husseinnasser.com

Introduction to NGINX

NGINX Web Server / Reverse Proxy

Agenda

- What is NGINX?
- Current & Desired Architecture
- Layer 4 and Layer 7 Proxying in NGINX
- TLS Termination vs TLS Passthrough
- Timeouts in NGINX
- Example
  - NGINX as a Web Server, Layer 7 and Layer 4 Proxy
  - Enable HTTPS, TLS 1.3 & HTTP/2 on NGINX
- Summary

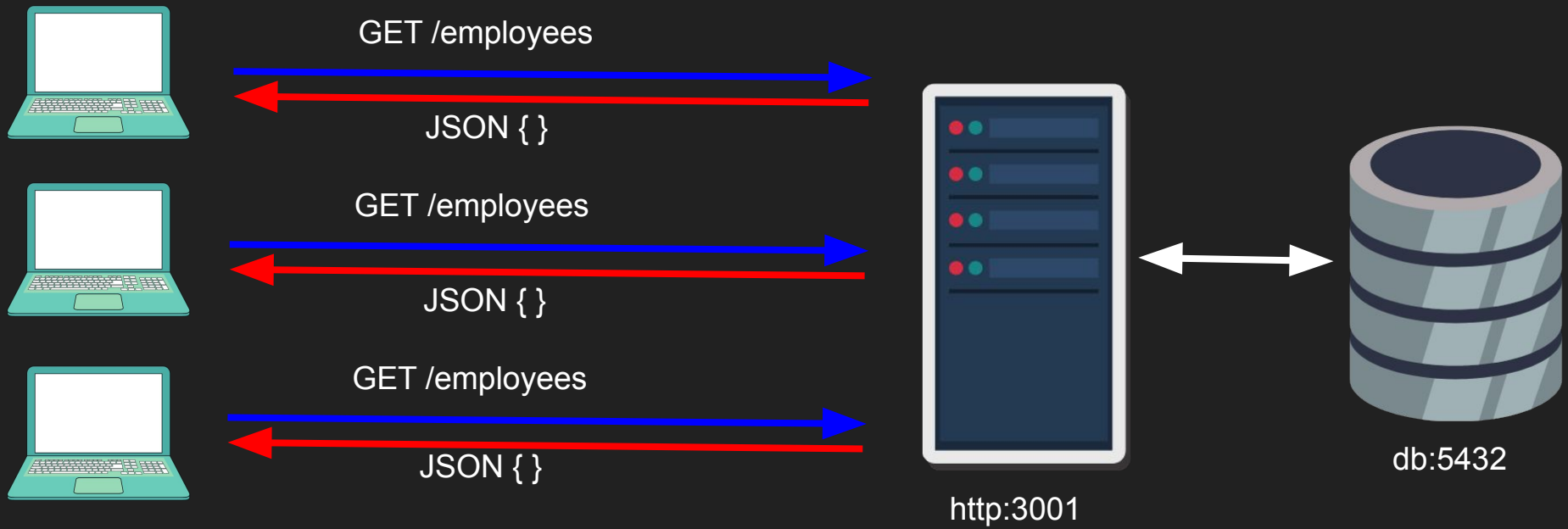
What is NGINX?

What is NGINX?

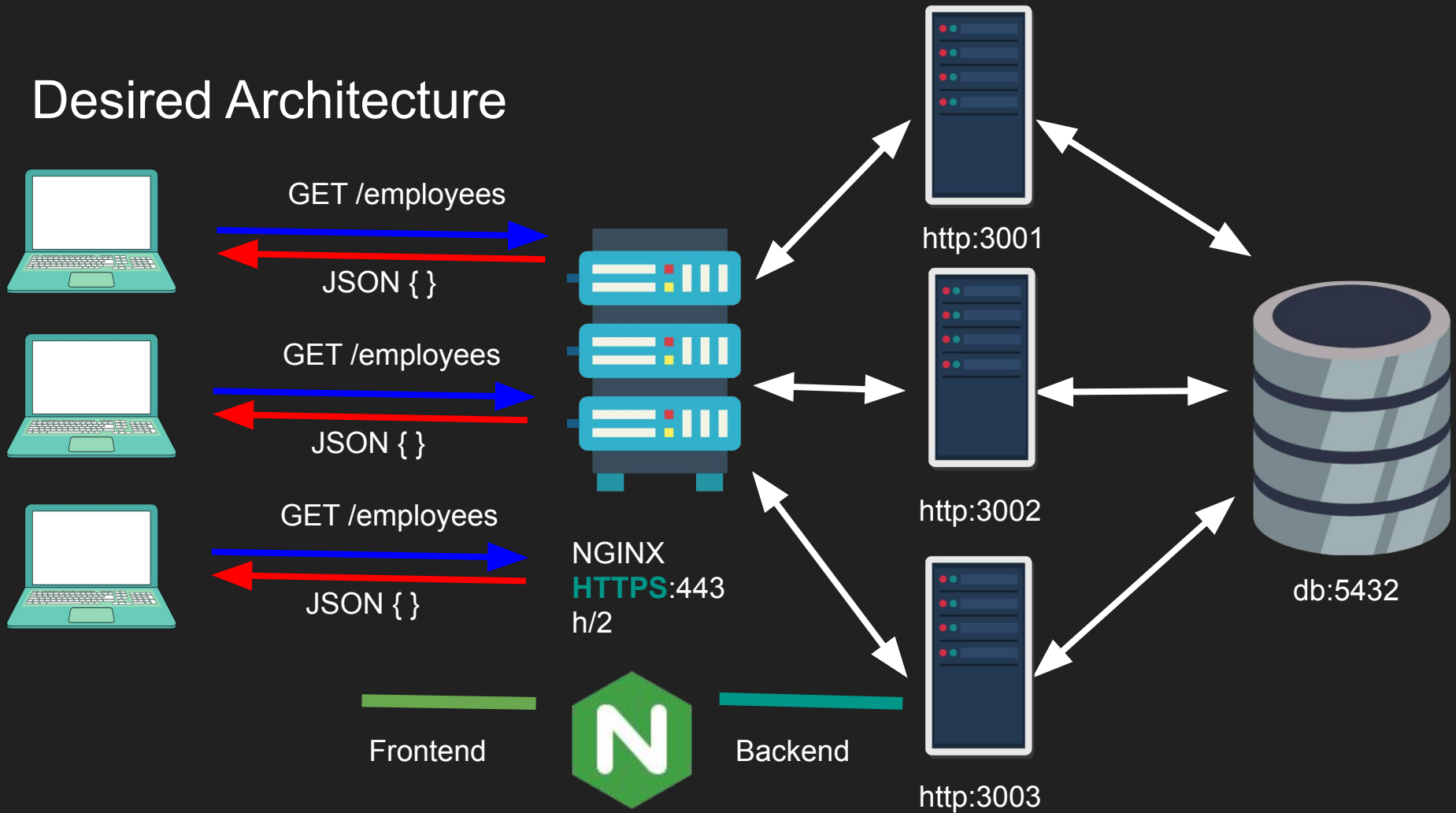
- Web Server
  - Serves web content
- Reverse Proxy
  - Load Balancing
  - Backend Routing
  - Caching
  - API Gateway

Current vs Desired Architecture

Current Architecture



Desired Architecture



NGINX Layer 4 vs Layer 7 proxying

Layer 4 and Layer 7

- Layer 4/7 refers to OSI model layers
- In Layer 4 we see TCP/IP stack only nothing about the app, we have access to
  - Source IP, Source Port
  - Destination IP, Destination Port
  - Simple packet inspection (SYN/TLS hello)
- In Layer 7 we see the application, HTTP/ gRPC etc..
  - We have access to more context
  - I know where the client is going, which page they are visiting
  - Require decryption

Layer 4 and Layer 7 proxying in NGINX

- NGINX can operate in Layer 7 (e.g. http) or Layer 4 (tcp)
- Layer 4 proxying is useful when NGINX doesn’t understand the protocol (MySQL database protocol)
- Layer 7 proxying is useful when NGINX want to share backend connections and cache results
- Using **stream** context it becomes a layer 4 proxy
- Using **http** context it becomes a layer 7 proxy

TLS Termination vs TLS Passthrough

TLS

- TLS stands for Transport Layer Security
- It is a way to establish end-to-end encryption between one another
- Symmetric encryption is used for communication (client/server has the same key)
- Asymmetric encryption is used initially to exchange the symmetric key (diffie hellman)
- Server (sometimes even the client) need to authenticate themselves by supplying a certificate signed by a certificate authority

TLS Termination

- NGINX has TLS (e.g. HTTPS) backend is not (HTTP)
- NGINX terminates TLS and decrypts and send unencrypted.
- NGINX is TLS and backend is also TLS (HTTPS)
- NGINX terminates TLS, decrypted, optionally rewrite and then re-encrypt the content to the backend.
- NGINX NGINX can look at the L7 data, re-write headers, cache **but** needs to share the backend certificate or at least has its own

TLS Passthrough

- Backend is TLS
- NGINX proxies/streams the packets directly to the backend.
- The TLS handshake is forwarded all the way to the backend.
- Just like a tunnel
- No caching, L4 check only, **but** more secure, NGINX doesn’t need the backend certificate.

NGINX Timeouts

NGINX Timeouts

Frontend Timeouts

- client_header_timeout
- client_body_timeout
- send_timeout
- keepalive_timeout
- lingering_timeout
- resolver_timeout

Backend Timeouts

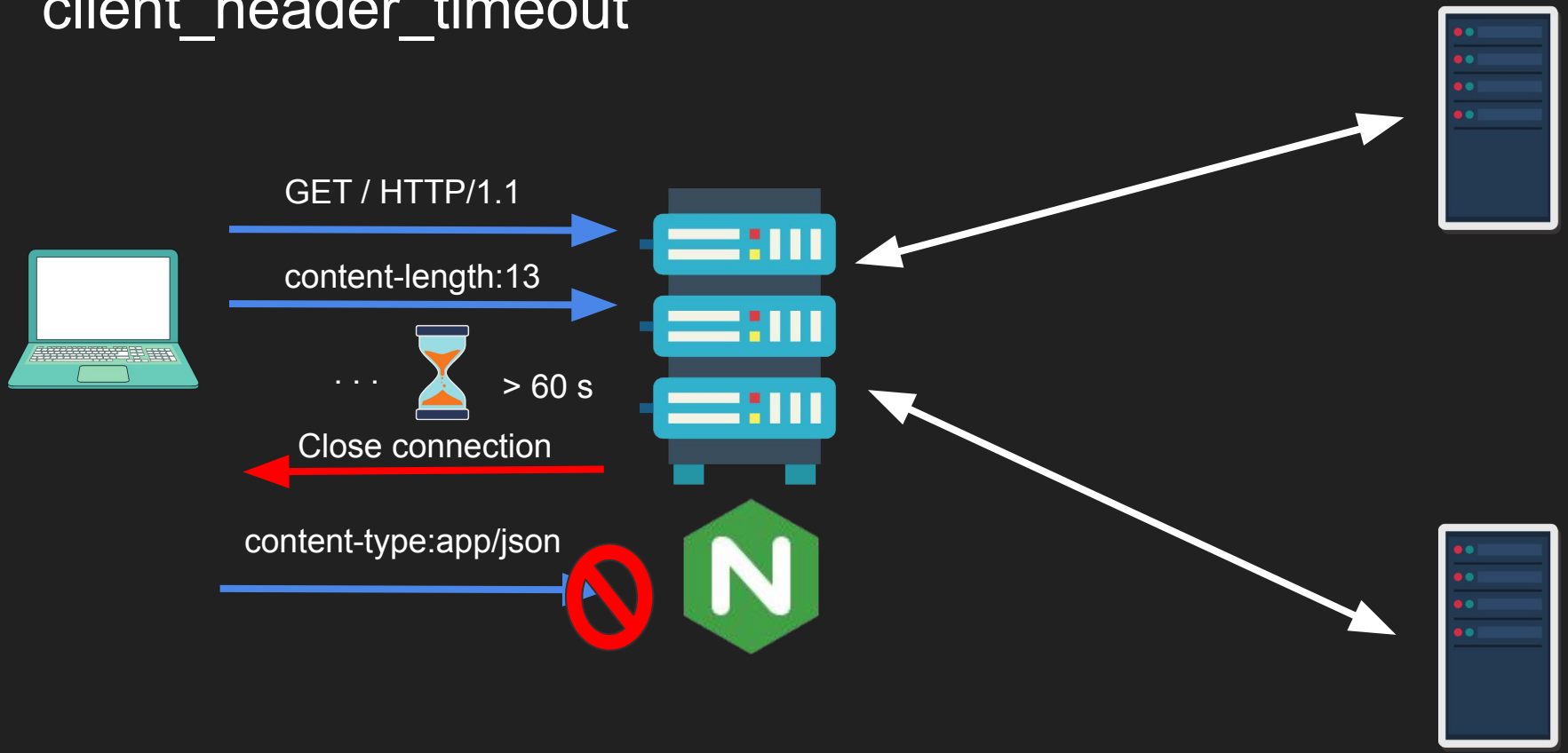
- proxy_connect_timeout
- proxy_send_timeout
- proxy_read_timeout
- keepalive_timeout
- proxy_next_upstream_timeout

NGINX Frontend Timeouts

Frontend Timeouts

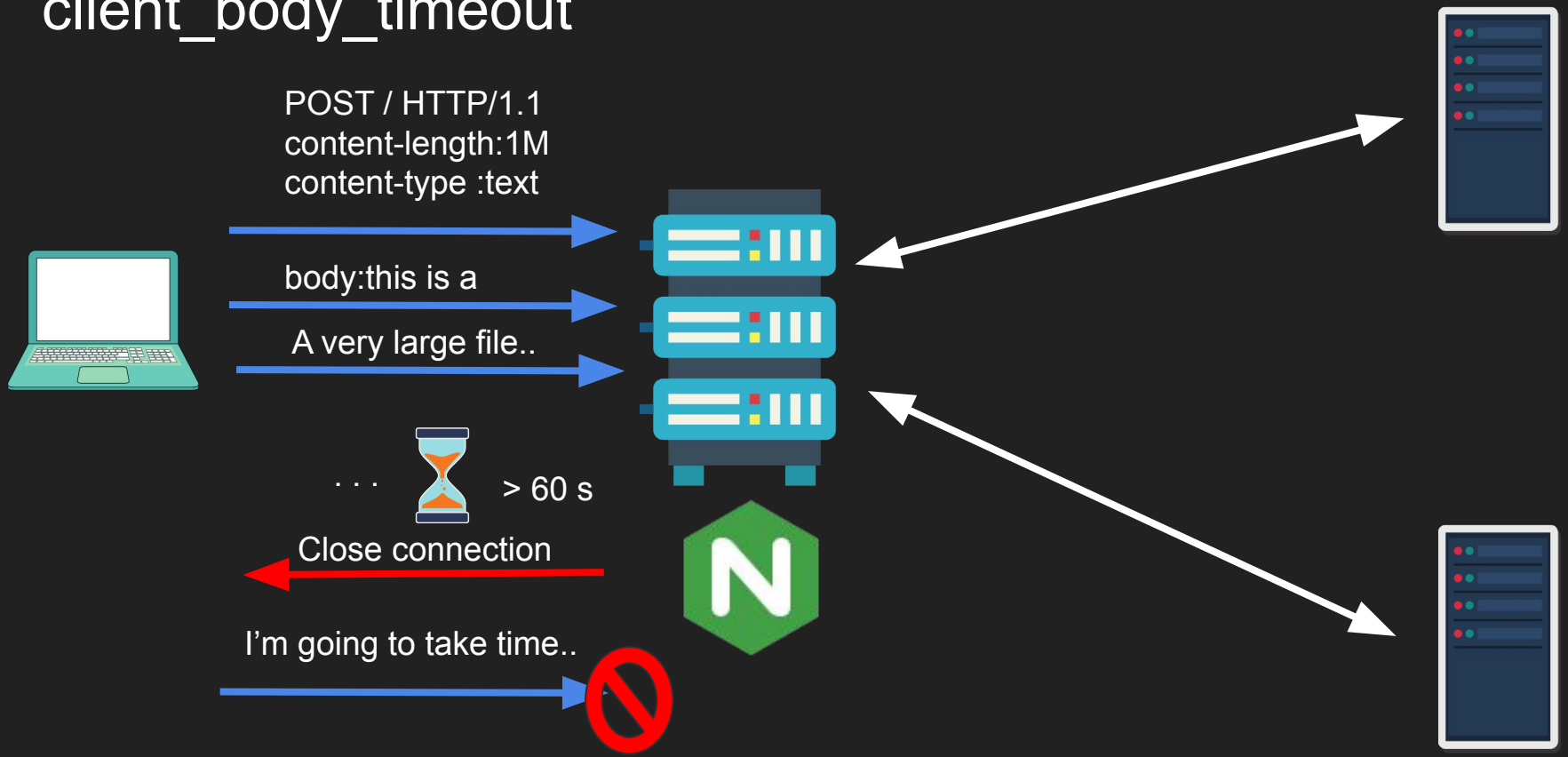
- client_header_timeout
- client_body_timeout
- send_timeout
- keepalive_timeout
- lingering_timeout
- resolver_timeout

client_header_timeout



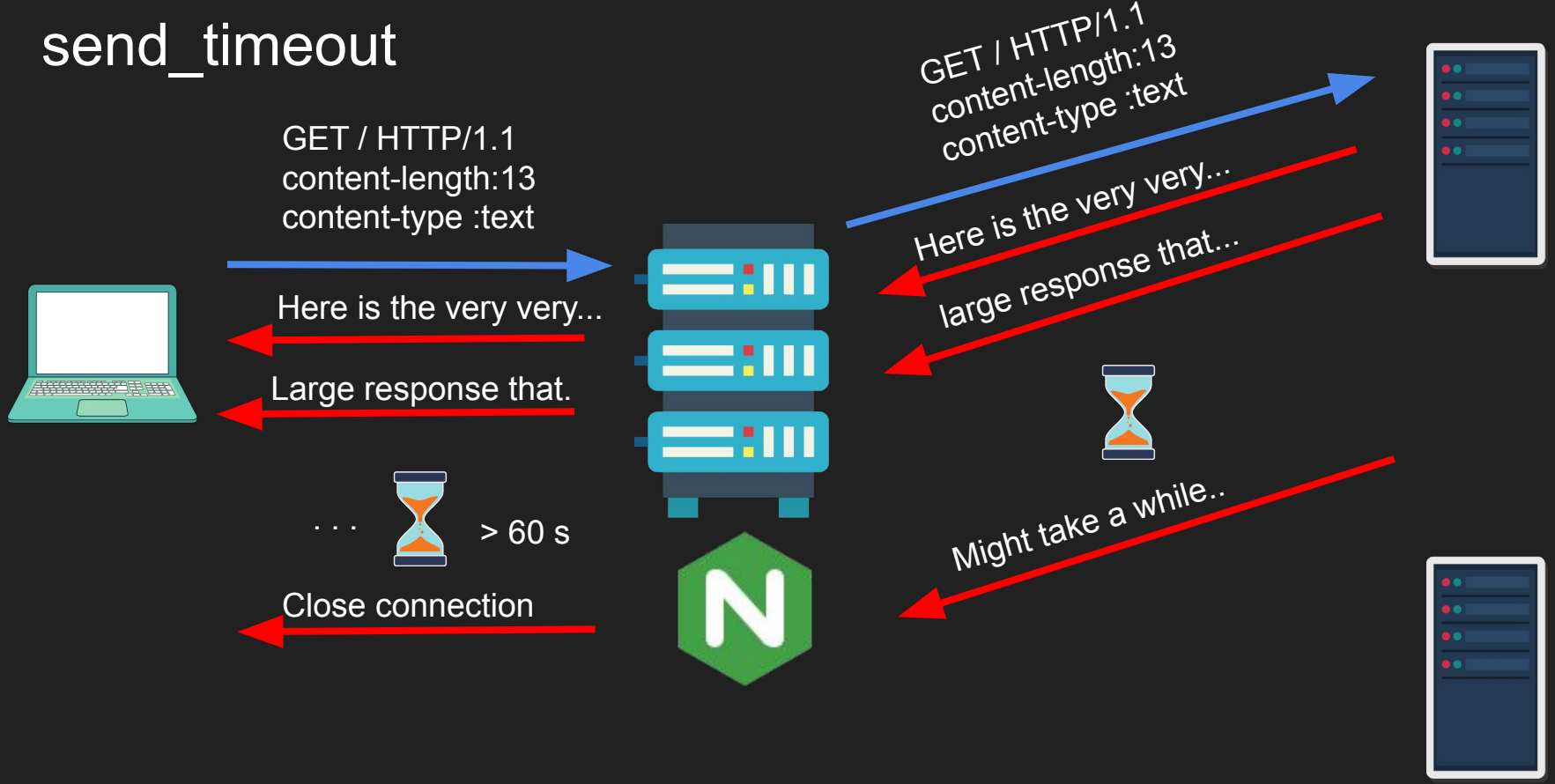
Defines a timeout for reading client request header. If a client does not transmit the entire header within this time, the request is terminated with the 408 (Request Time-out) error. Default 60s

client_body_timeout



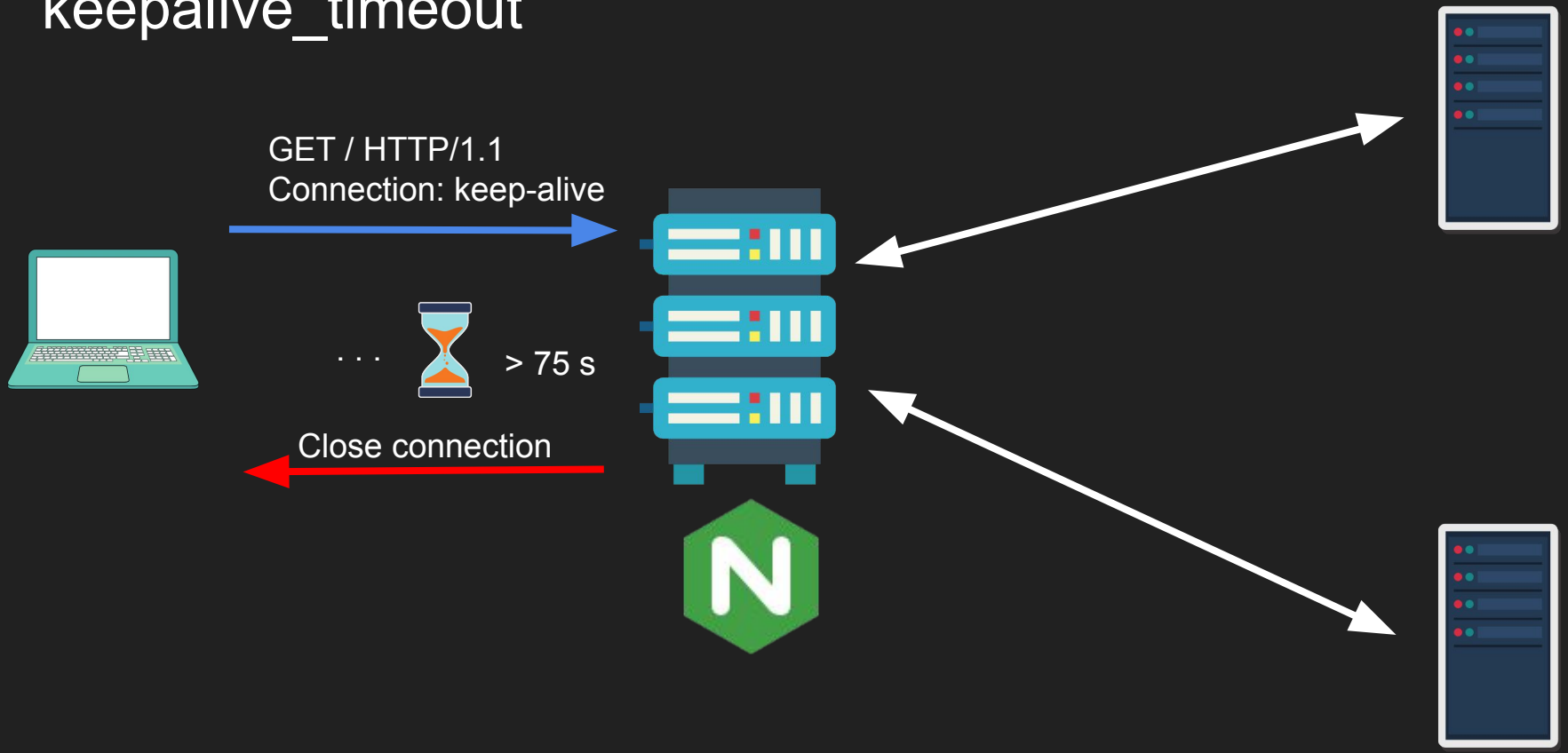
Defines a timeout for reading client request body. The timeout is set only for a period between two successive read operations, not for the transmission of the whole request body. If a client does not transmit anything within this time, the request is terminated with the 408 (Request Time-out) error. Default 60s

send_timeout



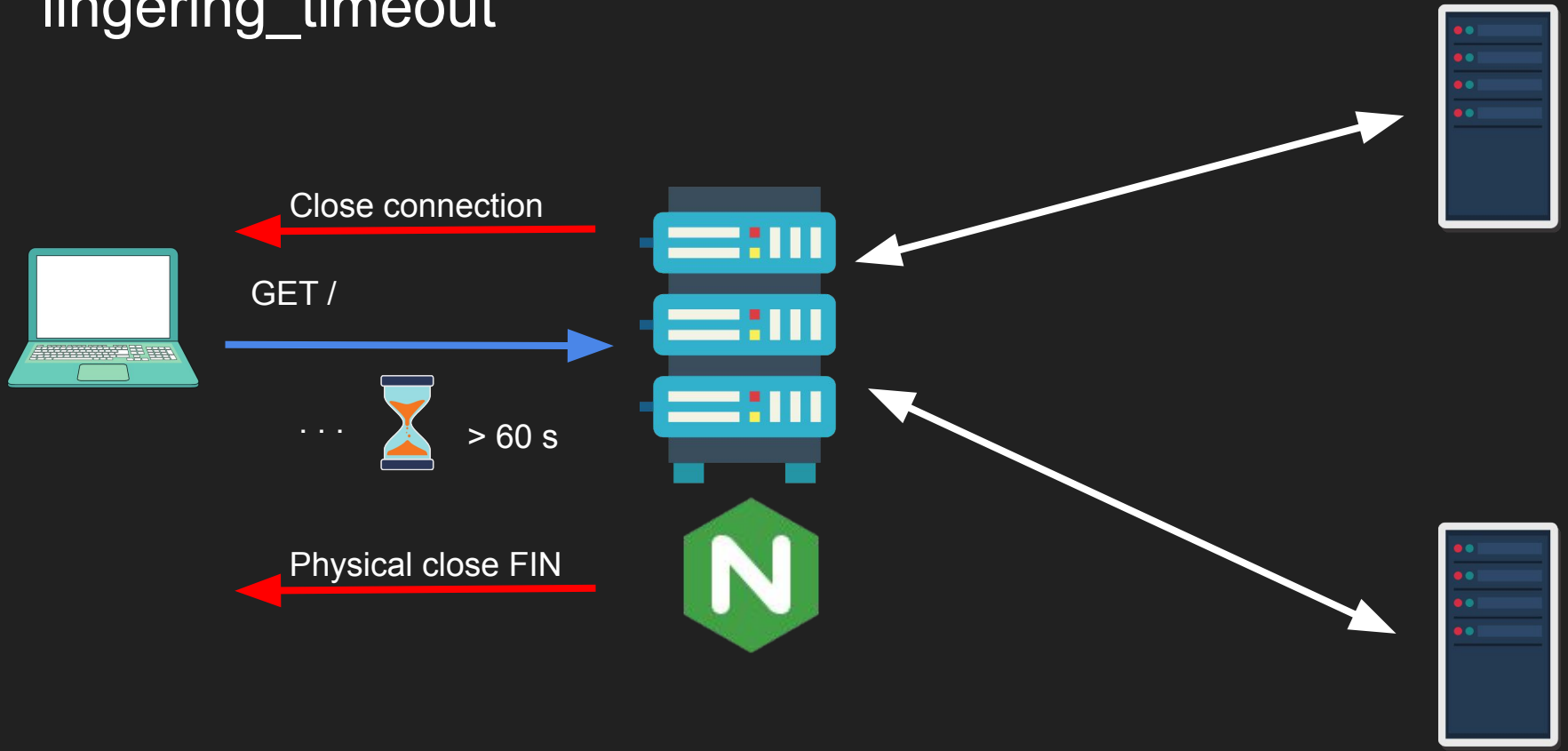
Sets a timeout for transmitting a response to the client. The timeout is set only between two successive write operations, not for the transmission of the whole response. If the client does not receive anything within this time, the connection is closed. (Default 60s)

keepalive_timeout



The first parameter sets a timeout during which a keep-alive client connection will stay open on the server side. The zero value disables keep-alive client connections. The optional second parameter sets a value in the “Keep-Alive: timeout=time” response header field. Two parameters may differ. (default 75 seconds)

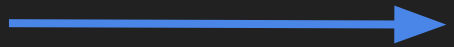
lingering_timeout

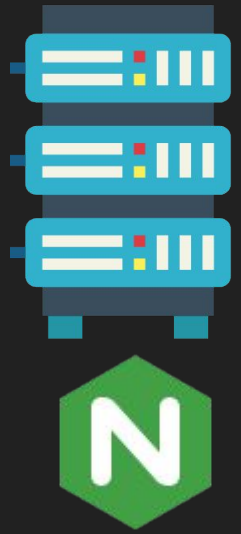


When lingering_close is in effect, this directive specifies the maximum waiting time for more client data to arrive. If data are not received during this time, the connection is closed. Otherwise, the data are read and ignored, and nginx starts waiting for more data again. The “wait-read-ignore” cycle is repeated, but no longer than specified by the lingering_time directive.

resolver_timeout

GET / HTTP/1.1
Connection: keep-alive

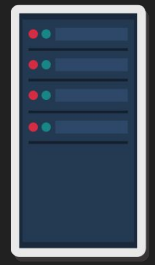




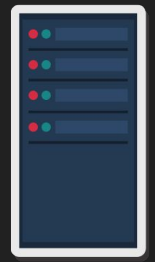
What is the IP of
server1.test.com?

. . .

> 30 s



server1.test.com



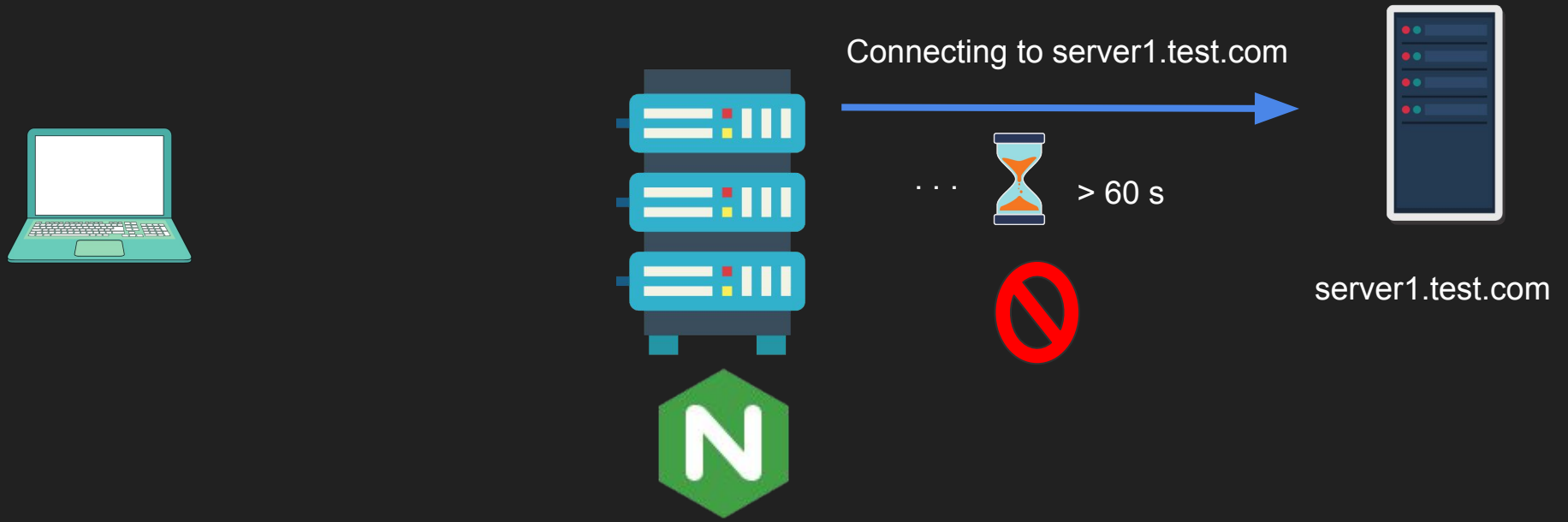
Sets a timeout for name resolution, 30 seconds

NGINX Backend Timeouts

Backend Timeouts

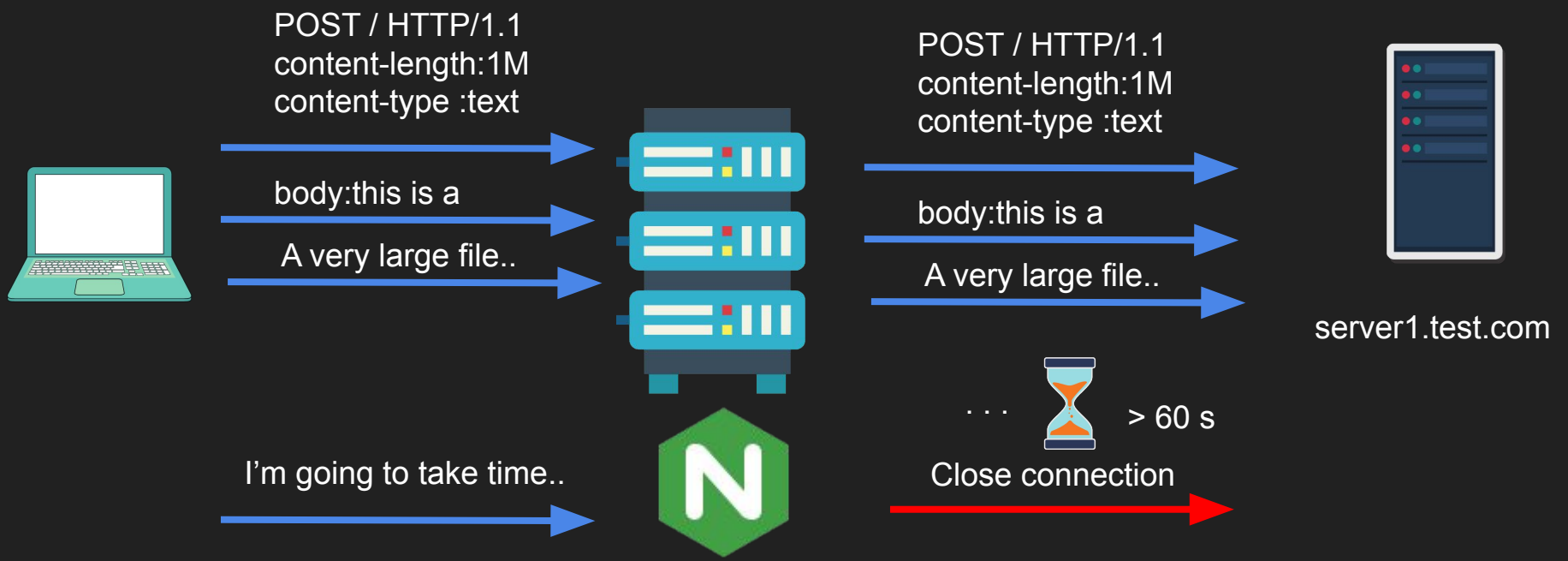
- proxy_connect_timeout
- proxy_send_timeout
- proxy_read_timeout
- keepalive_timeout
- proxy_next_upstream_timeout

proxy_connect_timeout



Defines a timeout for establishing a connection with a proxied server. It should be noted that this timeout cannot usually exceed 75 seconds.

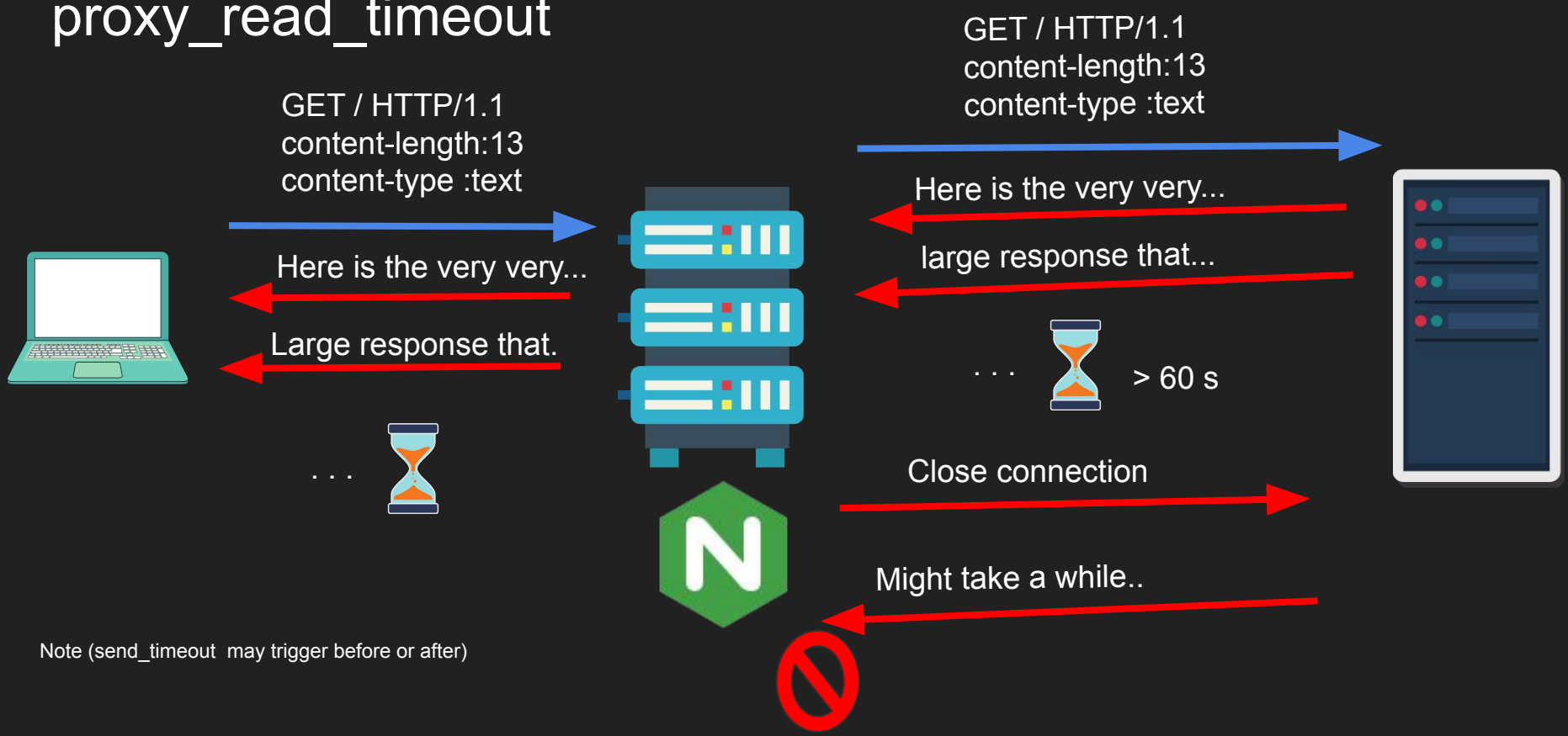
proxy_send_timeout



Note (body_timeout may trigger before) so we don't even bother the backend..

Sets a timeout for transmitting a request to the proxied server. The timeout is set only between two successive write operations, not for the transmission of the whole request. If the proxied server does not receive anything within this time, the connection is closed.

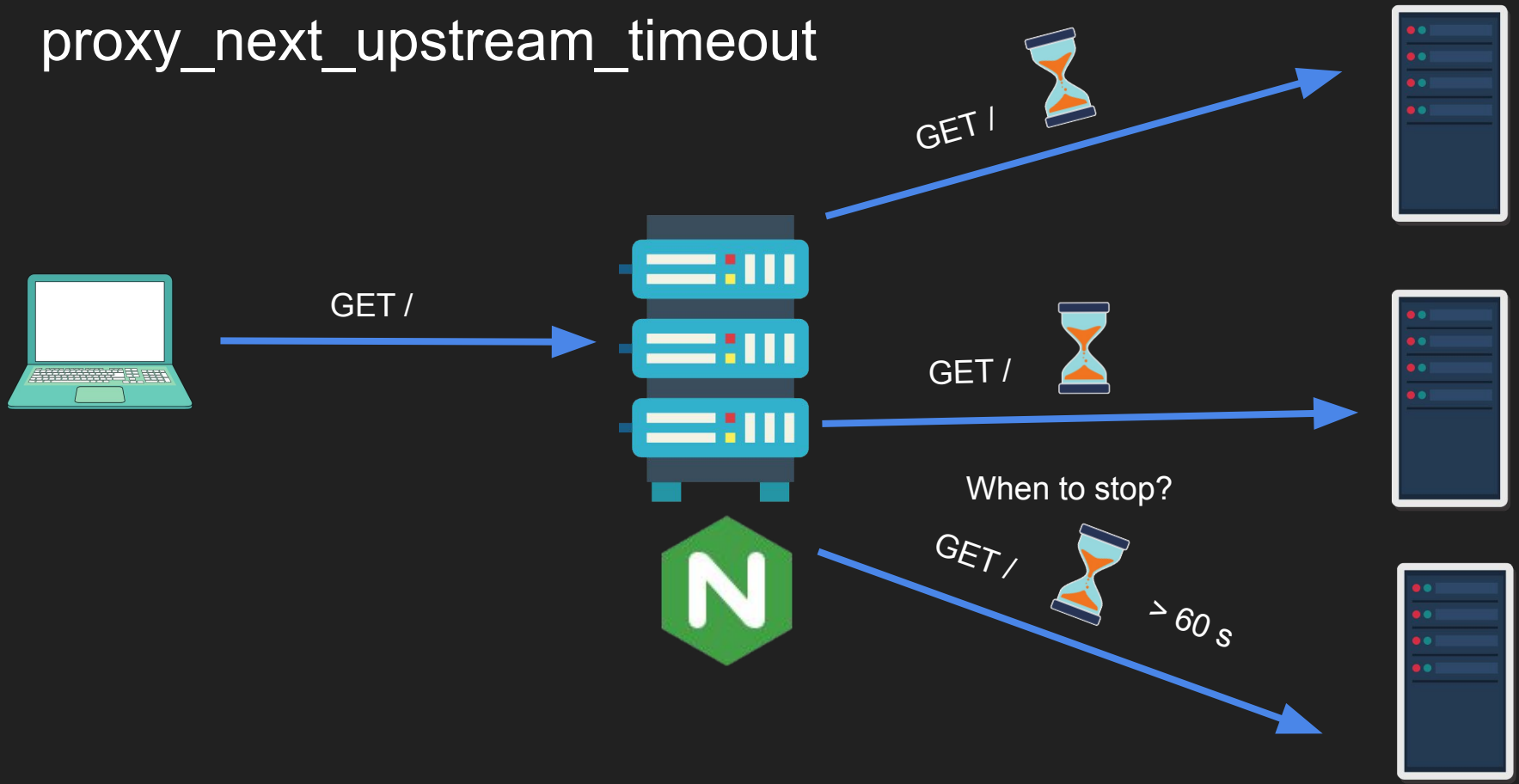
proxy_read_timeout



Note (send_timeout may trigger before or after)

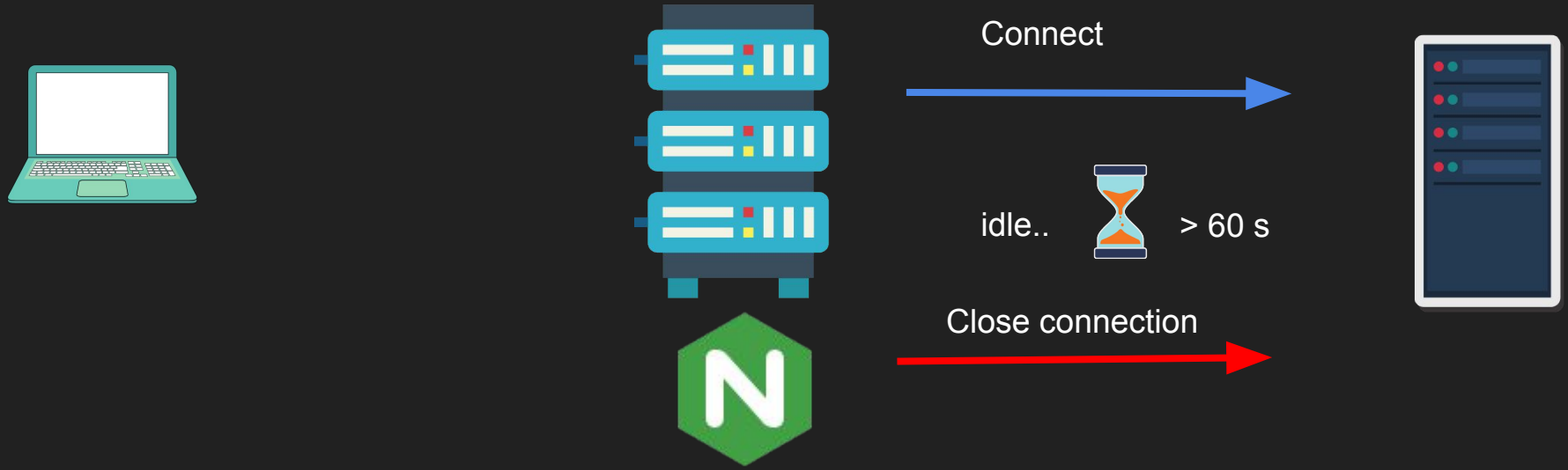
Defines a timeout for reading a response from the proxied server. The timeout is set only between two successive read operations, not for the transmission of the whole response. If the proxied server does not transmit anything within this time, the connection is closed.

proxy_next_upstream_timeout



Limits the time during which a request can be passed to the next server. The 0 value turns off this limitation. Default 0

Keepalive_timeout (Backend)



Sets a timeout during which an idle keepalive connection to an upstream server will stay open.

Example

- Install NGINX (mac)
- NGINX as a Web Server
- NGINX as a Layer 7 Proxy
  - Proxy to 4 backend NodeJS services (docker)
  - Split load to multiple backends (app1/app2)
  - Block certain requests (/admin)
- NGINX as a Layer 4 Proxy
- Enable HTTPS on NGINX (lets encrypt)
- Enable TLS 1.3 on NGINX
- Enable HTTP/2 on NGINX

NGINX
Internal Architecture

Master process

NGINX spins up a worker process per CPU core by default

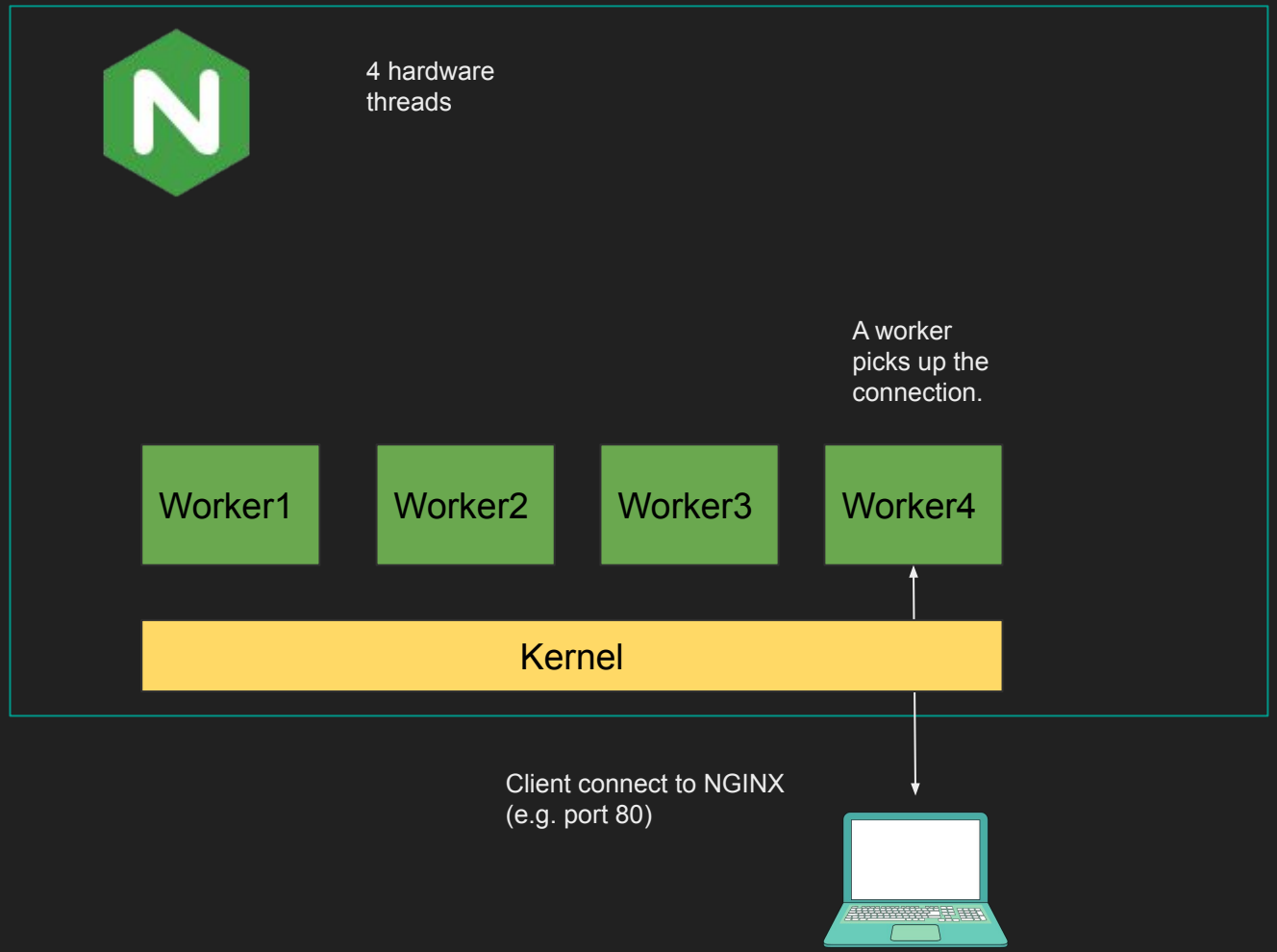
Worker1

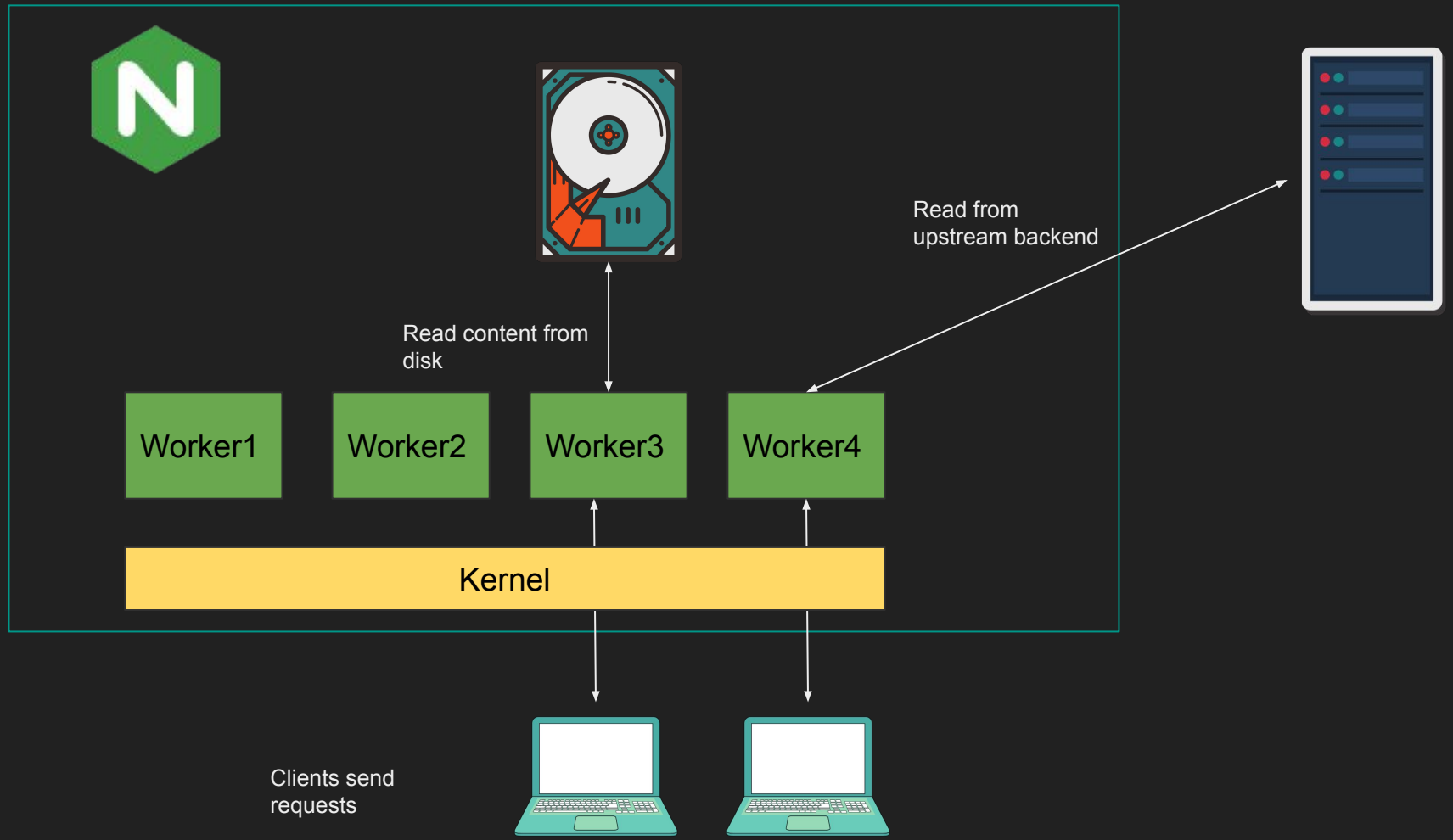
Worker2

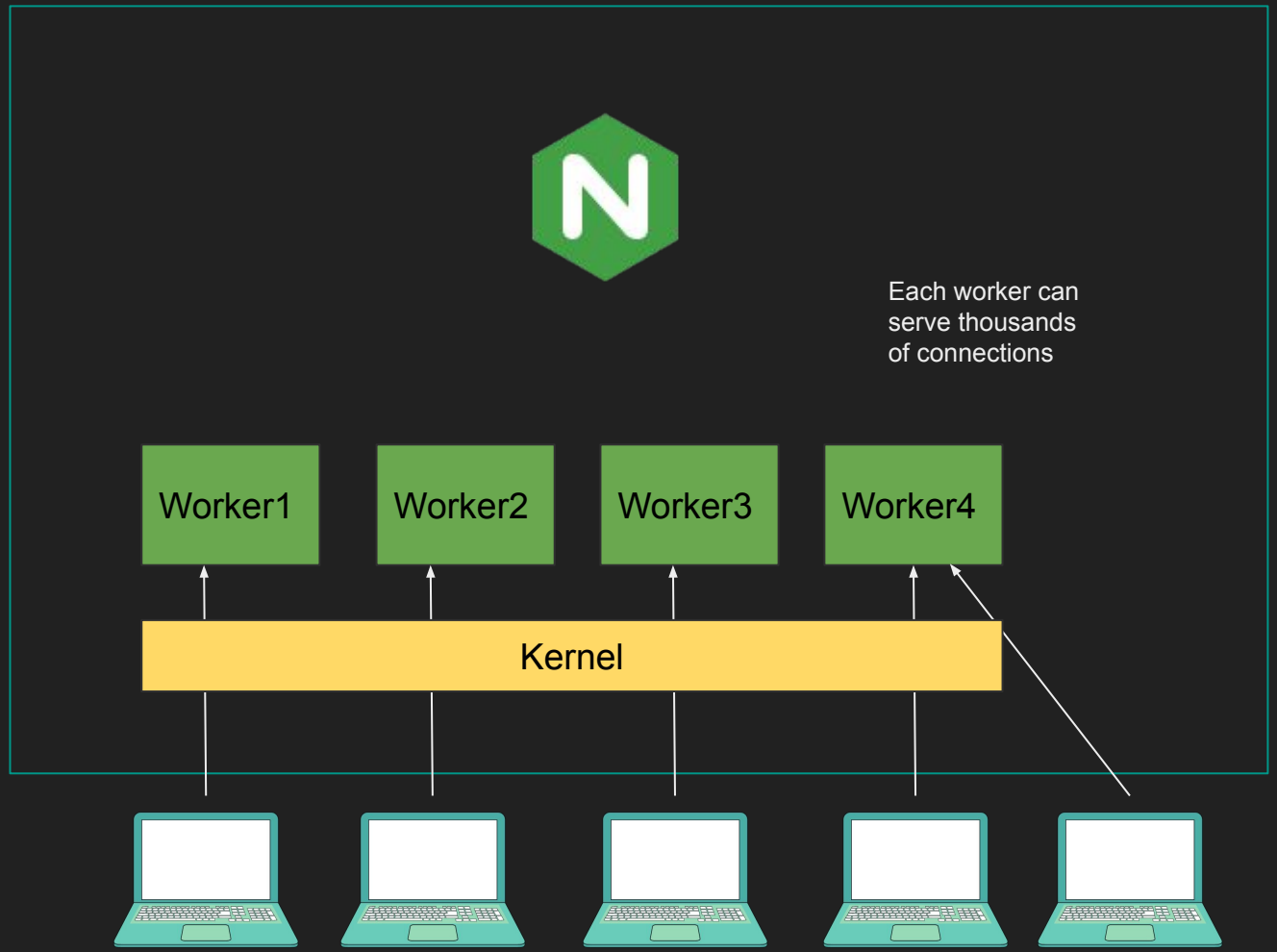
Worker3

Workern

Child processes


4 hardware threads
A worker picks up the connection.
Worker1
Worker2
Worker3
Worker4
Kernel
Client connect to NGINX (e.g. port 80)


Read from
upstream backend
Read content from
disk
Worker1
Worker2
Worker3
Worker4
Kernel
Clients send
requests


Each worker can
serve thousands
of connections
Worker1
Worker2
Worker3
Worker4
Kernel

Summary

- What is NGINX?
- Current & Desired Architecture
- Layer 4 and Layer 7 Proxying in NGINX
- TLS Termination vs TLS Passthrough
- Timeouts in NGINX
- Example
  - NGINX as a Web Server, Layer 7 Proxy & Layer 4 Proxy
  - Enable HTTPS, TLS 1.3 & HTTP/2 on NGINX
- Summary