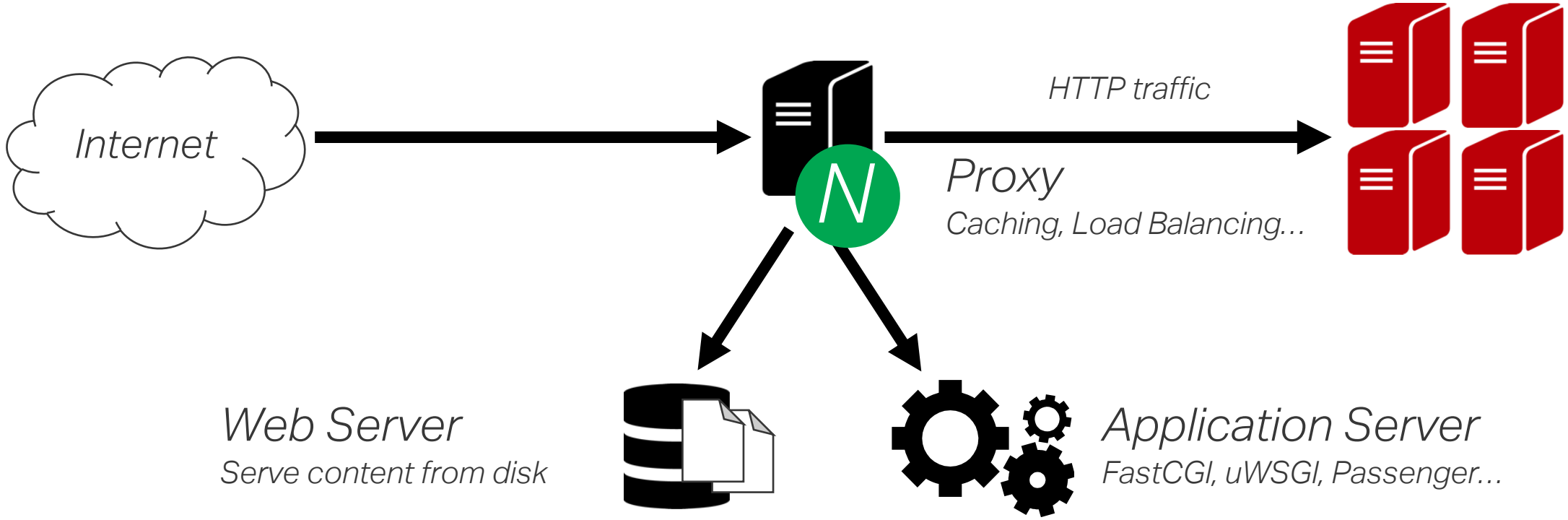
AGILITY

Exploring njs: NGINX JavaScript

Doug Gallarda
Senior Solutions Engineer, Eastern US

Matt Kryshak
Senior Solutions Engineer, Western US

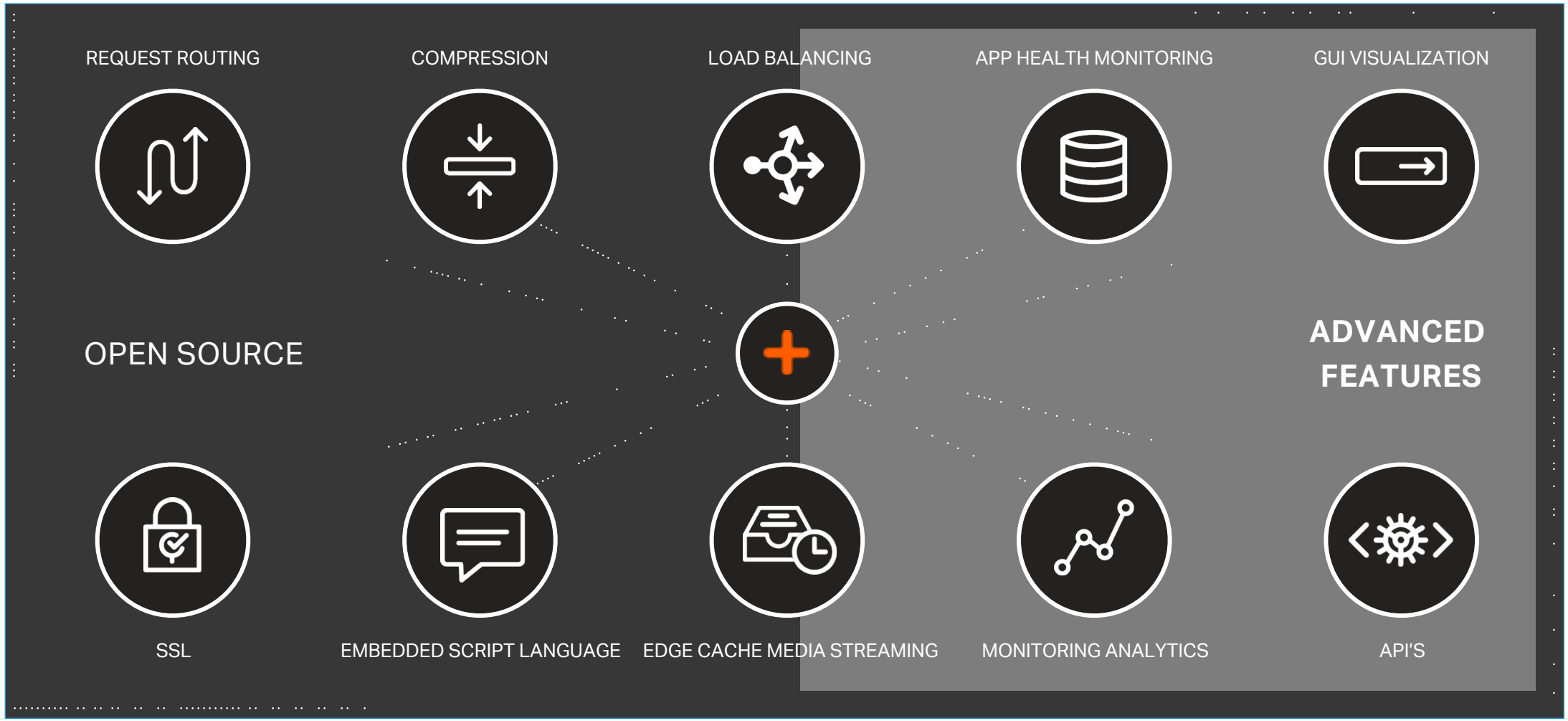
What is NGINX?



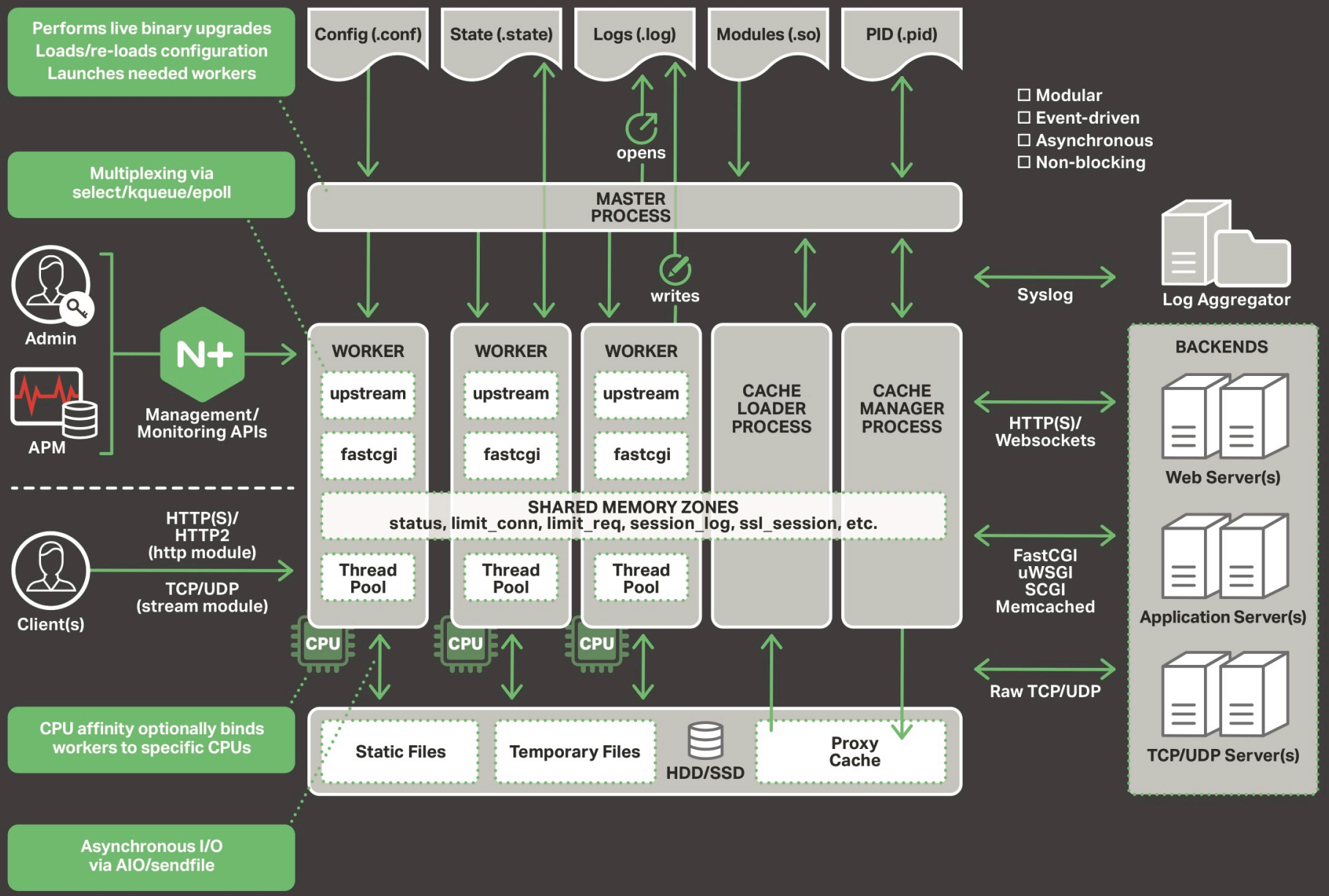
Advanced Features:

- ☑ *Application Acceleration*
- ☑ *SSL and HTTP2 termination*
- ☑ *Performance Monitoring*
- ☑ *High Availability*
- ☑ *Bandwidth Management*
- ☑ *Content-based Routing*
- ☑ *Request Manipulation*
- ☑ *Response Rewriting*
- ☑ *Authentication*
- ☑ *Video Delivery*
- ☑ *Mail Proxy*
- ☑ *GeoLocation*

NGINX and NGINX Plus



The Powerful and Efficient Architecture of NGINX



Example NGINX Configuration (/etc/nginx/nginx.conf)

PROCESSING REQUESTS AND RESPONSES

```
server {
    listen       80;
    server_name  localhost;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }

    location /app/ {
        proxy_pass http://backend1;
        proxy_set_header Host $host;
    }
}
```

Blocks

Directives

Variables

Variables and Directives

NGINX CONFIGURATION

$gzip_ratio
$host
$hostname (ngx_http_core_module)
$hostname (ngx_stream_core_module)
$http2
$http_
$https
$invalid_referer
$is_args
$jwt_claim_
$jwt_header_
$limit_conn_status (ngx_http_limit_conn_module)
$limit_conn_status (ngx_stream_limit_conn_module)
$limit_rate
$limit_req_status
$memcached_key
$modern_browser

gzip_vary
hash (ngx_http_upstream_module)
hash (ngx_stream_upstream_module)
health_check (ngx_http_upstream_hc_module)
health_check (ngx_stream_upstream_hc_module)
health_check_timeout
hls
hls_buffers
hls_forward_args
hls_fragment
hls_mp4_buffer_size
hls_mp4_max_buffer_size
http
http2_body_preread_size
http2_chunk_size
http2_idle_timeout
http2_max_concurrent_pushes

NGINX is Modular

ngx_http_core_module
ngx_http_access_module
ngx_http_addition_module
ngx_http_api_module
ngx_http_auth_basic_module
ngx_http_auth_jwt_module
ngx_http_auth_request_module
ngx_http_keyval_module
ngx_http_limit_conn_module
ngx_http_limit_req_module
ngx_http_log_module
ngx_http_map_module
ngx_http_memcached_module
ngx_http_mirror_module
ngx_http_mp4_module

ngx_stream_access_module
ngx_stream_geo_module
ngx_stream_geoip_module
ngx_stream_js_module
ngx_stream_keyval_module
ngx_stream_limit_conn_module
ngx_stream_log_module
ngx_stream_map_module
ngx_stream_proxy_module
ngx_stream_realip_module
ngx_stream_return_module
ngx_stream_split_clients_module
ngx_stream_ssl_module
ngx_stream_ssl_preread_module
ngx_stream_upstream_module
ngx_stream_upstream_hc_module
ngx_stream_zone_sync_module

Extending NGINX with Custom Modules

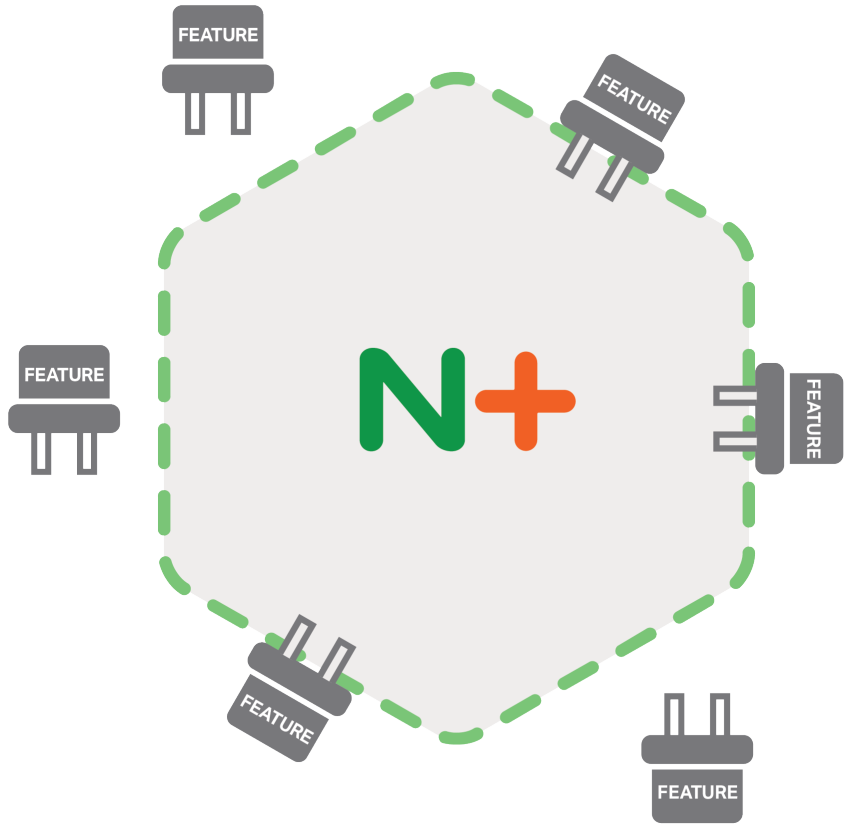
NGINX is extended by writing a custom module

Custom modules are written in C

Static Modules are compiled into the NGINX binary

Dynamic Modules can be loaded on demand

- `load_module modules/ngx_http_js_module.so;`



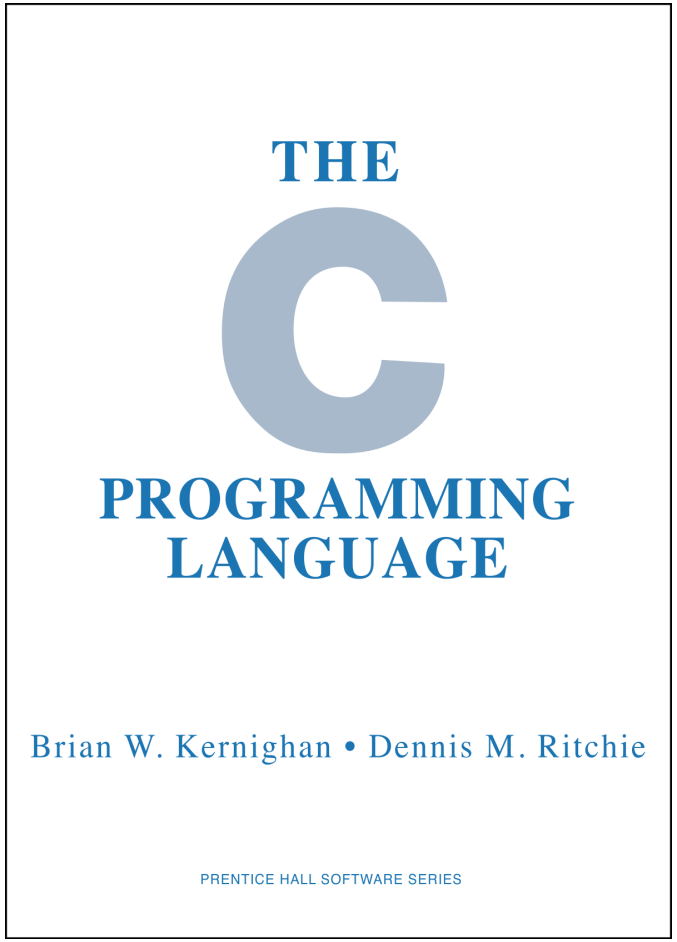
C is Hard

Programming in C is hard compared to more modern programming languages

Requires low level knowledge of the internal workings of NGINX

Imposes a significant barrier to entry

Most customization needs are simple extensions better done through a **scripting language**



Scripting NGINX

Perl

- Well known scripting language
- Great for string handling
- Worker process is blocked for long running tasks
- Error conditions cause worker process to abort

LUA

- Not well known to most programmers
- Arcane syntax compared to other languages
- Popular amongst NGINX user base

Choosing JavaScript

A GOOD FIT FOR NGINX

The programming language of the web

Well known to most programmers

C-like syntax

- Same style as NGINX config files

Event-driven programming

- Matches the event-driven nature of NGINX

JavaScript is the Programming Language of the Web

MANY IMPLEMENTATIONS IN THE WILD

Client-Side:

Server-Side:

njs: JavaScript Specialized for NGINX Configuration

INFORMS AND EXTENDS THE PROCESSING OF REQUESTS AND RESPONSES

Standards compliant JavaScript, but without the stuff we didn't need

- ECMAScript 5.1 compliant with some ECMAScript 6 and later extensions

No persistent JavaScript VM that depends on garbage collection to survive

- NGINX JavaScript runtime environment lives and dies with each request
- Single-threaded bytecode execution designed for quick initialization and disposal

Uses compiled C code in the NGINX server over JavaScript interpreted code where possible

- Non-blocking code execution through NGINX's event-driven schedular
- Presents a JavaScript interface to native NGINX built-in operations and data structures
- Closely integrated with NGINX request processing phases

Optimizing for Performance

FAST CREATION AND DESTRUCTION OF VMS

Bytecode compilation at start time

```
>> 1+1*2
```

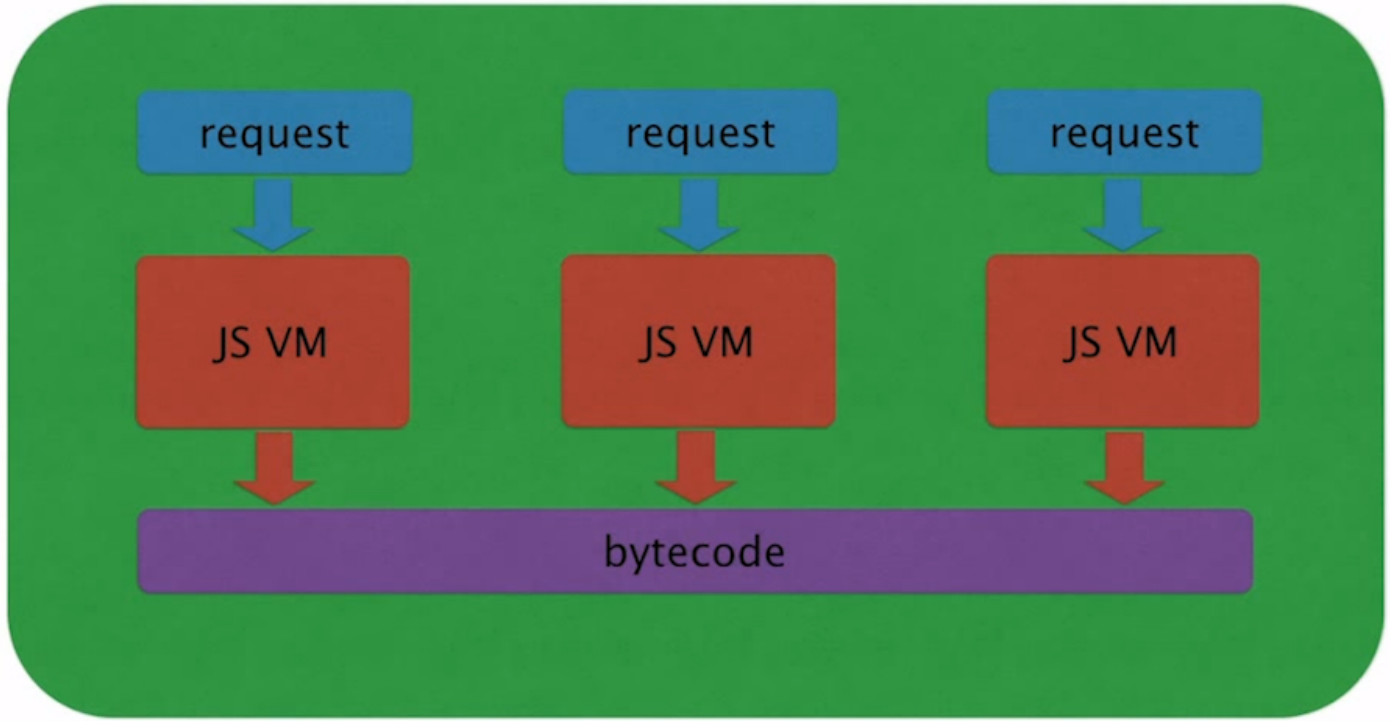
00000 MULTIPLY	1652F30 1652E10 1652F20
00040 ADD	1652F30 1652E10 1652F30
00080 STOP	1652F30

Copy-on-write cloning of compiled VM for each request

No JIT compilation

Register vs Stack based VM

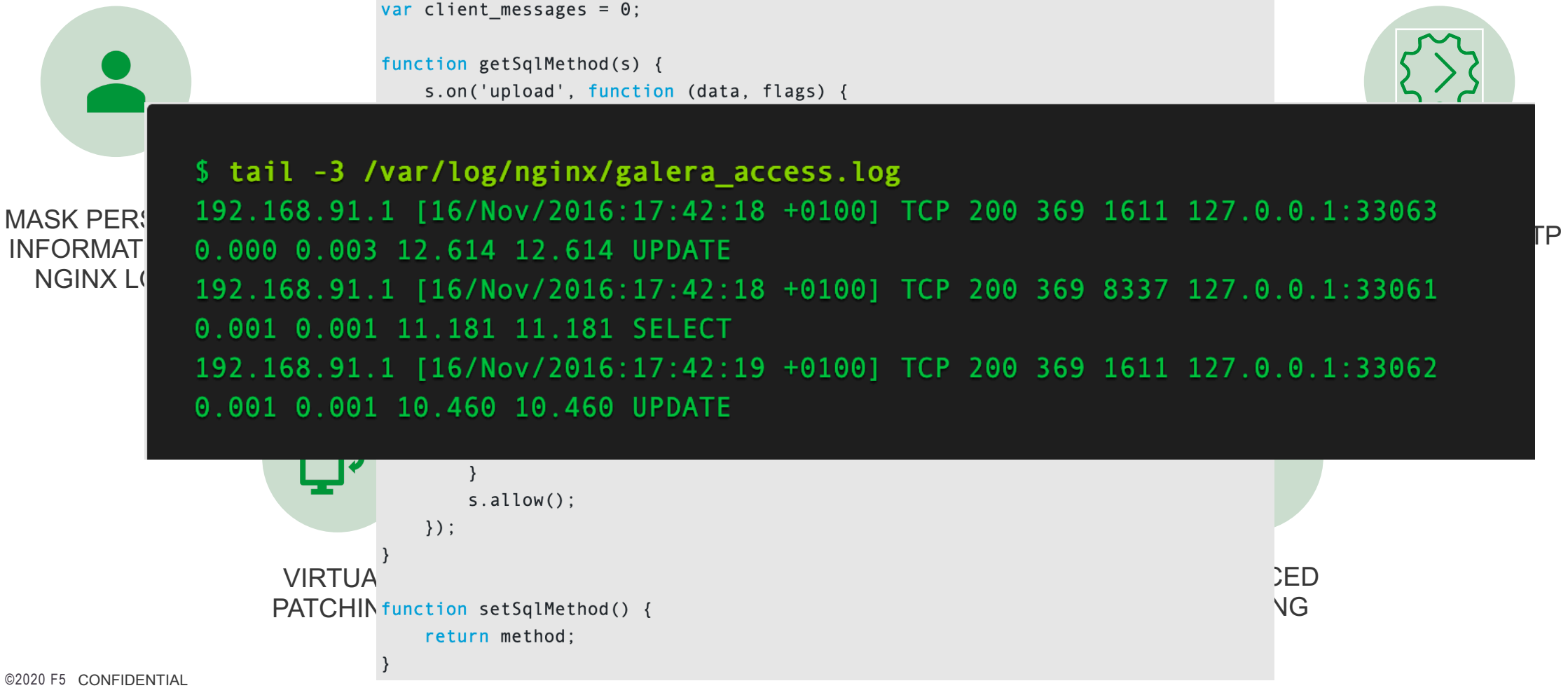
UTF8 vs UTF16 strings



njs is Fast


times/s
1000000
100000
10000
1000
V8/6.8
SpiderMonke
duktape/2.3
njs/0.2.4

Example use cases



Playing with njs

Installing njs: (Included in Docker Image)

```
yum install nginx-plus-module-njs
```

Enabling njs in your NGINX config:

```
load_module modules/ngx_stream_js_module.so;
load_module modules/ngx_http_js_module.so;
```

Command line tool:

```
$ njs
interactive njs 0.4.1
```

Importing njs into NGINX Config

AS OF NJS 0.4.0

```
load_module modules/ngx_http_js_module.so;

 http {

   js_import example.js;

   server {
     listen 80;

     location / {
       js_content example.hello;
     }
   }
}
```

Enable JavaScript Module

Import JavaScript Code

Invoke JavaScript Function

Including a Single njs File into NGINX Config

NOW DEPRECATED

```
js_include example.js;

server {
  listen 80;

  location / {
    js_content hello;
  }
}
```

Only One File Allowed

One Namespace

NGINX Config with Inline JavaScript (nginScript)

NO LONGER SUPPORTED

```
location / {
    js_run "
    var res;
    res = $r.response;
    res.status = 200;
    res.send('Hello World!');
    res.finish();
    ";
}
```

Hello World in njs *(Content Handler)*

USING JS_CONTENT

nginx.conf

```
load_module modules/ngx_http_js_module.so;

 http {

   js_import example.js;

   server {
     listen 80;


     location /version {
        js_content example.version;
     }


     location / {
       js_content example.hello;
     }
  }
}
```

example.js

```
function version(r) {
  r.return(200, njs.version);
}


function hello(r) {
  r.return(200, "Hello world!\n");
}

export default {hello, version};
```

Making NGINX Data Available to **njs** JavaScript

BY PASSING A REQUEST OBJECT (R)

HTTP Method:	`r.method`
HTTP URI:	`r.uri`
Query String:	`r.args{}`
HTTP Headers:	`r.headersIn{}`
HTTP Version:	`r.httpVersion`
Client Address:	`r.remoteAddress`
NGINX Variables:	`r.variables{}`

Send Response:	`r.return()`
Web Request:	`r.subrequest()`
Redirect Request:	`r.internalRedirect()`
Write info log:	`r.log()`
Send Headers:	`r.sendHeader()`
Send Body Part:	`r.send()`
Finish Sending:	`r.finish()`

Hello World in njs *(Variable Handler)*

USING JS_SET

nginx.conf

```
http {

  js_import main from example.js;
  js_set $encoded_foo main.encoded_foo;
  js_set $greeting main.hello;

  server {
    listen 80;

    location / {
       proxy_pass http://example.com?foo=$encoded_foo;
    }


    location /hello {
      return 200 $greeting;
    }
  }
}
```

example.js

```
function encoded_foo(r) {
  return encodeURIComponent('foo & bar?');
}


function hello(r) {
  return "Hello world!\n";
}

export default {hello, encoded_foo};
```

Advanced: Promise and Arrow Functions

FETCHING A TOKEN FOR ACCESS TO AN UPSTREAM SERVER

```
function content(r) {
   r.subrequest(r, '/auth')
   .then(reply => JSON.parse(reply.responseBody))
   .then(response => {
       if (!response['token']) {
           throw new Error("token is not available");
       }
       return token;
   })
  .then(token => {
      r.subrequest('/backend', `token=${token}`)
      .then(reply => r.return(reply.status, reply.responseBody));
  })
  .catch(_ => r.return(500));
}
```

Where JavaScript Plugs Into NGINX Request Processing

NGINX Processing Phase	HTTP Module	Stream Module
Access – Network connection access control		js_access
Pre-read – Read/write body		js_preread
Filter – Read/write body during proxy		js_filter
Content – Send response to client	js_content	
Log / Variables – Evaluated on demand	js_set	js_set

Using njs for Access Control

USING JS_ACCESS

nginx.conf

```
load_module modules/ngx_stream_js_module.so;


 events {}


 http {
   js_import stream.js;

   server {
     listen 12346;

     js_access stream.access;

       }
}
```

stream.js

```
function access(s) {
  if (s.remoteAddress.match('^192.*')) {
    s.abort();
    return;
  }
  s.allow();
}
```

Using njs for TCP/UDP Load Balancing

USING JS_PREFETCH

```
function getClientId(s) {
    s.on('upload', function (data, flags) {
        if ( data.length == 0  ) {
            return;
        } else {
                client_id_str = data.toString().match(/(?<=Call-ID: )[^\r]+/);
        }
        s.allow();
    });
}
```

Lab Modules

5.1. Getting Started
5.2. Hello World [hello]
5.3. Decode URI [decode_uri]
5.4. Extract JWT Payload into NGINX Variable [jwt]
5.5. Subrequests join [join_subrequests]
5.6. Secure hash [secure_link_hash]
5.7. File IO [file_io]
5.8. Complex redirects using njs file map. [complex_redirects]
5.9. Injecting HTTP header using stream proxy [stream/inject_header]
5.10. Generating JWT token [gen_hs_jwt]
5.11. Choosing upstream in stream based on the underlying protocol [stream/detect_http]

To The Lab!!

Agility Class Lab Guide

Learn More About njs

The njs home page on nginx.org: https://nginx.org/en/docs/njs/

Installing njs on NGINX Plus: https://docs.nginx.com/nginx/admin-guide/dynamic-modules/nginscript/

Blog posts on njs: https://www.nginx.com/blog/introduction-nginscript/

Github repo with njs example code: https://github.com/xeioex/njs-examples

Let F5 Know What You Think!

TAKE OUR SURVEY AND WIN A FREE PAIR OF FLIP-FLOPS

Thank  You!