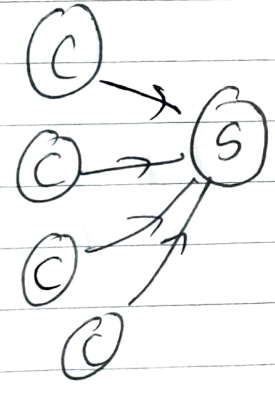
Load Balancers

Load Balancer → (i) A server that sits b/w a set of clients and set of servers which distributes load / reroutes traffic evenly across all servers in our set.
(balances workload)

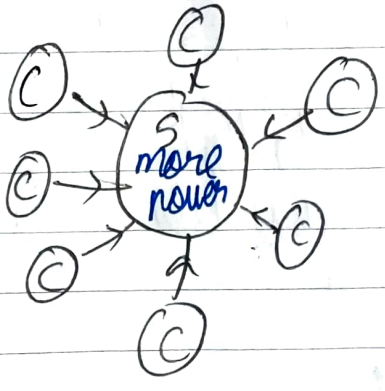
(ii) This prevents 1 server from getting too overloaded by too many client reqs.

(iii) Also clients do not know abt load balancers' existence so load balancer ≡ reverse proxy

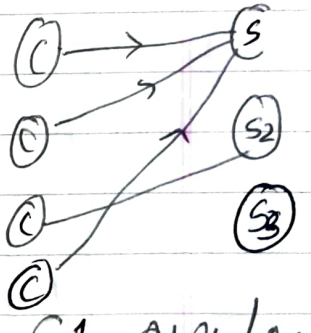
Case 1: Server overload due to lotta reqs.



Case 2: Vertical scaling: (power of server ↑ but still 1 old power, so overload

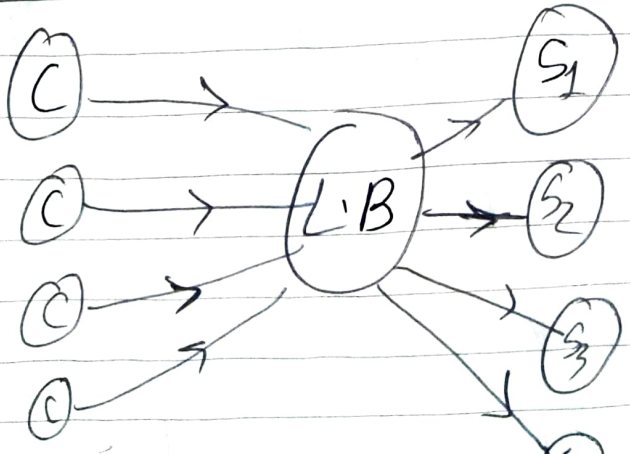


Case 3: Horizontal scaling
Multiple servers but still most client reqs to 1 server.



S1: overloaded

Case 4: Load Balancer



reqs evenly distri.
① No overload of a server
② Throughput ↑
③ Latency ↓ (no server clogged)

Other places where load balancin can be done:

1) b/w servers and dB
2) at D.N.S layer i.e. same service → multip servers so L.B. directs you to multip servers.

eg:- 'amazon' → single domain gets multip IP addresses (1 for .com, 1 for .in) for multip servers

amazon.com → Amazon D.N.S. in the US. → US server

amazon.in → Amazon D.N.S. in India → Indian server

eg2: Netflix US vs Netflix India

eg on opp page → we are accessin diff. IP addresses of google.com i.e. diff. servers and we get allotted the server related to our specific IP address (abbr. IPA) by the load balancer

NOTE Hardware L.B. → physical machine dedicated to load balancin

Software L.B. → not physical machines (eg:- a cloud services L.B. → cloud servers)
use: more power, customiza"

SOFTWARE L.B.

1) How does L.B. know which servers it can access?

(i) Sys. designers config. L.B and server to know abt each other

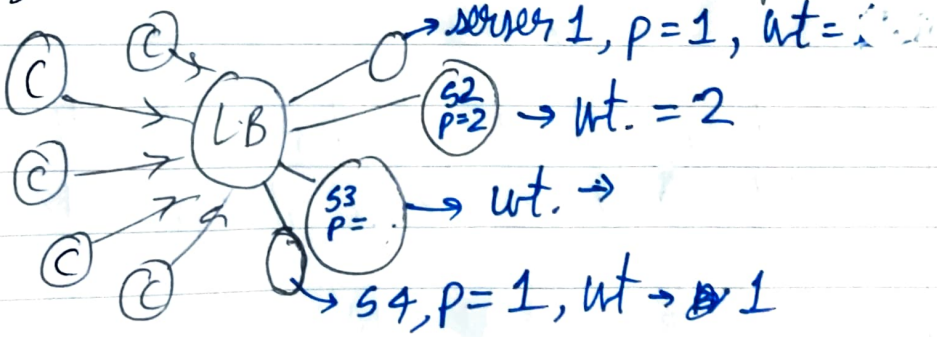
(ii) On addin / remov'n old server, it registers/ deregisters itself w.r.t. LB

2) Ways to select servers to direct load to

(1) Random redirec^n → No logic used. Just random selec^n of server from available servers
↓
by chance, one server may get overloaded

(2) Round Robin → L.B. goes thru all servers in order and allots a batch of reqs. to a server and so on until all servers hv equal load so for new batch of reqs., it'll again start frm 1st server
↓
server's power not optimally used.

Let size of server ∝ its power (p)



(3) Weighted Round Robin : L.B. goes as per round robin approach but if a weight is added on 2nd server, L.B. will send it a couple more reqs.
∴ Power of each server ⇒ optimized

Request Batch	Round Robin (RR)	Wtd. R-R.
1	S1	S1
2	S2	S2 } avg wt = 2
3	S3	S2 }
4	S4	S3
5	S1	S3 } wt = 3
6	S2	S3 }
7	S3	S4

(3) Performance/load Based : L.B. will do performance/health checks on servers

- how much traffic a server is handlin at any given time
- how long a server is takin to respond to traffic
- how many resources is the server usin
- etc.

Based on these checks, it allotes reqs (more reqs. low high performance servers

(4) IP based server selen : On gettin requests from clients L.B. hashes the IP address of the client, and dependin on this hashed value, the client's req. gets directed to a specific server

Use :- Caching

eg :- You're cachin results of certain reqs

in your servers, it's helpful to redirect the client to a server in which the response to ... request has already been cached

eg:- i) Client 1 searches for "cheese pizza"
~~ii) ... knows that the response to this request is ... in server 1 ...~~
ii) L.B. hashes down IPA of client to say "1" → basically a UID
iii) Now, it sees that in server S3 a result for a prev req. made with UID = 1 is cached and ready
iv) L.B. redirects client 1's req. to server S3
v) Client 1 gets his/her response super fast.

(5) Path-based → L.B. selects server or set of servers based on path of the reqs.

eg:- Algo Expert has servers S1, S2, S3 .... S25

All reqs related to payments → S1, S2 .... S9

All reqs related to running code → S10 - S18

All reqs related to watching vids → S19 - S21, S22, S24, S25

All reqs related to "about" page → S23

Date:
M T W T F S S

Use:

1) Deployment of big change to any service (eg:- running code service) affects only the servers allotted to this service.

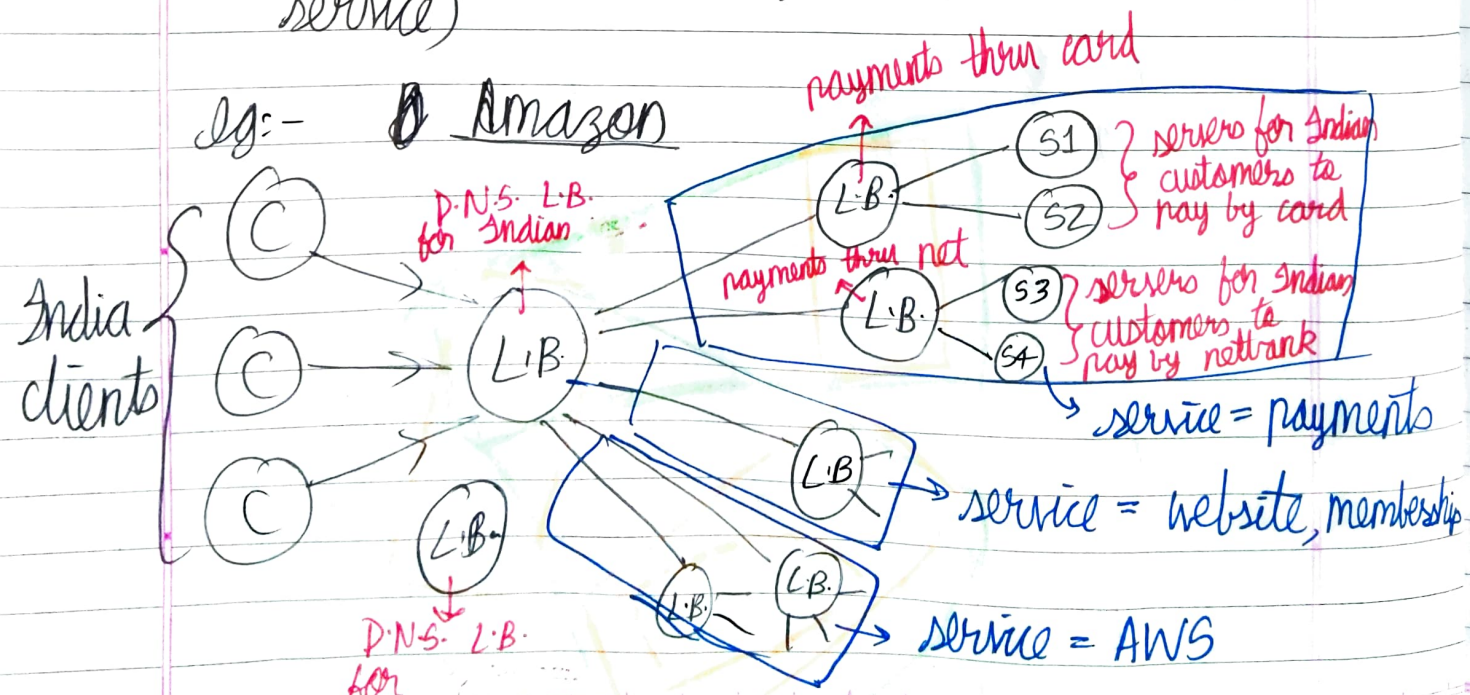
2) If one set of servers for a service start failing atleast other services are still up and running. (unaffected)

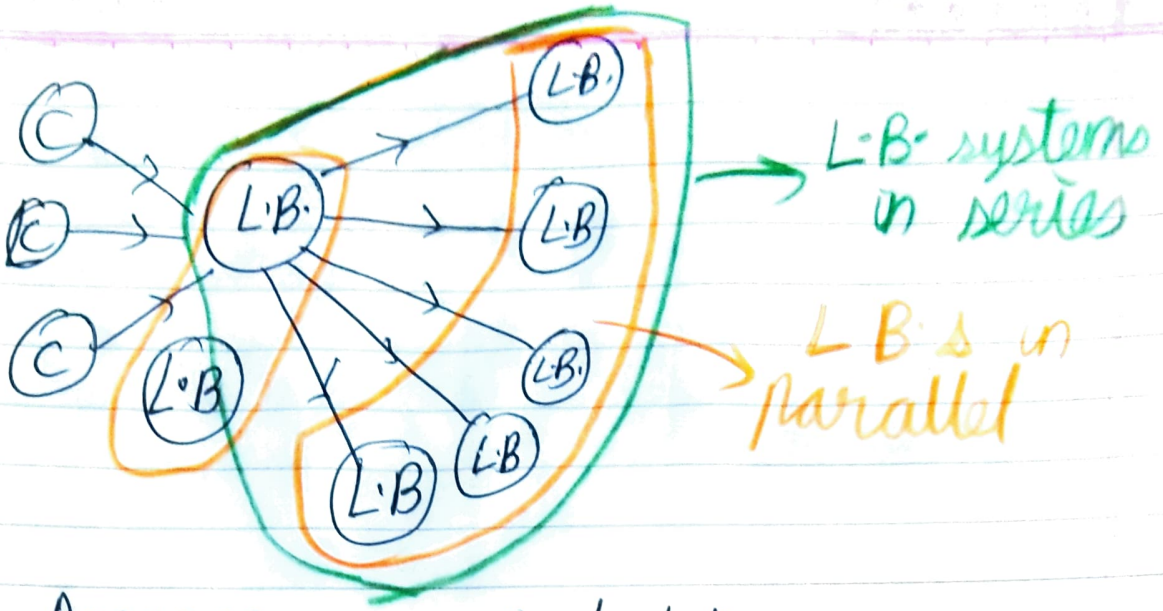
MULTIPLE LOAD BALANCERS

1) In series → to keep dividing a batch of reqs into smaller batches based on some criteria (eg:- path) on the req.

2) In parallel → To (1) just prevent 1 L.B. itself from getting overloaded or (2) Redirect to diff. servers based on req. criteria or server criteria (eg:- performance, diff kinda service)

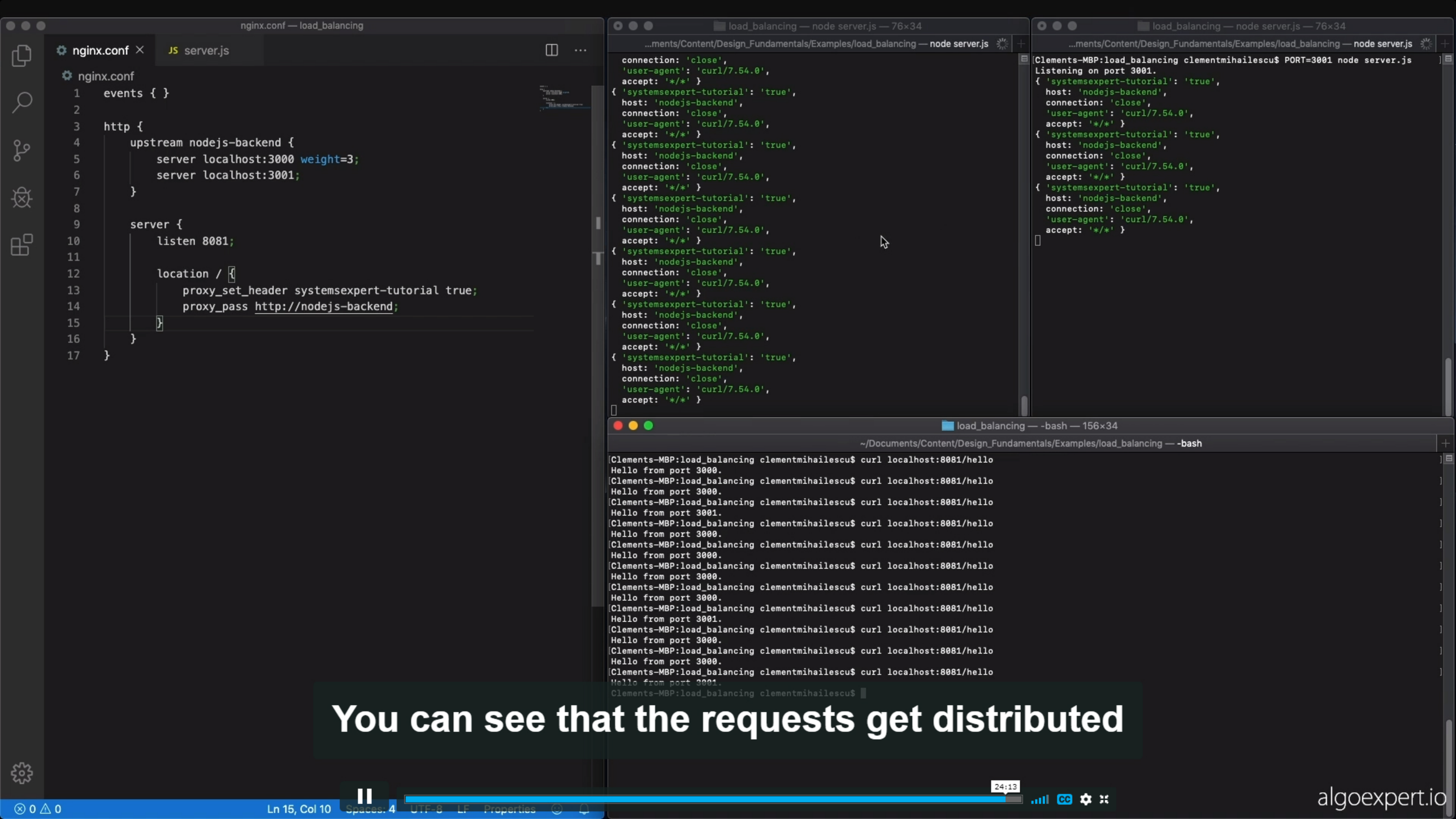
eg:- Amazon





Amazon eg:- Just L·B·s

code on opp. page → shows wted R·R· selec^{n} meth.


nginx.conf — load_balancing
nginx.conf
server.js
nginx.conf
events { }
http {
upstream nodejs-backend {
server localhost:3000 weight=3;
server localhost:3001;
}
server {
listen 8081;
location / {
proxy_set_header systemsexpert-tutorial true;
proxy_pass http://nodejs-backend;
}
}
}
load_balancing — node server.js — 76×34
...ments/Content/Design_Fundamentals/Examples/load_balancing — node server.js
load_balancing — node server.js — 76×34
...ments/Content/Design_Fundamentals/Examples/load_balancing — node server.js
Clements-MBP:load_balancing clementmihailescu$ PORT=3001 node server.js
Listening on port 3001.
load_balancing — -bash — 156×34
~/Documents/Content/Design_Fundamentals/Examples/load_balancing — -bash
Clements-MBP:load_balancing clementmihailescu$ curl localhost:8081/hello
Hello from port 3000.
Hello from port 3001.
You can see that the requests get distributed
24:13
Ln 15, Col 10
algoexpert.io