

Web Server from Scratch

Greg Schafer
Sept 24, 2015

Why

Understand foundation



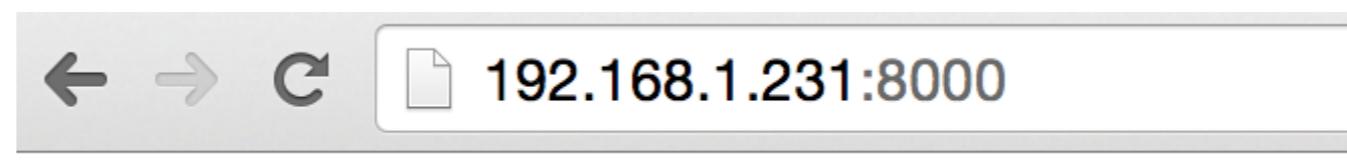
Implementation

- Python modules
 - socket
 - socketserver
 - http.server

Sidenote: http.server

python2 -m BaseHTTPServer

python3 -m http.server



- [.git/](#)
- [.gitignore](#)
- [.server.py.swp](#)
- [LICENSE](#)
- [README.md](#)

Basic server
using socket

socket - Echo Server

```
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.bind(('0.0.0.0', 8000))
    sock.listen()
    conn, addr = sock.accept()
    print('connected by', addr)
    while True:
        data = conn.recv(1024)
        if not data: break
        conn.sendall(data)
    conn.close()
```

socket - Echo Server

```
> python3 server.py
connected by ('127.0.0.1', 46784) > telnet localhost 8000
                                         Trying 127.0.0.1...
                                         Connected to localhost.
                                         Escape character is '^]'.
                                         server will echo everything I enter
                                         server will echo everything I enter
                                         hurray
                                         hurray
                                         ^]

telnet> Connection closed.
```

- Won't accept any connections other than the first

socket - Multi-connection Echo Server

```
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.bind(('0.0.0.0', 8000))
    sock.listen()
    while True:
        conn, addr = sock.accept()
        print('connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data: break
            conn.sendall(data)
        conn.close()
```

Let's make it
speak



HTTP

```
> telnet google.com 80
Trying 216.58.217.46...
Connected to google.com.
Escape character is '^]'.
HEAD / HTTP/1.1
```

```
HTTP/1.1 200 OK
Date: Tue, 22 Sep 2015 04:40:43 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See http://www.google.com/support/
accounts/bin/answer.py?hl=en&answer=151657 for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie:
PREF=ID=1111111111111111:FF=0:TM=1442896843:LM=1442896843:V=1:S=GBKCIpFhpuQ
xr2L6; expires=Thu, 31-Dec-2015 16:02:17 GMT; path=/; domain=.google.com
Set-Cookie:
NID=71=jj3ch6SpQ_DNSDs5iTfTv_0AsGYZIq2faSMMbk0_KEC9ulJ CtXLmwYXs4a0nvMHuXCd1
y--grI_af0Ied9n-QMG2DtAEshxHqitm2VFR1M-
fAQ49mJYanihx2R0tZxwZP8DUFCLUvNzhPS07jqnaRxqjfshgiptB; expires=Wed, 23-
Mar-20y
Transfer-Encoding: chunked
Accept-Ranges: none
Vary: Accept-Encoding
```

socket - HTTP Server

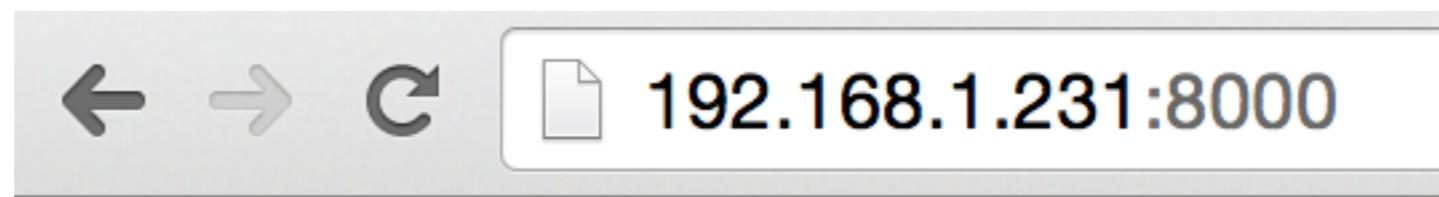
```
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.bind(('0.0.0.0', 8000))
    sock.listen()
    while True:
        conn, addr = sock.accept()
        print('connected by', addr)

        data = conn.recv(1024)
        print('received', data)

        response = b'HTTP/1.1 200 OK\n\nHello World!\n\n'
        conn.sendall(response)
        conn.close()
```

socket - HTTP Server



```
> python3 server.py
connected by ('192.168.1.114', 53276)
received b'GET / HTTP/1.1\r\nHost:
192.168.1.231:8000\r\nConnection: keep-alive\r
\nAccept: text/html,application/xhtml+xml,
application/xml;q=0.9,image/webp,*/*
;q=0.8\r\nUpgrade-Insecure-Requests: 1\r
\nUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_10_4) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/45.0.2454.93 Safari/537.36\r
\nAccept-Encoding: gzip, deflate, sdch\r
\nAccept-Language: en-US,en;q=0.8\r\n\r\n'
```

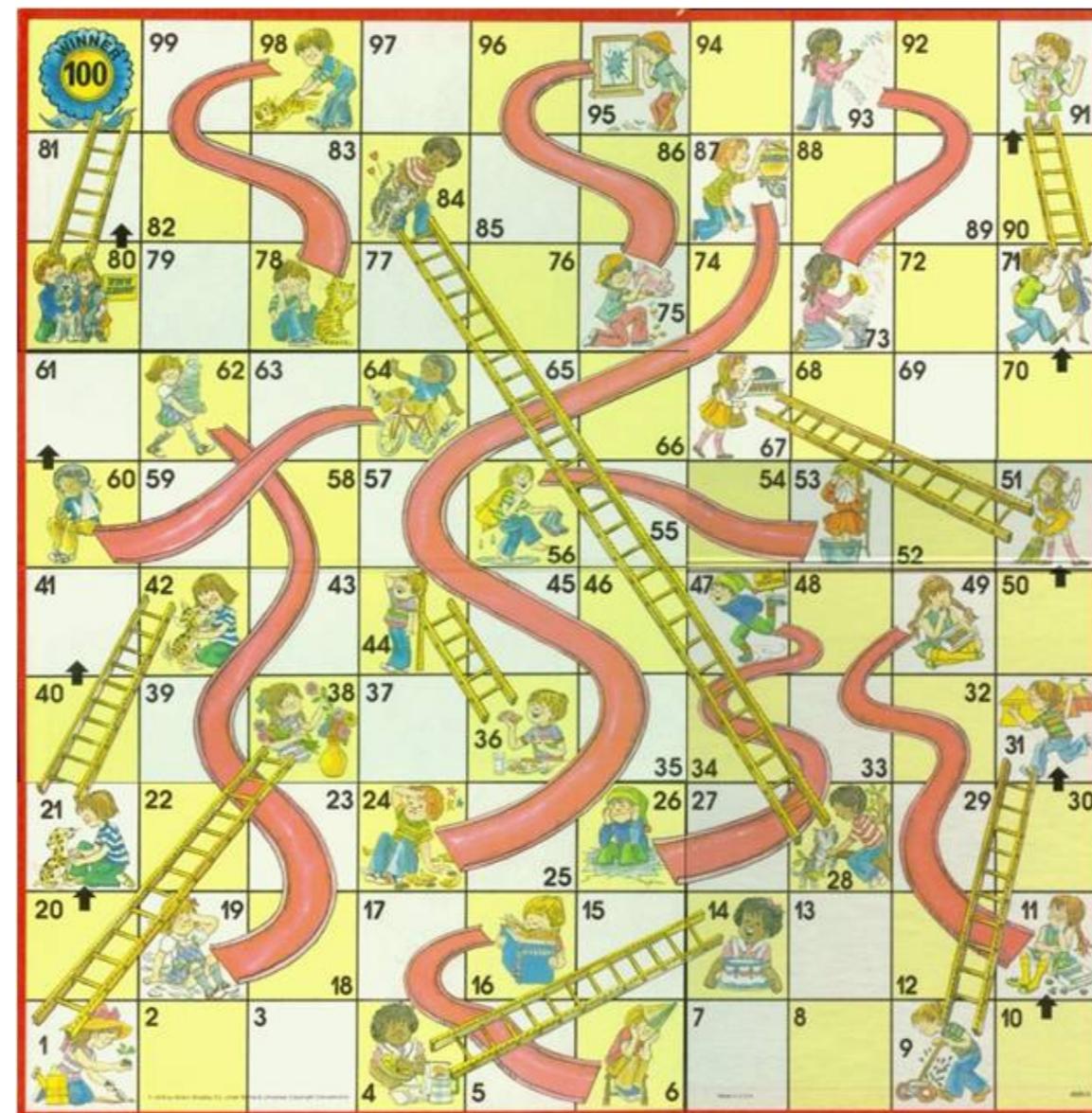
Parsing



```
data = conn.recv(1024)
content = data.decode('utf-8')
lines = content.split('\r\n')
method, path, protocol = lines[0].split()
print('method:', method)
print('path:', path)
print('protocol:', protocol)
headers = lines[1:-1]
print('headers:\n\t' + '\n\t'.join(headers))
```

```
connected by ('192.168.1.114', 64955)
method: GET
path: /
protocol: HTTP/1.1
headers:
    Host: 192.168.1.231:8000
    Connection: keep-alive
    Cache-Control: max-age=0
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    Upgrade-Insecure-Requests: 1
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/61.0.3163.100 Safari/537.36
    Accept-Encoding: gzip, deflate, sdch
    Accept-Language: en-US,en;q=0.8
    Cookie: csrftoken=zxsPglk4cMjRPQSMe9x0d
```

Routing and Views



```
from collections import namedtuple

Request = namedtuple('Request', ['method', 'path', 'headers'])

def hello_view(request):
    return '''HTTP/1.1 200 OK\r\n\r\nHello World!'''

def echo_headers(request):
    lines = ''.join('<p>{}</p>'.format(h) for h in request.headers)
    return '''HTTP/1.1 200 OK\r\n\r\n<html>{}</html>'''.format(lines)

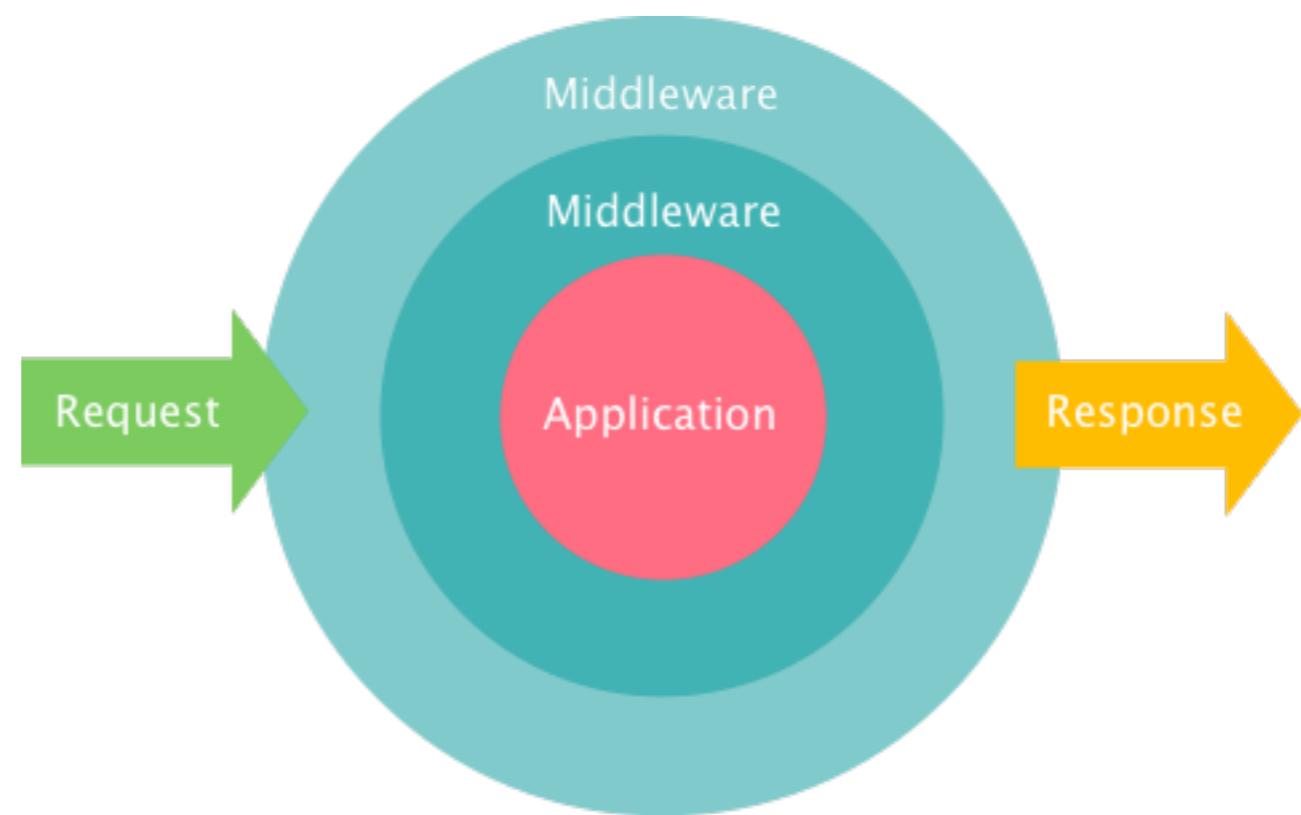
def make_404(request):
    return '''HTTP/1.1 404 Not Found\r\n\r\nResource doesn't exist'''
```

(After
parsing
code)

```
request = Request(method, path, headers)
if path == '/':
    response = hello_view(request)
elif path == '/headers':
    response = echo_headers(request)
else:
    response = make_404(request)
response = response.encode('utf-8')

conn.sendall(response)
```

Middleware



```
cache = {}

def cache_response(request, response):
    cache[request.path] = response
    return response

def return_cached(request):
    return cache.get(request.path)

request_middleware = ['return_cached']
response_middleware = ['cache_response']

    response = None
    request = Request(method, path, headers)
    for layer in request_middleware:
        response = locals()[layer](request)
        if response: break

    if not response:
        print('performing render')
        if path == '/':
            response = hello_view(request)
        elif path == '/headers':
            response = echo_headers(request)
        else:
            response = make_404(request)
        response = response.encode('utf-8')

    for layer in response_middleware:
        response = locals()[layer](request, response)
```

Concurrency



```
time.sleep(15)
response = '''HTTP/1.1 200 OK\r\n\r\nHello World!'''
response = response.encode('utf-8')
```

| Name | Method | Status | Type | Initiator | Size | Time | Timeline - Start Time |
|---------------|--------|--------|---------|-----------|------|---------|--|
| 192.168.1.231 | GET | 200 | docu... | Other | 31 B | 15.02 s | <div style="width: 15.02 s; background-color: #00A000; height: 10px;"></div> |
| 192.168.1.231 | GET | 200 | docu... | Other | 31 B | 28.70 s | <div style="width: 28.70 s; background-color: #00A000; height: 10px;"></div> |
| 192.168.1.231 | GET | 200 | docu... | Other | 31 B | 42.45 s | <div style="width: 42.45 s; background-color: #00A000; height: 10px;"></div> |

1. Forking/threading pre-2012 apache
2. Event-driven nginx
3. Coroutines

Forking

```
def handle_request(client_connection):
    data = conn.recv(1024)
    content = data.decode('utf-8')

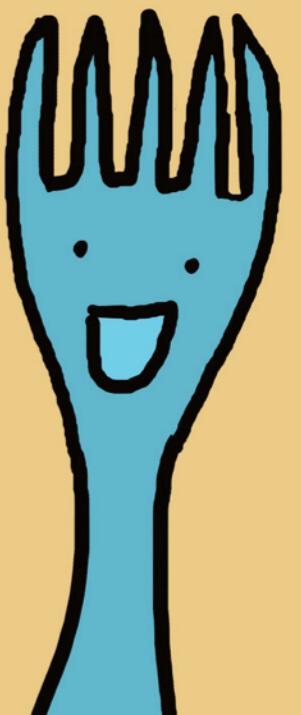
    time.sleep(15)
    response = '''HTTP/1.1 200 OK\r\n\r\nHello World!'''
    response = response.encode('utf-8')

    conn.sendall(response)
    conn.close()

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as sock:
    sock.bind(('0.0.0.0', 8000))
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.listen()
    while True:
        conn, addr = sock.accept()
        print('connected by', addr)

        pid = os.fork()
        if pid == 0: # child
            sock.close() # close child copy
            handle_request(conn)
            conn.close()
            os._exit(0) # child exits
        else: # parent
            conn.close() # close parent copy
```

I'm
forking
adorable!

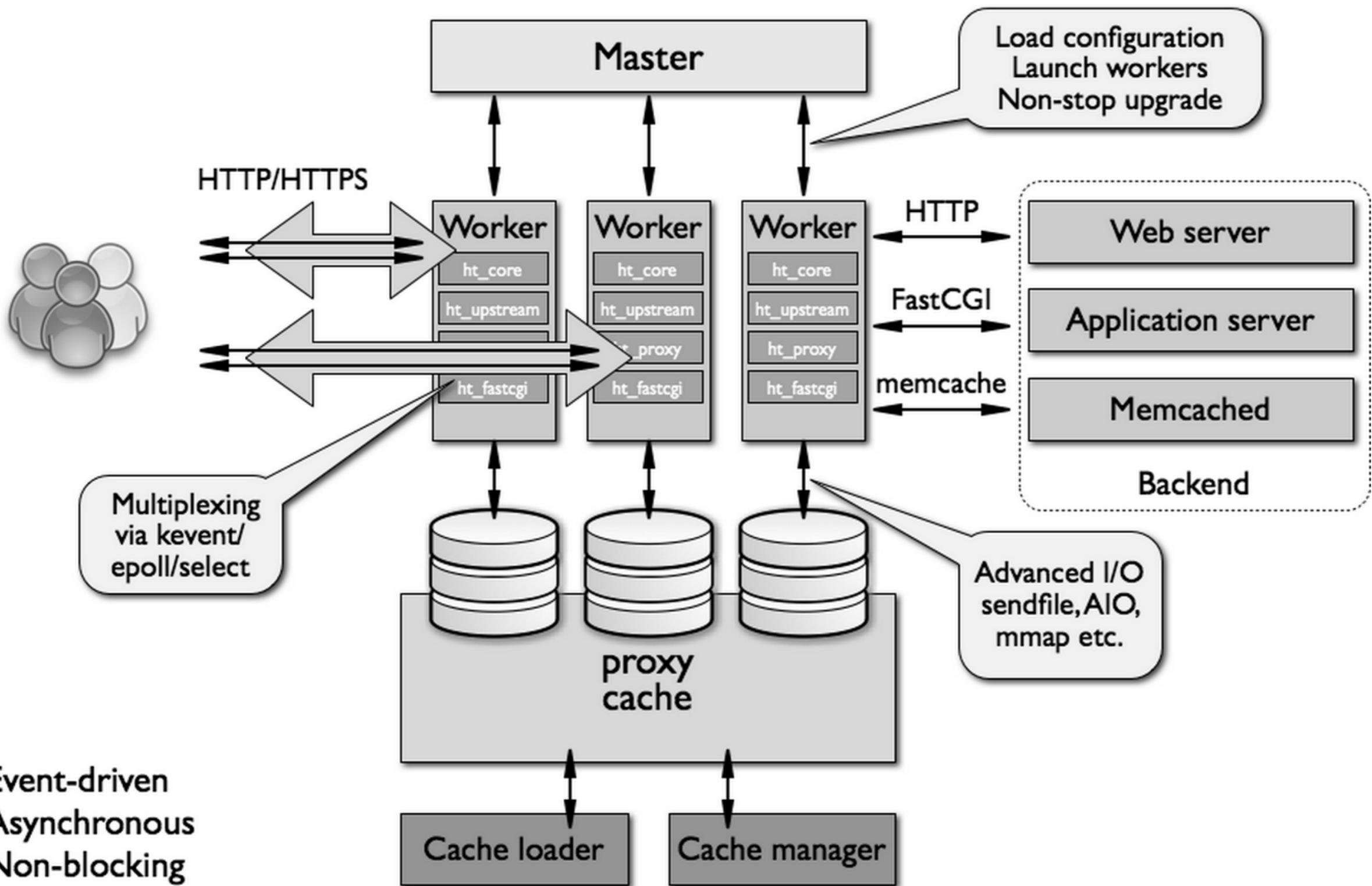


| Name | Method | Status | Type | Initiator | Size | Time | Timeline - Start Time | |
|---------------|--------|--------|---------|-----------|------|---------|-----------------------|---------|
| 192.168.1.231 | GET | 200 | docu... | Other | 31 B | 15.02 s | | 20.00 s |
| 192.168.1.231 | GET | 200 | docu... | Other | 31 B | 15.01 s | | 20.00 s |
| 192.168.1.231 | GET | 200 | docu... | Other | 31 B | 15.03 s | | 20.00 s |

```
10535 0.3 0.3 31668 7232 pts/0 S+ 05:49 0:00 python3 server.py
10537 0.0 0.2 31668 4756 pts/0 S+ 05:49 0:00 python3 server.py
10540 0.0 0.2 31668 4756 pts/0 S+ 05:49 0:00 python3 server.py
10541 0.0 0.2 31668 4756 pts/0 S+ 05:49 0:00 python3 server.py
```

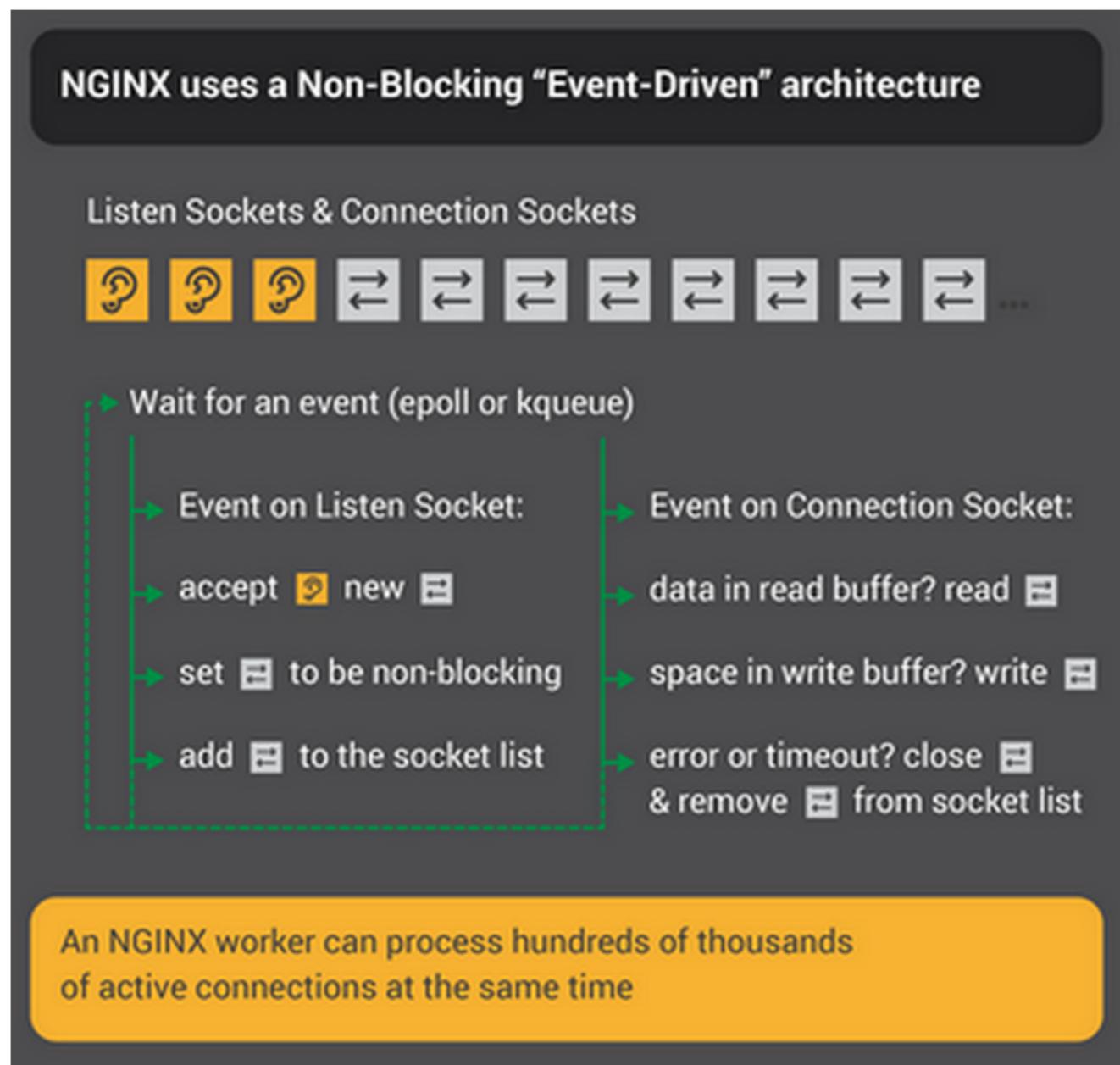
```
10535 0.1 0.3 31668 7232 pts/0 S+ 05:49 0:00 python3 server.py
10537 0.0 0.0 0 0 pts/0 Z+ 05:49 0:00 [python3] <defunct>
10540 0.0 0.0 0 0 pts/0 Z+ 05:49 0:00 [python3] <defunct>
10541 0.0 0.0 0 0 pts/0 Z+ 05:49 0:00 [python3] <defunct>
```

Event-driven



- Event-driven
- Asynchronous
- Non-blocking

Event-driven



Coroutines

```
import asyncio

async def handle_request(reader, writer):
    print('handling request')
    data = await reader.read(1024)
    content = data.decode('utf-8')

    addr = writer.get_extra_info('peernode')
    print('received', content, 'from', addr)

    print('send hello world')
    writer.write(b'''HTTP/1.1 200 OK\r\n\r\nHello World!''')
    await writer.drain()

    print('close socket')
    writer.close()

loop = asyncio.get_event_loop()
coro = asyncio.start_server(handle_request, host='0.0.0.0', port=8000, loop=loop)
server = loop.run_until_complete(coro)

print('Serving on {}'.format(server.sockets[0].getsockname()))
try:
    loop.run_forever()
except KeyboardInterrupt:
    pass

server.close()
loop.run_until_complete(server.wait_closed())
loop.close()
```

Higher-Level Modules

```
from http.server import HTTPServer, SimpleHTTPRequestHandler

class MyRequestHandler(SimpleHTTPRequestHandler):
    def do_GET(self):
        self.send_response(200)
        self.end_headers()
        self.wfile.write('Hello World!'.encode('utf-8'))

server_address = ('', 8000)
httpd = HTTPServer(server_address, MyRequestHandler)
httpd.serve_forever()
```

Higher-Level Modules

