NGINX Web Server

SOFTWARE ARCHITECTURE

HOOMAN BEHNEJAD FARD

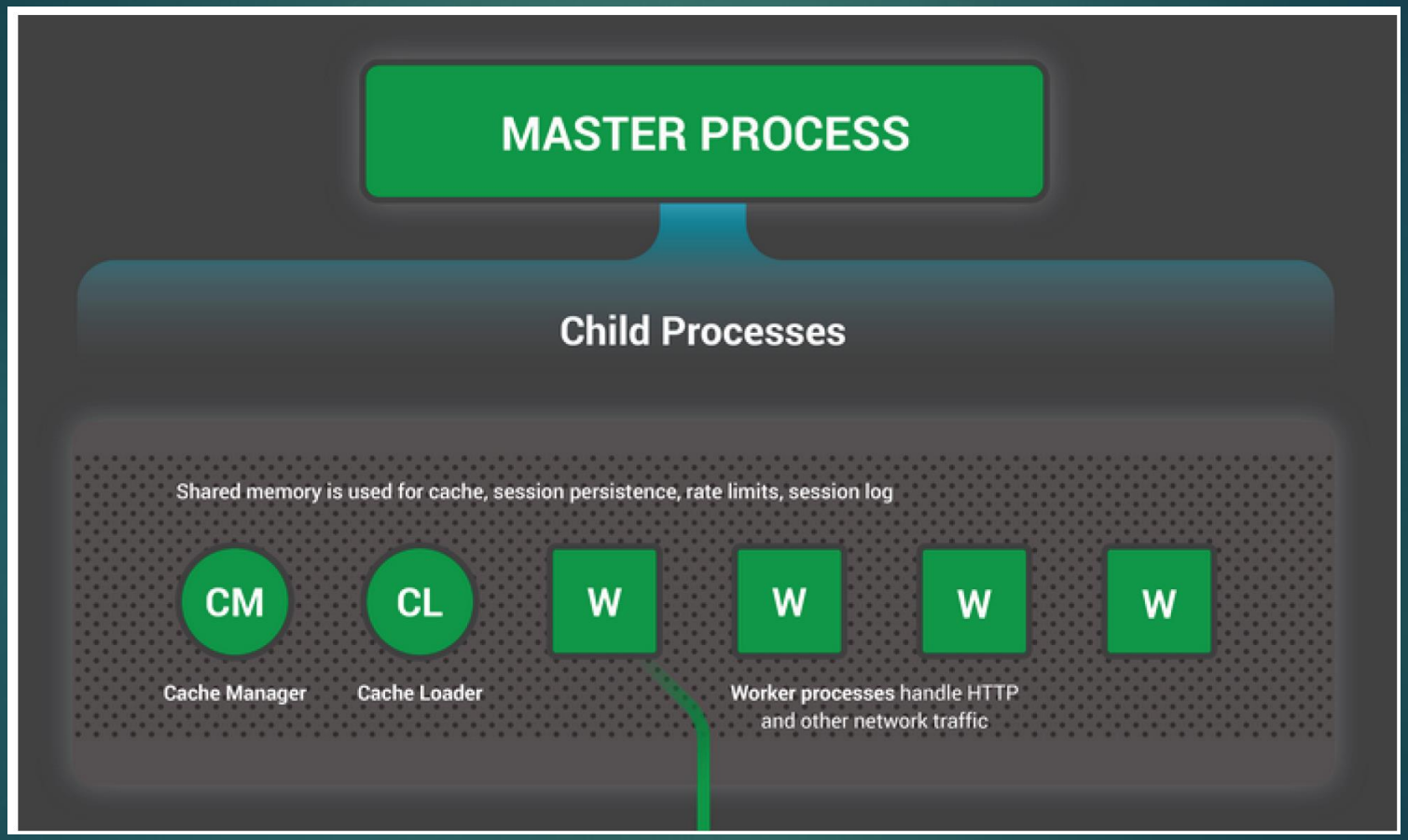
Overview

- nginx were written in 2002.
- In 2004 it was released to the public under the two-clause BSD license.
- was meant to be a specialized tool to achieve more performance.
- Advanced event-based mechanisms.
- modular, event-driven, asynchronous, non-blocking.
- uses event notification mechanisms

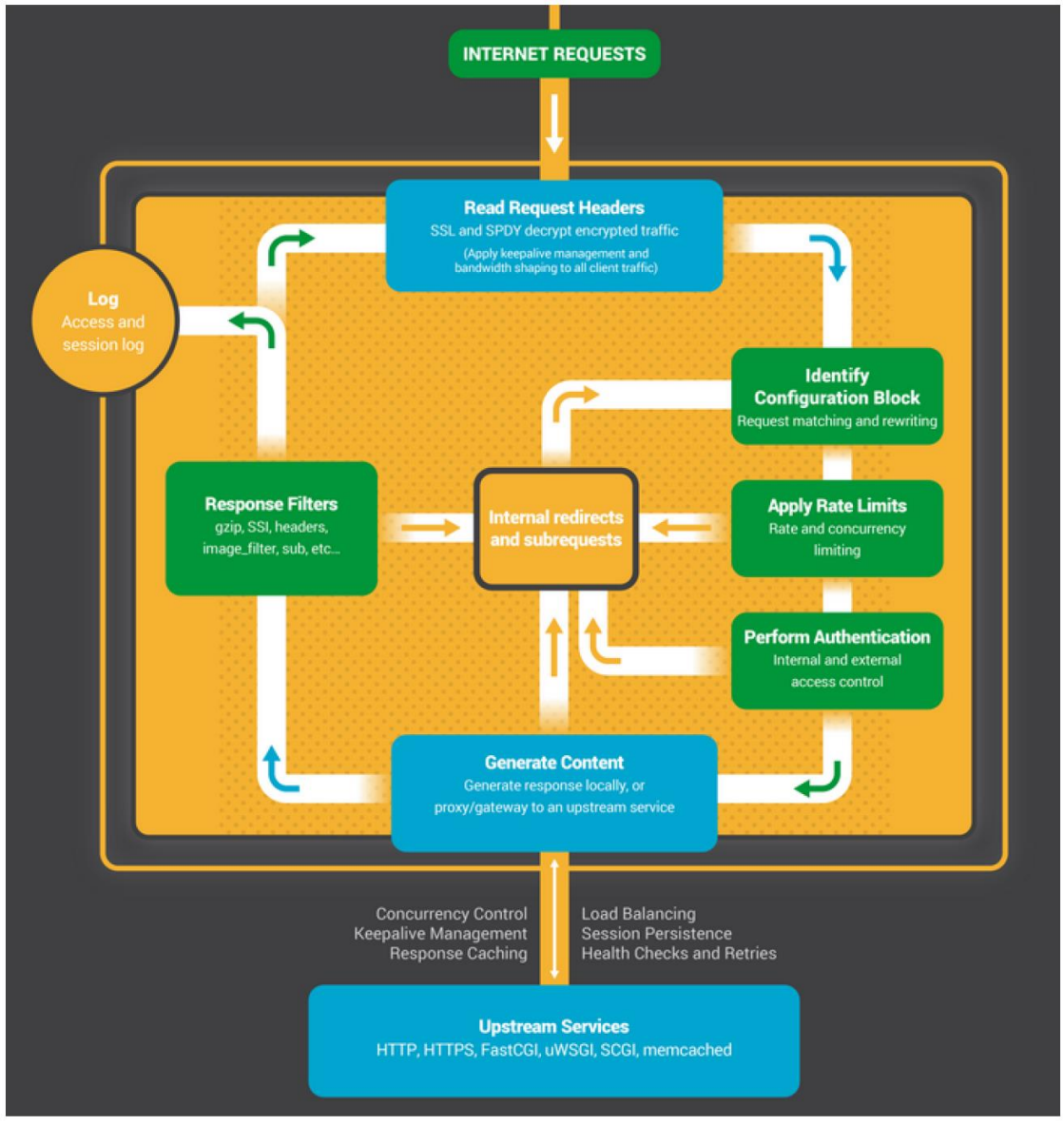
NGINX Focuses

- High performance
- High concurrency
- Low memory usage

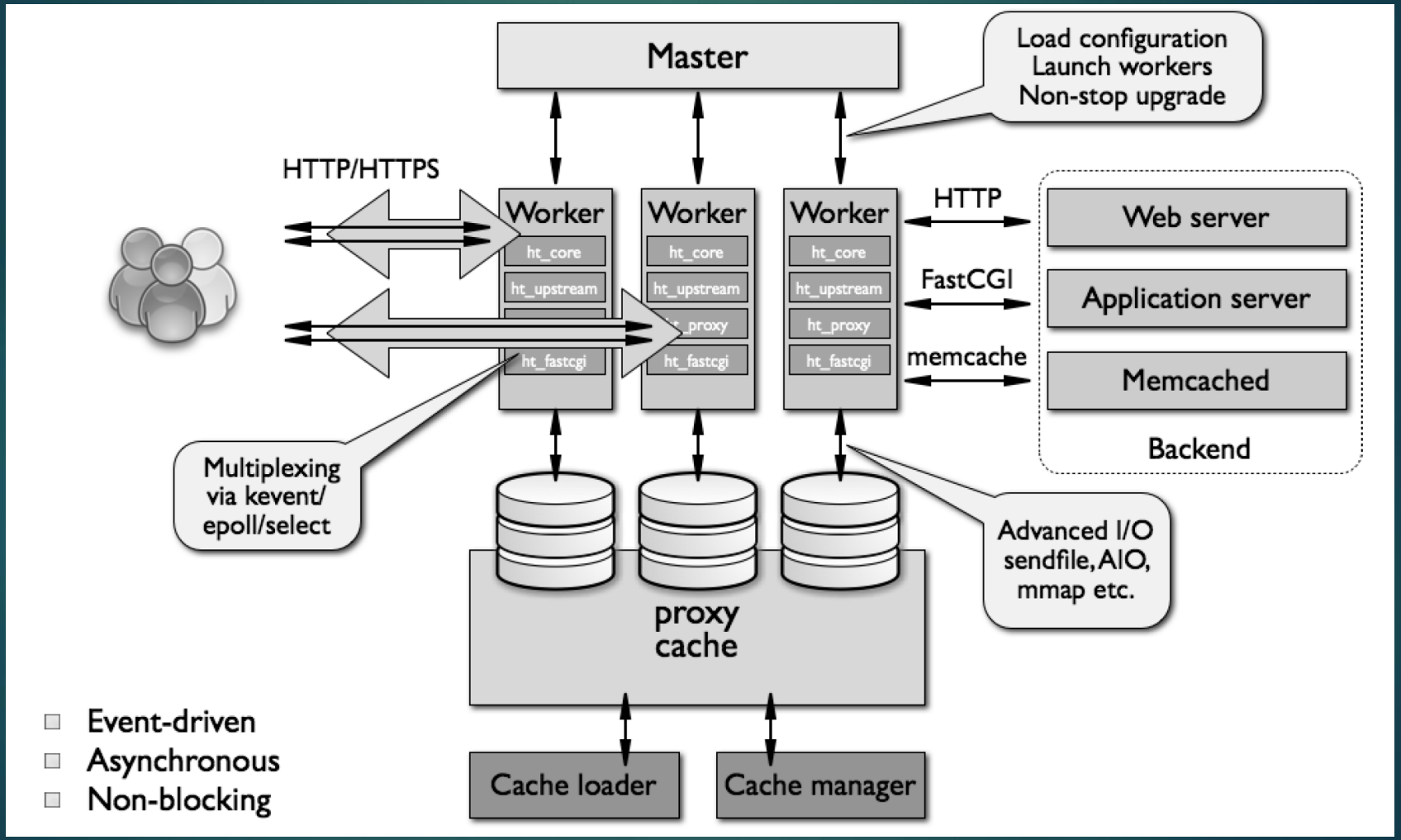
nginx's process model



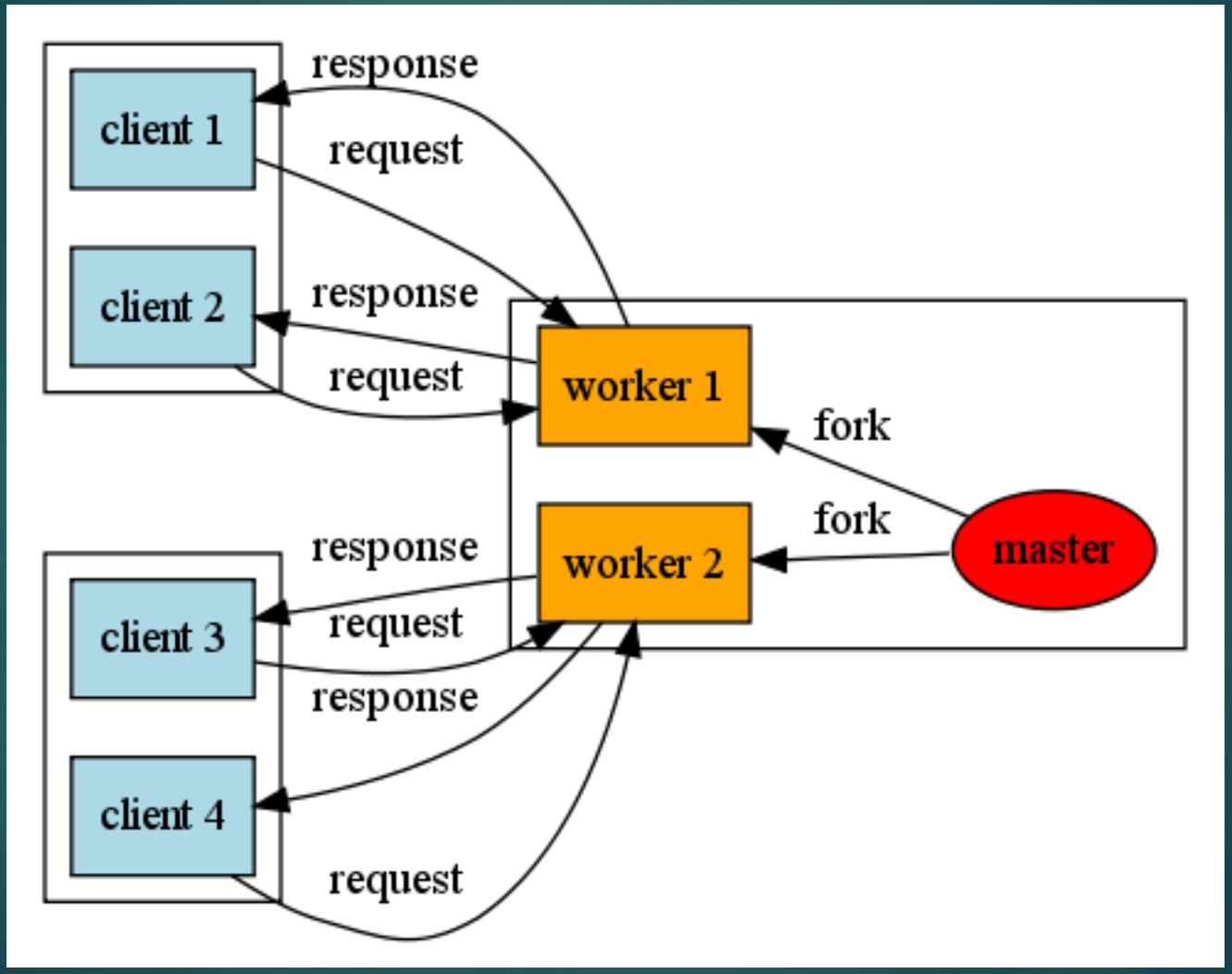
Request Flow

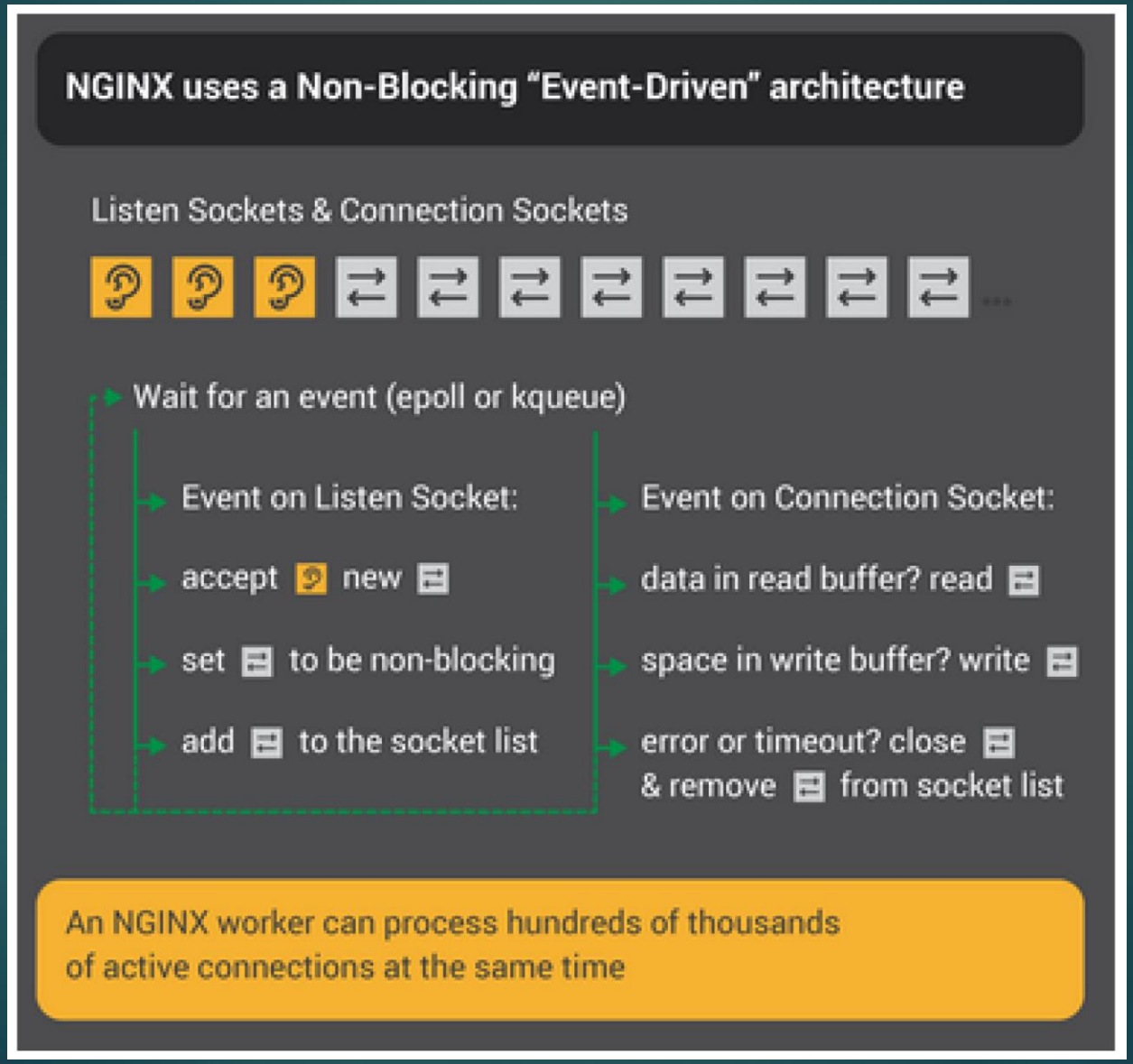


nginx's architecture

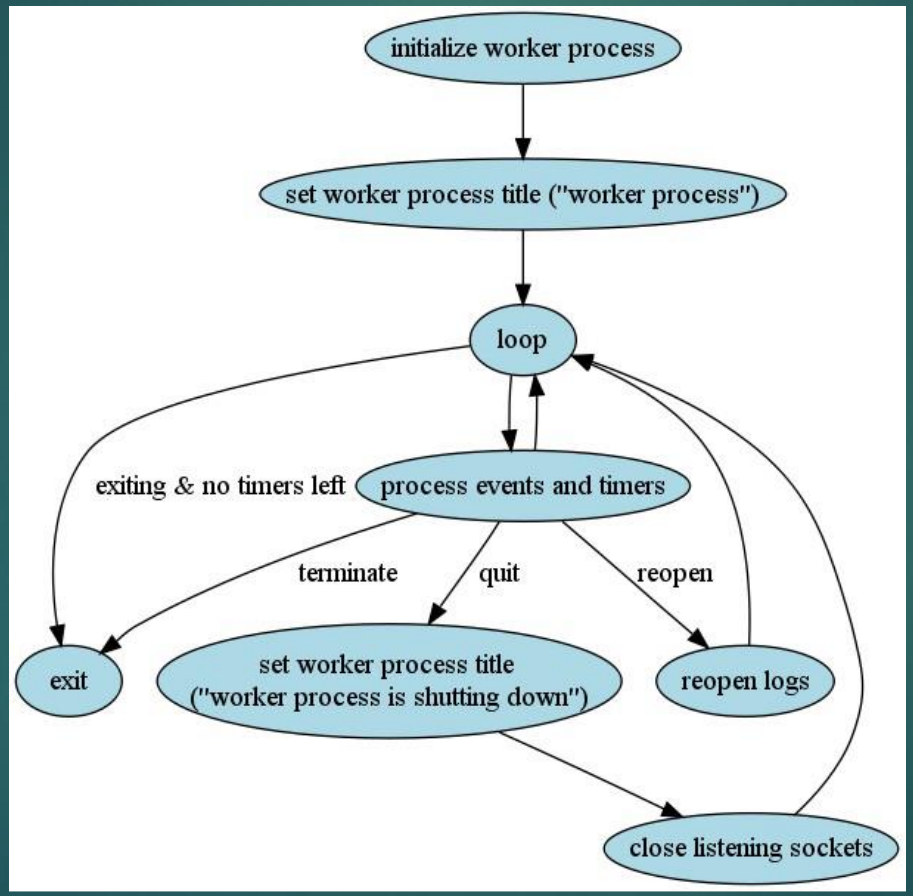


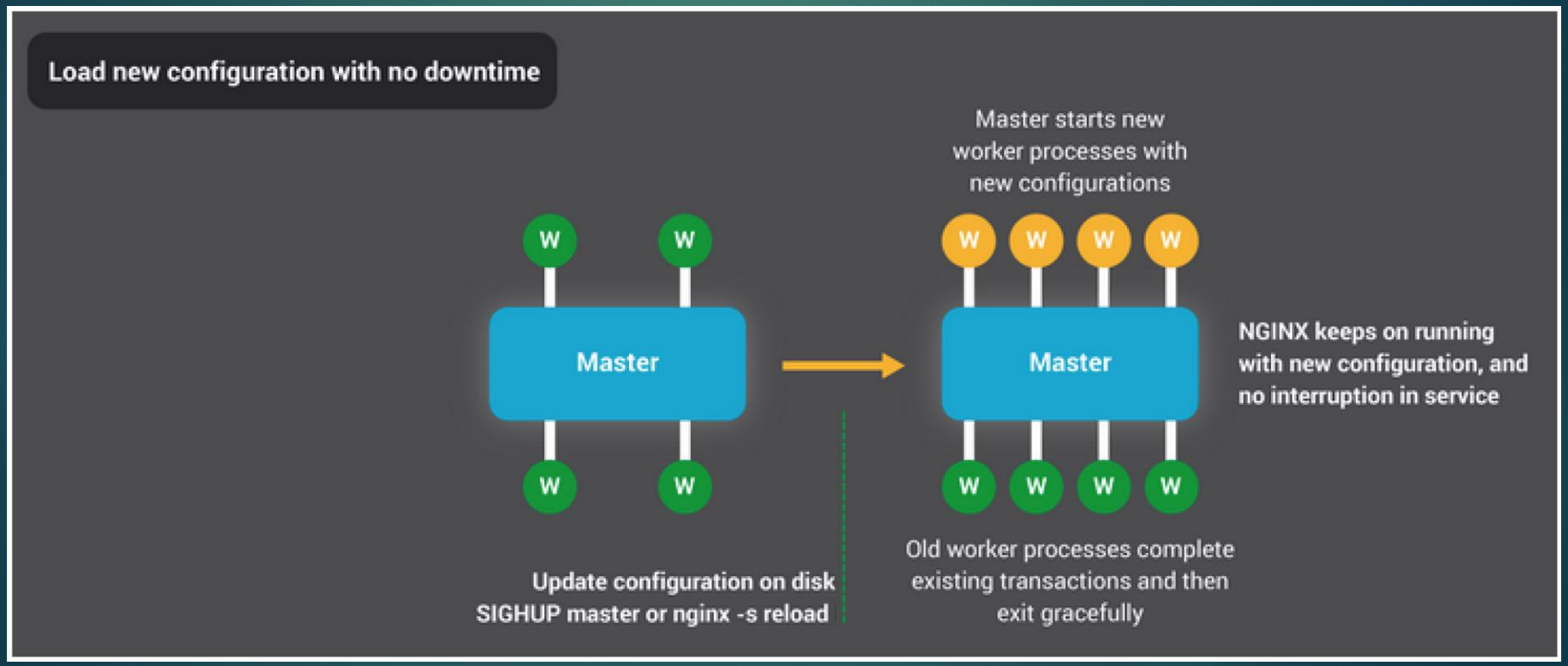
Master and Workers




NGINX uses a Non-Blocking "Event-Driven" architecture
Listen Sockets & Connection Sockets
Wait for an event (epoll or kqueue)
Event on Listen Socket:
accept new
set to be non-blocking
add to the socket list
Event on Connection Socket:
data in read buffer? read
space in write buffer? write
error or timeout? close
& remove from socket list
An NGINX worker can process hundreds of thousands
of active connections at the same time

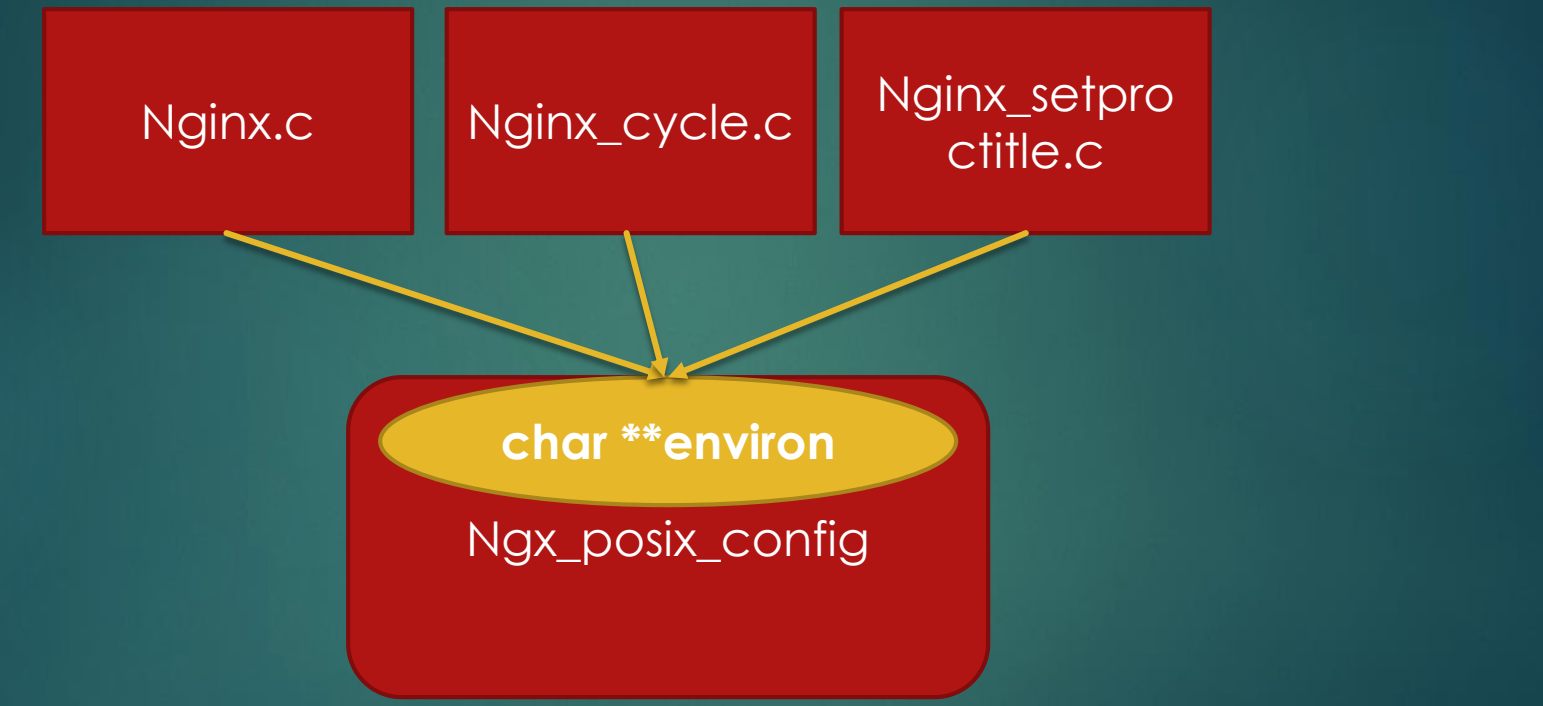
Worker Process Cycle




Load new configuration with no downtime
Master starts new worker processes with new configurations
W
W
W
W
Master
Master
W
W
W
W
W
W
W
W
NGINX keeps on running with new configuration, and no interruption in service
Update configuration on disk
SIGHUP master or nginx -s reload
Old worker processes complete existing transactions and then exit gracefully

- modules are compiled along with the core at build stage.
- statically assigns each client to a specific worker thread at connection establishment.

```
struct ngx_module_s {
    ngx_uint_t            ctx_index;
    ngx_uint_t            index;
    char                 *name;
    ngx_uint_t            spare0;
    ngx_uint_t            spare1;
    ngx_uint_t            version;
    const char           *signature;
    void                 *ctx;
    ngx_command_t        *commands;
    ngx_uint_t            type;
    ngx_int_t           (*init_master)(ngx_log_t *log);
    ngx_int_t           (*init_module)(ngx_cycle_t *cycle);
    ngx_int_t           (*init_process)(ngx_cycle_t *cycle);
    ngx_int_t           (*init_thread)(ngx_cycle_t *cycle);
    void                (*exit_thread)(ngx_cycle_t *cycle);
    void                (*exit_process)(ngx_cycle_t *cycle);
    void                (*exit_master)(ngx_cycle_t *cycle);
    uintptr_t             spare_hook0;
    uintptr_t             spare_hook1;
    uintptr_t             spare_hook2;
    uintptr_t             spare_hook3;
    uintptr_t             spare_hook4;
    uintptr_t             spare_hook5;
    uintptr_t             spare_hook6;
    uintptr_t             spare_hook7;
};
```

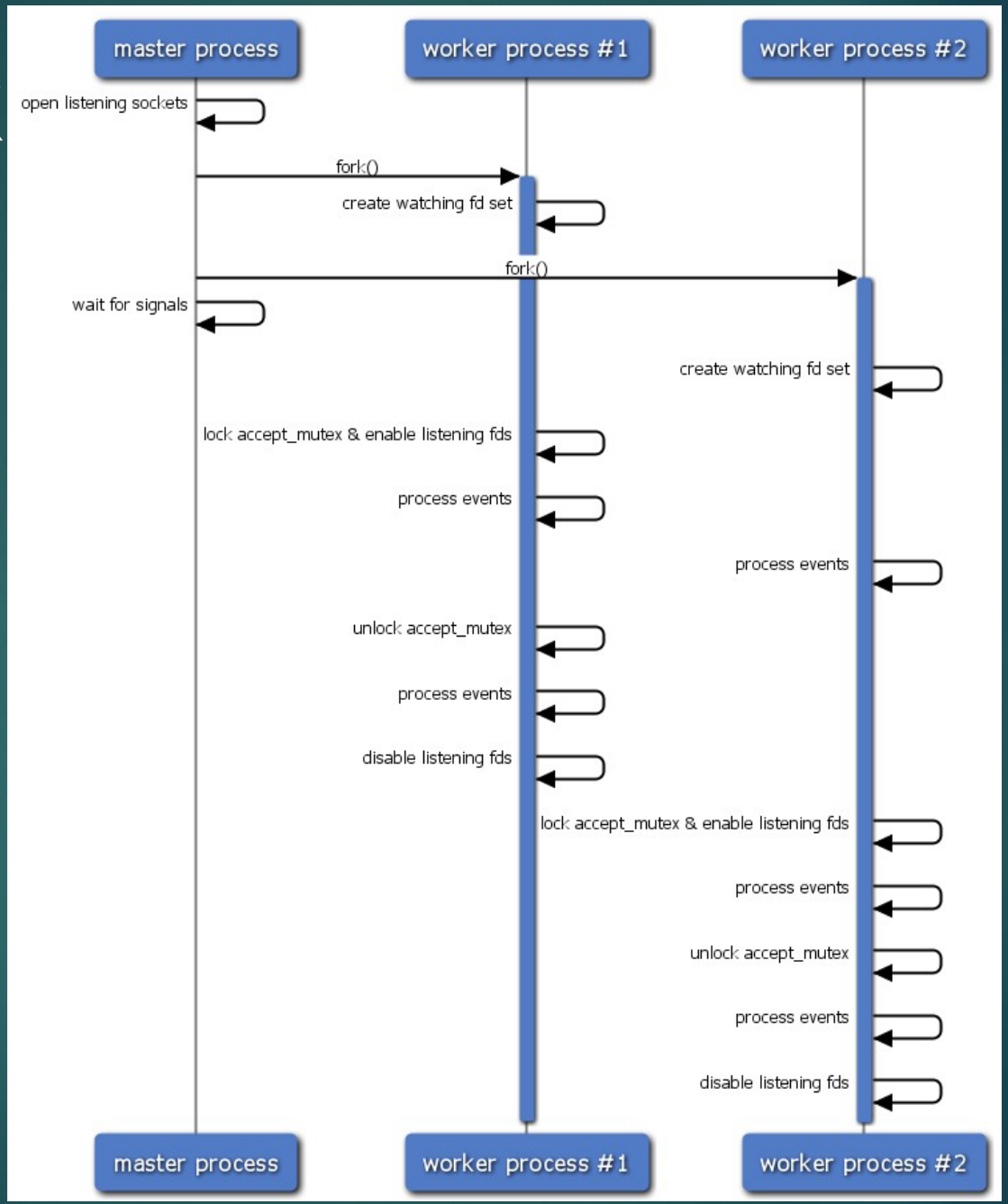


Char **ngx_os_environ, int ngx_kqueue, int ngx_eventfd,
aio_context_t ngx_aio_ctx, char *ngx_os_argv_last

Inter-process Communication

- Signals
- Shared memory
  - Connection counter
  - stat
  - atomic & spinlock
  - Mutex

Mutex


master process
worker process #1
worker process #2
open listening sockets
fork()
create watching fd set
fork()
wait for signals
create watching fd set
lock accept_mutex & enable listening fds
process events
process events
unlock accept_mutex
process events
disable listening fds
lock accept_mutex & enable listening fds
process events
unlock accept_mutex
process events
disable listening fds
master process
worker process #1
worker process #2

Data Structures

- Abstract data types
  - Array
  - List
  - Queue
  - Hash table
  - Red black tree
  - Radix tree
- Characteristic
  - keep interfaces clean
  - Efficient

General Module Interface

- Context
  - index & ctx_index
- Directives
- Type
  - core/event/http/mail
- Hooks
  - init_master
    - called at master process initialization
  - init_module
    - called when the module is loaded
  - init_process
    - called at worker process initialization
  - exit_process
    - called at worker process termination
  - exit_master
    - called at master process termination

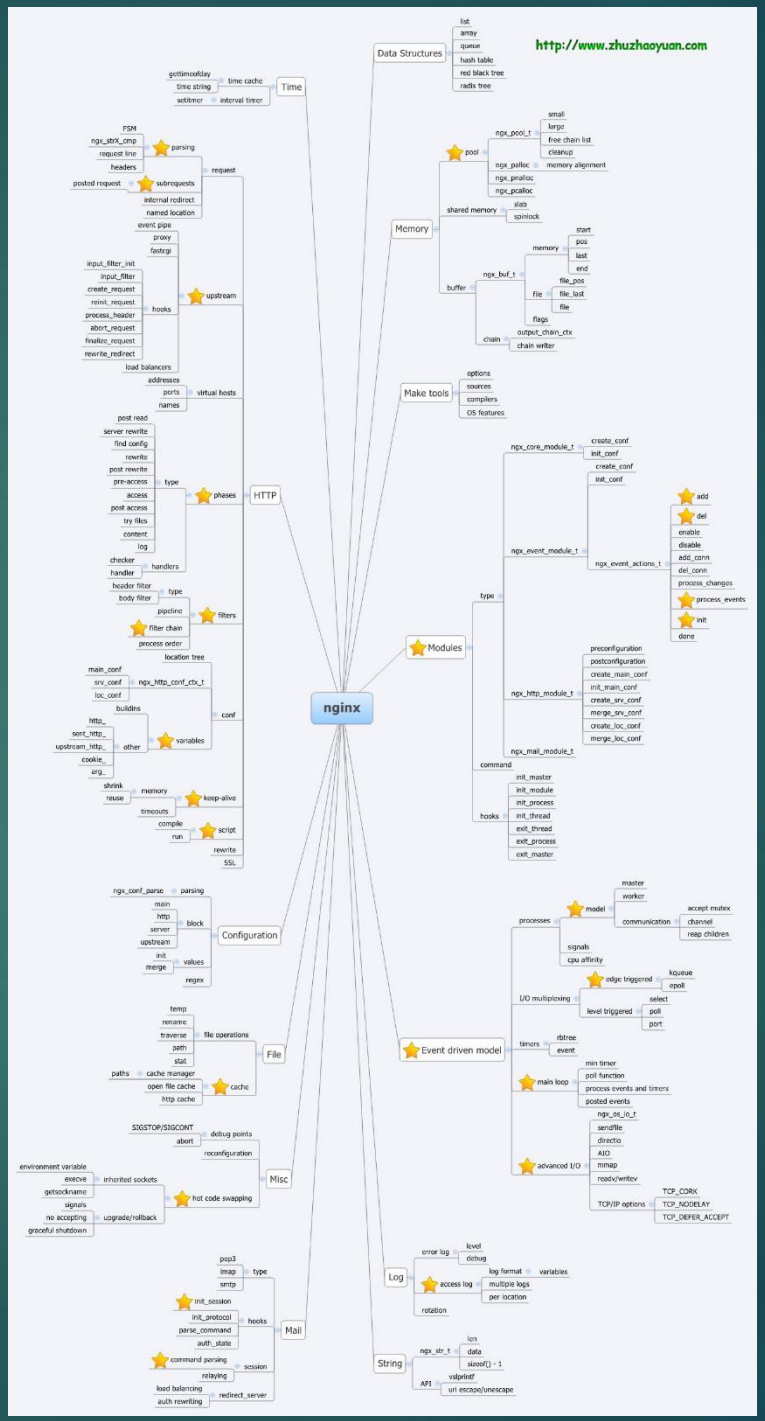
Event Module Interface

- Name
- Hooks
  - create_conf
  - init_conf
  - event_actions
    - add
    - del
    - enable
    - disable
    - add_conn
    - del_conn
    - process_changes
    - process_events
    - init
    - done

Code Organization

- core/
  - The backbone and infrastructure
- event/
  - The event-driven engine and modules
- http/
  - The HTTP server and modules
- mail/
  - The Mail proxy server and modules
- misc/
  - C++ compatibility test and the Google perftools module
- os/
  - OS dependent implementation files

Big Picture


http://www.zhuzhaoyuan.com
nginx
HTTP
Configuration
File
Misc
Mail
Time
Data Structures
Memory
Make tools
Modules
Event driven model
Log
String