

# Application Monitor

## Comprehensive report

Amit Kumar Mallik 19D070007

Bhavnoor Singh Marok 190050027

Jayesh Singla 190050053

Shrey Singla 190050114

3 April, 2022

CS 387 : Database and Information Systems Lab

## Description

This is an application monitor that monitors an application running in parallel over multiple machines. There is a controller machine which runs the InfluxDB application monitor. The client machines sends the data in log format to the application monitor using Telegraf. We then analyze the data in the controller machine using InfluxDB and generate a dashboard for CPU statistics, memory usage, disk usage, postgres. Using the analysis, we can suggest changes to the distribution algorithm of the controller machine to improve performance.

One might think that this task can be performed by performing a time-series analysis on a simple set of log files. However, not using the rich set of features and optimisations provided by databases will be sub-optimal. Database systems take care of optimisations and data integrity, thereby saving the efforts of programmers, and also reducing the possibility of bugs.

We have used a time-series database InfluxDB combined with Telegraf to monitor the data streams from respective machines. Telegraf provides a nice API to send data from an application to another, across various machines using plugins. InfluxDB is a good choice as it is optimized for time-series data analysis and can be integrated with Telegraf easily.

## Intended Users

1. This project can be used by companies hosting multiple servers. Using our project, they can figure out if load is unbalanced among their servers. It could also predict that certain server is going to throttle in some time which would indicate the companies to redirect the load from this server. It could also analyze the performance of different parts like databases, hard resources such as CPU etc and thus find out the bottleneck which needs to be improved.
2. This could also be used more specifically by database administrators to view database specific statistics and keep track of the database usage by different connections.

## Use cases

### Login on the central node

#### 1. Description

On startup of our application, the user will be presented with a login page, where he must enter the credentials given to him by the application admin. The password for admin is admin.

#### 2. Trigger

This event will be triggered by human user on clicking on the login button

#### 3. Primary Actor

Admin or users of the application

#### 4. Inputs

The inputs required will be-

- Username
- Password

**Constraints:** The username, password should be valid credentials supplied by the admin

#### 5. Main success path

If the user supplies valid credentials, the use case will succeed

**6. Preconditions**

The user should not already be logged in

**7. Exceptions**

An error will be raised if the user provides wrong credentials and they will stay on the login page

**8. Post conditions**

On success, the user will be redirected to the dashboard page

## Registration of user by administrator

**1. Description**

The admin can register a user by creating credentials and assigning group(s) to the user based on his permission level

**2. Trigger**

This event will be triggered by admin of our application

**3. Primary Actor**

Admin

**4. Inputs**

The inputs required will be-

- Username
- Password
- Confirm Password
- Group(s)

**Constraints:** The username should not exist before and the (hash of) passwords must match

**5. Main success path**

User is added to the users table.

**6. Preconditions**

The user should be the administrator

**7. Exceptions**

An error will be raised if the username already exists

**8. Post conditions**

On success, a new user will be created in the User table

## Adding of node

**1. Description**

The admin will add a new node to the table of nodes and assign set of groups to that node. It will also alter the relation between nodes and groups appropriately. The node is identified by its IP.

**2. Trigger**

This event will be triggered by admin of the application

**3. Primary Actor**

Admin

**4. Inputs**

- IP
- Name
- Groups

### **Constraints**

The IP shouldn't exist already. The groups should be present in the groups table.

#### **5. Main Success Path**

The node will be added to the nodes table.

#### **6. Preconditions**

The user should be the administrator

#### **7. Exceptions**

A pop up of error will be shown if the constraints are not met.

#### **8. Post conditions**

On success, a new node will be setup to be tracked

## **Adding of group**

#### **1. Description**

The admin will add a new group to the table of groups. The group is identified by its group ID.

#### **2. Trigger**

This event will be triggered by admin of the application

#### **3. Primary Actor**

Admin

#### **4. Inputs**

- Name
- Description

### **Constraints**

Null (A unique ID is auto generated)

#### **5. Main Success Path**

The group will be added to the groups table.

#### **6. Preconditions**

The user should be the administrator

#### **7. Exceptions**

A pop up of error will be shown if the constraints are not met.

#### **8. Post conditions**

On success, a new group will be setup to be tracked

## **Adding of database**

#### **1. Description**

The admin will add a new database to the table of databases. The database is identified by its database ID.

#### **2. Trigger**

This event will be triggered by admin of the application

**3. Primary Actor**

Admin

**4. Inputs**

- Name
- Description
- Node

**Constraints**

The node must be a valid node. (A unique ID is auto generated).

**5. Main Success Path**

The database will be added to the databases table with the selected node

**6. Preconditions**

The user should be the administrator

**7. Exceptions**

A pop up of error will be shown if the constraints are not met.

**8. Post conditions**

On success, a new database will be setup to be tracked

**De-registration of user by administrator****1. Description**

The admin can de-register a user

**2. Trigger**

This event will be triggered by admin of our application

**3. Primary Actor**

Admin

**4. Inputs**

The inputs required will be-

- Username

**Constraints:** The username should exist before

**5. Main success path**

Delete the user from the users table

**6. Preconditions**

Only admin can delete users.

**7. Exceptions**

An error will be raised if the username doesn't exists

**8. Post conditions**

On success, the user will be deleted from the User table

## Deletion of node

### 1. Description

The admin will delete a node from the table of nodes. It will also alter the relation between nodes and groups appropriately. The node is identified by its IP.

### 2. Trigger

This event will be triggered by admin of the application

### 3. Primary Actor

Admin

### 4. Inputs

- IP

### Constraints

The IP should exist already.

### 5. Main Success Path

The node will be deleted from the nodes table.

### 6. Preconditions

The user should be the administrator

### 7. Exceptions

A pop up of error will be shown if the constraints are not met.

### 8. Post conditions

On success, the node will be deleted

## Deletion of group

### 1. Description

The admin will delete a group from the table of groups.

### 2. Trigger

This event will be triggered by admin of the application

### 3. Primary Actor

Admin

### 4. Inputs

- Name

### Constraints

The group name should exists in the group table

### 5. Main Success Path

The group will be deleted from the groups table.

### 6. Preconditions

The user should be the administrator

### 7. Exceptions

A pop up of error will be shown if the constraints are not met.

### 8. Post conditions

On success, the group will be deleted.

## Deletion of database

### 1. Description

The admin will delete a database from the table of databases.

### 2. Trigger

This event will be triggered by admin of the application

### 3. Primary Actor

Admin

### 4. Inputs

- Name
- Node

### Constraints

The node must be a valid node. The name must exist in the databases table.

### 5. Main Success Path

The database will be deleted from the databases table

### 6. Preconditions

The user should be the administrator

### 7. Exceptions

A pop up of error will be shown if the constraints are not met.

### 8. Post conditions

On success, the database will be deleted

## Choice of nodes, databases, metrics etc to track

### 1. Description

The user will be given the choice of what nodes, metrics, start time, window period, aggregate function and database connections to track.

### 2. Trigger

This event will be triggered when the user clicks on the dashboard icon on the left panel

### 3. Primary Actor

End user of our application

### 4. Inputs

The inputs required will be -

- Start time
- Choice of node or database statistics
- Nodes/Databases to track
- Metrics
- Aggregate function
- Window period

**Constraints:** The nodes or databases must be present in the respective tables. The metrics and aggregate function must be supported by influxdb. The user must be authorised to view the nodes he chooses according to his group.

**5. Main success path**

The parameters are stored locally so that the graph can be made.

**6. Preconditions**

The user should be logged in

**7. Exceptions**

An error will be raised if the user tries to choose invalid hosts or hosts that it is not authorized to view

**8. Post conditions**

On success, the user will be presented the graphs showing different statistics.

## The Dashboard

**1. Description**

The user will be showed the statistics of the databases/nodes selected. It will be in the form of line charts. The user will be provided various filters as well so that he can choose what to see in the dashboard.

**2. Trigger**

This event will be triggered by the user when he has chosen the databases to monitor or the nodes to track

**3. Primary Actor**

End user of the application

**4. Inputs**

The inputs required will be -

- Click event which gets the data from start time till current time and generates the graph (refresh button)

**Constraints:** Null

**5. Main success path**

The operation will be a success always.

**6. Preconditions**

The user should be logged in

**7. Exceptions**

No exception would be generated

**8. Post conditions**

On success, the user will be shown the graphs of statistics of the database/nodes he chose to monitor.

## Creating alerts

**1. Description**

The user will create alerts to detect some anomalous behaviour in the statistics such as a high % of CPU, memory being used etc.

**2. Trigger**

End user clicks on "Create Alert" button

**3. Primary Actor**

End user

#### 4. Inputs

The inputs required are

- Name
- Description
- Choice of node or database
- Metric
- Nodes/Databases
- type of threshold (above, below, in range, out of range)
- Priority
- Alert message

**5. Constraints** The metrics should be a valid metric, the nodes must be present in the nodes table.

#### 6. Main success path

The alert is created i.e. the information is inserted into the alerts table

#### 7. Preconditions

The host should be logged in

#### 8. Exceptions

A pop up of error will be shown if the constraints are not met.

#### 9. Post conditions

A card for the alert is created

## Deletion of alerts

#### 1. Description

The user will delete an alert. They will no longer be checked on the data.

#### 2. Trigger

End user clicks on "Delete" button of a specific alert card

#### 3. Primary Actor

End user

#### 4. Inputs

The inputs required are

- Click activity

#### 5. Constraints Null

#### 6. Main success path

The alert is deleted i.e. the information is removed from the alerts table and the alert is no longer checked on the time series data.

#### 7. Preconditions

The host should be logged in

#### 8. Exceptions

A pop up of error will be shown if the constraints are not met.

#### 9. Post conditions

A card for the alert is destroyed

## Show of alerts

### 1. Description

The user will be displayed the history of unacknowledged alerts which were raised in the past.

### 2. Trigger

End user clicks on "Show" button of a specific alert card

### 3. Primary Actor

End user

### 4. Inputs

The inputs required are

- Click activity

### 5. Constraints Null

### 6. Main success path

The list of unacknowledged alerts is shown

### 7. Preconditions

The host should be logged in

### 8. Exceptions

A pop up of error will be shown if the constraints are not met.

### 9. Post conditions

The list is shown to the user

## Acknowledgement of alerts

### 1. Description

The user can acknowledge that the user has seen and responded/ignored the alert which was shown to the user

### 2. Trigger

End user clicks on "Acknowledge" button in the list of alerts

### 3. Primary Actor

End user

### 4. Inputs

The inputs required are

- Click activity

### 5. Constraints Null

### 6. Main success path

Alert is acknowledged and the acknowledged field of alert in the alertLogs table is set to 1.

### 7. Preconditions

The host should be logged in

### 8. Exceptions

A pop up of error will be shown if the constraints are not met.

### 9. Post conditions

The alert is acknowledged

## Detailed Logical Schema

The following tables will be present in the Postgres database:

1. Node:

- IP varchar(20)
- Name varchar(20)

2. Group:

- Name varchar(20)
- Description varchar(500)

3. User:

- username varchar(20)
- password varchar(20)

4. NodeGroup:

- IP references table Node varchar(20)
- Name references table Group varchar(20) not null

5. User-Group:

- username references table User varchar(20)
- group\_name references table Group varchar(20)

6. Database:

- database\_id varchar(40) NOT NULL
- IP references Node varchar(20)
- name varchar(20)
- Description varchar(500)

7. NodeAlert:

- AlertID int
- username references User varchar(20)
- priority int CHECK(priority > 4)
- query varchar(500)
- threshold\_low float NOT NULL
- threshold\_high float
- type varchar(20) check(type in ('ABOVE', 'BELOW', 'IN\_RANGE', 'OUT\_OF\_RANGE'))
- message varchar(500)

8. DatabaseAlert

- AlertID int
- username references User varchar(20)
- priority int
- query varchar(500)
- threshold\_low float

- threshold\_high float
- type varchar(20) check(type in ('ABOVE', 'BELOW', 'IN\_RANGE', 'OUT\_OF\_RANGE'))
- message varchar(500)

#### 9. Sessions

- SessionID varchar(64)
- username references User varchar(20)
- login\_time DATETIME

## Integrity constraints

- The primary key attributes are by default not allowed to be NULL.
- The node name is allowed to be NULL as every node may not have a name.
- The second threshold value in alerts can be NULL as in case of alerts of type 'ABOVE' and 'BELOW'.
- In the table DatabaseStats, the tuple (node\_ip, database) must reference the primary key of Database Table which will have to be implemented manually in InfluxDB as it doesn't support foreign key constraints.

## Test results

The figure consists of three vertically stacked screenshots of a web application titled "ApplicationMonitor".

**Screenshot 1: Login Page**  
The top screenshot shows the login page. It features a dark-themed header with the "ApplicationMonitor" logo. Below the header is a central login form with a heart rate monitor icon. The form includes fields for "Username" (containing "admin") and "Password" (containing several dots). A blue "Log in" button is at the bottom. A small "Login Failed" modal window is displayed in the upper right corner with the message "Please check your username and password".

**Screenshot 2: Dashboard Page**  
The middle screenshot shows the dashboard page. The header includes the "ApplicationMonitor" logo and a "ClusterMonitor" icon. The left sidebar has links for "Dashboard", "Alerts", and "Settings". The main area has sections for "Node statistics" and "Database statistics", both with dropdown menus. A large central area contains a "Create some cells" button and a small server icon with a speech bubble. Buttons for "Refresh" and "AutoRefresh" are at the top right.

**Screenshot 3: Settings Page**  
The bottom screenshot shows the settings page. The header includes the "ApplicationMonitor" logo and a "ClusterMonitor" icon. The left sidebar has links for "Dashboard", "Alerts", and "Settings". The main area has tabs for "Nodes", "Group", "Users", and "Database", with "Nodes" selected. It includes fields for "IP:" and "Name:", a "Groups:" dropdown (set to "Please select"), and an "Add" button. Below is a table with four rows:

IP	Name	Action
1	Node1	⋮
2	Node2	⋮
3	Node3	⋮
4	Node4	⋮

The screenshots illustrate the steps to add a new node and a new group in the ClusterMonitor application.

**Step 1: Adding a New Node**

In the first screenshot, the "Nodes" tab is selected. A modal dialog is open for adding a new node. The IP field contains "10.9.70.34" and the Name field contains "Amit". Under Groups, there is a list of existing groups: Group1, Group3, Group5, and Group2. A blue "Add" button is visible at the bottom of the dialog. Below the dialog, a table lists existing nodes:

IP	Name
10.9.70.59	Shrey
10.9.70.123	Jayesh

**Step 2: Success Confirmation**

In the second screenshot, a success message is displayed: "Success Node added successfully". The node "Amit" has been added to the list of nodes.

**Step 3: Adding a New Group**

In the third screenshot, the "Group" tab is selected. A modal dialog is open for adding a new group. The Name field contains "Group6" and the Description field contains "My new group". A blue "Add" button is visible at the bottom of the dialog. Below the dialog, a table lists existing groups:

Name	Description
Group1	
Group2	
Group3	
Group4	
Group5	
Group6	

The screenshots illustrate the process of managing users and databases in the ClusterMonitor application.

**User Registration:**

The first screenshot shows the "Users" settings page. A success message indicates "Node added successfully". The form fields are:

- Username: jayesh
- Password: ..... (redacted)
- Confirm Password: ..... (redacted)
- Groups: Group1, Group5

**Database Creation:**

The second screenshot shows the "Database" settings page. A success message indicates "Database added successfully". The form fields are:

- Name: lab5db
- Description: Database for lab3
- Node: 10.9.70.59

**Database Listing:**

The third screenshot shows the "Database" settings page with a list of databases. The table includes columns: Name, IP of server node, and Description. The list shows the following entries:

Name	IP of server node	Description
Database1	1	Database1 Description
Database2	1	Database2 Description
Database25	10	Database25 Description
Database26	10	Database26 Description
lab3db	10.9.70.123	Db 3 description
lab2db	10.9.70.59	Db description
Database3	2	Database3 Description
Database4	2	Database4 Description
Database5	2	Database5 Description
Database6	3	Database6 Description

Pagination controls at the bottom indicate pages 1, 2, 3, and 4.

The image consists of three vertically stacked screenshots of a web-based monitoring application named "ClusterMonitor".

**Screenshot 1: Dashboard**

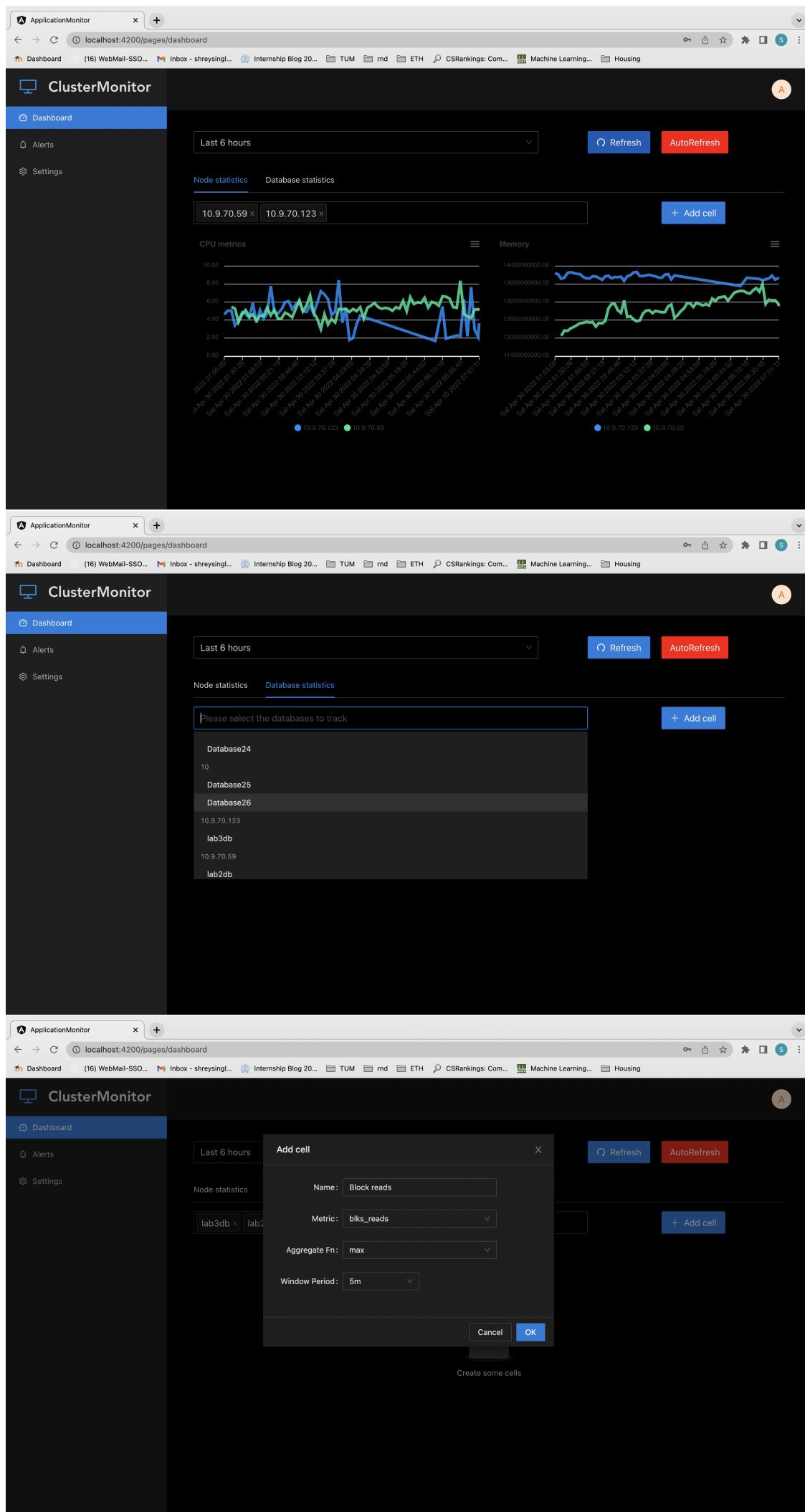
The dashboard shows a dark-themed interface with a sidebar on the left containing "Dashboard", "Alerts", and "Settings". The main area displays "Node statistics" for two nodes: "10.9.70.59" and "10.9.70.123". A central message says "Create some cells" with a "Create some cells" button. At the top right are "Refresh" and "AutoRefresh" buttons.

**Screenshot 2: Adding a CPU metrics cell**

A modal dialog titled "Add cell" is open. It contains fields for "Name: CPU metrics", "Metric: cpu", "Aggregate Fn: mean", and "Window Period: 5m". Below the dialog is a "Create some cells" message. At the bottom are "Cancel" and "OK" buttons.

**Screenshot 3: Adding a Memory cell**

A modal dialog titled "Add cell" is open. It contains fields for "Name: Memory", "Metric: mem", "Aggregate Fn: mean", and "Window Period: 5m". Below the dialog is a line chart showing memory usage over time for node "10.9.70.59". At the bottom are "Cancel" and "OK" buttons.



**ClusterMonitor Dashboard**

Last 6 hours

Node statistics Database statistics

lab3db lab2db

+ Add cell

Block reads Number of tuples fetched

Block reads chart (left): Y-axis ranges from 0.00 to 25000.00. X-axis shows time from Sat Apr 30 2022 01:05:00 to Sat Apr 30 2022 07:52:00. Two bars are shown: 10.9.70.59\_lab2db (blue) and 10.9.70.123\_lab3db (green). Both bars are at approximately 20000.00.

Number of tuples fetched chart (right): Y-axis ranges from 0.00 to 8000000.00. X-axis shows time from Sat Apr 30 2022 01:05:00 to Sat Apr 30 2022 07:52:00. Two bars are shown: 10.9.70.59\_lab2db (blue) and 10.9.70.123\_lab3db (green). Both bars are at approximately 6000000.00.

**ClusterMonitor Alerts**

+ Create Alert

**Create Alert**

Name: Alert1

Description: Alert on CPU

Entity: node

Metric: cpu

Nodes: 10.9.70.59

Threshold Type: ABOVE

Threshold: 10

Priority: CRIT

Alert message: Your CPU utilisation is high!

Cancel OK

The screenshots illustrate the ClusterMonitor application's interface for managing alerts. The application has a dark-themed UI with a sidebar containing 'Dashboard', 'Alerts' (which is selected), and 'Settings'. A 'Create Alert' button is located in the top right.

**Screenshot 1:** Shows two alerts: 'Alert1' (Alert on CPU, 0 unacknowledged alerts) and 'Alert2' (Another Alert on CPU, 0 unacknowledged alerts). Both alerts have a red border.

Alert ID	Type	Count
Alert1	Alert on CPU	0
Alert2	Another Alert on CPU	0

**Screenshot 2:** Shows three alerts: 'Alert1' (Alert on CPU, 0 unacknowledged alerts), 'Alert2' (Another Alert on CPU, 0 unacknowledged alerts), and 'Alert3' (Minor alert on CPU, 2 unacknowledged alerts). 'Alert3' has a blue border.

Alert ID	Type	Count
Alert1	Alert on CPU	0
Alert2	Another Alert on CPU	0
Alert3	Minor alert on CPU	2

**Screenshot 3:** A detailed view of 'Alert3' showing two entries in a table. Each entry includes a timestamp, the alert type ('CPU'), and an 'Acknowledge' button.

Timestamp	Type	Action
Apr 30, 2022, 7:26:47 AM	CPU	Acknowledge
Apr 30, 2022, 7:28:00 AM	CPU	Acknowledge

**Alert3 Details:**

- Timestamp: Apr 30, 2022, 7:26:47 AM
- Type: CPU
- Action: Acknowledge

**Alert3 Details:**

- Timestamp: Apr 30, 2022, 7:28:00 AM
- Type: CPU
- Action: Acknowledge

The screenshots illustrate the ClusterMonitor application's alert management features:

- Screenshot 1: Alert Acknowledgment**  
A modal window titled "Alert3" shows an alert entry for "Alert1" (Alert on CPU). It includes a timestamp ("Apr 30, 2022, 7:26:47 AM") and an "Acknowledge" button. Two success notifications are visible on the right: "Alert acknowledged successfully" (green checkmark) and "Alert acknowledged successfully" (green checkmark).
- Screenshot 2: Alert List**  
The main dashboard displays three alerts:
  - Alert1**: Alert on CPU, 0 unacknowledged alerts.
  - Alert3**: Minor alert on CPU, 6 unacknowledged alerts.
  - Alert2**: Another Alert on CPU, 1 unacknowledged alerts.A "+ Create Alert" button is located at the top right.
- Screenshot 3: Alert Deletion Confirmation**  
A confirmation dialog box asks, "Do you want to delete this alert?" with "Cancel" and "OK" buttons. The background shows the same alert list as Screenshot 2.

## 1 Further Improvements

1. Predictive analysis - Currently, the user needs to have the domain knowledge to get reasonable value of threshold. Our application monitor could suggest some values of threshold to the user.
2. Save dashboard - Currently, the user has to make the dashboard everytime the user restarts our application monitor. We could save the user preferences for quick access.

## Github Link

<https://github.com/jayeshs999/ApplicationMonitor.git>