Using `ngx_lua` in UPYUN 2

Monkey Zhang (timebug)

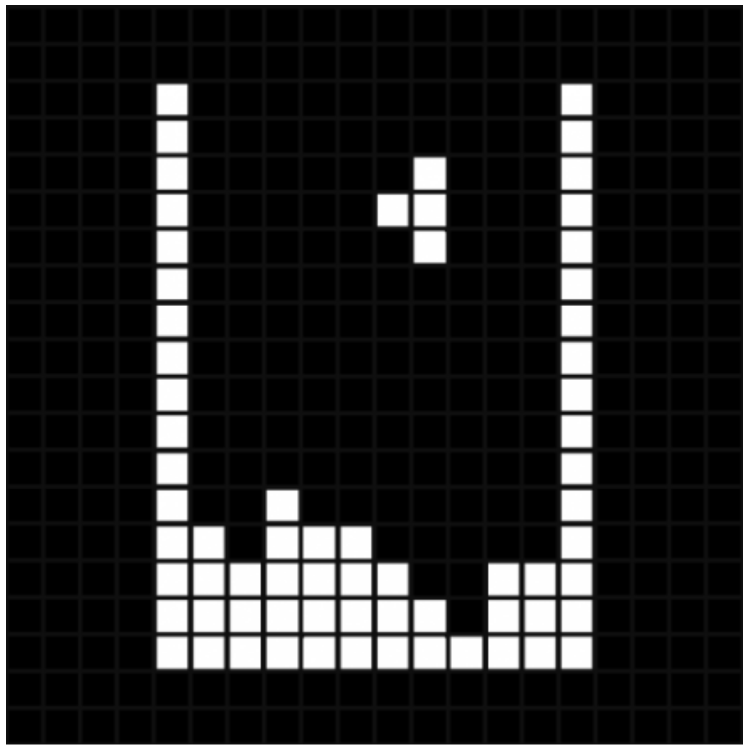
2015.11 @ Beijing OpenResty Con

云存储

云处理

云分发

upyun



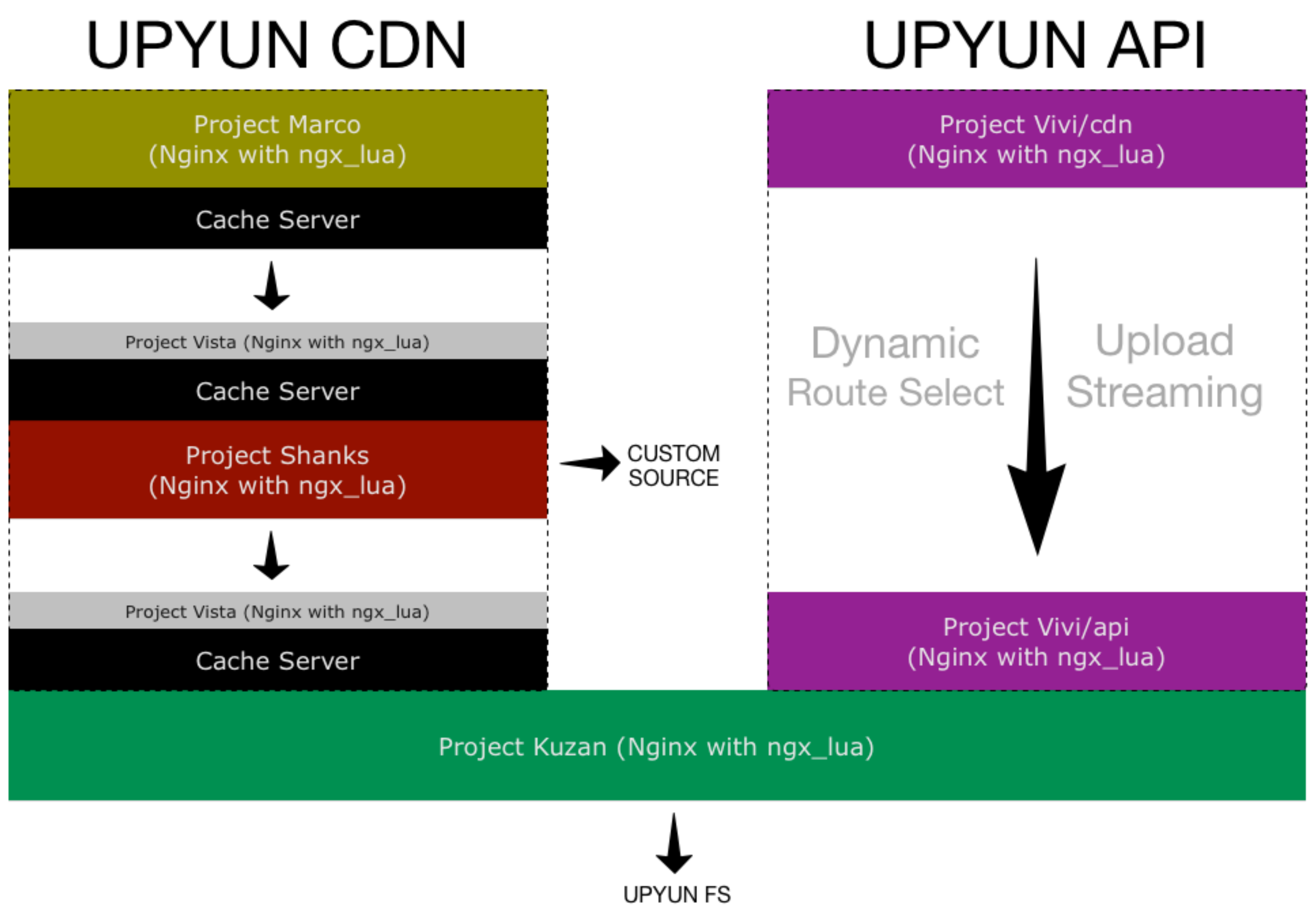
A Systems Engineer at UPYUN

- ★ Email: timebug.info@gmail.com
- ★ Github: https://github.com/timebug

```
$ ./configure --prefix=/opt/nginx \
    --add-module=/path/to/lua-nginx-module
```

```
http {
    server {
        listen 8080;

        location /add {
            set $res '';

            rewrite_by_lua '
                local a = tonumber(ngx.var.arg_a) or 0
                local b = tonumber(ngx.var.arg_b) or 0
                ngx.var.res = a + b
            ';

            content_by_lua '
                ngx.say(ngx.var.res)
            ';
        }
    }
}
```

```
$ curl 'http://localhost:8080/add?a=6&b=7'
13
```



UPYUN CDN & API is built on top of NGINX with ngx_lua

Why not use OpenResty?

40000+ lines Lua

lua-resty-sniff

lua-resty-dbcache

lua-resty-limit-req

lua-resty-combo

lua-resty-httppipe

lua-resty-**httproxy**

lua-resty-anticc

lua-resty-checkups

lua-resty-rewrite

lua-resty-argutils

lua-resty-17monip

...

Project Structure:

NGINX with ngx_lua

```
~/project/upyun/marco

├── Makefile
├── README.md
├── addons
|   └── ngx_upxxx_module
├── deps
├── nginx
|   ├── app
|   |   ├── etc
|   |   |   └── config.lua
|   |   ├── lib
|   |   |   └── resty
|   |   |       └── httpipe.lua
|   |   └── src
|   |       ├── modules
|   |       |   └── referer.lua
|   |       ├── marco_init.lua
|   |       └── marco_log.lua
|   └── conf
|       └── nginx.conf
├── patches
├── tests
└── util
    ├── deps
    ├── lua-releng
    └── ver.cfg
```

make install

```
/usr/local/marco

├── luajit
└── nginx
    ├── app
    |   ├── etc
    |   |   └── config.lua
    |   ├── lib
    |   |   ├── cjson.so
    |   |   └── resty
    |   |       └── httpipe.lua
    |   └── src
    |       ├── modules
    |       |   └── referer.lua
    |       ├── marco_init.lua
    |       └── marco_log.lua
    ├── conf
    |   └── nginx.conf
    ├── html
    ├── logs
    └── sbin
        └── nginx
```

Project Structure:

Quick Start & Run

make deps

make configure

make

make install

```
util/ver.cfg

V_PCRE=8.34
V_NGINX=1.7.10
V_LUAJIT=2.1-20150223
V_LUA_CJSON=2.1.0
V_NGX_DEVEL_KIT=0.2.19
V_NGX_LUA_MODULE=0.9.15
```

```
Makefile

INSTALL_LIBDIR=$(PREFIX)/nginx/app/lib/
configure: deps luajit
        @echo "==== Configuring Nginx $(V_NGINX) ===="
        cd $(NGINX_DIR) && ./configure \
                --with-pcre=$(ROOTDIR)/deps/pcre-$(V_PCRE) \
                --with-ld-opt="-Wl,-rpath,$(LUAJIT_LIB),-rpath,$(INSTALL_LIBDIR)" \
                --add-module=$(ROOTDIR)/deps/ngx_devel_kit-$(V_NGX_DEVEL_KIT) \
                --add-module=$(ROOTDIR)/deps/lua-nginx-module-$(V_NGX_LUA_MODULE) \
                --prefix=$(PREFIX)/nginx
        @echo "==== Successfully configure Nginx $(V_NGINX) ===="
```

Project Structure:

Development & Test

make dev

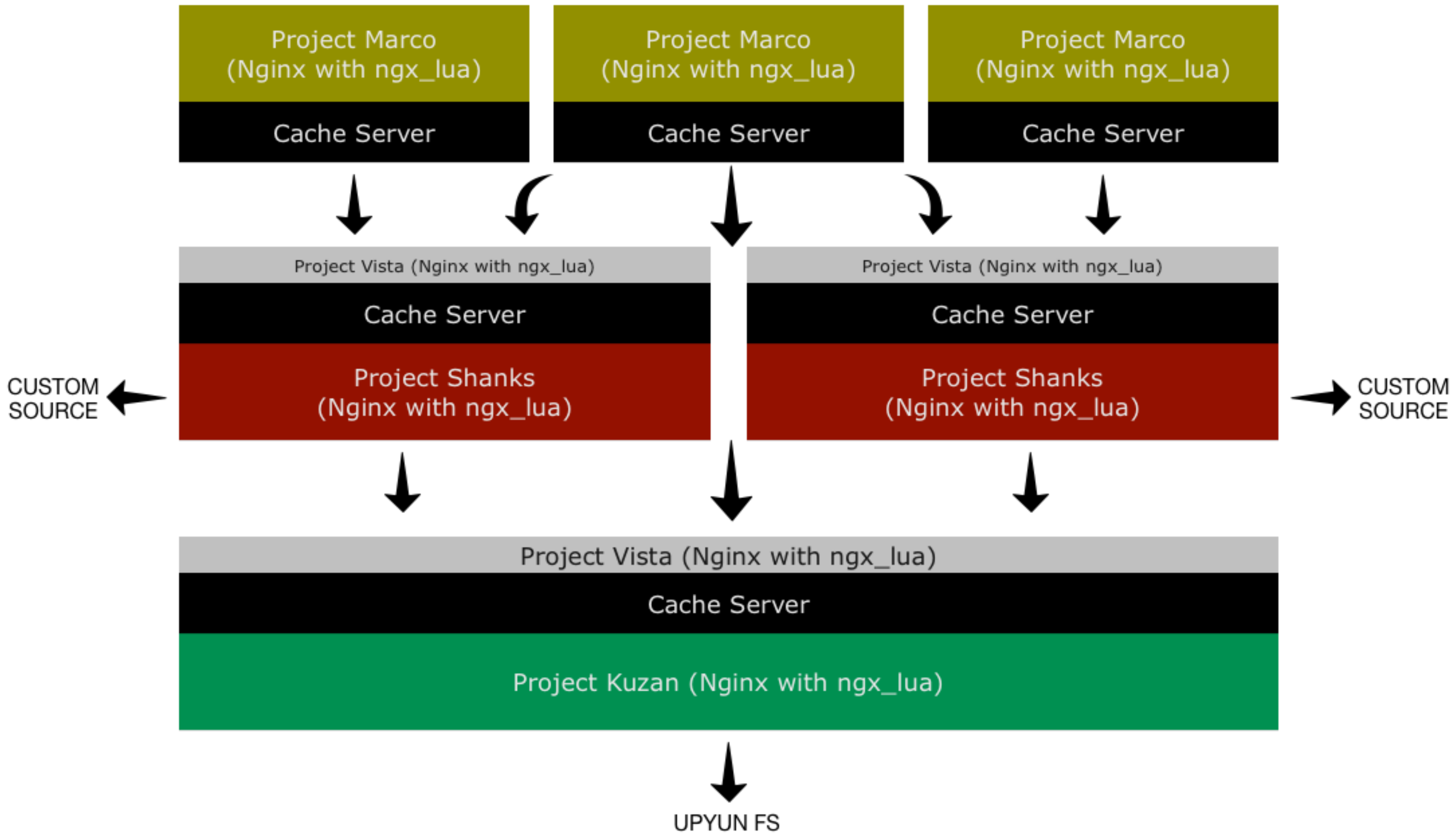
make test

```
Makefile

test:
     util/lua-releng
     py.test tests/test_marco.py
```

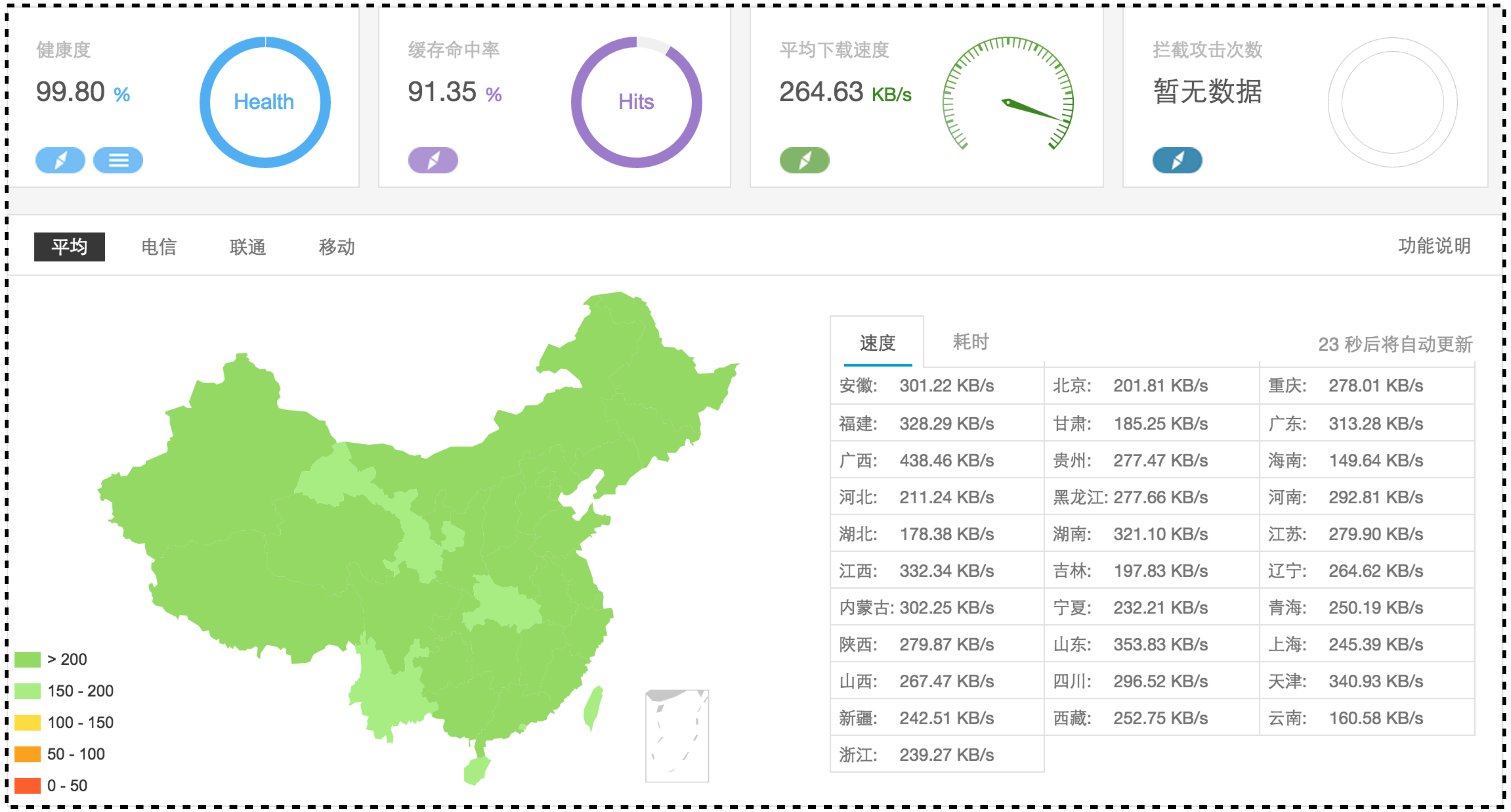
```
tests/test_marco.py

class TestMarco(unittest.TestCase):

    @no_error_log(["error"])
    @grep_error_log(level=["info"],
                    log_pattern="SSL_do_handshake[(][)] failed",
                    log_out=["SSL_do_handshake() failed"])
    def test_ssl_handler_no_certificate(self):
        fake_resp = self.curl_ssl(sni="fake.com", verbose=True)
        self.assertTrue("alert handshake failure" in fake_resp)
```

nginx.conf	service
server_name *.b0.upaiyun.com	Custom Domain Binding
valid_referers, **allow, deny**	**Custom Antileech Rules and Redirect:** ip, user-agent, referer, token etc.
expires 7d	**Custom Cache Control:** support specific URI rules etc.
ssl_certificate* **ssl_stapling***	Custom SSL
upstream { server 127.0.0.1 }	Custom CDN Origin: support multi-network routing etc.
max_fails=3 fail_timeout=30s **health_check (*)**	Custom Health Check Strategy: passive, active
round-robin, ip_hash, **hash (1.7.2+)**	Custom Load Balancing Strategy
rewrite	Custom URL rewrite
…	…

UPYUN CDN


Project Marco
(Nginx with ngx_lua)
Cache Server
Project Marco
(Nginx with ngx_lua)
Cache Server
Project Marco
(Nginx with ngx_lua)
Cache Server
Project Vista (Nginx with ngx_lua)
Cache Server
Project Shanks
(Nginx with ngx_lua)
Project Vista (Nginx with ngx_lua)
Cache Server
Project Shanks
(Nginx with ngx_lua)
CUSTOM SOURCE
CUSTOM SOURCE
Project Vista (Nginx with ngx_lua)
Cache Server
Project Kuzan (Nginx with ngx_lua)
UPYUN FS

130+ Edge Nodes



健康度 99.80 % Health

缓存命中率 91.35 % Hits

平均下载速度 264.63 KB/s

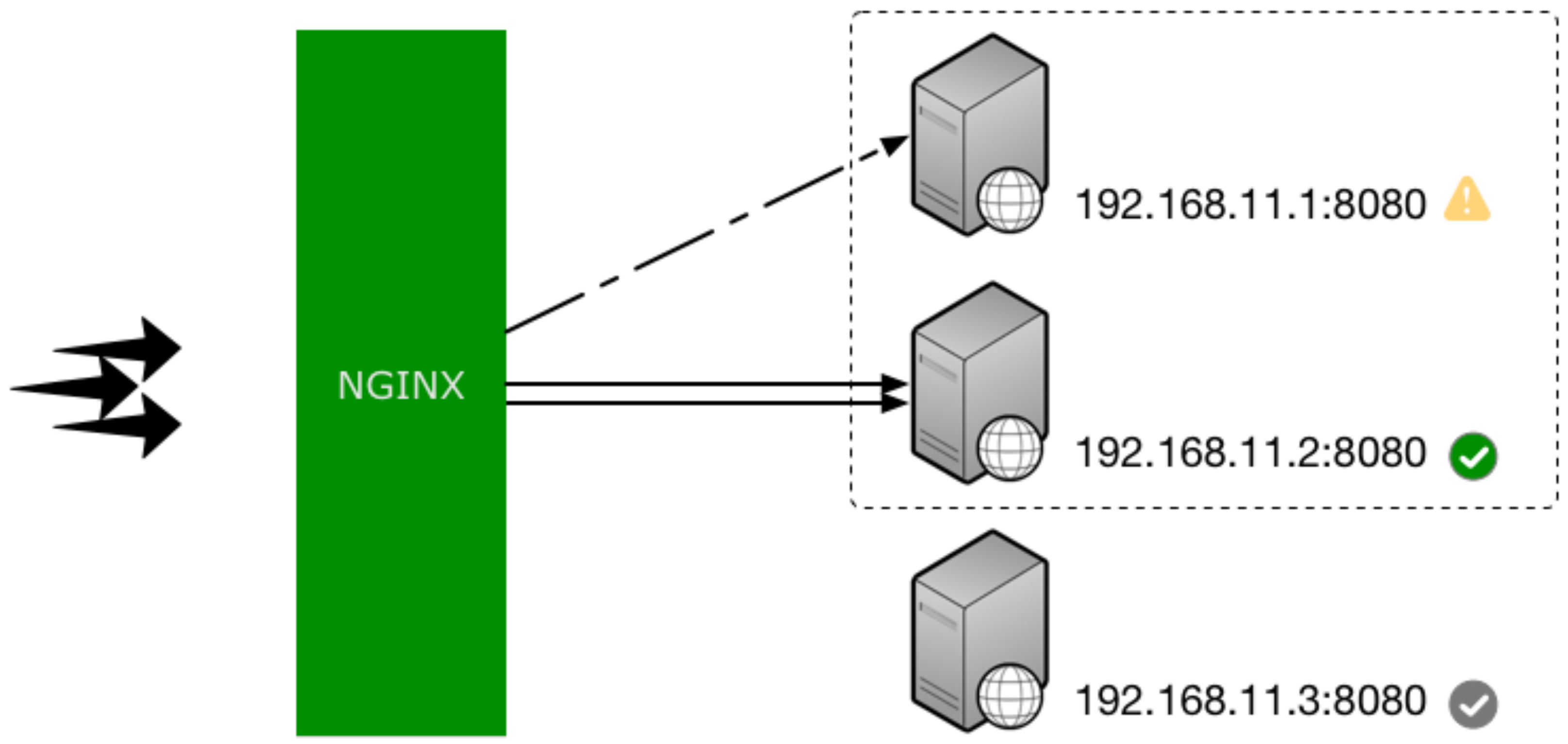
拦截攻击次数 暂无数据

平均 电信 联通 移动 功能说明

🟩 > 200
🟩 150 - 200
🟨 100 - 150
🟧 50 - 100
🟥 0 - 50

速度 耗时 23 秒后将自动更新

安徽: 301.22 KB/s	北京: 201.81 KB/s	重庆: 278.01 KB/s
福建: 328.29 KB/s	甘肃: 185.25 KB/s	广东: 313.28 KB/s
广西: 438.46 KB/s	贵州: 277.47 KB/s	海南: 149.64 KB/s
河北: 211.24 KB/s	黑龙江: 277.66 KB/s	河南: 292.81 KB/s
湖北: 178.38 KB/s	湖南: 321.10 KB/s	江苏: 279.90 KB/s
江西: 332.34 KB/s	吉林: 197.83 KB/s	辽宁: 264.62 KB/s
内蒙古: 302.25 KB/s	宁夏: 232.21 KB/s	青海: 250.19 KB/s
陕西: 279.87 KB/s	山东: 353.83 KB/s	上海: 245.39 KB/s
山西: 267.47 KB/s	四川: 296.52 KB/s	天津: 340.93 KB/s
新疆: 242.51 KB/s	西藏: 252.75 KB/s	云南: 160.58 KB/s
浙江: 239.27 KB/s		



```
upstream blog.upyun.com {
    server 192.168.11.1:8080 weight=1 max_fails=10 fail_timeout=30s;
    server 192.168.11.2:8080 weight=2 max_fails=10 fail_timeout=30s;

    server 192.168.11.3:8080 weight=1 max_fails=10 fail_timeout=30s backup;

    proxy_next_upstream error timeout http_500;
    proxy_next_upstream_tries 2;
}
```

Lua Upstream Configuration:

lua-resty-**checkups**

```lua
-- app/etc/config.lua

_M.global = {
    checkup_timer_interval = 5,
    checkup_timer_overtime = 60,
}

_M.api = {
    timeout = 2,
    typ = "general", -- http, redis, mysql etc.

    cluster = {
        {   -- level 1
            try = 2,
            servers = {
                { host = "192.168.11.1", port = 8080, weight = 1 },
                { host = "192.168.11.2", port = 8080, weight = 2 },
            }
        },
        {   -- level 2
            servers = {
                { host = "192.168.11.3", port = 8080, weight = 1 },
            }
        },
    },
}
```

cosocket

↓

http

redis

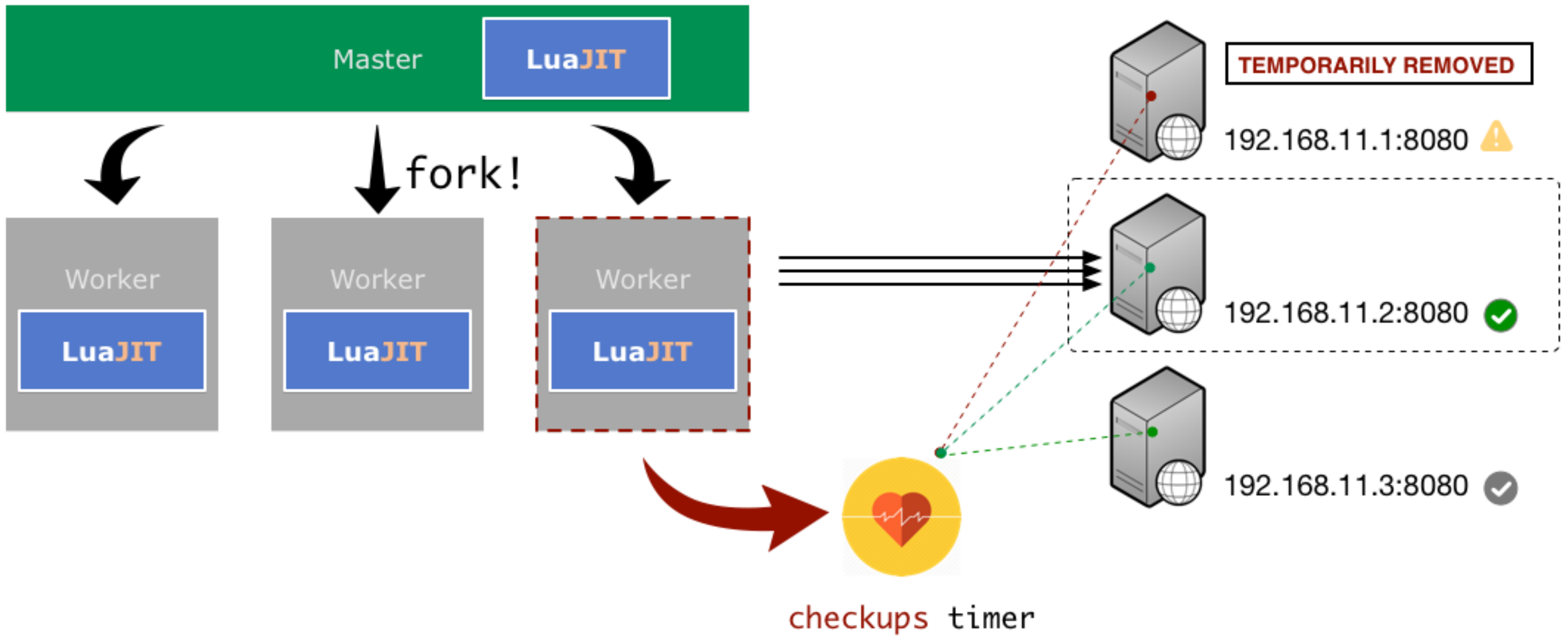
mysql

mem**cached**

...

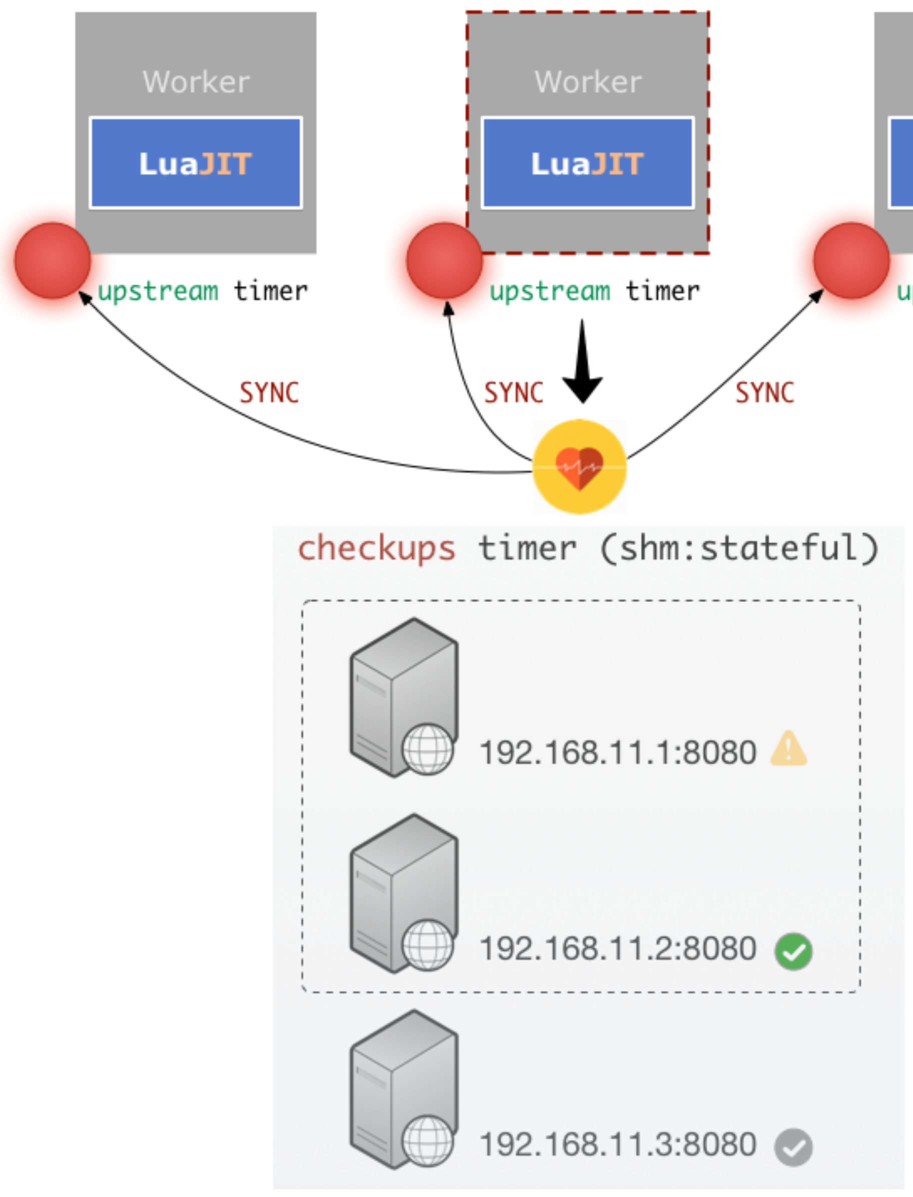
Lua Upstream Health Checks:

lua-resty-checkups



```
access_by_lua '
    local checkups = require "resty.checkups"

    -- only one timer is active among all the nginx workers
    checkups.create_checker()
';
```

Lua Upstream Health Checks: checkups with nginx.conf

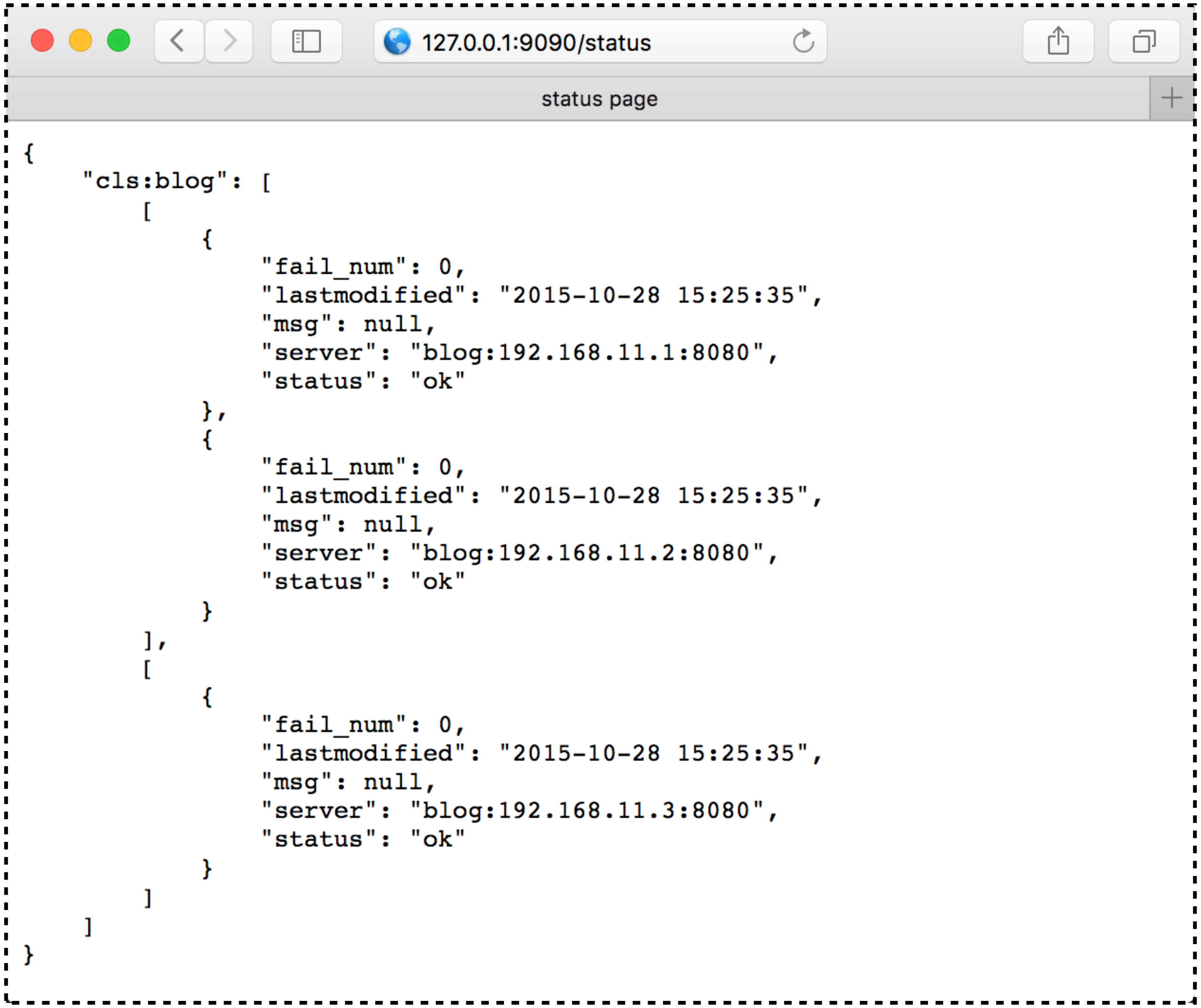


```
-- app/etc/config.lua

_M.global = {
    checkup_timer_interval = 5,
    checkup_timer_overtime = 60,

    ups_status_sync_enable = true,
    ups_status_timer_interval = 2,
}

_M.blog = {
    cluster = {
        {   -- level 1
            try = 2,
            upstream = "blog.upyun.com",
        },
        {   -- level 2
            upstream = "blog.upyun.com",
            upstream_only_backup = true,
        },
    },
}
```

lua-upstream-nginx-module

Lua Upstream Health Checks:

checkups with status page

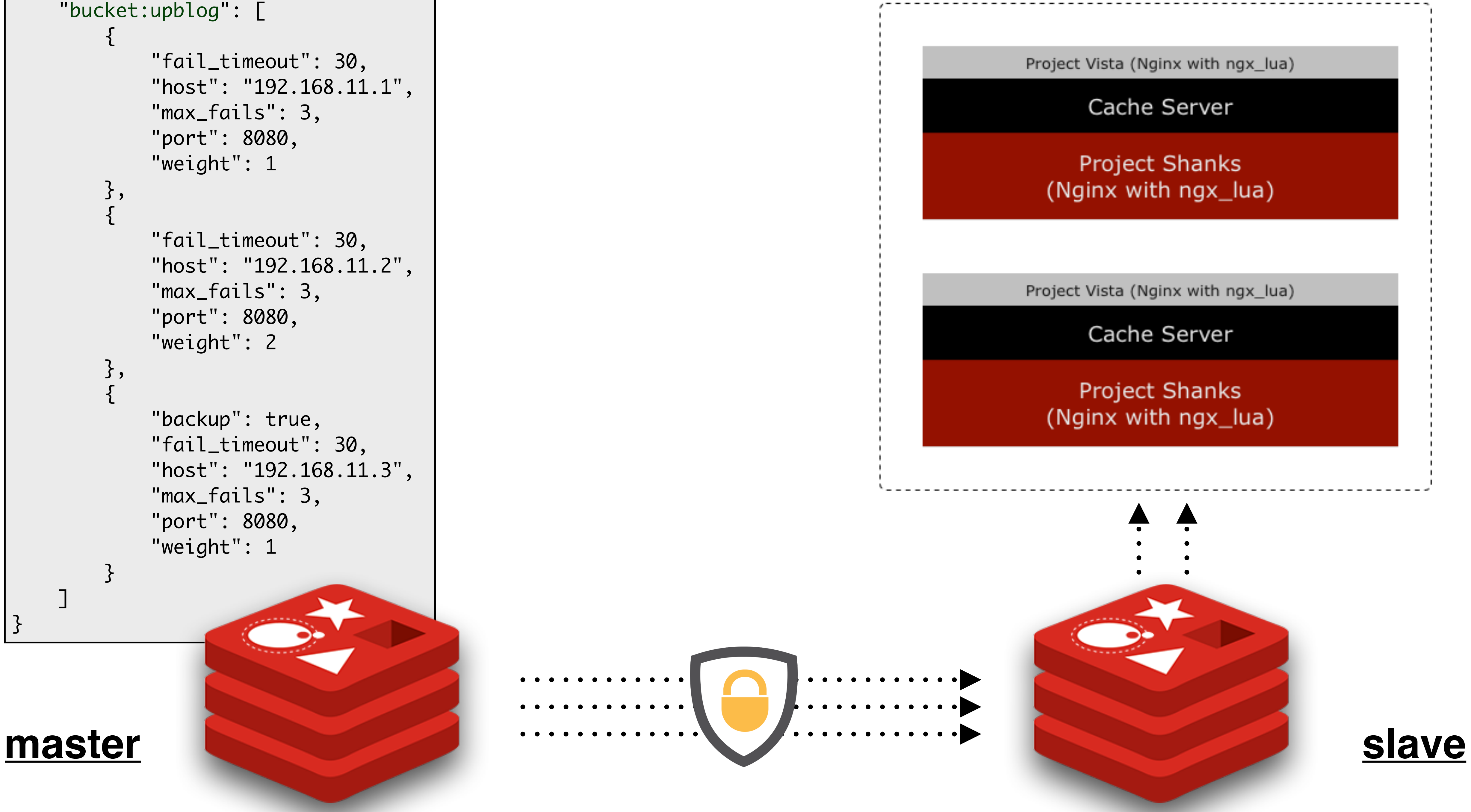


```
{
    "cls:blog": [
        [
            {
                "fail_num": 0,
                "lastmodified": "2015-10-28 15:25:35",
                "msg": null,
                "server": "blog:192.168.11.1:8080",
                "status": "ok"
            },
            {
                "fail_num": 0,
                "lastmodified": "2015-10-28 15:25:35",
                "msg": null,
                "server": "blog:192.168.11.2:8080",
                "status": "ok"
            }
        ],
        [
            {
                "fail_num": 0,
                "lastmodified": "2015-10-28 15:25:35",
                "msg": null,
                "server": "blog:192.168.11.3:8080",
                "status": "ok"
            }
        ]
    ]
}
```

Lua Upstream Dynamically:

Configure Everything as **JSON**

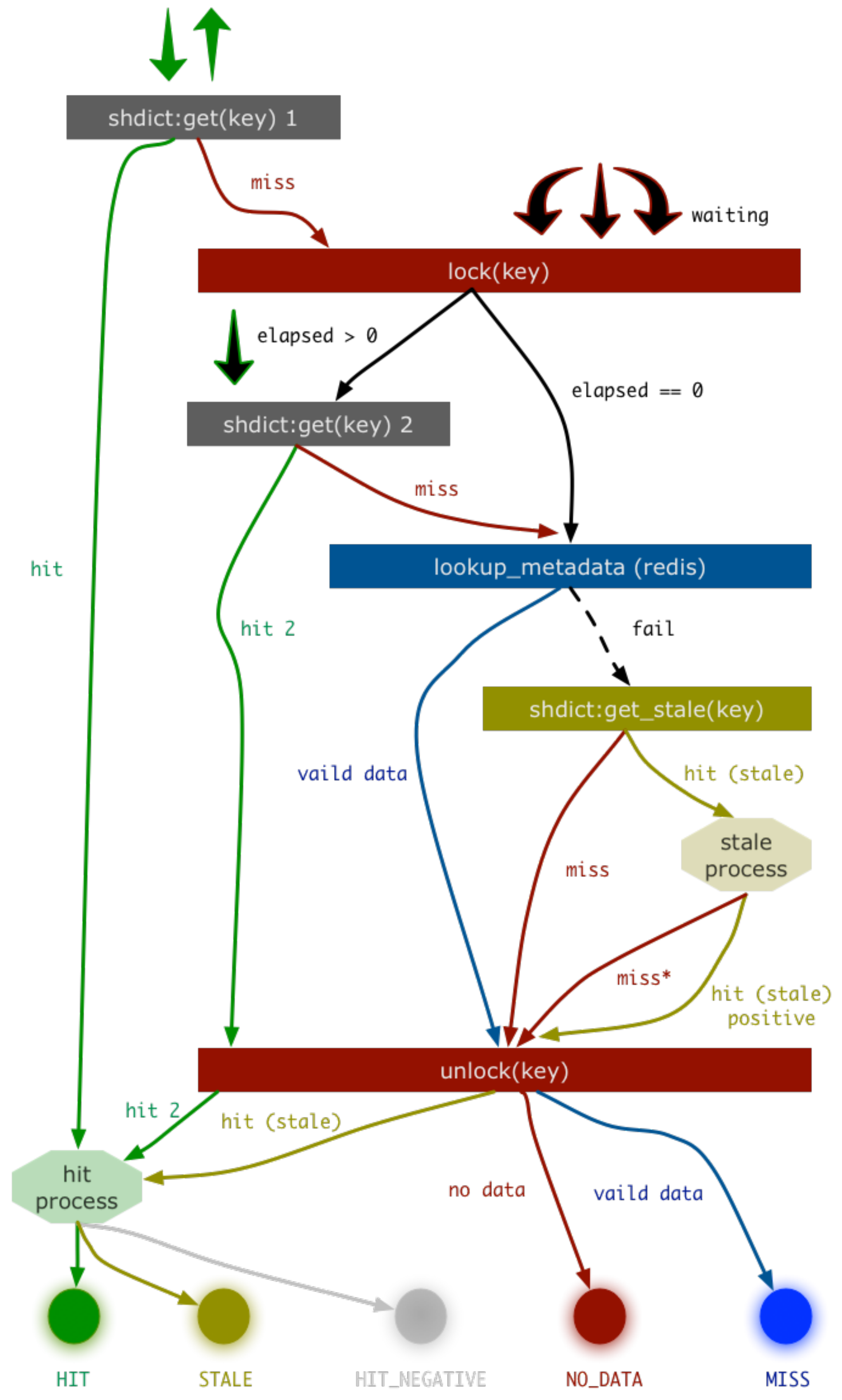
```
{
    "bucket:upblog": [
        {
            "fail_timeout": 30,
            "host": "192.168.11.1",
            "max_fails": 3,
            "port": 8080,
            "weight": 1
        },
        {
            "fail_timeout": 30,
            "host": "192.168.11.2",
            "max_fails": 3,
            "port": 8080,
            "weight": 2
        },
        {
            "backup": true,
            "fail_timeout": 30,
            "host": "192.168.11.3",
            "max_fails": 3,
            "port": 8080,
            "weight": 1
        }
    ]
}
```



Lua Metadata Cache:

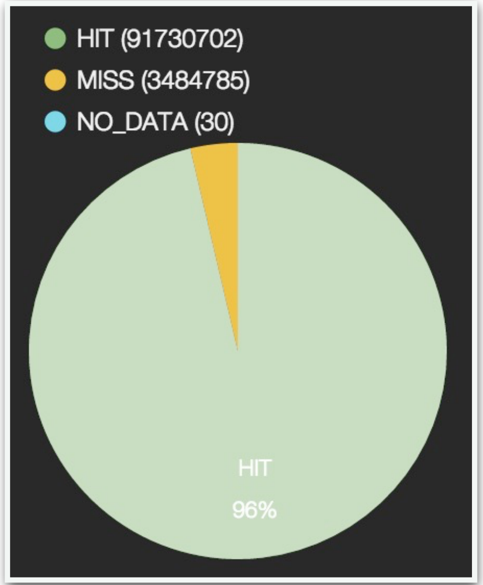
lua-resty-shcache

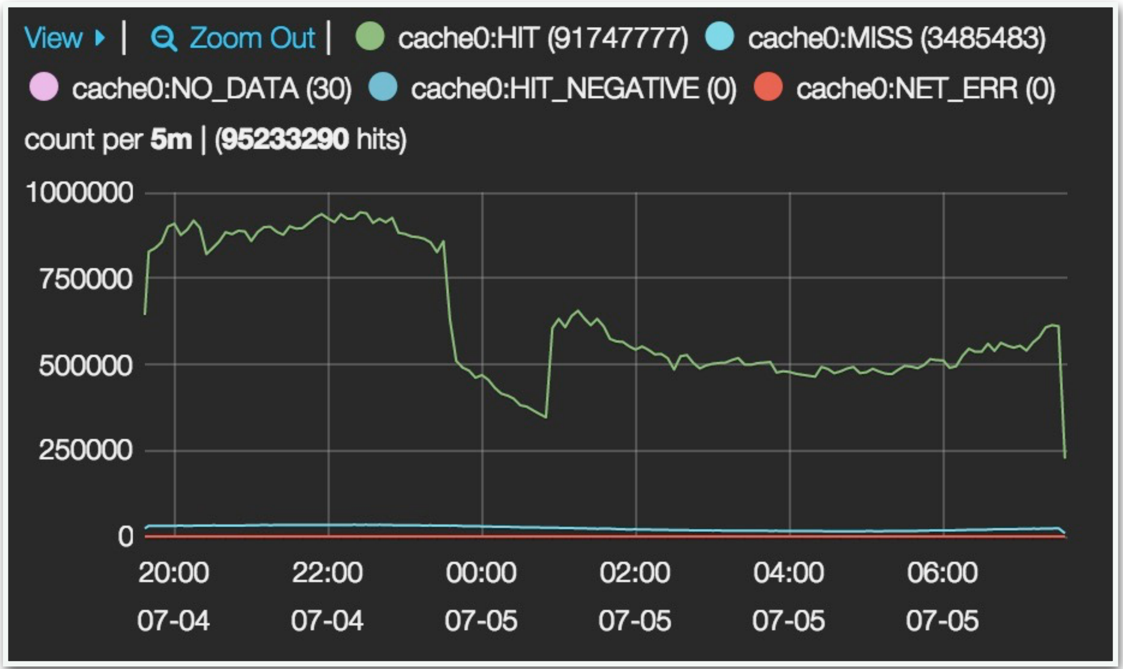
```
-- app/src/modules/metadata.lua

local shcache = require "resty.shcache"

function _M.get_metadata(bucket)
    local lookup_metadata = function ()
        -- fetch from redis
        return res
    end

    local cache_data = shcache:new(
        ngx.shared.metadata,
        { external_lookup = lookup_metadata,
          encode = cmsgpack.pack,
          decode = cmsgpack.unpack,
        },
        { positive_ttl = cache_positive_ttl,
          negative_ttl = cache_negative_ttl,
          name = "metadata",
        })

    -- local key = ...

    local data, _ = cache_data:load(key)
    if not data then
        return
    end

    return data
end
```



Lua Metadata Cache:

lua-resty-dbcache

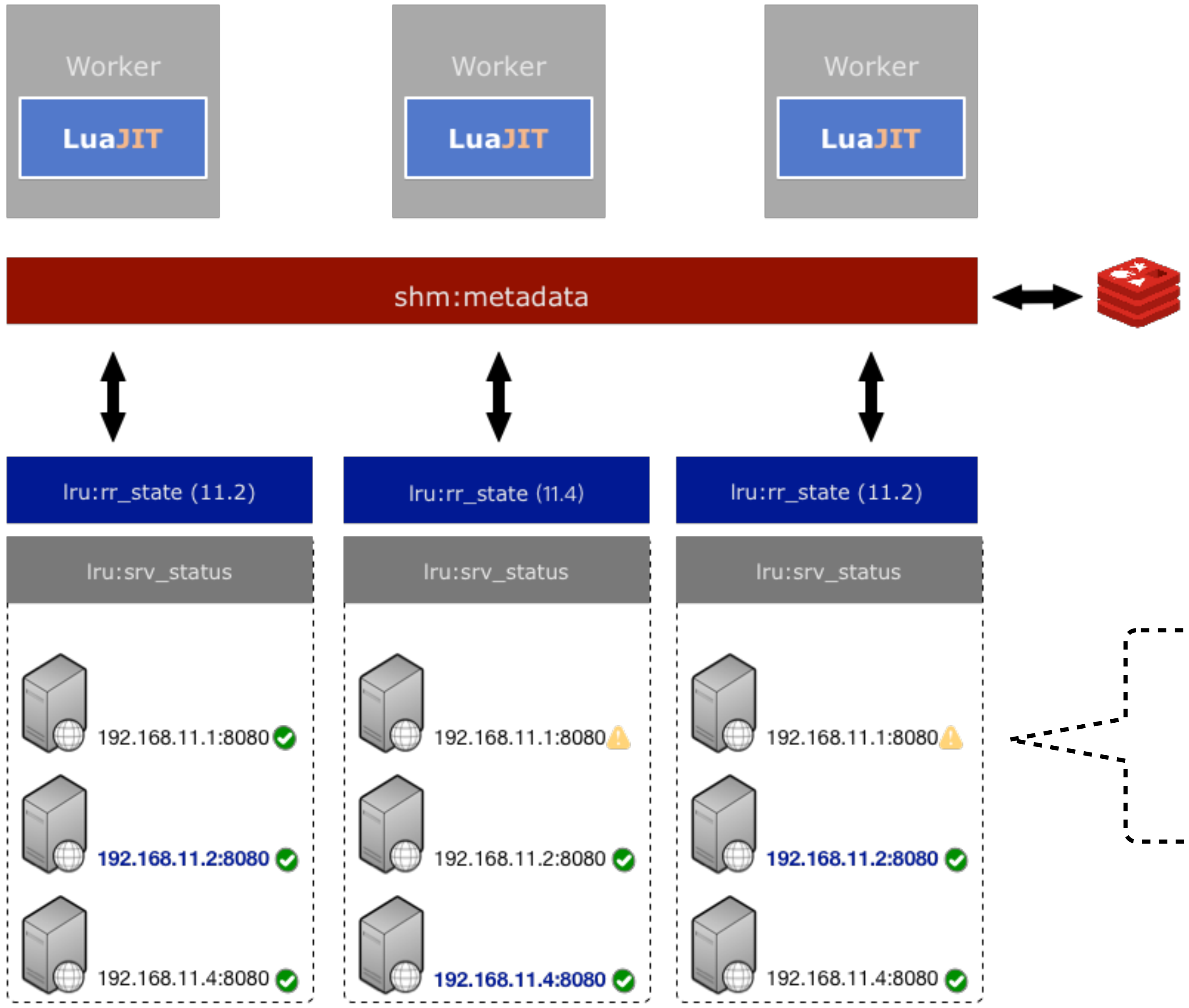




- ★ **HIT**
- ★ **STALE**
- ★ **HIT_NEGATIVE**
- ★ **NO_DATA**
- ★ **MISS**
- ★ **??? (NET_ERR)**

Lua Upstream Dynamically:

maintaining **internal** state



```
{
    "bucket:upblog": [ . . . ]
}
```

lua-resty-shcache

lua-resty-lrucache

```
max_fails=10
fail_timeout=30s
```

Lua Upstream **Load Balancing**:

round-robin with weight

```lua
function _M.reset_round_robin_state(cls)
    local rr = { index = 0, current_weight = 0 }
    rr.gcd, rr.max_weight, rr.weight_sum = _M.calc_gcd_weight(cls.servers)
    cls.rr = rr
end
```

```lua
cluster = {
    {
        servers = {
            { host = "127.0.0.1", port = 12351, weight = 1 },
            { host = "127.0.0.1", port = 12352, weight = 4 },
            { host = "127.0.0.1", port = 12353, weight = 3 },
            { host = "127.0.0.1", port = 12355, weight = 6 },
        }
    }
}
```

rr.index = 0
rr.current_weight = 0
rr.gcd = 1
rr.max_weight = 6
rr.weight_sum = 14

Lua Upstream **Load Balancing**:

round-robin with weight

local try_servers_by_round_robin = function(cls, verify_server_status, callback)

```
local bad_servers = {}

for i = 1, #cls.servers, 1 do
    local srv, index, err = _M.select_round_robin_server(cls, verify_server_status, bad_servers)
    if not srv then
        return nil, err
    else
        local res, _ = callback(srv)

        if res then
            if srv.effective_weight ~= srv.weight then
                srv.effective_weight = srv.weight
                _M.reset_round_robin_state(cls)
            end
            return res
        end

        if srv.effective_weight > 1 then
            srv.effective_weight = floor(sqrt(srv.effective_weight))
            _M.reset_round_robin_state(cls)
        end

        bad_servers[index] = true
    end
end
```

Lua Upstream **Load Balancing**:

round-robin with weight

function _M.select_round_robin_server(cls, verify_server_status, bad_servers)

```
local rr = cls.rr
local servers = cls.servers

local index = rr.index
local current_weight = rr.current_weight
local gcd = rr.gcd
local max_weight = rr.max_weight
local weight_sum = rr.weight_sum

local failed = 1
```

```
repeat

until failed > weight_sum
```

TALK IS CHEAP

Lua Upstream **Load Balancing**:

round-robin with weight

```
index = index % #servers + 1
if index == 1 then
    current_weight = current_weight - gcd
    if current_weight <= 0 then current_weight = max_weight end
end

local srv = servers[index]
if srv.effective_weight >= current_weight then
    cls.rr.index, cls.rr.current_weight = index, current_weight
    if not bad_servers[index] then
        if verify_server_status then
            if verify_server_status(srv) then
                return srv, index
            else
                if srv.effective_weight > 1 then
                    srv.effective_weight, index, current_weight, failed_count = 1, 0, 0, 0
                    _M.reset_round_robin_state(cls)
                    gcd, max_weight, weight_sum = cls.rr.gcd, cls.rr.max_weight, cls.rr.weight_sum
                end
                failed = failed + 1
            end
        else
            return srv, index
        end
    else
        failed = failed + 1
    end
end
```

repeat . . . . until failed > weight_sum

Lua Upstream **Load Balancing**:

round-robin with weight

```
local verify_server_status = function(srv)
    local peer_key = _gen_key(srv)

    local peer_status = cjson.decode(state:get(PEER_STATUS_PREFIX .. peer_key))
    if peer_status == nil or peer_status.status ~= _M.STATUS_ERR then
        return true
    end

    return
end
```

- ★ **STATUS_OK = 0**
- ★ **STATUS_UNSTABLE = 1**
- ★ **STATUS_ERR = 2**

Lua Upstream **Load Balancing**:

=== TEST 1: round-robin single level

```
--- http_config eval
"$::HttpConfig" . "$::InitConfig"
--- config
    location = /t {
        content_by_lua '
            local checkups = require "resty.checkups"
            checkups.create_checker()
            ngx.sleep(2)
            local dict = {
                [12351] = "A",
                [12352] = "B",
                [12353] = "C",
                [12355] = "E",
            }
            local cb_ok = function(srv)
                ngx.print(dict[srv.port])
                return 1
            end

            for i = 1, 30, 1 do
                local ok, err = checkups.ready_ok("single_level", cb_ok)
                if err then
                    ngx.say(err)
                end
            end
        ';
    }
--- request
GET /t
--- response_body: EEBEBCEBCEABCEEEBEBCEBCEABCEEE
```

```
_M.single_level = {
    cluster = {
        {
            servers = {
                { host = "127.0.0.1", port = 12351, weight = 1 },
                { host = "127.0.0.1", port = 12352, weight = 4 },
                { host = "127.0.0.1", port = 12353, weight = 3 },
                { host = "127.0.0.1", port = 12355, weight = 6 },
            }
        }
    }
}
```

EEBEBCEBCEABCE

......

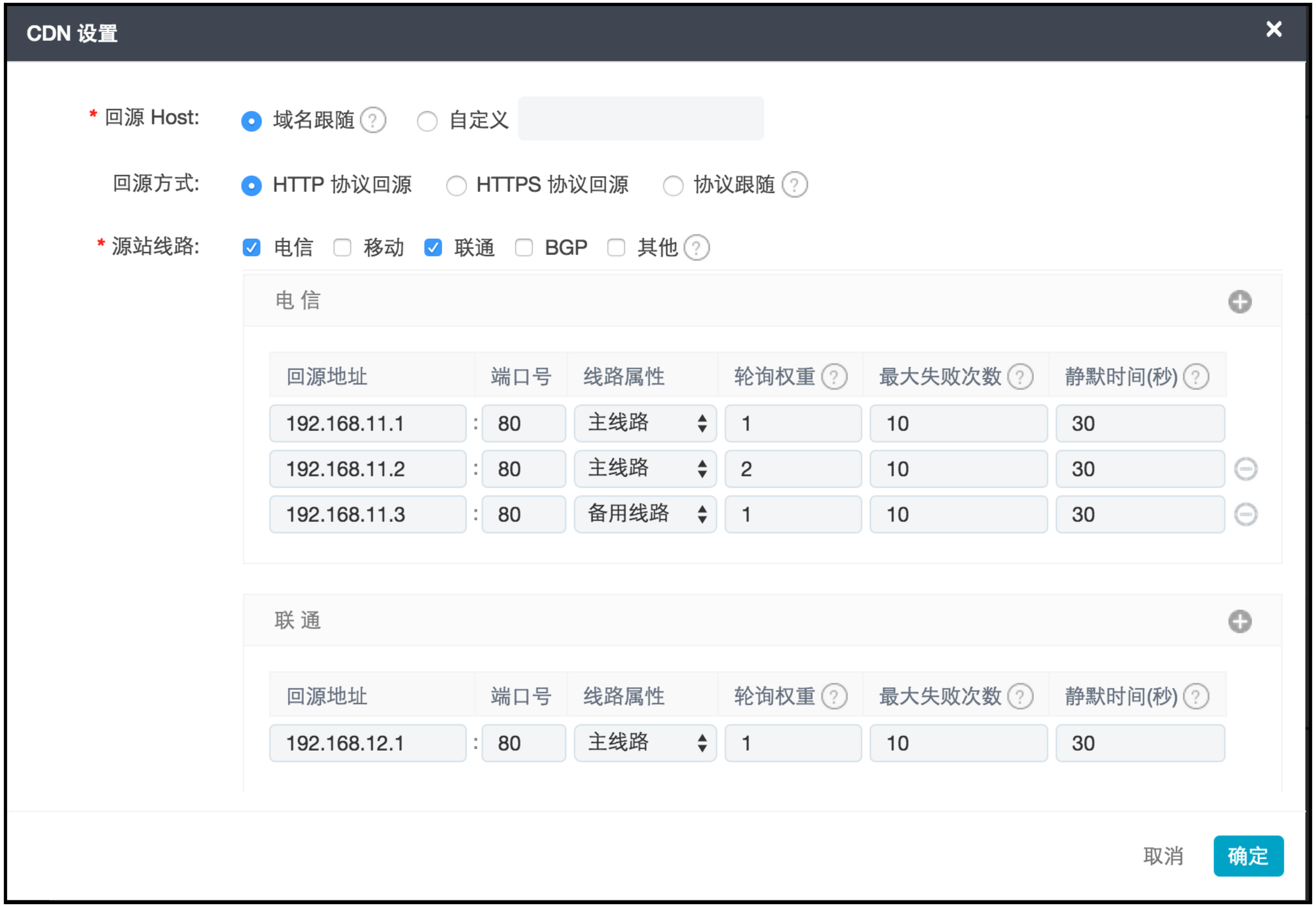
Lua Upstream **Load Balancing**:

consistent-hash **and more**

- ★ **try_servers_by_round_robin**
- ★ **try_cluster_by_round_robin**

```
cluster = {
    {
        servers = {
            { host = "127.0.0.1", port = 12351, weight = 1 },
            { host = "127.0.0.1", port = 12352, weight = 4 },
            { host = "127.0.0.1", port = 12353, weight = 3 },
            { host = "127.0.0.1", port = 12355, weight = 6 },
        }
    },
    {
        servers = {
            { host = "127.0.0.1", port = 12354, weight = 1 },
            { host = "127.0.0.1", port = 12356, weight = 2 },
        }
    }
}
```

- ★ **try_servers_by_consistent_hash**
- ★ **try_cluster_by_consistent_hash**



CDN 设置

* 回源 Host: [✓ 域名跟随] 自定义

回源方式: [✓ HTTP 协议回源] HTTPS 协议回源 协议跟随

* 源站线路: ✓ 电信 移动 ✓ 联通 BGP 其他

电 信

回源地址	端口号	线路属性	轮询权重	最大失败次数	静默时间(秒)
192.168.11.1	80	主线路	1	10	30
192.168.11.2	80	主线路	2	10	30
192.168.11.3	80	备用线路	1	10	30

联 通

回源地址	端口号	线路属性	轮询权重	最大失败次数	静默时间(秒)
192.168.12.1	80	主线路	1	10	30

取消 确定

tianchaijz:

"$WHEN($MATCH($_URI, '^/foo/.*'))$ADD_REQ_HEADER(X-Foo, bar)"

Marco: **I GOT IT !**

Edge Server

Lua Custom URL **rewrite**:

lua-resty-rewrite | variables

$_METHOD **$_SCHEME**

$_HOST $_POST_name $_SYM_sym

$_HOST_n $_HEADER_name

$_COOKIE_name $_URI

$_GET_name **$_RANDOM_n** $_RANDOM

$_QUERY

Lua Custom URL **rewrite**:

lua-resty-rewrite | functions

$ENCODE_BASE64(E)

$ALL(E1, E2, ...)

$UPPER(E)

$DECODE_BASE64(E)

$ANY(E1, E2, ...)

$LOWER(E)

$WHEN(E1, E2, ...)

$SUB(E1, from, to)

$PCALL(E)

$MATCH(E1, E2)

$GT(E1, E2)

$ADD_REQ_HEADER(E1, E2)

$GE(E1, E2)

$DEL_REQ_HEADER(E1)

$EQ(E1, E2)

$ADD_RSP_HEADER(E1, E2)

Lua Custom URL **rewrite**:

lua-resty-rewrite | break

```
rewrite /download/(.*)/(.*)   /$1/$2.mp3?_session=$_COOKIE_id?
rewrite /download/(.*)/(.*)   /$1/$2.mp3?user=$_HOST_1 break
. . .
```

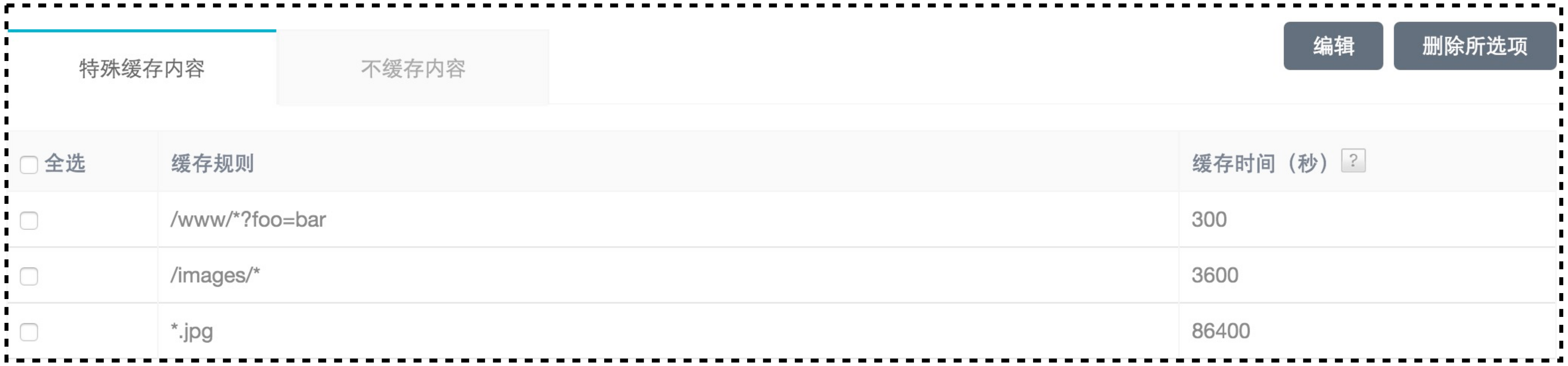
See More: https://github.com/upyun/docs/issues/5 http://io.upyun.com/2015/03/09/hello-world/?foo=bar

[scheme] [host] [path] [query]

Lua Custom **Cache-Control**:

Using specific URI rules

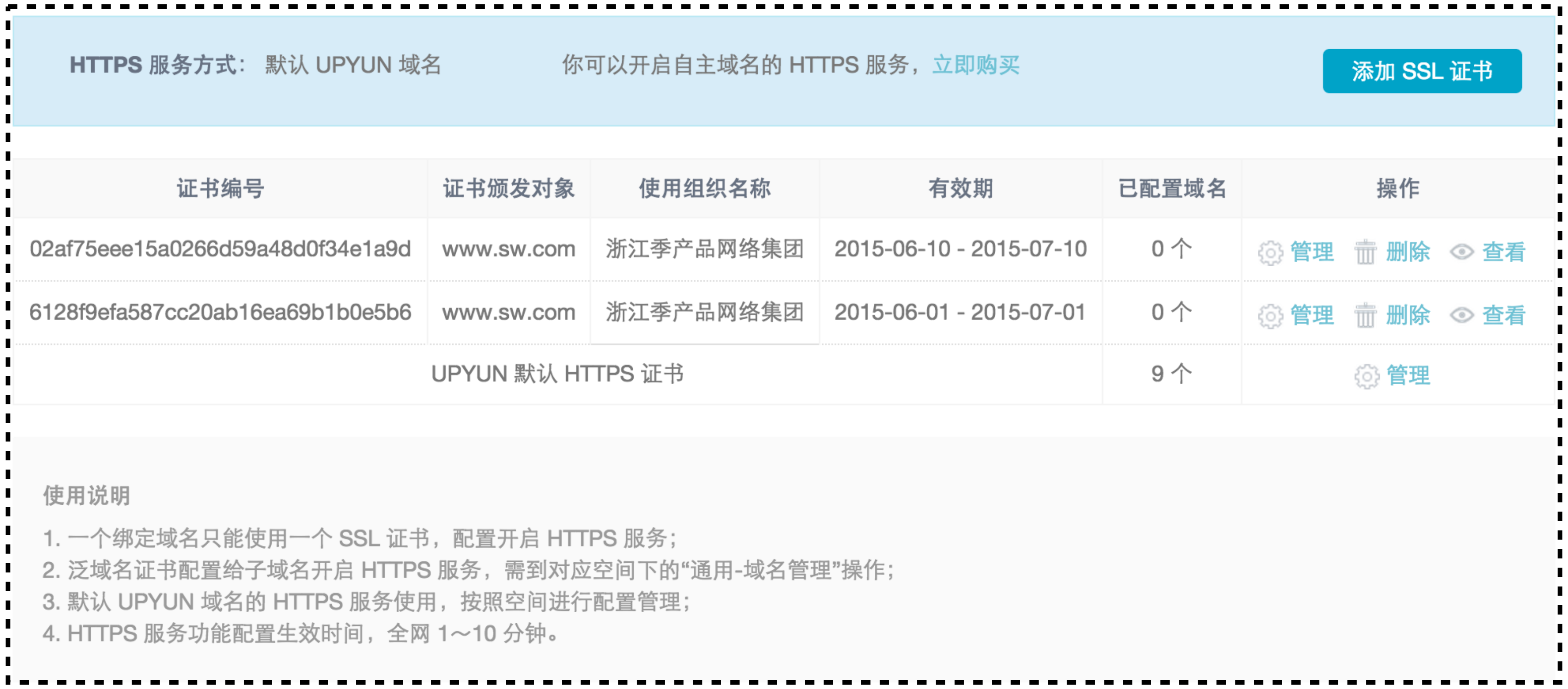
```
location ^~ /www/ {
    if ($query_string ~* "foo=bar") {
        expires 300s;
    }
}

location ^~ /images/ {
    expires 1h;
}

location ~* \.jpg$ {
    expires 1d;
}
```



全选	缓存规则	缓存时间（秒）
☐	/www/*?foo=bar	300
☐	/images/*	3600
☐	*.jpg	86400

Lua Custom SSL:

Certificates Load & OCSP stapling


HTTPS 服务方式： 默认 UPYUN 域名　　你可以开启自主域名的 HTTPS 服务，立即购买　　添加 SSL 证书

证书编号	证书颁发对象	使用组织名称	有效期	已配置域名	操作
02af75eee15a0266d59a48d0f34e1a9d	www.sw.com	浙江季产品网络集团	2015-06-10 - 2015-07-10	0 个	管理 删除 查看
6128f9efa587cc20ab16ea69b1b0e5b6	www.sw.com	浙江季产品网络集团	2015-06-01 - 2015-07-01	0 个	管理 删除 查看
UPYUN 默认 HTTPS 证书				9 个	管理

使用说明

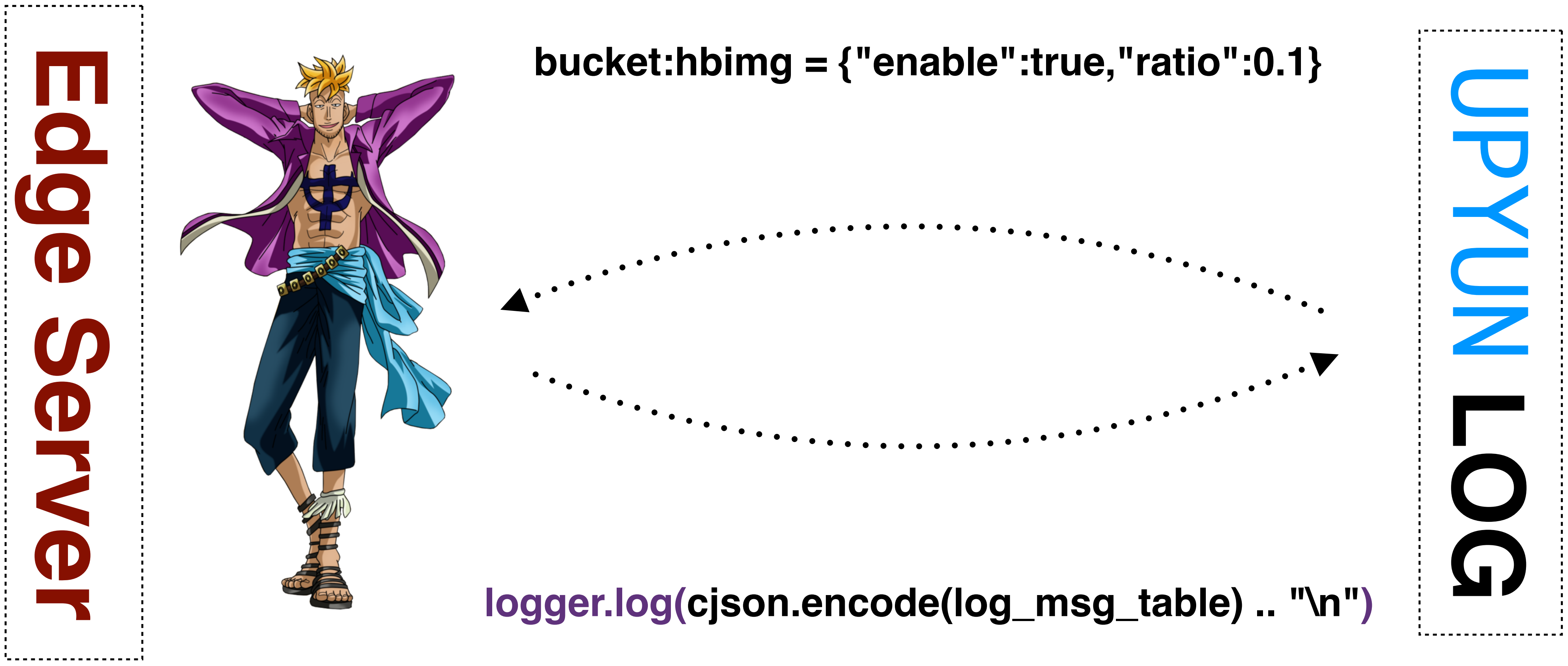
1. 一个绑定域名只能使用一个 SSL 证书，配置开启 HTTPS 服务；
2. 泛域名证书配置给子域名开启 HTTPS 服务，需到对应空间下的“通用-域名管理”操作；
3. 默认 UPYUN 域名的 HTTPS 服务使用，按照空间进行配置管理；
4. HTTPS 服务功能配置生效时间，全网 1～10 分钟。

```
server {
    listen 443 ssl;
    server_name upyun.com;

    ssl_certificate       upyun.com.pem;
    ssl_certificate_key   upyun.com.key;

    ssl_stapling on;
    ssl_stapling_verify on;
    ssl_trusted_certificate /etc/ssl/private/ca-certs.pem;
}
```

Lua Custom **Logging**:

lua-resty-logger-socket

```
log_format combined '$remote_addr - $remote_user [$time_local] '
                    '"$request" $status $body_bytes_sent '
                    '"$http_referer" "$http_user_agent"';

server {
    access_log /path/to/access.log combined buffer=4096;
    . . .
}
```

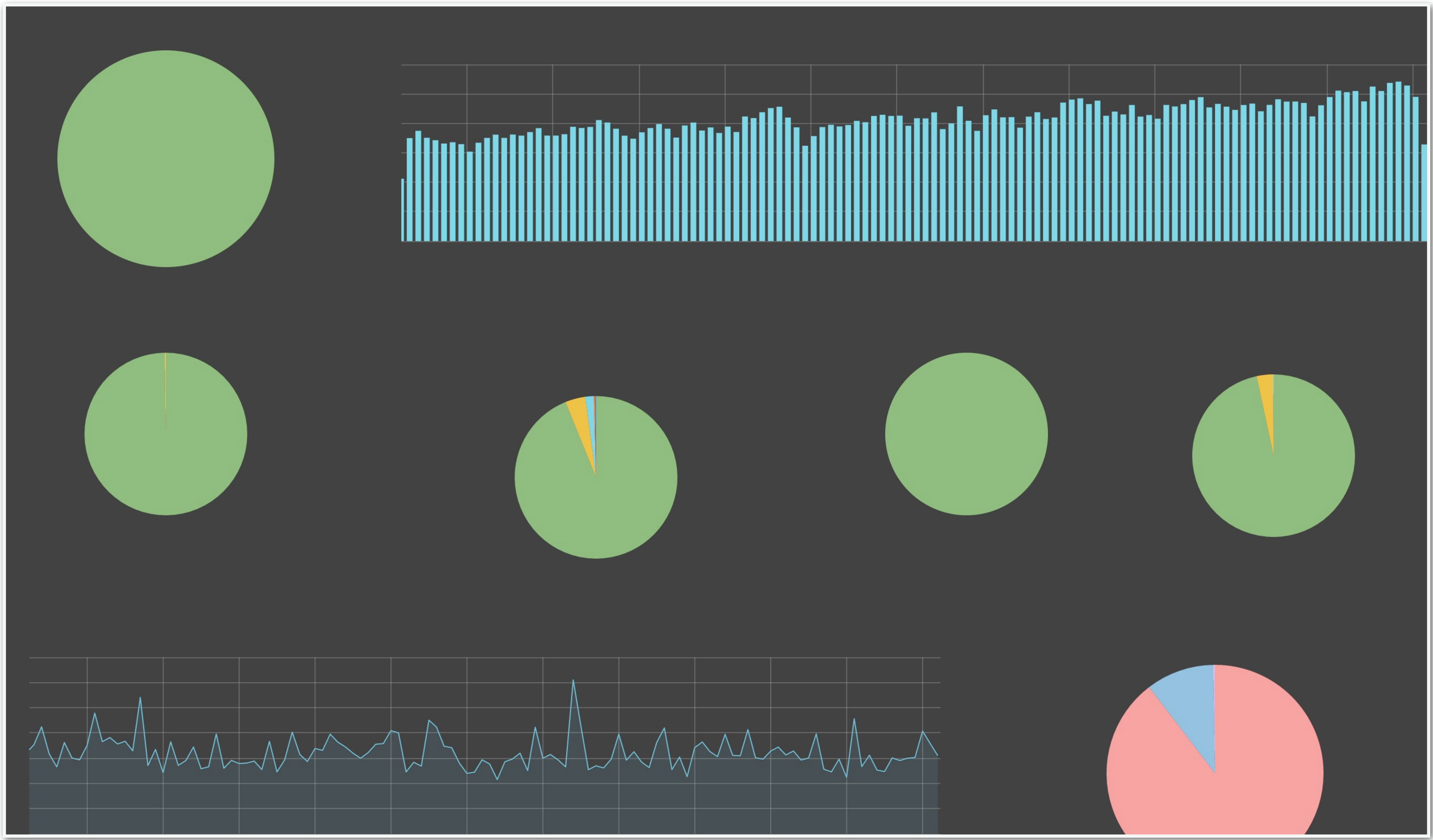


bucket:hbimg = {"enable":true,"ratio":0.1}

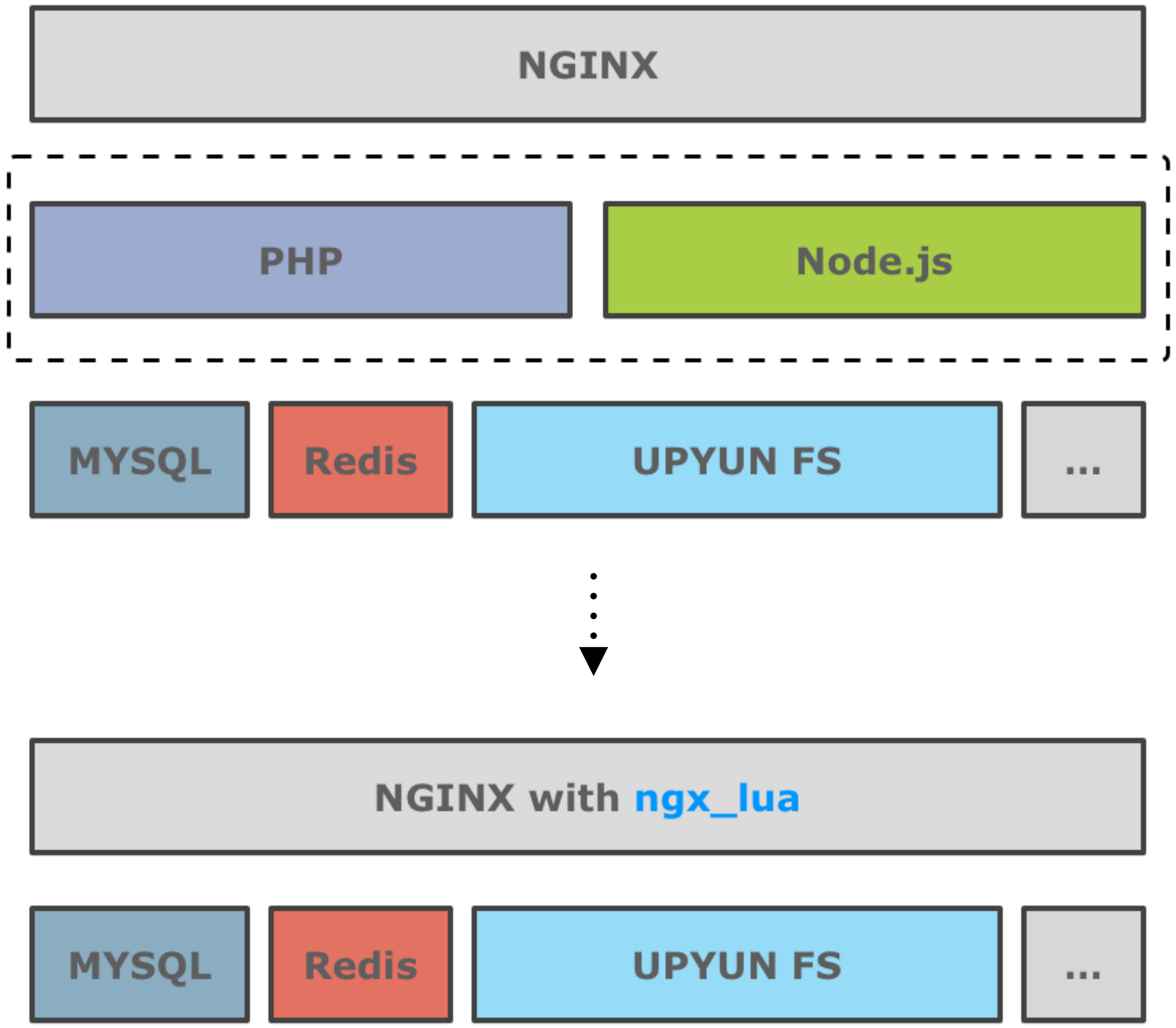
logger.log(cjson.encode(log_msg_table) .. "\n")

UPYUN LOG Platform:

HAProxy + Heka + Kafka + Elasticsearch + Kibana

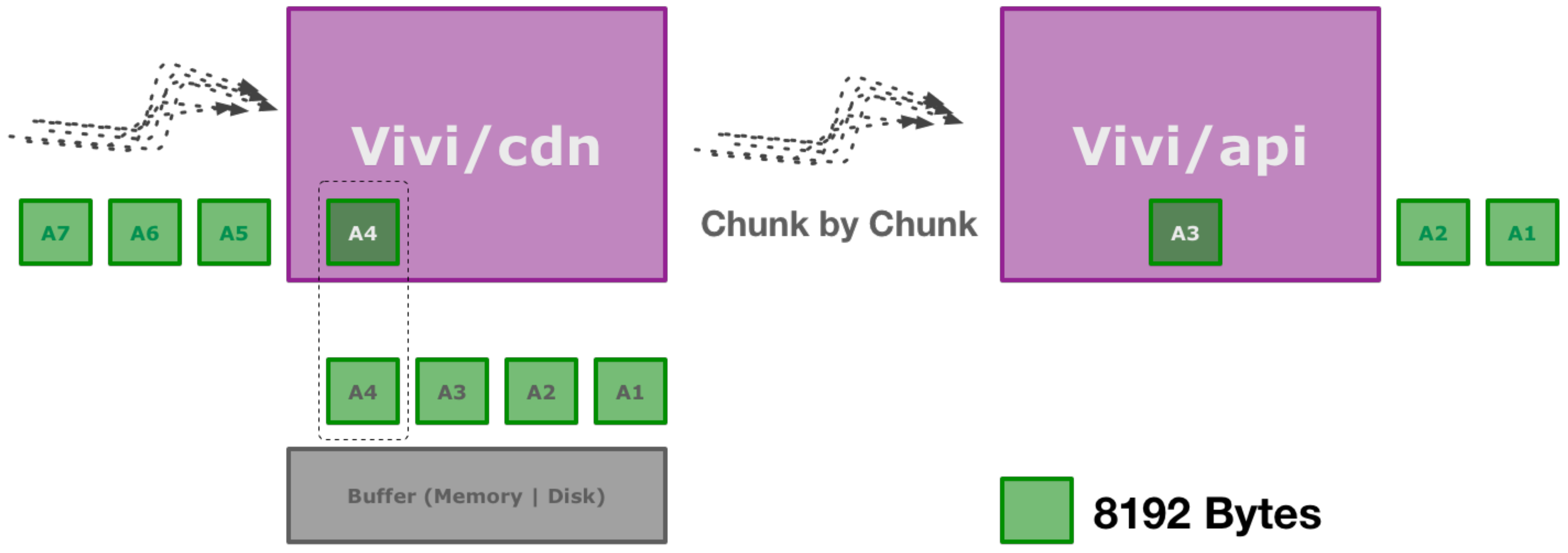


UPYUN API


NGINX
PHP
Node.js
MYSQL
Redis
UPYUN FS
...
NGINX with ngx_lua
MYSQL
Redis
UPYUN FS
...

```
location /upload {
    proxy_request_buffering off;
    . . .
}
```

Lua Streaming Upload



```
ngx.req.init_body()
ngx.req.append_body(chunk)
ngx.req.finish_body()
```

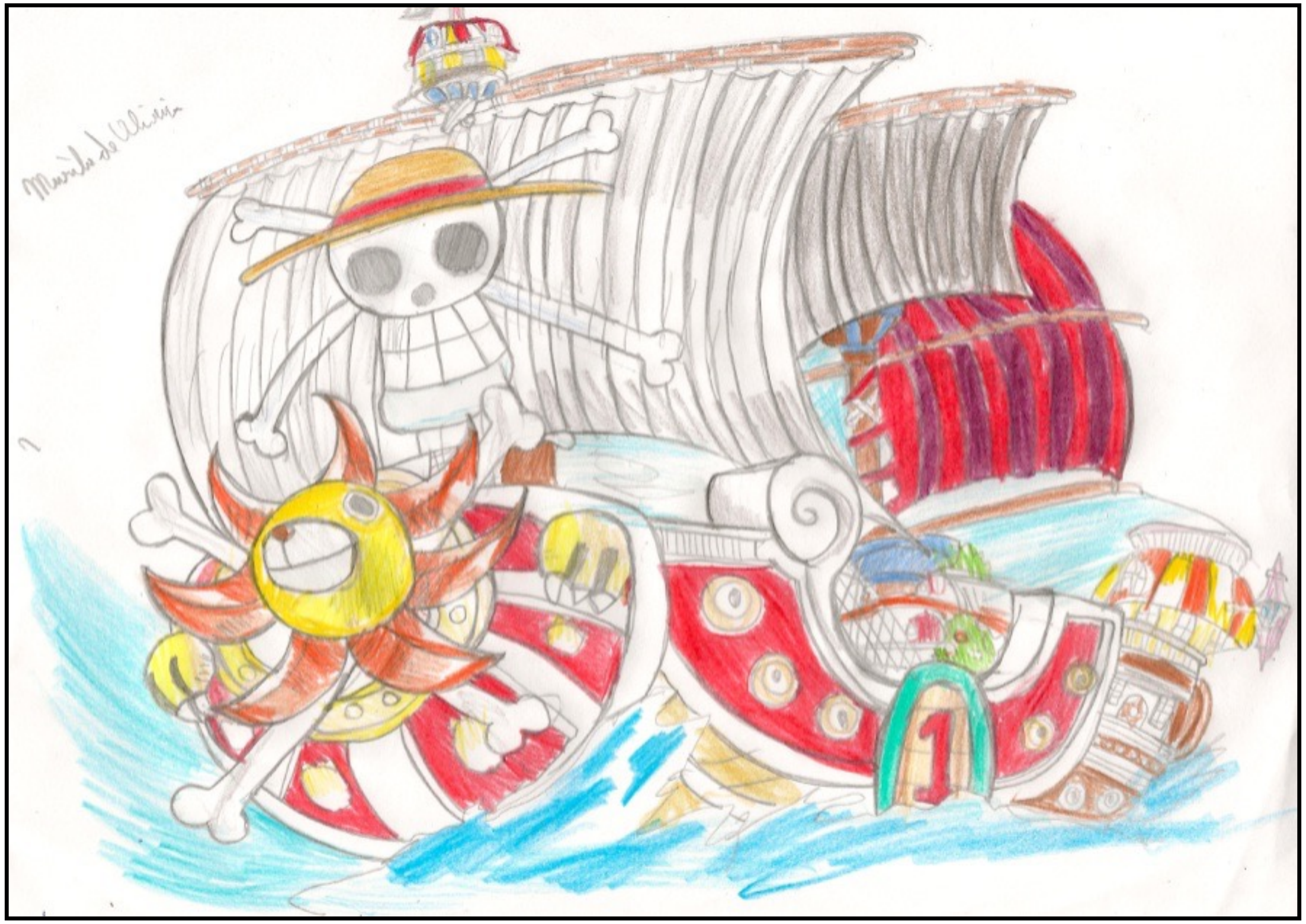
Lua CDN

Lua WAF

Lua SSL

Lua API

Join our team



Nginx ngx_lua agentzh Lua

blog.cloudflare.com Openresty **Lua**JIT Github

Maxim Dounin Igor Sysoev chaoslawful

https://groups.google.com/forum/#!forum/OpenResty

Ansible Michael Pall Open source

Thanks

…

Q & A