLES           AGE   VERSION
cantcode-virtualbox      Ready    <none>          32d   v1.28.1
ramyashri                Ready    control-plane   32d   v1.28.0
sujitha-virtual-machine  Ready    <none>          32d   v1.28.0
ubuntulinux              Ready    <none>          32d   v1.28.0
root@ramyashri:/home/ramya#
```

Figure 2: Screenshot of our cluster nodes

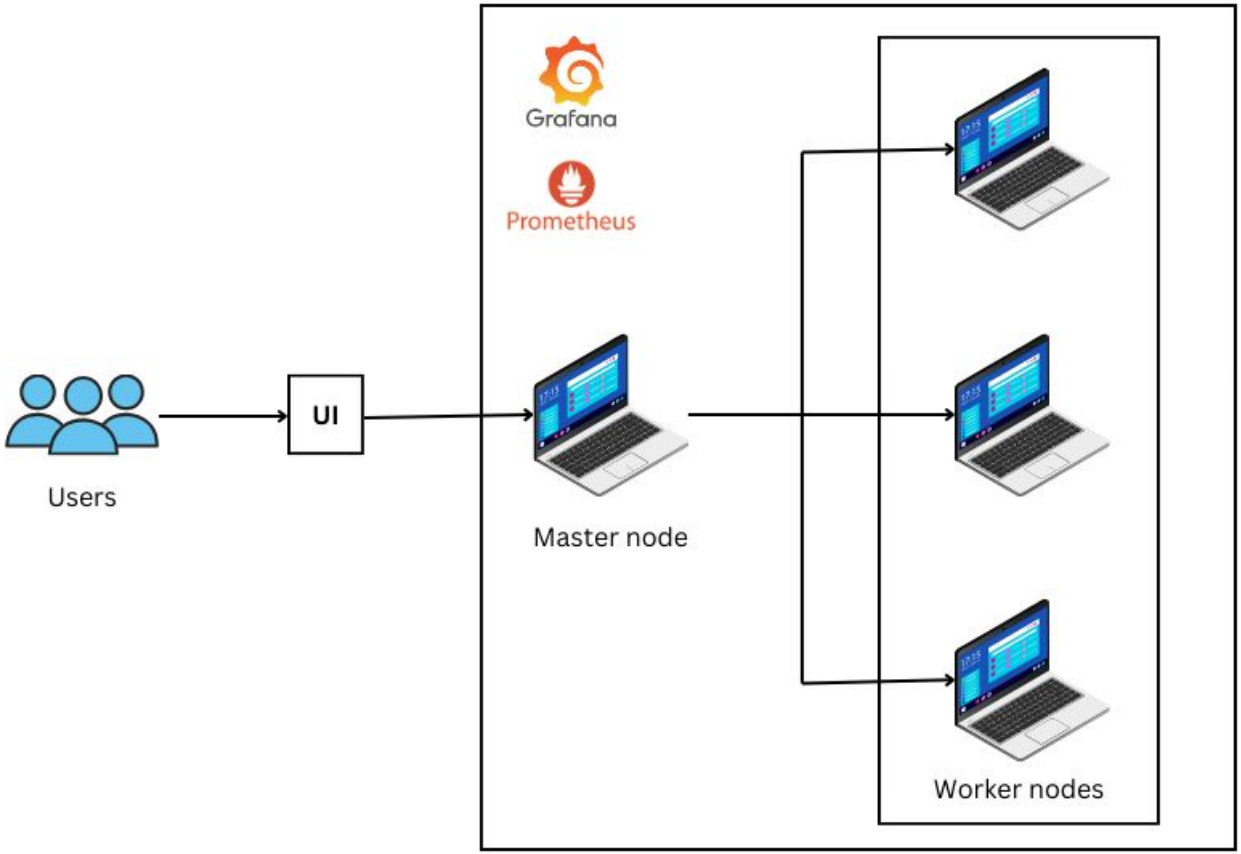


Figure 3: Cluster architecture diagram

Gaps in the existing kubernetes architecture

- Automatic deployment of the cluster is not possible.
- Kubernetes does not offer event based vertical scaling.
- Kubernetes does not strategically assign applications to the nodes with minimal latency.

To address these gaps, we have implemented some solutions, which are discussed in the upcoming slides.

Implementation

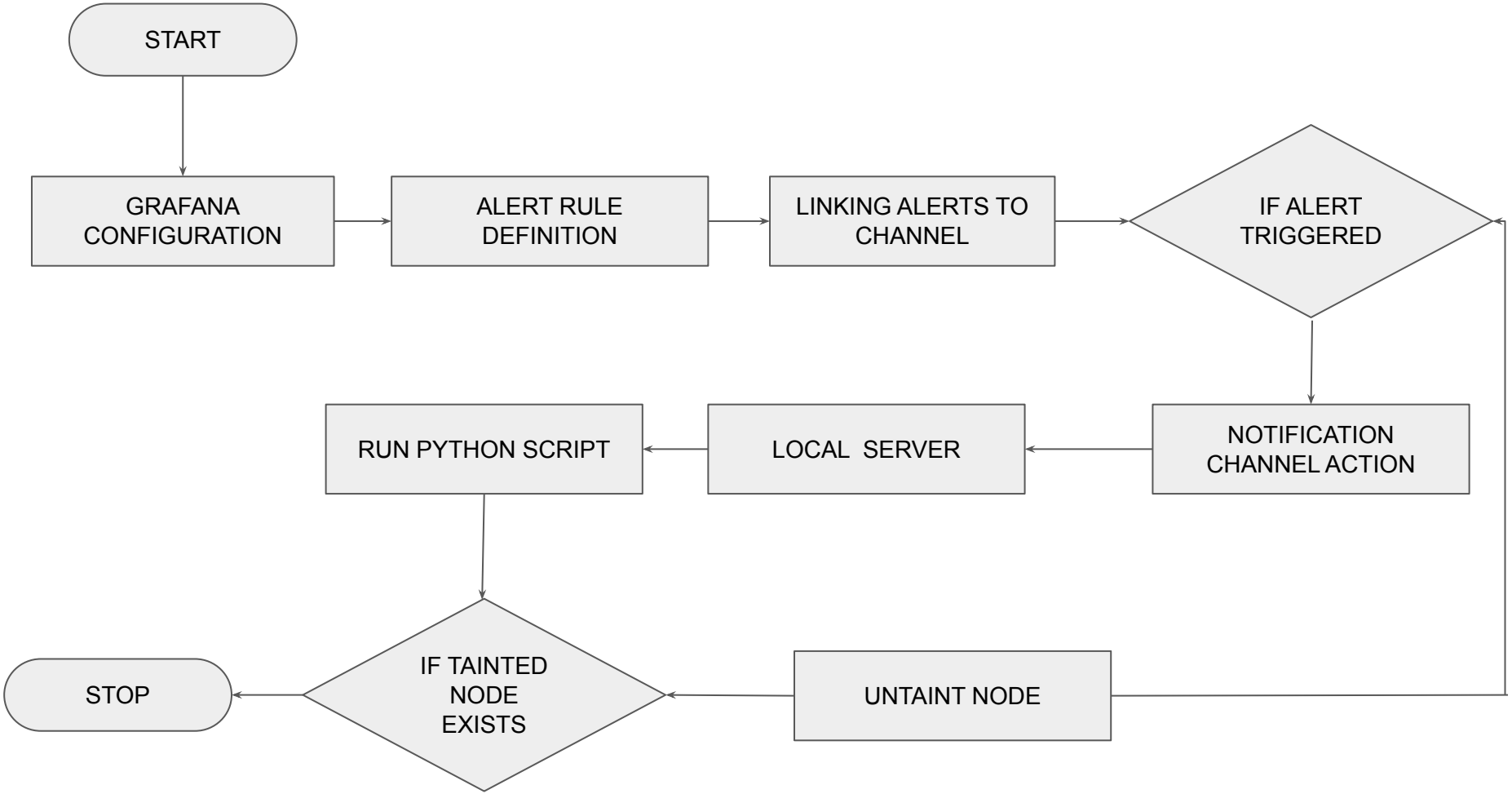
1) Automatic Deployment:

- For a worker to join the cluster, it must have a token which is only generated after the master node is initialized. Kubernetes does not offer any in built communication channel to send the tokens to the worker nodes which makes it tedious.
- To make the addition of worker node more automated and efficient, we are running a code using client-server architecture for the distribution file containing necessary tokens and commands.

2) Event based vertical scaling

In our Kubernetes edge cluster with limited resources: efficient resource utilization is a priority. To implement this, we came up with this strategy:

- Only the required nodes are untainted at any given time.
- We incorporated Prometheus, to keep monitoring our edge cluster at all times.
- Grafana is connected to the prometheus data source of our cluster.


START
GRAFANA CONFIGURATION
ALERT RULE DEFINITION
LINKING ALERTS TO CHANNEL
IF ALERT TRIGGERED
NOTIFICATION CHANNEL ACTION
LOCAL SERVER
RUN PYTHON SCRIPT
IF TAINTED NODE EXISTS
STOP
UNTAINT NODE

- We created a nodeJS server which listens to any alerts from Grafana.
- A notification channel is created in Grafana, specifying the method of communication for alert notification.
- Then, we configured alerts by writing multiple queries in grafana after which the alerts are linked to notification channel.
- As soon as an alert is triggered, the nodeJS server will recognise it and run the python script to automatically untaint the tainted nodes.

3) Wireless, distributed low latency cluster

Our entire edge infrastructure operates wirelessly, utilizing laptops as nodes within the cluster.

It is feasible for all nodes to be dispersed across different geographical locations, as long as they are connected to the the same network. This mirrors the functionality of a distributed edge cluster.

To guarantee swift responses for users, it is essential to minimize the latency within the cluster.

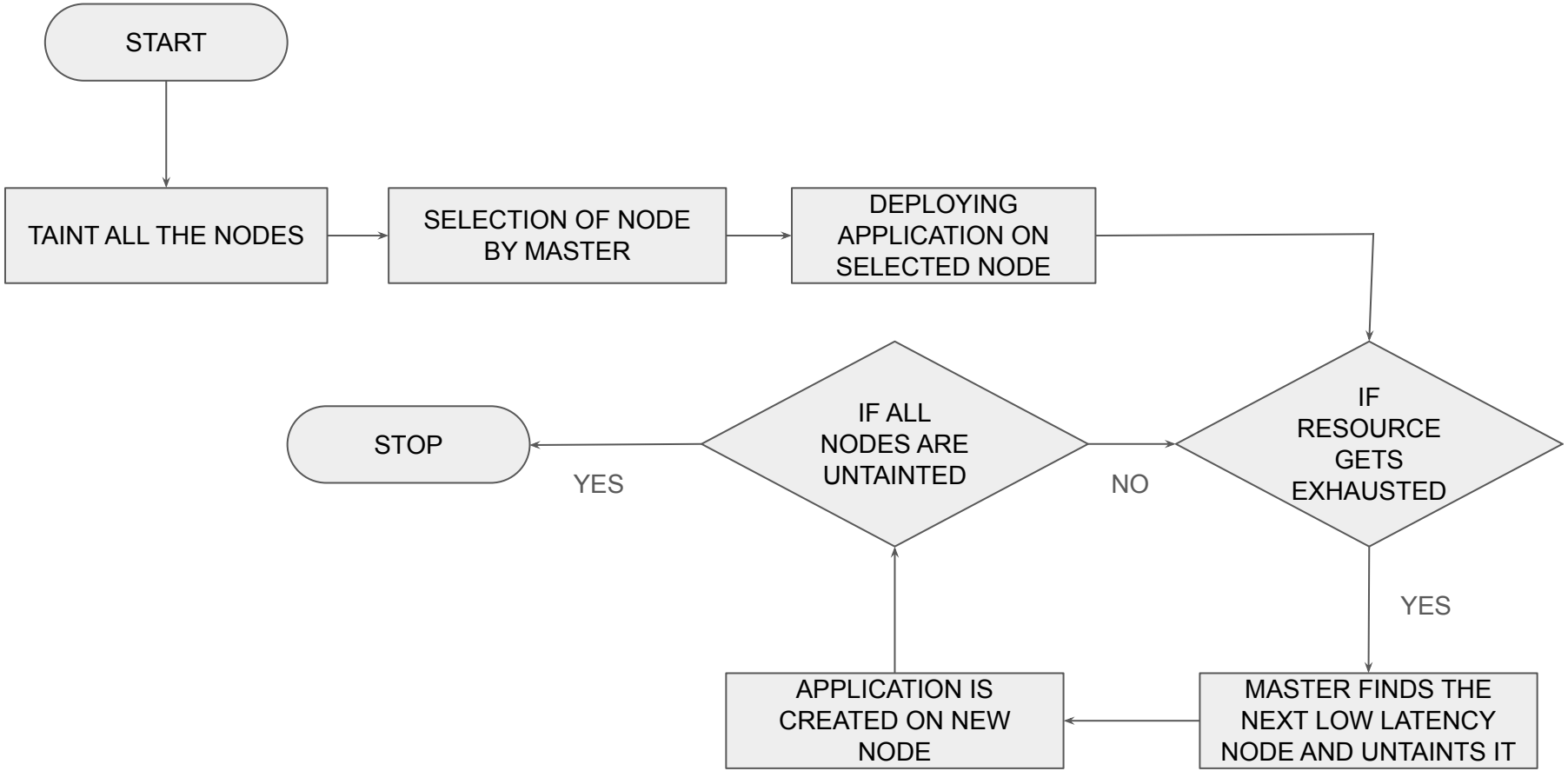
So, we came up with an approach to minimise the latency in our cluster.

Since the nodes are distributed, the worker node with minimal latency from the master node is to be found.

An instance of the application is then created on that node.

Heavy influx in requests results in exhaustion of node resources.

So, as soon as that happens, our algorithm checks for the node with next minimal latency and creates another instance of the application on that node.


START
TAINT ALL THE NODES
SELECTION OF NODE BY MASTER
DEPLOYING APPLICATION ON SELECTED NODE
IF RESOURCE GETS EXHAUSTED
NO
YES
IF ALL NODES ARE UNTAINTED
YES
STOP
MASTER FINDS THE NEXT LOW LATENCY NODE AND UNTAINTS IT
APPLICATION IS CREATED ON NEW NODE

All the worker nodes in the cluster are initially tainted.

The master node runs an algorithm to find a tainted worker node with the lowest latency among all the worker nodes.

To keep track of resource utilization in the cluster, Grafana and Prometheus are used. If any resource utilization exceeds a certain threshold, alerts are sent to the local nodeJS server.

As soon as the alert is received, the master runs the algorithm to find the tainted node with next minimum latency.

This process essentially schedules the applications in nodes with minimal latency.

POSSIBLE USE CASES OF OUR CLUSTER:

DRONE APPLICATIONS:

- Given the wireless capability of our edge nodes, we have the potential for node mobility, allowing each node to function as a drone and execute computations.
- While we have not explored the mobility aspect of our wireless network, there's an opportunity to establish a low-latency cluster tailored for drone operations.

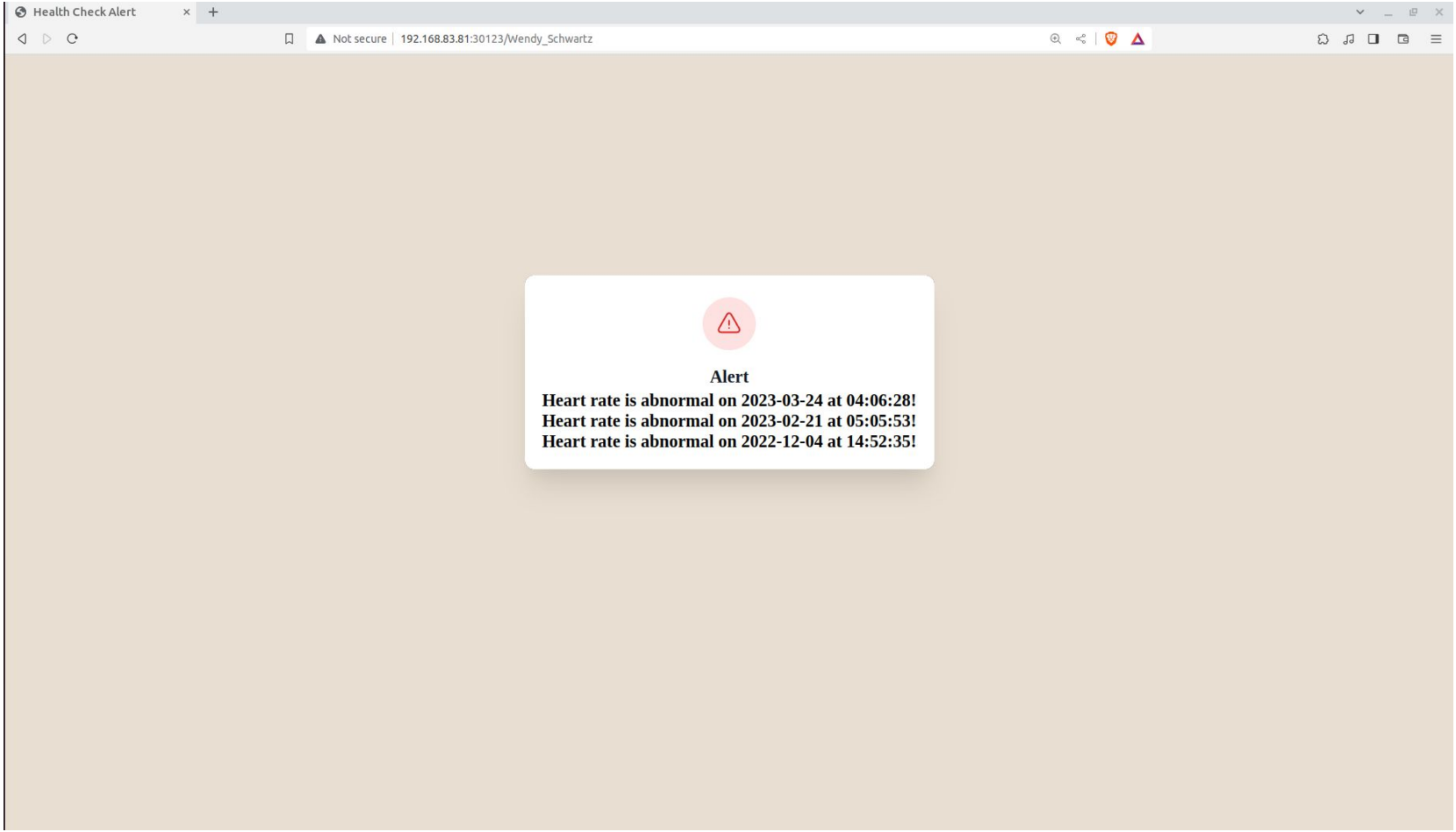
SAMPLE APPLICATION

We have created a sample application using NodeJS and MongoDB to test the features of our cluster.

In this application, we are assuming that a sensor generates data encompassing fields such as name, date, time, and heart rate. This data is then transmitted to a MongoDB pod deployed within the edge cluster.

To facilitate the heart rate monitoring process, we've implemented an Express.js code in a separate pod.

The application pod is exposed to users through a NodePort service, providing external accessibility. A client can access their heart rate monitoring results using the url http://masterip:portnumber/patientname. Kubernetes services are employed to establish connectivity between the Express.js application pod and the MongoDB pod, enabling seamless access to health-related data stored in the ”healthcheck” database.


Health Check Alert
Not secure | 192.168.83.81:30123/Wendy_Schwartz
Alert
Heart rate is abnormal on 2023-03-24 at 04:06:28!
Heart rate is abnormal on 2023-02-21 at 05:05:53!
Heart rate is abnormal on 2022-12-04 at 14:52:35!