Class Activity – Load Balancing HTTP Requests

- Consider a cloud with 4 machines that accepts HTTP requests. Each requests triggers some "heavy" processing, and one server would not be enough to satisfy the demand – hence, we need to spread the requests between the 4 node servers.
- We will not be using a real could – you will be emulating this setup on your local machine.

Class Activity – Load Balancing HTTP Requests (2)

- wasteCycles.js is a Node program that serves the following HTTP requests:
  - /cycles: occupies the CPU for one second
  - (any string ending with .html or .js): returns the contents of the file (as you did in class)
  - Due to the single-threadness of Node, if you execute more than one request per second, the program will delay the processing of further requests
  - The program takes as input a port number to listen to.
- waste.js is a Node program that submits a request to *wasteCycles.js* every 800ms.
- Upon running waste.js, you will notice that the processing time goes well above 1000ms

Your task:

- You should launch four instances of wasteCycles.js to emulate four cloud servers
  - node wasteCycles.js 8080, node wasteCycles.js 8081, node wasteCycles.js 8082, node wasteCycles.js 8083
- Implement loadBalancer.js , a Node program accepts requests on port 8079 and submits them in a round-robin fashion to one of the node servers (wasteCycles)
- To test, modify the port number in waste.js to be 8079
- If your implementation is correct, the processing time should be stable around 1000ms