Introduction

- We have 2 Machines(which I created using docker) running same service each and 1 Monitor(also on docker) which monitors resource availability and send requests accordingly.
- So, At any point of time we need to know the state of Machines and wheather services are running or not in them.
- I used node js to create server over python because node js is inherently asynchronous which means when I use Ping-Echo technique, I do not have to wait for one Echo for other to come after, and node is faster.

Machines

- To know if a Machine is running or not, I used `ping` in linux (which performs similar role to Ping-Echo technique by sending ICMP ECHO_REQUEST to hosts, which reply back).
- To know if a service is running or not, I used Ping-Echo technique and created `/ping` endpoint. If we send GET request to this endpoint, it sends "pong" back. By this, I determine if the service is running.
- In `/` endpoint of server, I calculated sum and random if query contains 2 numbers, by arguments `first` and `second`.

Monitor

- At any time, Monitor needs to keep track of what is the state of system and whom to send request.
- Since state can change any time, I decided on an interval `pollFreq` at which Monitor will use `ping` and `/ping` to decide what is the current state. Since `ping` and `http request` depend on reply for completion, I had to keep timeouts for them to update state continuously.
- If Monitor recieves request, it checks current state and sends request accordingly. If S1 is running, send request to S1 and if it fails(it can happen that in this timeframe state changed), send to S2.
- To dynamically update webpage, I used sockets to create a connection with client and send status updates so I do not have to keep suppling whole pages.

Steps to run

- Go to Machine1 folder and build image. `docker build -t machine1`. Do same for Machine2(name `machine2`) and Monitor(name `monitor`).
- Create a network that accepts 1.2.3.* IPs `docker network create --subnet 1.2.3.0/24 a1_net`
- Create containers from images.

```
docker container run --rm -d --net a1_net --ip 1.2.3.100 --name machine1 machine1
docker container run --rm -d --net a1_net --ip 1.2.3.200 --name machine2 machine2
docker container run --rm -d --net a1_net --ip 1.2.3.150 --name monitor monitor
```

- Monitor and Machines are running at `1.2.3.150:3000`, `1.2.3.100:3000` and `1.2.3.200:3000` respectively. Use `docker pause` to stop service, and `docker stop` to turn machine off.