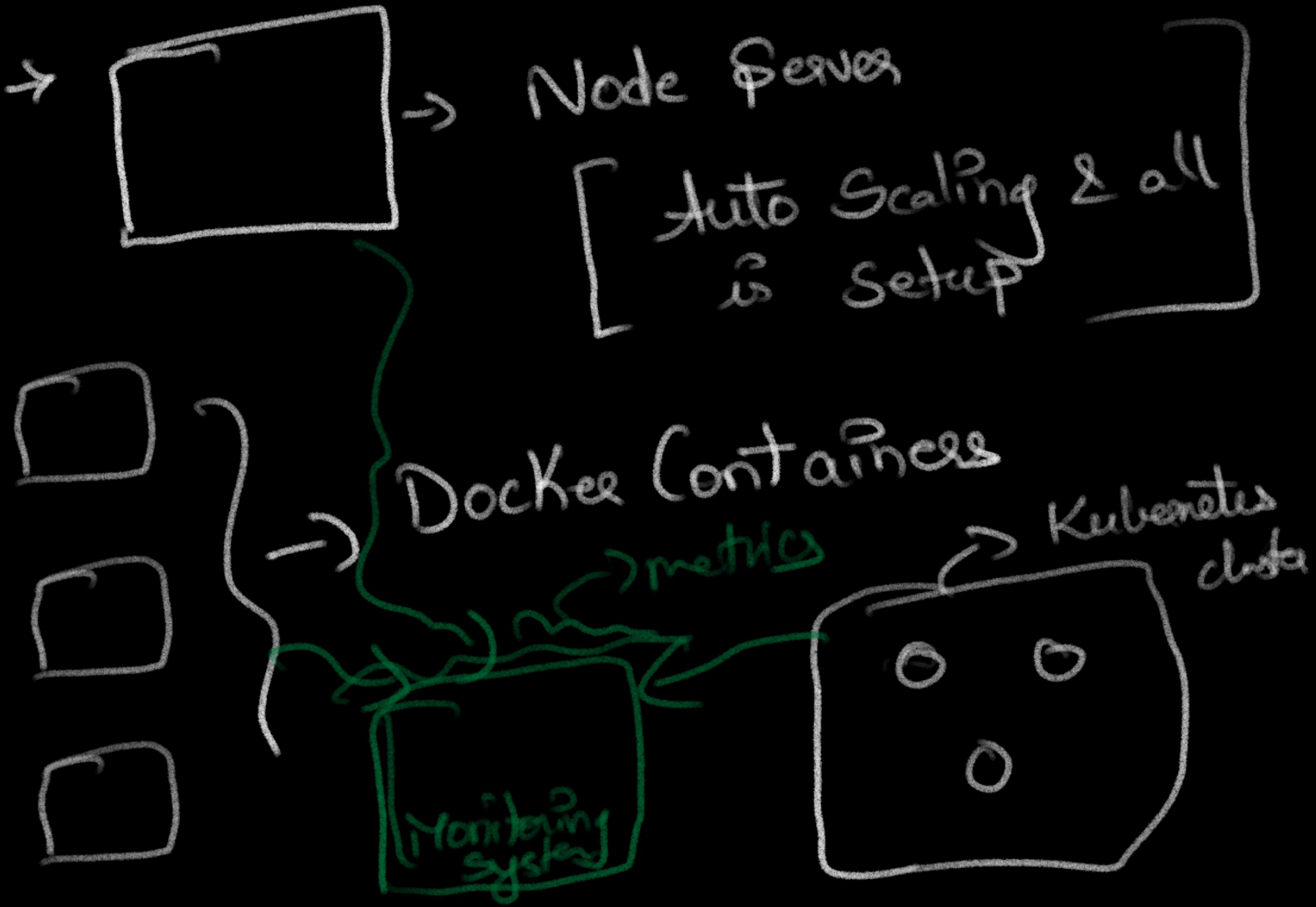
Monitoring with Grafana, Prometheus & loki



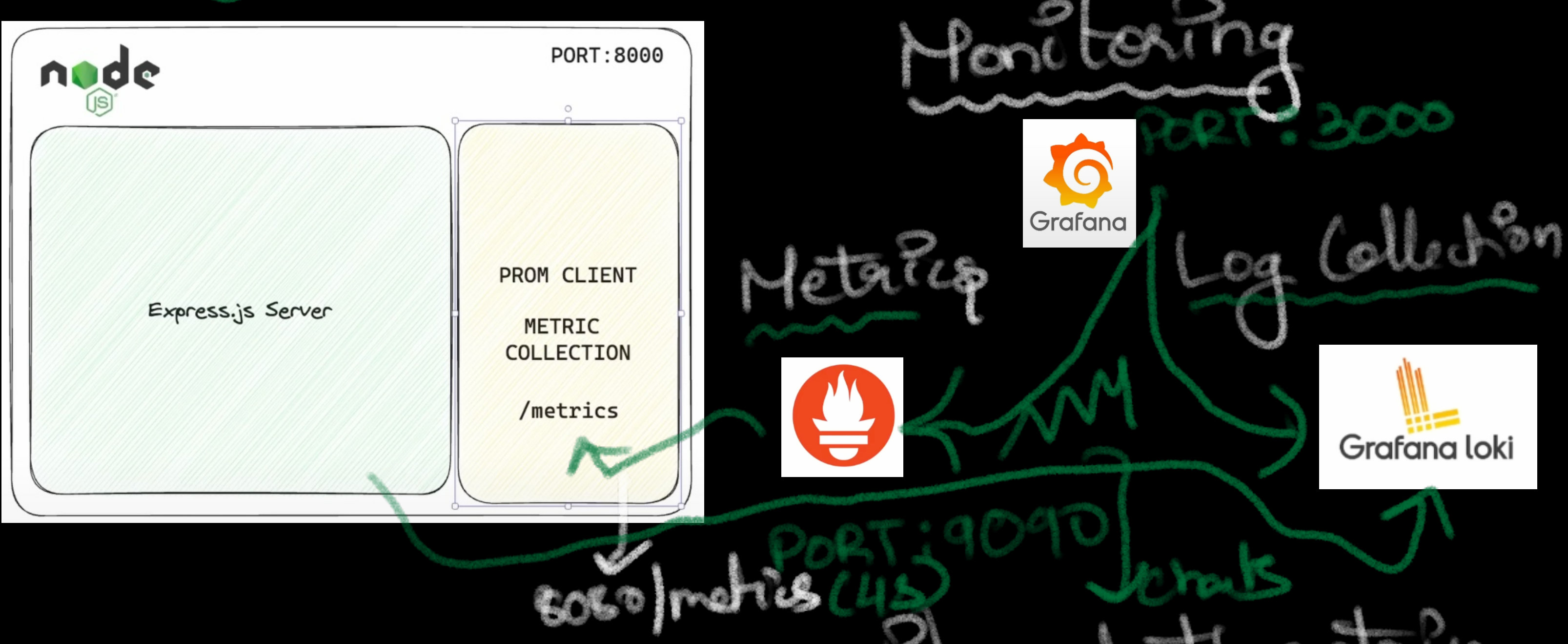
→ Now, Consider you have deployed the app & it's up & running.

→ Now, you've went to sleep & the morning you've received message that the application was slow & throwing few errors.

→ So, To monitor if and when it is down, you need to setup a monitoring system.

→ So, You need to set a Central Monitoring system & get the metrics of all your server, containers & clusters.

→ For this, we will be using Grafana, Prometheus & Loki



→ we have to monitor both metrics & logs

→ We will use prometheus to gather metrics & we will install a prometheus client over our server

→ Node : `npm i prom-client` & integrate code in node.

→ We then run a prometheus server on port 9090, this server will get the metrics of prom client

→ We will give a 4s time interval to scrape the data again & again

→ We then use grafana to visualize those metrics.

→ Grafana will interact with promethe
& then will give us charts

→ After Running grafana, the username & pass by default are admin then click skip then click 'add data source'

→ Choose prometheus & copy your Ip addr & prometheus port. If your running on your local machine, go to the Bottom hit Btn 'Save & test'.

→ Go to Home, Create a Dashboard, Add a Visualization & Select prometheus data source

→ Then go to metrics explore & select metric & run query.

→ After Running Query, you can choose the visualization type to time series, gauge & all

→ Instead of adding each manually you can google NodeJS grafana in google & go to grafana site & copy the dashboard ID.

→ Now, go to dashboard, click new & click import & paste ID & select data source & Run.

→ By default prometheus keeps 15 days data.

→ You can also create custom metrics & in the code attached in repo I have created a custom metric
[' Req & Resp time of each route']
& Total Requests .

→ We should select label Route!= /metrics route & add histogram for [' Req & Resp time of each route']

→ Run Grafana loki with docker &
[ ] to work with node

→ Save DashBoard & Create a data source with Loki, and now add visualization dashboard by Selecting loki and label (level) & (info) Since in code we use logger.info & Similar for (error) { logger.error}