# Nginx Notes

Paolo D'Apice

iZENEsoft (Shanghai) Co. Ltd.

February 13, 2012

# What's Nginx

## A definition

NGINX™

*Apache is like Microsoft Word, it has a million options but you only need six.*
*Nginx does those six things, and it does five of them 50 times faster than Apache.*

Chris Lea

# What's Nginx

## Another definition

*Batman is fast. Nginx is fast.*
*Batman fights crime. Nginx fights wasted CPU cycles and memory leaks.*
*Batman performs well under pressure. Nginx, for its part, excels under heavy server loads.*

*But Batman would be almost nothing without the Batman utility belt. Instead of a utility belt, Nginx has a module chain.*

Evan Miller

# How it works

## Modules

Module types:

- **handlers** process a request and produce an output
- **filters** manipulate the output produced by a handler
- **load-balancers** forward a request choosing between multiple backend servers

# Components

1. configuration structs
2. directives
3. context
4. definition
5. installation

# Components

## Configuration structs

- up to three
  - main
  - server
  - location
- populated by module directives

# Components

## Directives

- array of `ngx_command_t` structs
- used for module configuration

# Components

## Context

- an `ngx_http_module_t` struct
- contains function references for creating and merging configurations

# Components

## Definition

- an `ngx_module_t` struct named `ngx_<module name>_module`
- contains references to context, directives and other callbacks

# Components

## Installation

- depends on the module type
- see later . . .

# Handlers

Typically do four things:

1. get location configuration
2. generate appropriate response
3. send response header
4. send response body

# Handlers

## Upstream Handlers (Proxy)

- do little "real work"
- callbacks invoked when the upstream is ready

# Handlers

## Installation

- define the directive that enables the module
- get the "core" struct for the location
- assign a handler to it

# Filters

Manipulate responses generated by handlers

- header filters
- body filters

# Filters

## Header filters

Three basic steps:

1. decide whether to operate on the response
2. operate on the response
3. call the next filter

# Filters

## Body filters

- operate only on one buffer (chain link) at a time
  - overwrite
  - replace
  - insert new buffer before/after
- support incomplete buffer chain
- no nice high-level API

# Filters

## Installation

- post-configuration step
- insert the filter on the top of filter chain (LIFO)

# Load-Balancer

Decide which backend server will receive a particular request

1. enabling configuration directive
2. registration function
3. upstream initialization function
4. peer initialization function
5. load balancing function
6. peer release function

# Load-Balancer

## enabling directive

- should have the `NGX_UPS_CONF` flag set
- provide a pointer to a *registration function*

# Load-Balancer

## registration function

- define options within the `upstream` configuration
- register an *upstream initialization function*

# Load-Balancer

## upstream initialization function

- resolves the server names
- allocates sockets
- sets a callback to the *peer initialization function*

# Load-Balancer

## peer initialization function

- called *once* per request
- sets up data structures used by the *load balancing function*
  - struct named `ngx_http_upstream_<module>_peer_data_t`
- sets up two callbacks

  get the *load-balancing function*

  free the *peer release function*

# Load-Balancer

## load balancing function

- called *at least* per request
- defines the load-balancing policy using sockets

# Load-Balancer

## peer release function

- operates after the upstream connection take place
- at least must set `pc->tries` to zero

# Writing Nginx Modules

- no or little documentation
  - mostly in Russian or Chinese
  - *self documented* source code
- learn by example

# Writing Nginx Modules

- write you code
- write a `config` file
- compile Nginx with your module

# Writing Nginx Modules

- take a look at the hello-module
  - autotools adapted to CMake
  - C++ support[1]
  - Test::Nginx Perl framework
- stick to the naming standards

[1] Nginx is written in plain C

# References

Nginx Wiki

`http://wiki.nginx.org/Main`

Evan Miller's *definitive* guide

`http://www.evanmiller.org/nginx-modules-guide.htm`

Antoine Bonavita's blog about Nginx

`http://www.nginx-discovery.com`

# More References

- Taobao's Tengine
  `http://tengine.taobao.org`
- agentz's blog
  `http://agentzh.org`
- chaoslawful's blog
  `http://chaoslawful.iteye.com`
- Joshua Zhu's blog
  `http://blog.zhuzhaoyuan.com`