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**
   he 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 (aboe, 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

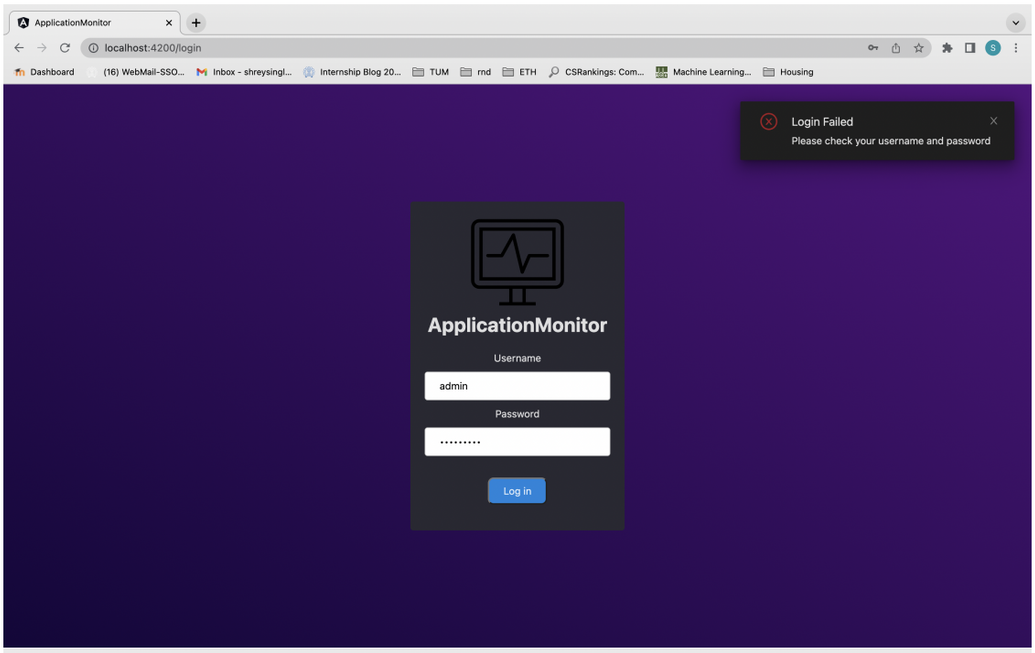
- threshold_high float
- type varchar(20) check(type in ('ABOVE', 'BELOW', 'IN_RANGE', 'OUT_OF_RANGE'))
- message varchar(500)

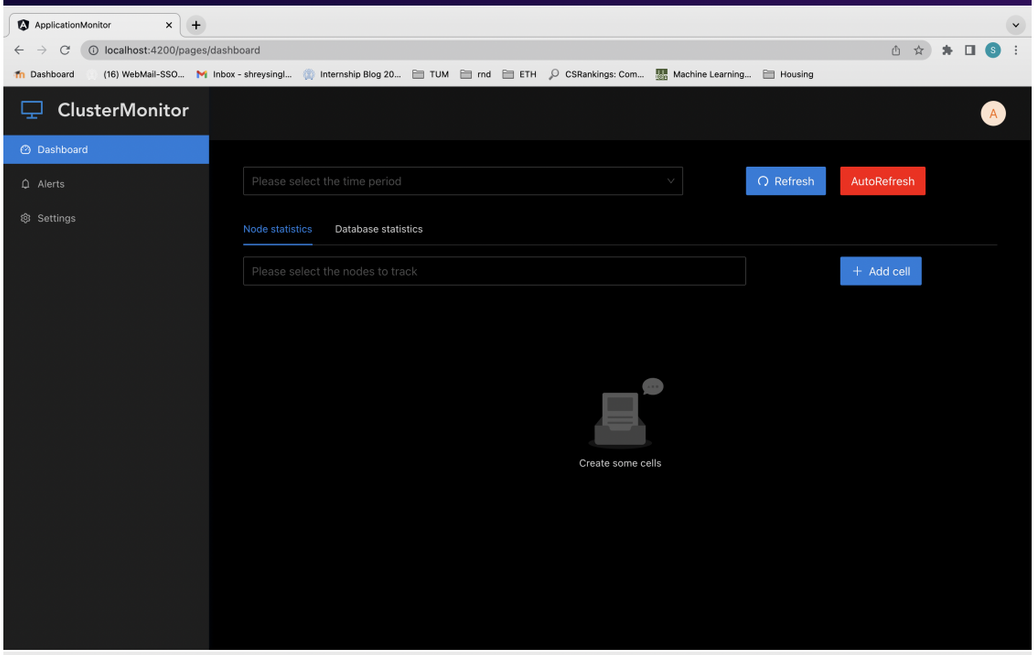
9. Sessions
    - SessionID varchar(64)
    - username references User varchar(20)
    - login_time DATETIME

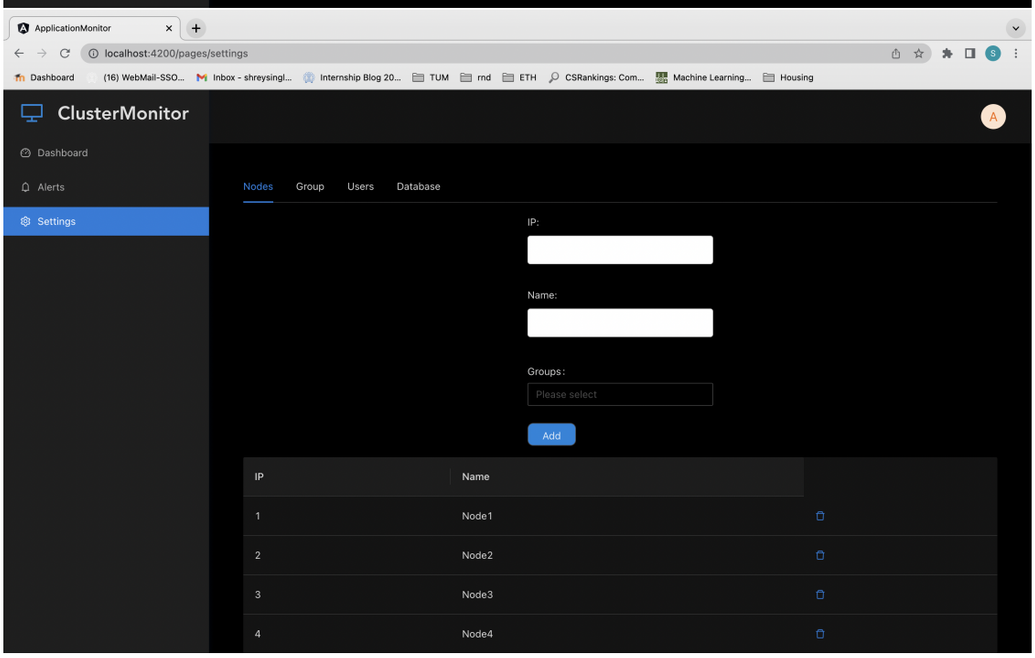
Integrity constraints

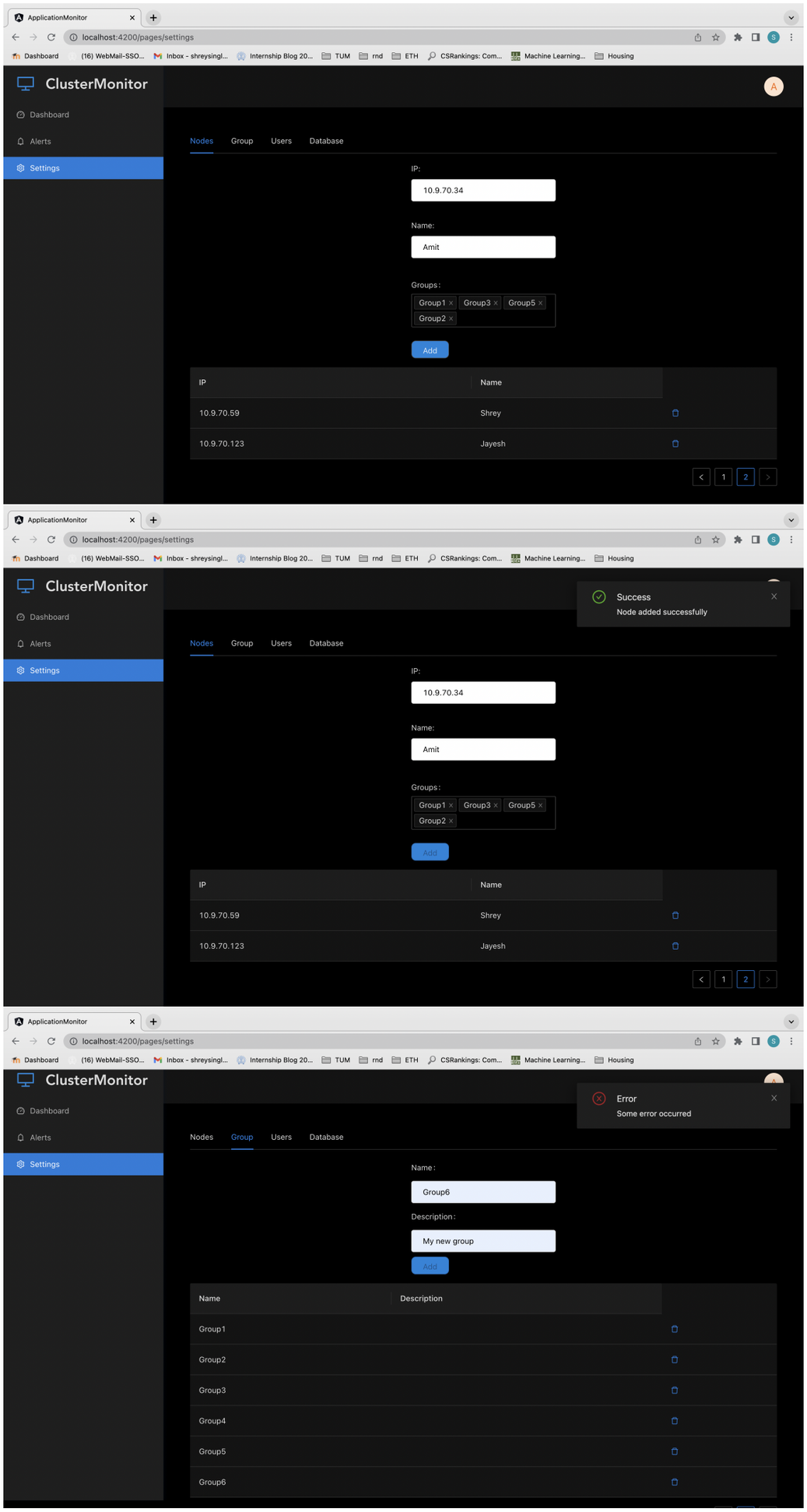
- The primary key attributes are by default not allowed to be NULL.
- The node name is allowed to be NULL as every node may not have a name.
- The second threshold value in alerts can be NULL as in case of alerts of type 'ABOVE' and 'BELOW'.
- In the table DatabaseStats, the tuple (node_ip,database) 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.

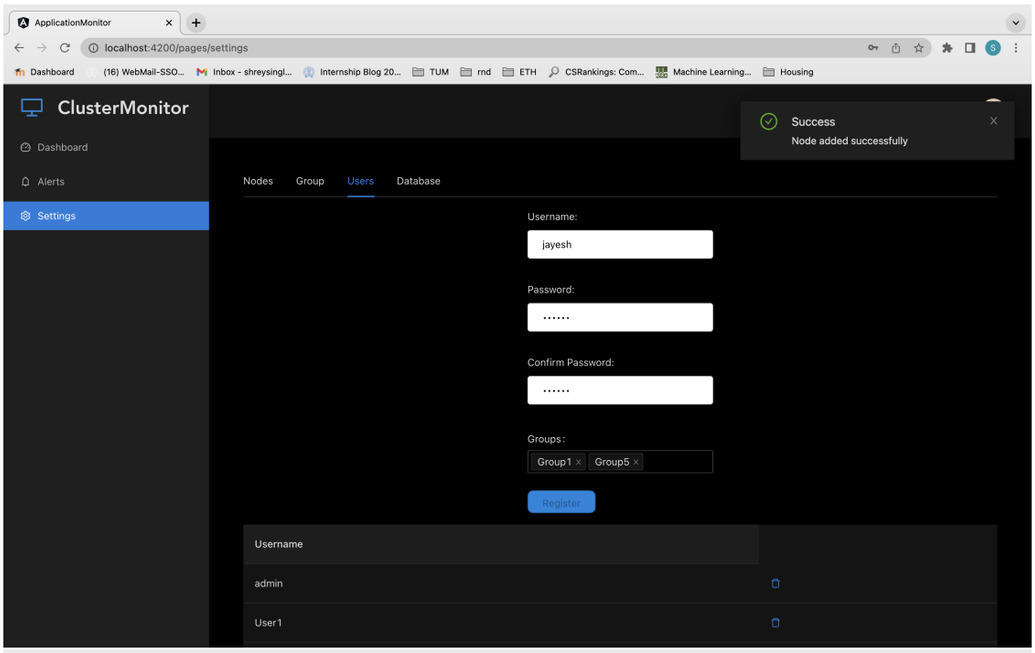
Test results

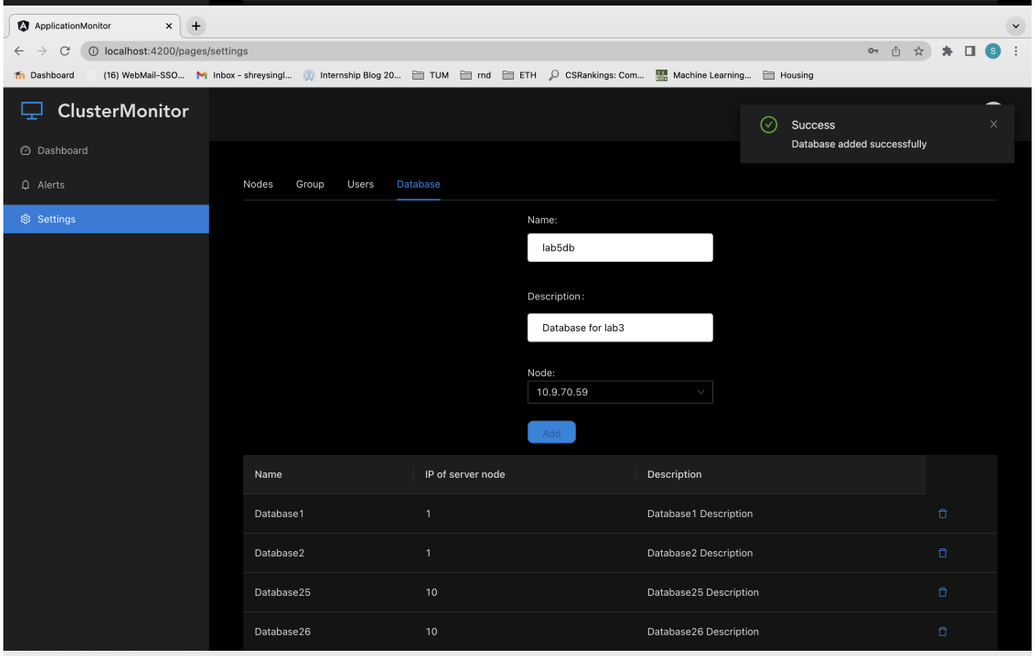


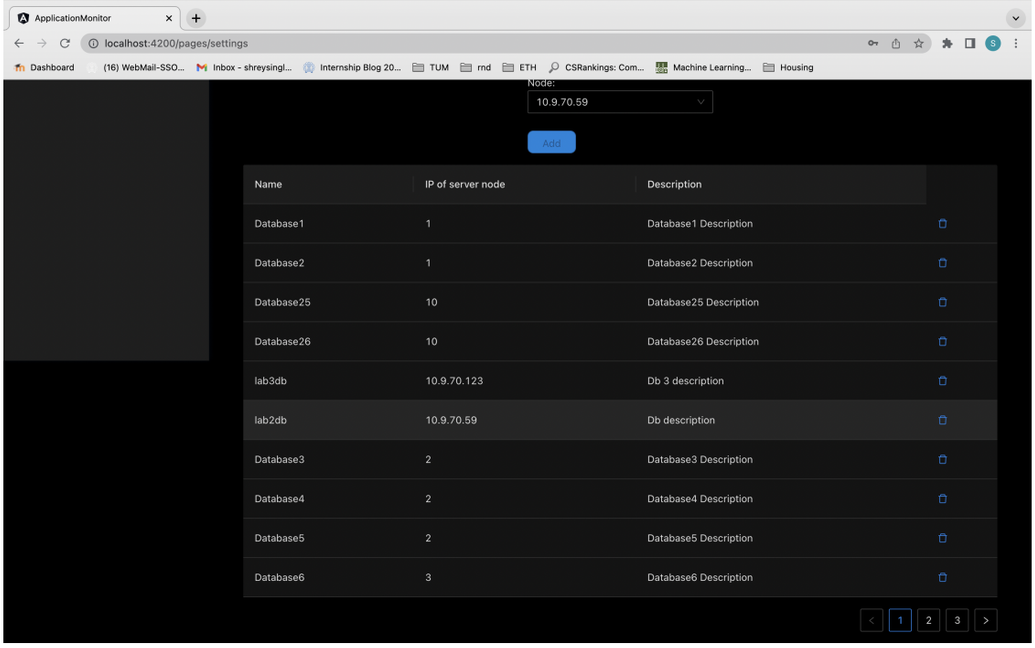


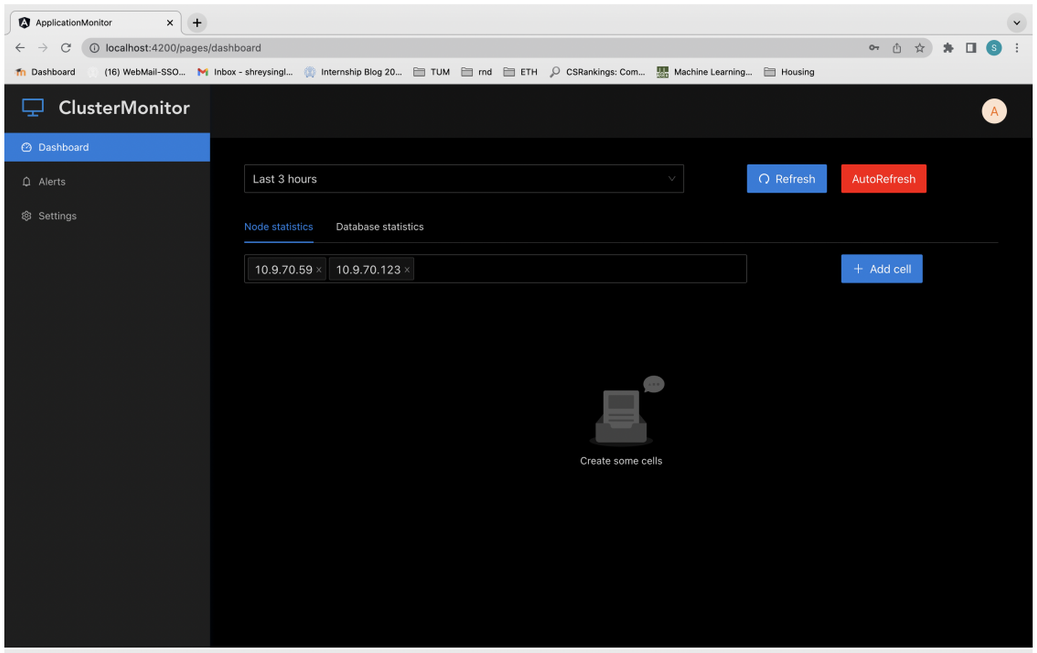


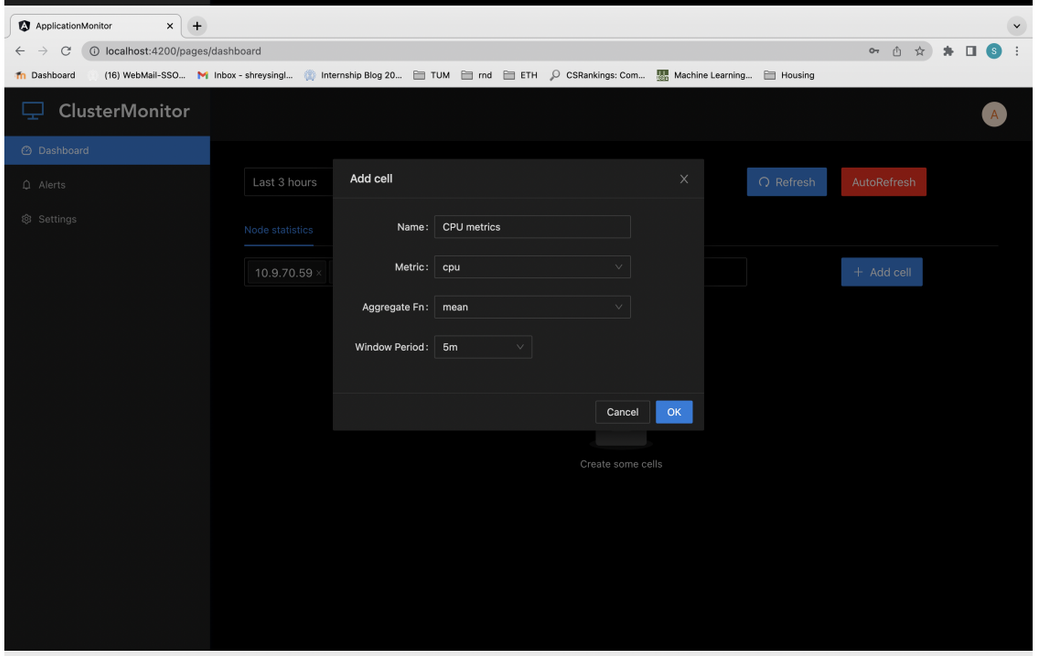

ApplicationMonitor
localhost:4200/pages/settings
ClusterMonitor
Dashboard
Alerts
Settings
Nodes
Group
Users
Database
IP:
10.9.70.34
Name:
Amit
Groups:
Group1
Group3
Group5
Group2
Add
IP
Name
10.9.70.59
Shrey
10.9.70.123
Jayesh
Success
Node added successfully
Error
Some error occurred
Description:
My new group
Group6
Group4

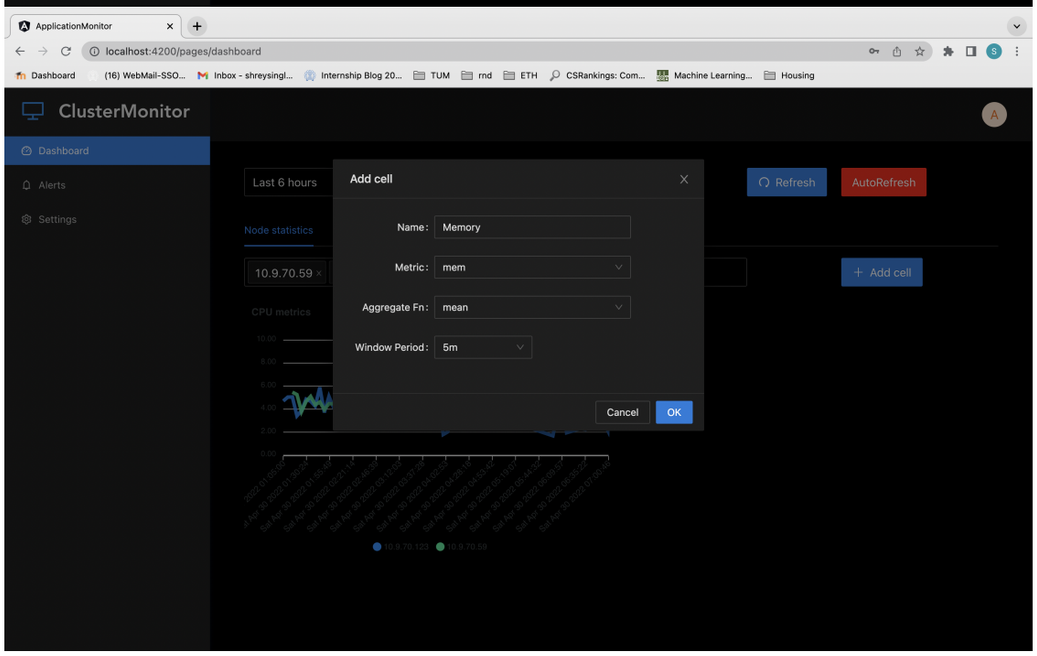

ApplicationMonitor
localhost:4200/pages/settings
ClusterMonitor
Dashboard
Alerts
Settings
Success
Node added successfully
Nodes
Group
Users
Database
Username:
jayesh
Password:
Confirm Password:
Groups:
Group1
Group5
Register
Username
admin
User1

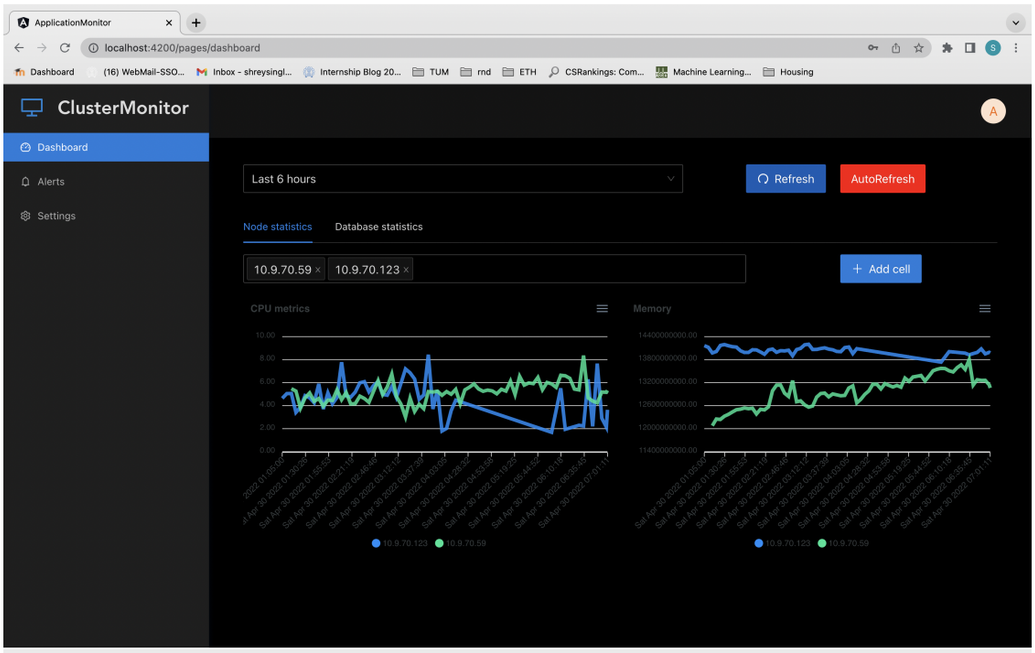

ApplicationMonitor
localhost:4200/pages/settings
ClusterMonitor
Dashboard
Alerts
Settings
Success
Database added successfully
Nodes
Group
Users
Database
Name:
lab5db
Description:
Database for lab3
Node:
10.9.70.59
Add
Name
IP of server node
Description
Database1
1
Database1 Description
Database2
1
Database2 Description
Database25
10
Database25 Description
Database26
10
Database26 Description

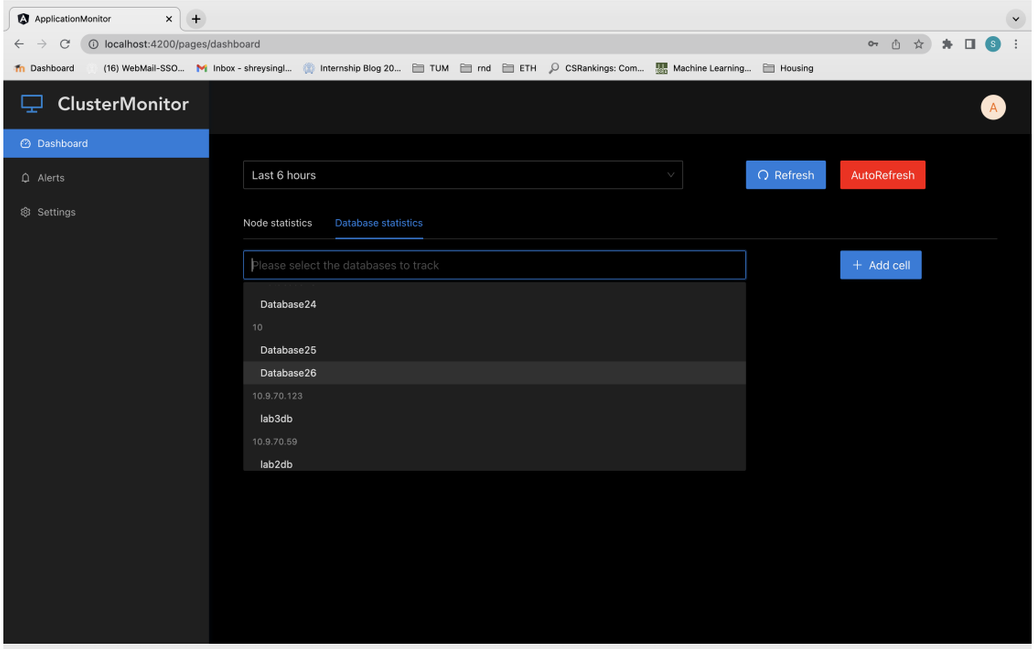

ApplicationMonitor
localhost:4200/pages/settings
Node:
10.9.70.59
Add
Name
IP of server node
Description
Database1
1
Database1 Description
Database2
1
Database2 Description
Database25
10
Database25 Description
Database26
10
Database26 Description
lab3db
10.9.70.123
Db 3 description
lab2db
10.9.70.59
Db description
Database3
2
Database3 Description
Database4
2
Database4 Description
Database5
2
Database5 Description
Database6
3
Database6 Description
1
2
3

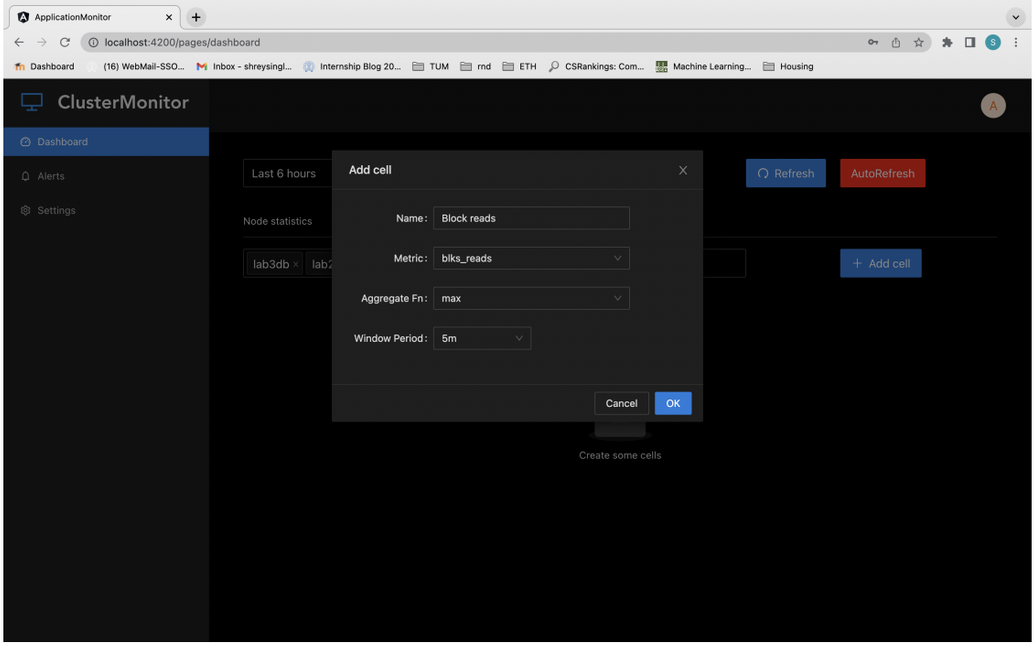

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 3 hours
Refresh
AutoRefresh
Node statistics
Database statistics
10.9.70.59
10.9.70.123
Add cell
Create some cells

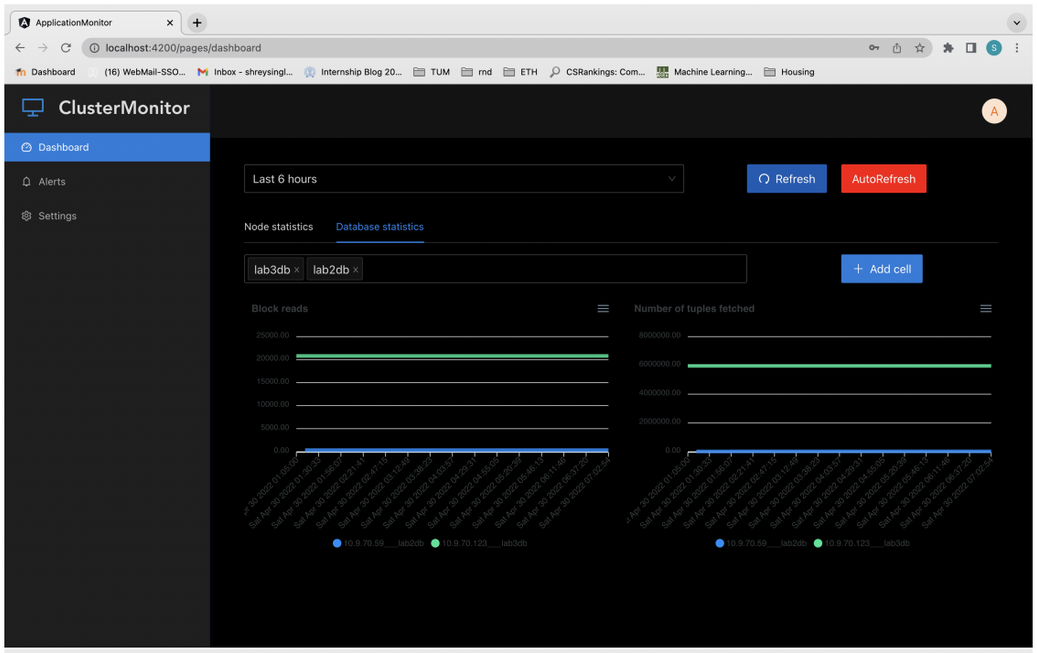

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 3 hours
Refresh
AutoRefresh
Node statistics
10.9.70.59
Add cell
Add cell
Name: CPU metrics
Metric: cpu
Aggregate Fn: mean
Window Period: 5m
Cancel
OK
Create some cells

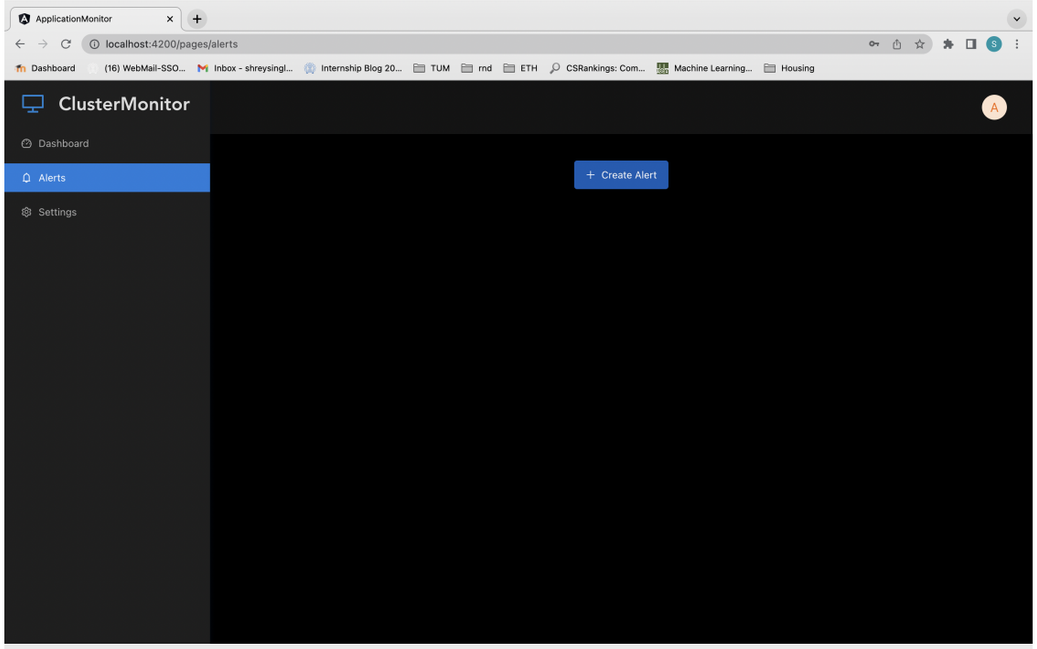

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 6 hours
Refresh
AutoRefresh
Node statistics
10.9.70.59
Add cell
Add cell
Name: Memory
Metric: mem
Aggregate Fn: mean
Window Period: 5m
Cancel
OK
CPU metrics

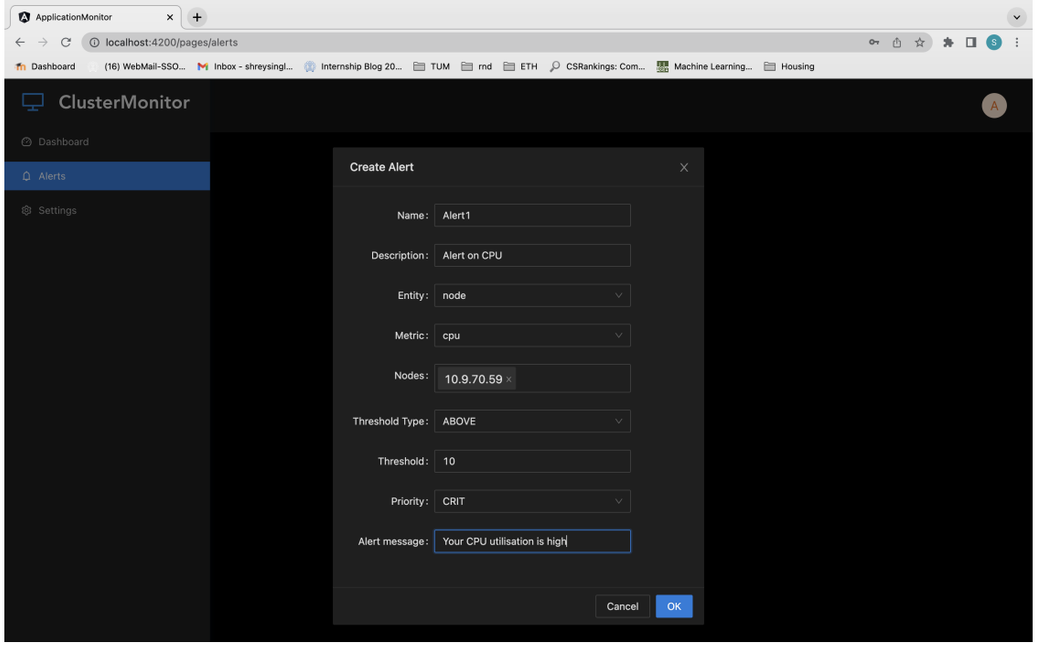

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 6 hours
Refresh
AutoRefresh
Node statistics
Database statistics
10.9.70.59
10.9.70.123
Add cell
CPU metrics
Memory

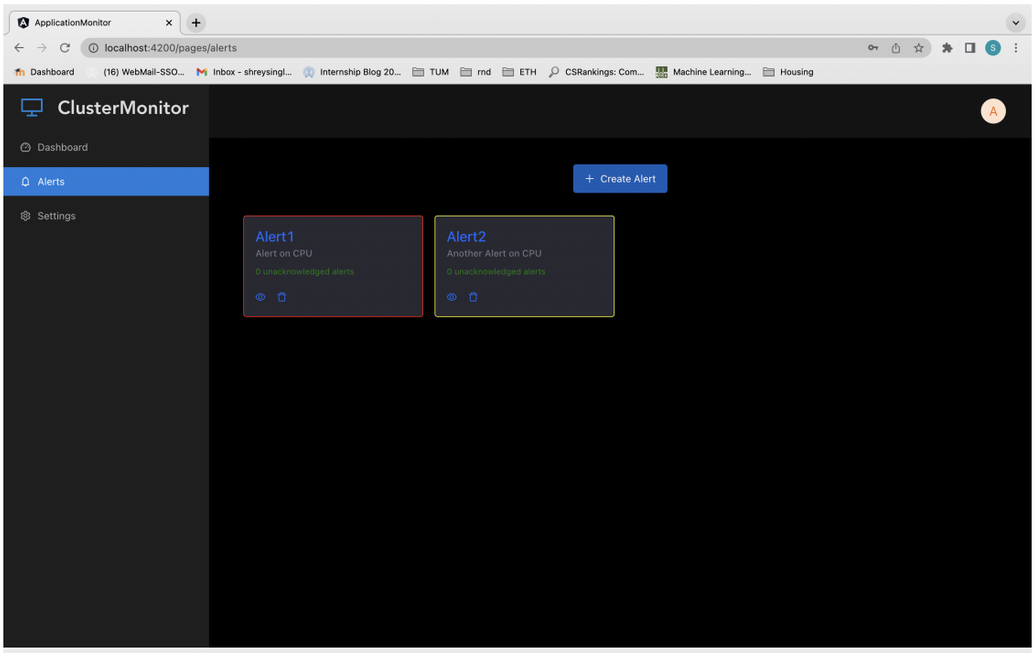

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 6 hours
Refresh
AutoRefresh
Node statistics
Database statistics
Please select the databases to track
Add cell
Database24
10
Database25
Database26
10.9.70.123
lab3db
10.9.70.59
lab2db

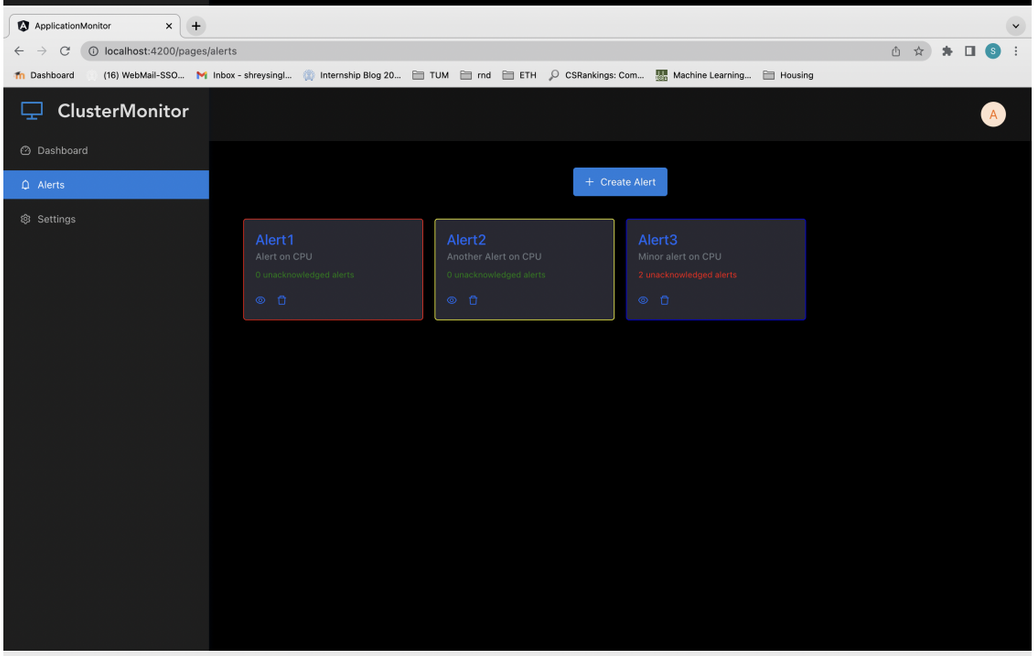

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 6 hours
Refresh
AutoRefresh
Node statistics
lab3db
Add cell
Add cell
Name: Block reads
Metric: blks_reads
Aggregate Fn: max
Window Period: 5m
Cancel
OK
Create some cells

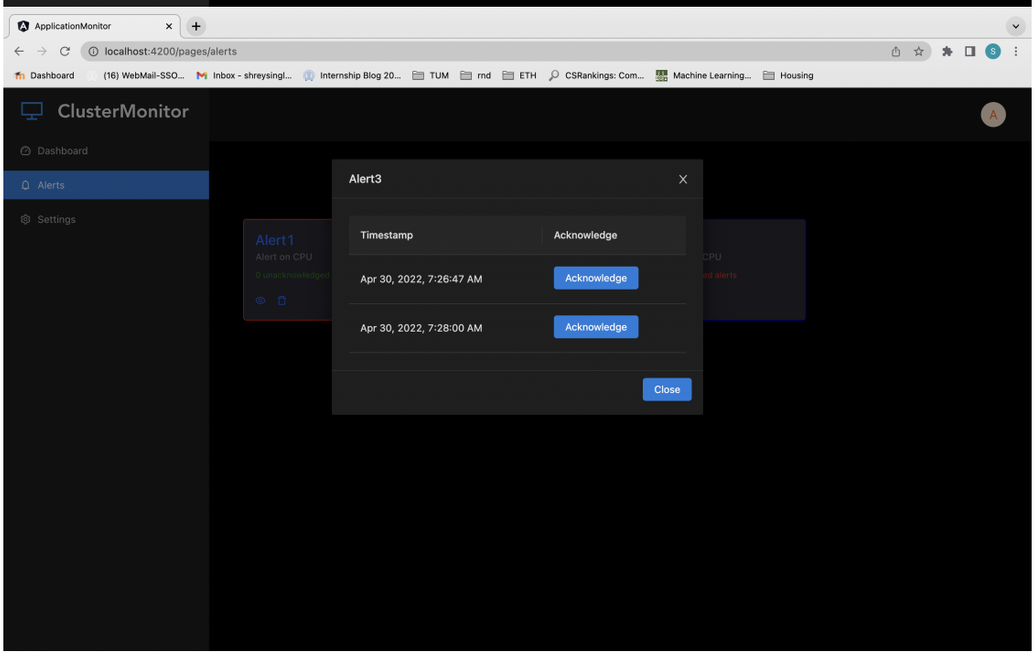

ApplicationMonitor
localhost:4200/pages/dashboard
ClusterMonitor
Dashboard
Alerts
Settings
Last 6 hours
Refresh
AutoRefresh
Node statistics
Database statistics
lab3db
lab2db
Add cell
Block reads
Number of tuples fetched

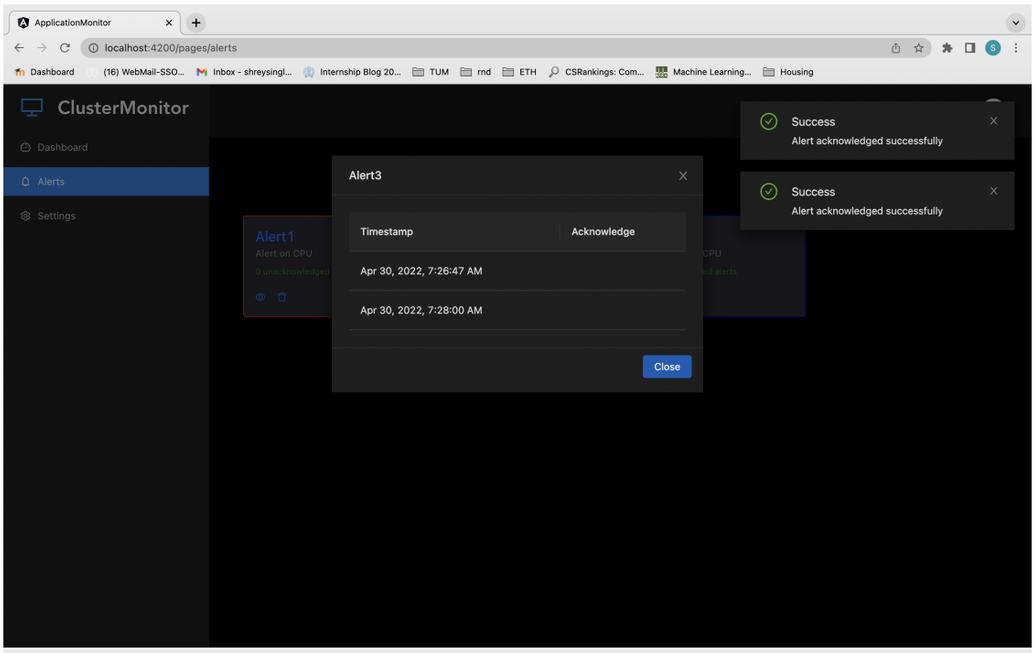

ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Create Alert

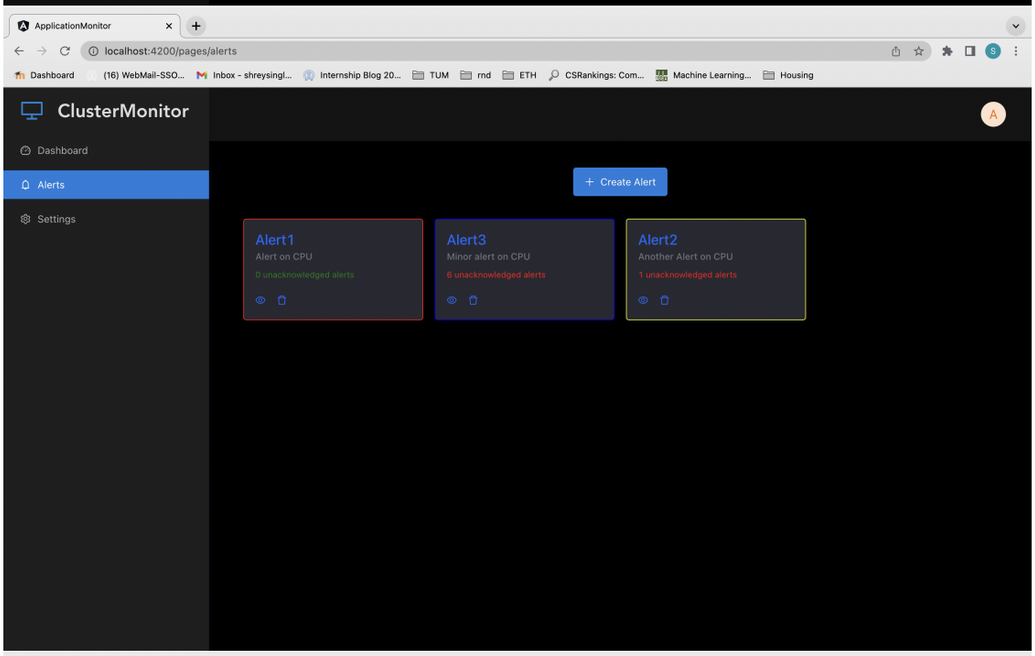

ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Create Alert
Name: Alert1
Description: Alert on CPU
Entity: node
Metric: cpu
Nodes: 10.9.70.59
Threshold Type: ABOVE
Threshold: 10
Priority: CRIT
Alert message: Your CPU utilisation is high
Cancel
OK

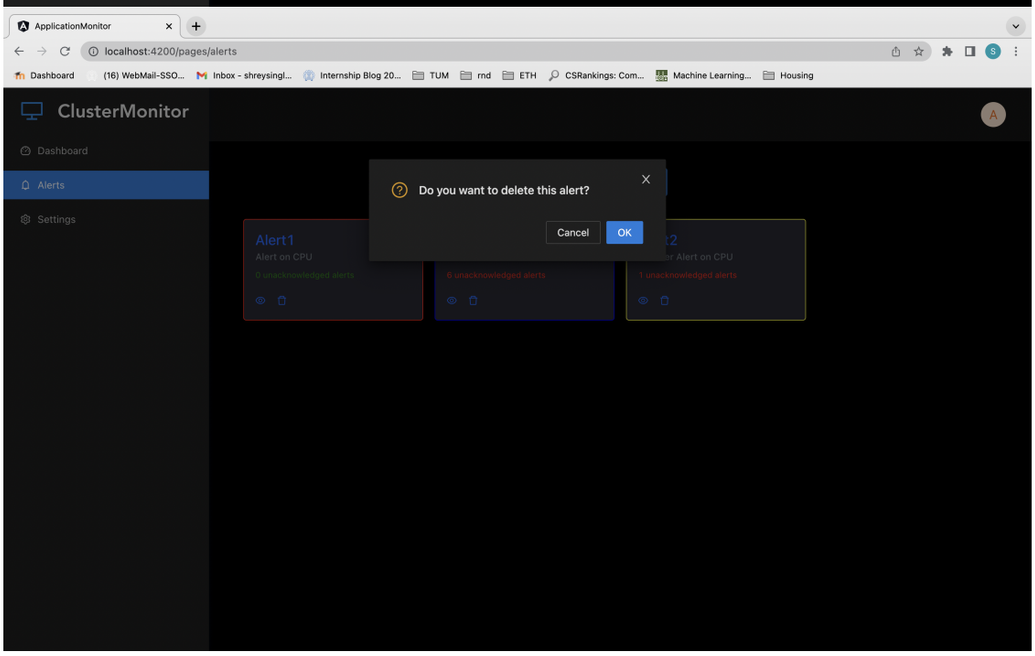

ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Create Alert
Alert1
Alert on CPU
0 unacknowledged alerts
Alert2
Another Alert on CPU
0 unacknowledged alerts


ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Create Alert
Alert1
Alert on CPU
0 unacknowledged alerts
Alert2
Another Alert on CPU
0 unacknowledged alerts
Alert3
Minor alert on CPU
2 unacknowledged alerts


ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Alert3
Timestamp
Acknowledge
Apr 30, 2022, 7:26:47 AM
Acknowledge
Apr 30, 2022, 7:28:00 AM
Acknowledge
Close


ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Success
Alert acknowledged successfully
Success
Alert acknowledged successfully
Alert3
Timestamp
Acknowledge
Apr 30, 2022, 7:26:47 AM
Apr 30, 2022, 7:28:00 AM
Close
Alert1
Alert on CPU


ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Create Alert
Alert1
Alert on CPU
0 unacknowledged alerts
Alert3
Minor alert on CPU
6 unacknowledged alerts
Alert2
Another Alert on CPU
1 unacknowledged alerts


ApplicationMonitor
localhost:4200/pages/alerts
ClusterMonitor
Dashboard
Alerts
Settings
Do you want to delete this alert?
Cancel
OK
Alert1
Alert on CPU
0 unacknowledged alerts

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