# Watchdog System - RTC System Monitoring Tool

## Tables of content :

### About Watchdog :

Watchdog is a tool that pulls real time metrics of a system (could be multiple) in order to monitor the resource consumption.

The advantage of using Watchdog is that it gives you a scope for utilizing CPU resources in a much more effective way.

It'll be feasible for anyone to manage their computer resources.

### Tech Stack

- Backend :
  - NodeJS
  - Socket.io
  - Express
  - MongoDB
  - Redis

- Frontend :
    - ReactJS
    - Socket.io client
    - ChartsJS

Note : To generate binaries for the system to be able to get metrics **pkg** has been used.

## Calculating CPU Load & Memory

### CPU Load :

start = {totalMs, idleMs}

end = {totalMs, idleMs}

idleDifference = end.idleMs - start.idleMs

totalDifference = end.totalMs - start.totalMs

cpuLoad = 100 - (100 * idleDifference) / totalDifference %

### Why start & end ?

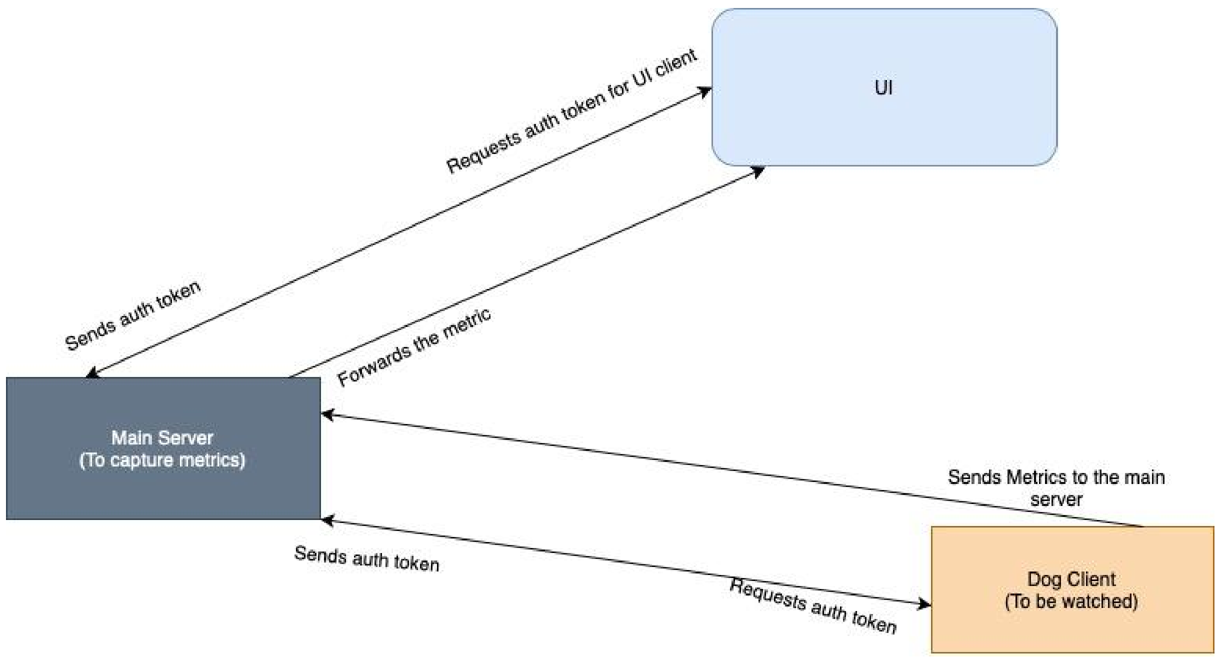
*After 100 milliseconds we compute the next average so that we can get the difference & calculate the percentage.*

### Memory :

usedMemory = totalMemory - freeMemory

memoryUsage = ( (usedMemory / totalMemory ) * 100 ) / 100

## Workflow of the tool :



## Workflow in details :

### Main Server :

The main server is using **Express** to serve with **cluster** module to spread work across all the **threads,** now it has one **endpoint** where client will request for an **auth token** & using that token live connection will be established between client & server.

Each client might not be forever connected to the server so to make sure that they get connected with the same worker thread we **hash** the **IP** with **farm hash** & using **sticky session** when the same client comes we connect it with the same worker thread.

The main server is ready to accept the metrics from the dog client & forward it to the UI client.

### Dog Server :

The server to be watched will be using **socket.io client** to listen to **“connect” event** which is getting emitted from main server then the dog server **emits 2 events** “**clientAuth**” which’ll be used for authorizing the dog client & 2nd one will be “**performanceData**” which is to compute & send all the metrics to the main server.**Main server will be listening for both of these events.**

**Libraries used for building tool :**

**Dog Server** : pkg, socket.io client & os

**Main Server** : socket.io, express, farmhash, socket.io/redisAdapter, crypto-js, jsonwebtoken, redis & ioredis

## Annexures :

### Why is NGINX not a fit ?

- Usually we use a **reverse proxy** server (**NGINX**) to for not exposing our server to public but here NGINX won’t be good fit at all the reason is that if we don’t expose the server to public the client would be the main server itself & everytime some dog client sends the metric the **IP will not be considered for the client but of the main server instead.**
- **Therefore,** the main server is deployed using **pm2** in the **background process.**

**How to use Nodejs workers in Cores (The worker module) ?**

- The **worker module** comes with Nodejs by default.
- To spread load across the workers check whether the primary thread is currently used or not if yes the use the “**cluster.fork()**” method in a loop till the **no. of cores the system has.**
- In case of any **worker thread process** getting killed the event “**exit**” comes handy which can **respawn** that thread again once its process has been killed.

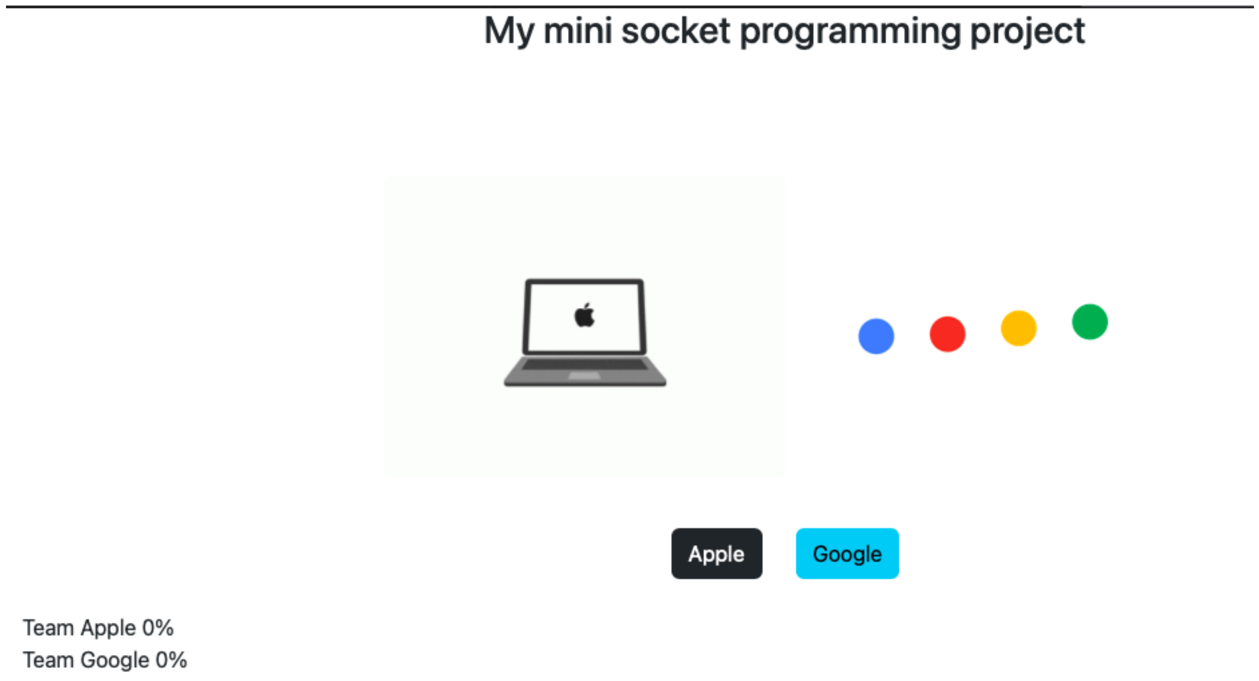
**How to make Nodejs binaries using pkg ?**

- **pkg** is a handy library that understands Nodejs environment & makes binaries for 3 systems by default for the same **CPU architecture.**

- Command to generate binaries : **pkg file_name.js**

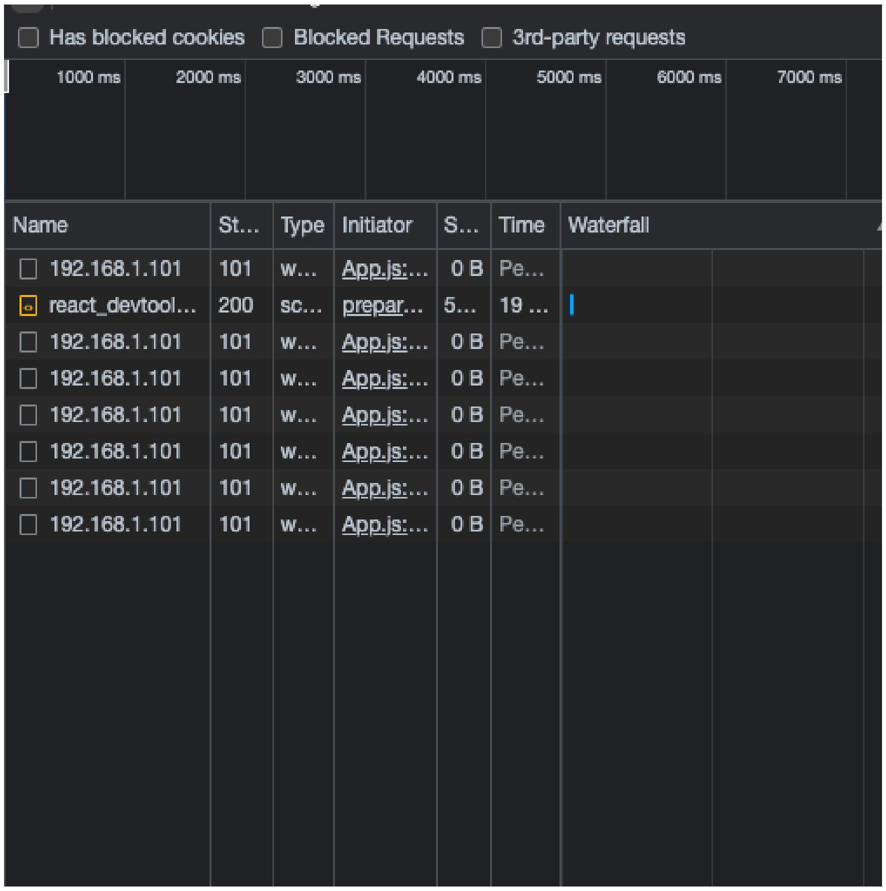
**A Demonstration for WebSocket polling with Sockets**

Down below is an polling app which shows the percentage of people voting for apple or google :



Team Apple 0%
Team Google 0%

When we choose our vote & go to the developer settings in the **Network Tab** it is visible that the protocol switches from **200** to **101** which is **switching protocol** the protocol gets updated to **websockets**



Has blocked cookies   Blocked Requests   3rd-party requests

1000 ms   2000 ms   3000 ms   4000 ms   5000 ms   6000 ms   7000 ms

| Name | St… | Type | Initiator | S… | Time | Waterfall |
|---|---|---|---|---|---|---|
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |
| react_devtool… | 200 | sc… | prepar… | 5… | 19 … | |
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |
| 192.168.1.101 | 101 | w… | App.js:… | 0 B | Pe… | |