Openresty 企业网关应用

--Kevin

Agenda

- About me
- Initial requirement: RBAC for downstream
- Event bus / Message bus / Cache bus / Communications
- EmmyLua introduction & HTTP RPC reference
- Additional cookie
- Q & A

About me

袁开 -- Kevin

13年互联网老兵

资深码农，深耕互联网视频领域

就职于华数传媒网络有限公司，新媒体事业群总架构师

感谢 Openresty社区, 感谢春哥, 给了我们如此优秀的开发平台

-- 不要怀疑 Openresty 不仅仅可以做调度, 做CDN, 做AB Testing,

做负载均衡, 做流量分发. 还可以做应用, 甚至作为核心身份认证系统

触达到每一次请求, 并且以极低的集成代价无缝接入到现有体系中.

我们花了6个月时间, 1个半工程师, 做了下面这个系统.

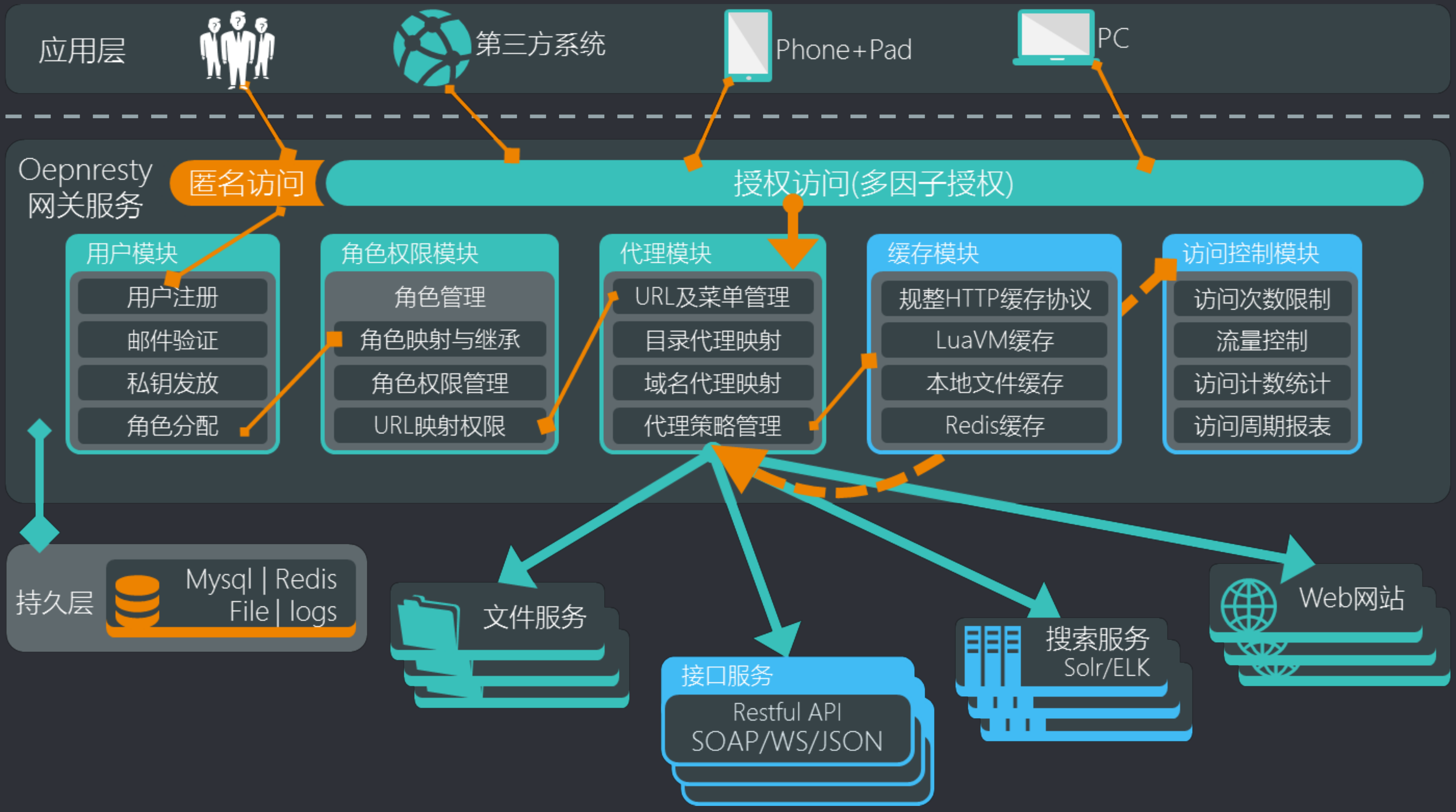
Initial requirement: RBAC for downstream

-- 改造老系统总是痛苦的事情， 但总得有人来做

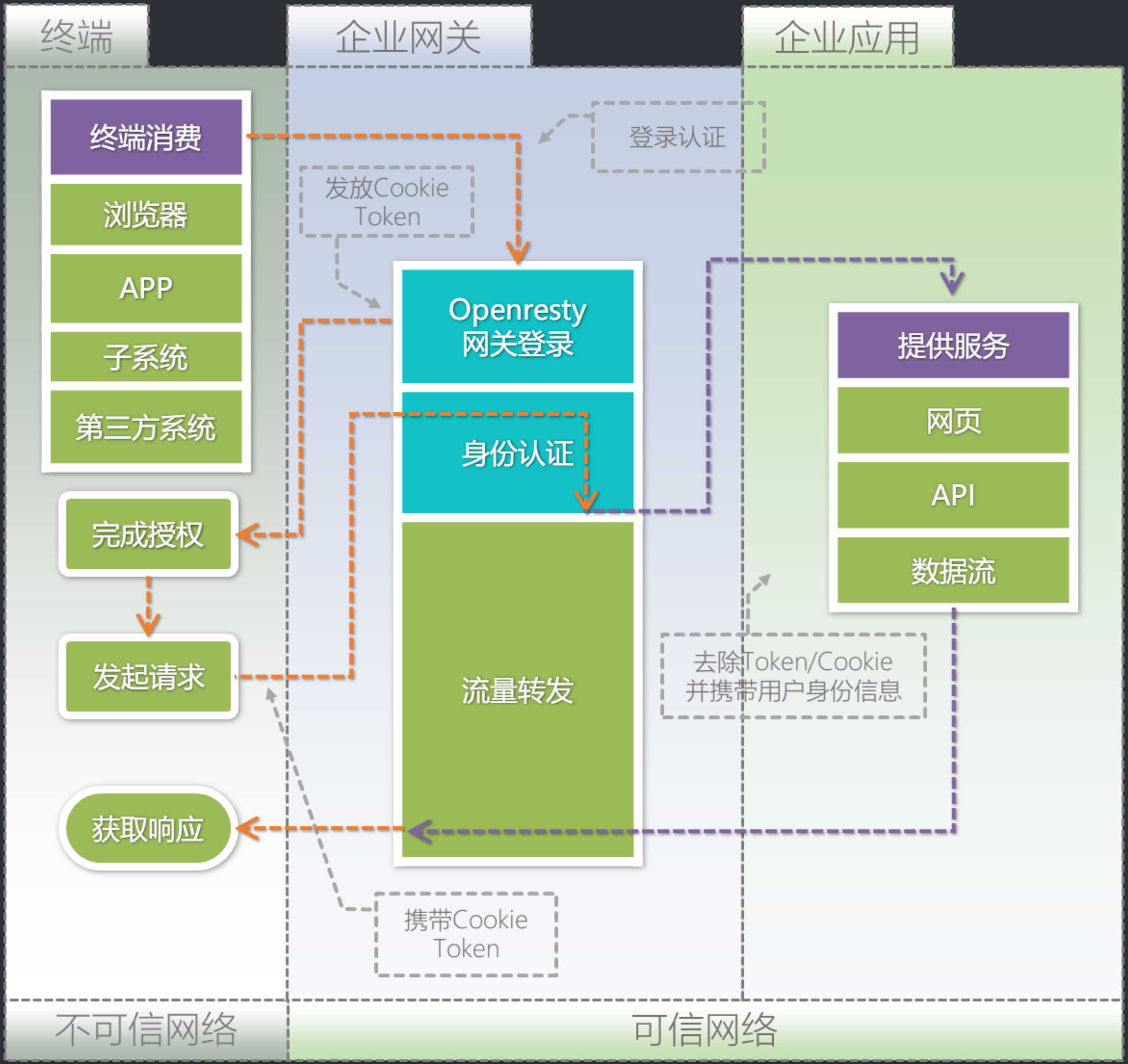
Initial requirement

要求:

- 提升安全级别，在线系统必须使用双因子认证
- 对后端文件分发做权限控制
- 对部分数据接口/文件接口做权限控制
- 对存量老系统做统一用户登录
- 不能影响现网用户使用，不能影响原有系统之间的相互接口调用。
- 能够针对散落在不同数据中心的服务同时做认证,并进行同步管理
- 如果可能，为老系统提供缓存服务。。。

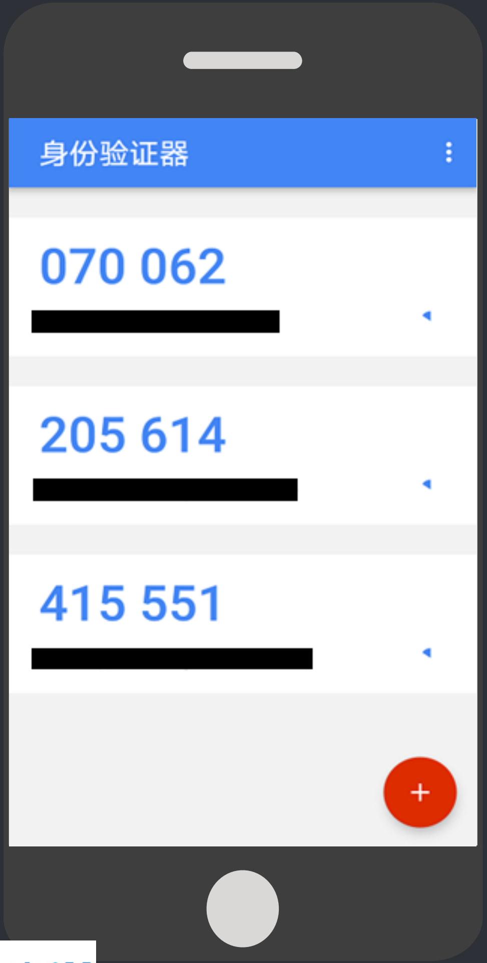

应用层
第三方系统
Phone+Pad
PC
Oepnresty
网关服务
匿名访问
授权访问(多因子授权)
用户模块
用户注册
邮件验证
私钥发放
角色分配
角色权限模块
角色管理
角色映射与继承
角色权限管理
URL映射权限
代理模块
URL及菜单管理
目录代理映射
域名代理映射
代理策略管理
缓存模块
规整HTTP缓存协议
LuaVM缓存
本地文件缓存
Redis缓存
访问控制模块
访问次数限制
流量控制
访问计数统计
访问周期报表
持久层
Mysql | Redis
File | logs
文件服务
接口服务
Restful API
SOAP/WS/JSON
搜索服务
Solr/ELK
Web网站

基本业务模型



终端
企业网关
企业应用
终端消费
浏览器
APP
子系统
第三方系统
完成授权
发起请求
获取响应
登录认证
发放Cookie Token
Openresty 网关登录
身份认证
流量转发
提供服务
网页
API
数据流
去除Token/Cookie 并携带用户身份信息
携带Cookie Token
不可信网络
可信网络

About MFA



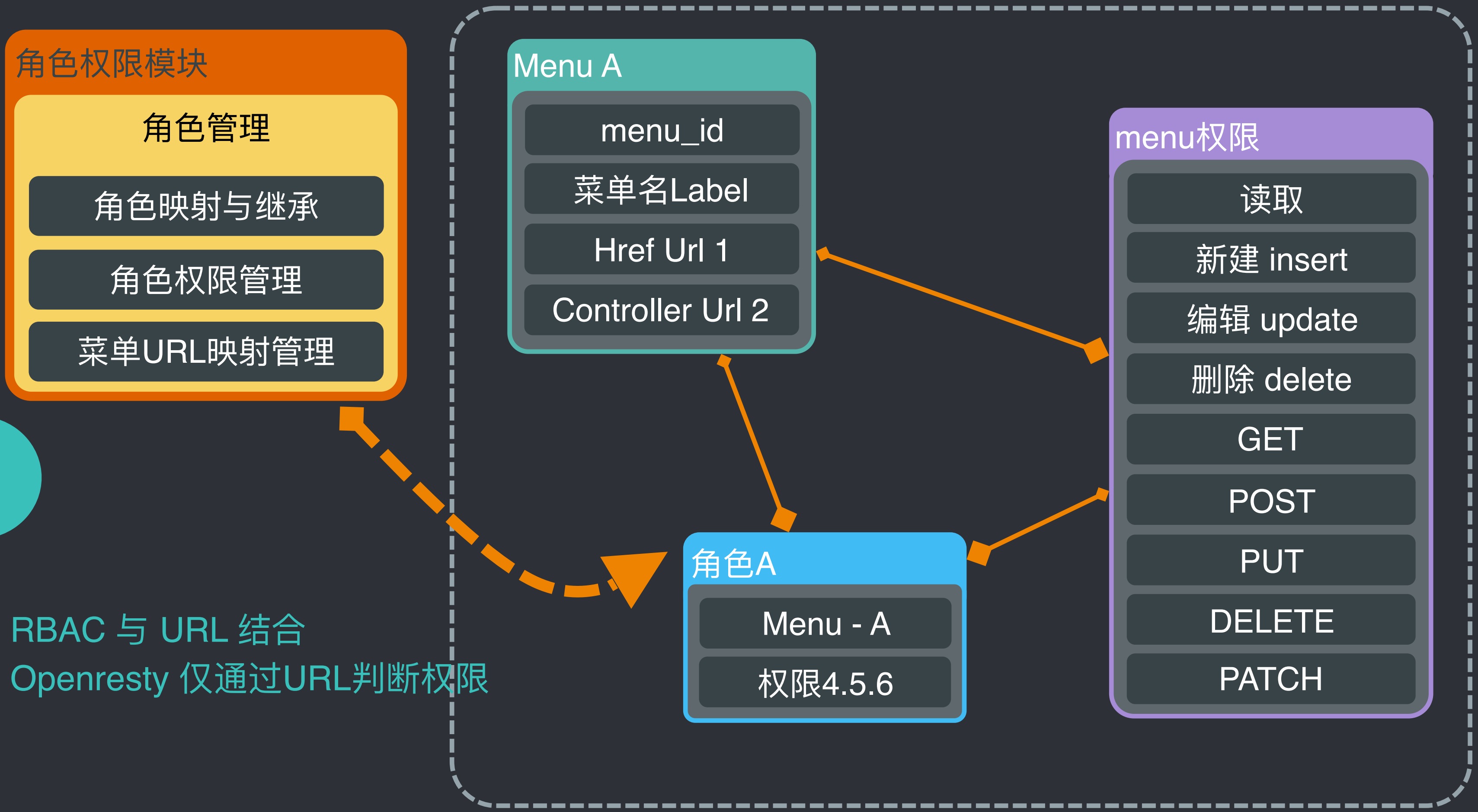
070 062
[redacted]

205 614
[redacted]

415 551
[redacted]

HTTPS + MFA 已经是标配

当然配合VPN 食用风味更佳



RBAC 与 URL 结合
Openresty 仅通过URL判断权限

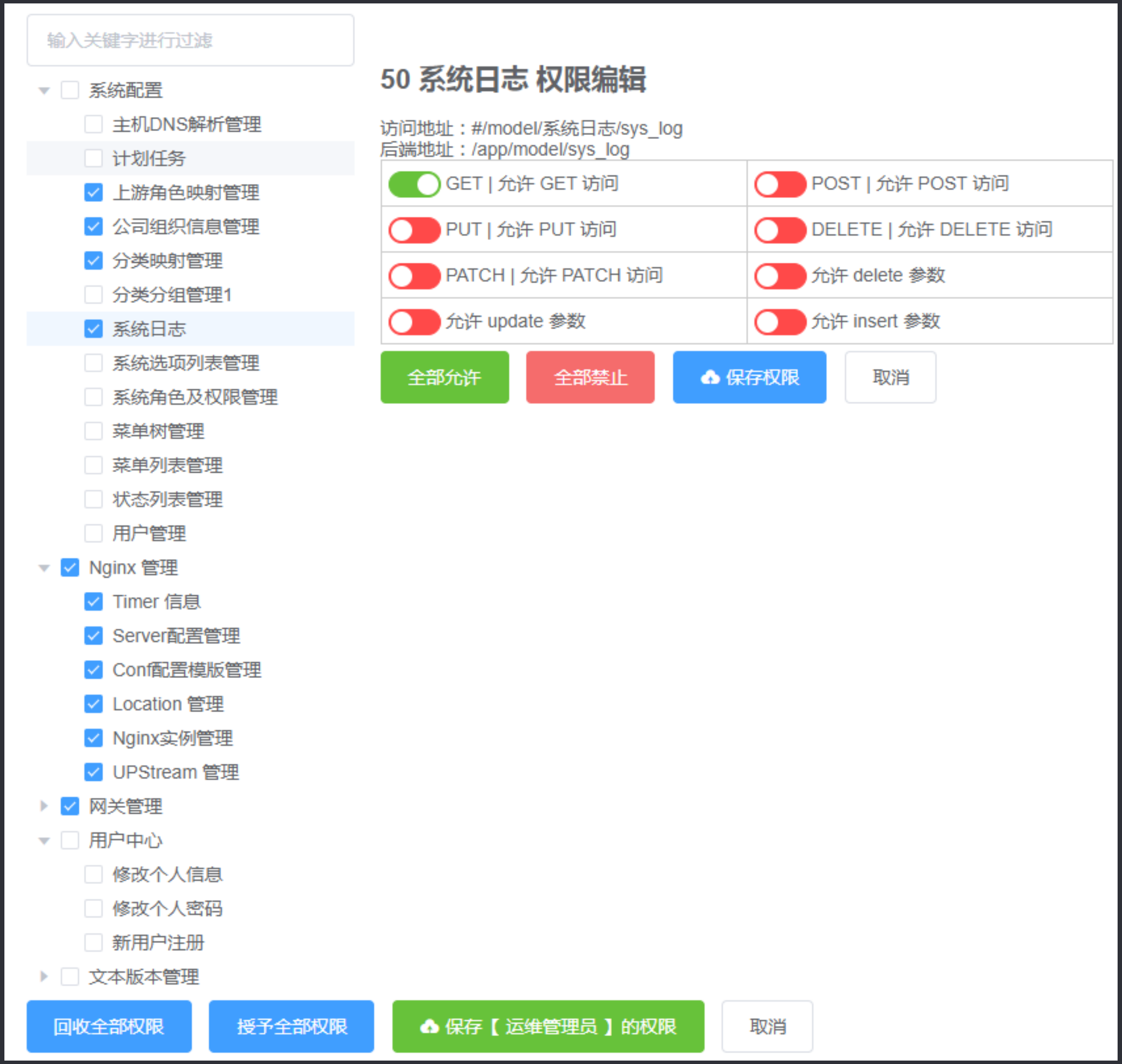
重要性么当然是最高的，资源么当然是没有的，上线么最好就明天！

喜提Openresty

DB:	lua-resty-mysql
Redis:	lua-resty-redis
Router:	lor.sumory.com / K-Router
Http:	lua-resty-http
Render:	lua-resty-template
Mail:	lua-resty-smtp
Shell:	lua-resty-exec
Cache:	lua-resty-tlc
Xpath:	lua-xpath + tidy-html
MQ:	Nchan

Element

RBAC 权限管理


输入关键字进行过滤

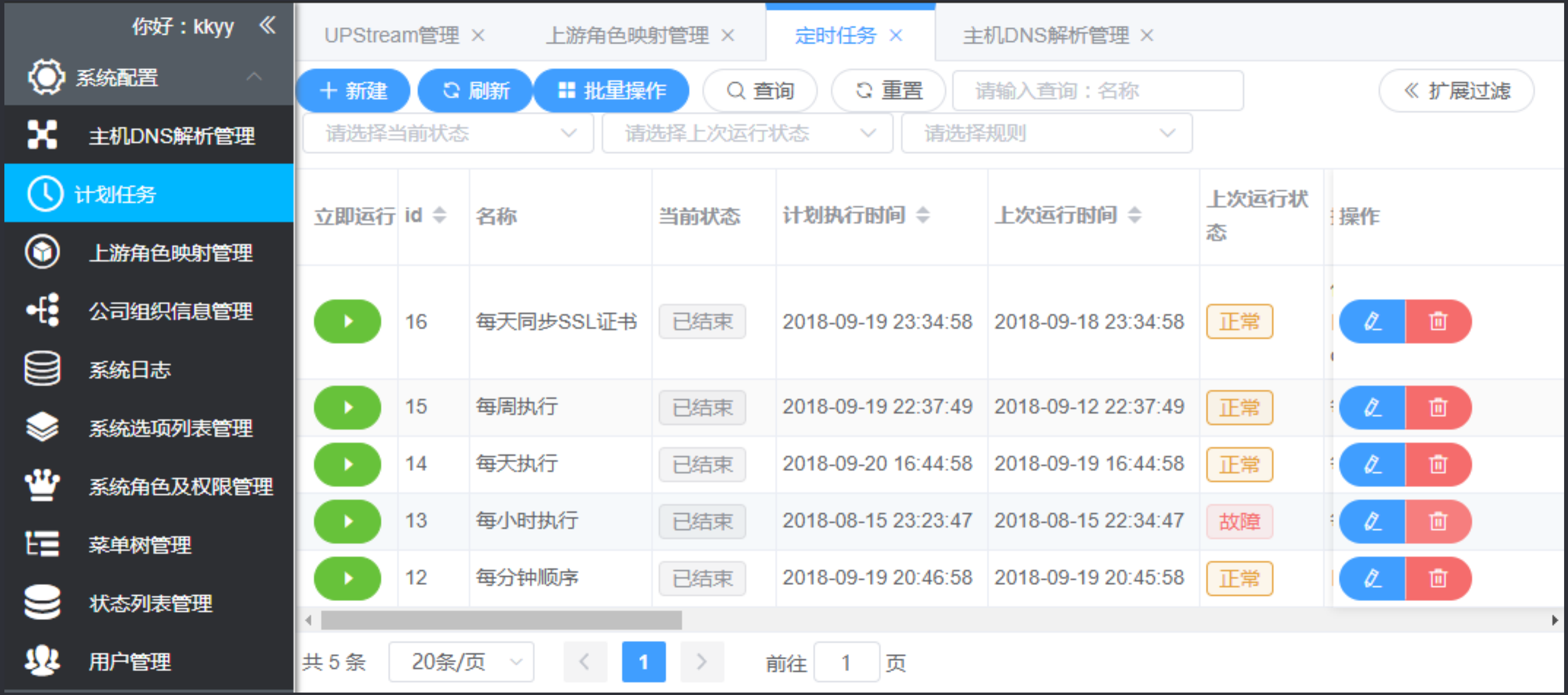
- ☐ 系统配置
  - ☐ 主机DNS解析管理
  - ☐ 计划任务
  - ✓ 上游角色映射管理
  - ✓ 公司组织信息管理
  - ✓ 分类映射管理
  - ☐ 分类分组管理1
  - ✓ 系统日志
  - ☐ 系统选项列表管理
  - ☐ 系统角色及权限管理
  - ☐ 菜单树管理
  - ☐ 菜单列表管理
  - ☐ 状态列表管理
  - ☐ 用户管理
- ✓ Nginx 管理
  - ✓ Timer 信息
  - ✓ Server配置管理
  - ✓ Conf配置模版管理
  - ✓ Location 管理
  - ✓ Nginx实例管理
  - ✓ UPStream 管理
- ✓ 网关管理
- ☐ 用户中心
  - ☐ 修改个人信息
  - ☐ 修改个人密码
  - ☐ 新用户注册
- ☐ 文本版本管理

50 系统日志 权限编辑

访问地址：#/model/系统日志/sys_log
后端地址：/app/model/sys_log

🟩 GET | 允许 GET 访问	🟥 POST | 允许 POST 访问
🟥 PUT | 允许 PUT 访问	🟥 DELETE | 允许 DELETE 访问
🟥 PATCH | 允许 PATCH 访问	🟥 允许 delete 参数
🟥 允许 update 参数	🟥 允许 insert 参数

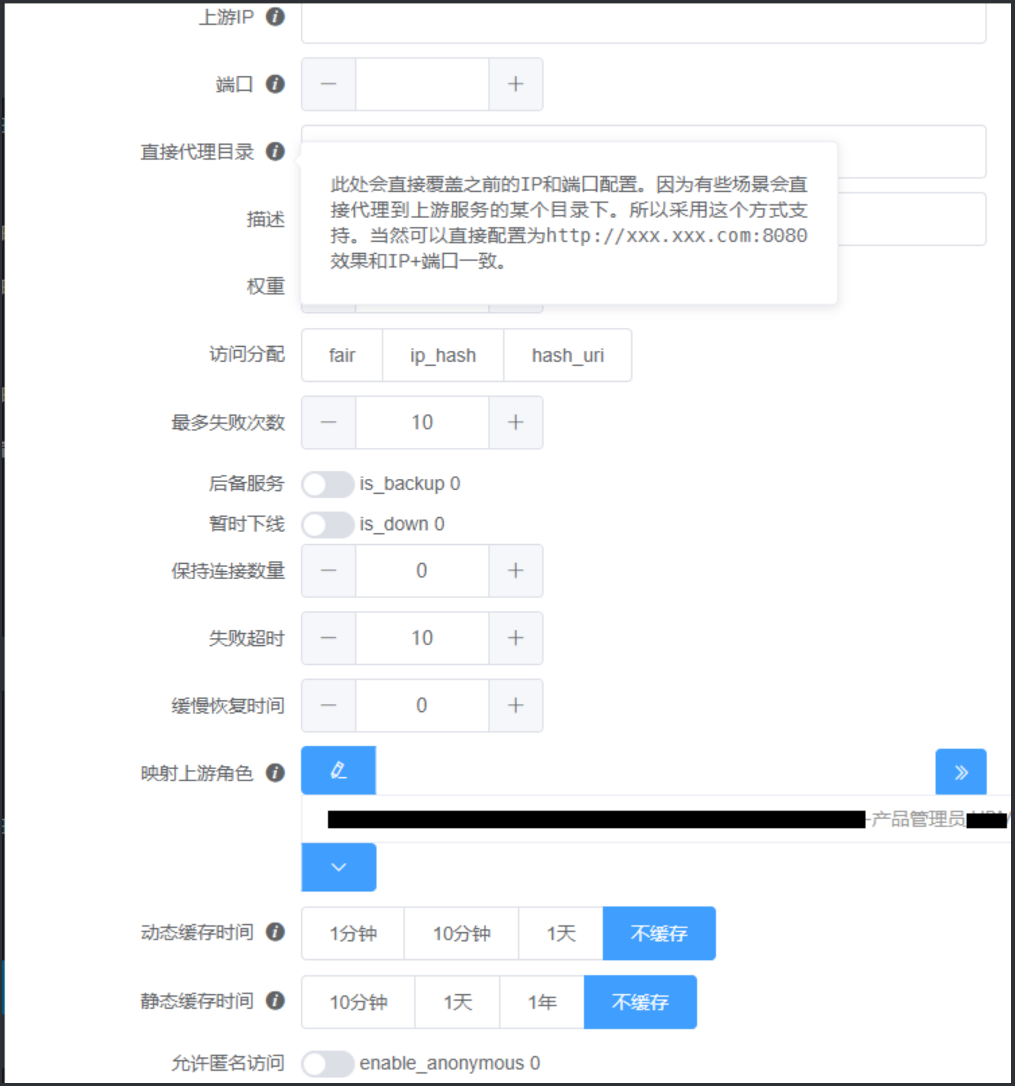
SSL 证书同步



立即运行	id	名称	当前状态	计划执行时间	上次运行时间	上次运行状态	操作
▶	16	每天同步SSL证书	已结束	2018-09-19 23:34:58	2018-09-18 23:34:58	正常	
▶	15	每周执行	已结束	2018-09-19 22:37:49	2018-09-12 22:37:49	正常	
▶	14	每天执行	已结束	2018-09-20 16:44:58	2018-09-19 16:44:58	正常	
▶	13	每小时执行	已结束	2018-08-15 23:23:47	2018-08-15 22:34:47	故障	
▶	12	每分钟顺序	已结束	2018-09-19 20:46:58	2018-09-19 20:45:58	正常	

共 5 条 20条/页 1 前往 1 页

上游管理，及角色映射



上游IP

端口

直接代理目录

此处会直接覆盖之前的IP和端口配置。因为有些场景会直接代理到上游服务的某个目录下。所以采用这个方式支持。当然可以直接配置为http://xxx.xxx.com:8080 效果和IP+端口一致。

描述

权重

访问分配: fair | ip_hash | hash_uri

最多失败次数: 10

后备服务: is_backup 0

暂时下线: is_down 0

保持连接数量: 0

失败超时: 10

缓慢恢复时间: 0

映射上游角色: [redacted] 产品管理员 [redacted]

动态缓存时间: 1分钟 | 10分钟 | 1天 | [✓ 不缓存]

静态缓存时间: 10分钟 | 1天 | 1年 | [✓ 不缓存]

允许匿名访问: enable_anonymous 0

Performance test in single Openresty box (VM)

```
wrk -R200000 -t8 -c800 -d30 -H "token: cuNH1lUCzu2otEGNHP" http://10.80.62.32/_counter
wrk -R200000 -t8 -c800 -d30 -H “token: cuNH1lUCzu2otEGNHP" http://10.80.62.32/app/v1/api/approot.tool/counter
wrk -R200000 -t8 -c800 -d30 -H “token: cuNH1lUCzu2otEGNHP" http://10.80.62.32/lor/hello/test
```

Simple request with token validation or login check could reach 80000+ req/s in production

QPS	1 core 1g	4 core 4g	4 ECS Core 8g	Action
Native	13497	45981	37153	count hit
k-router	12874	42867	32814	count hit HTTP RPC API
lor-router	4446	12626	12007	lor hello world

Keys to Performance

- Avoid using ngx.var system variables, Cache it within ngx.ctx whenever possible
- String variables will be cached within luajit. Hence, it will significant improve string manipulation performance
- String.byte has much better performance than string.find and ngx.re.find
- SharedDICT is much faster than Redis
- os.execute will cause blocking, using lua-resty-exec or lua-resty-shell instead
- As official says. Dispose ngx.timer once job done, trigger GC manually if necessary

感谢 Sumory

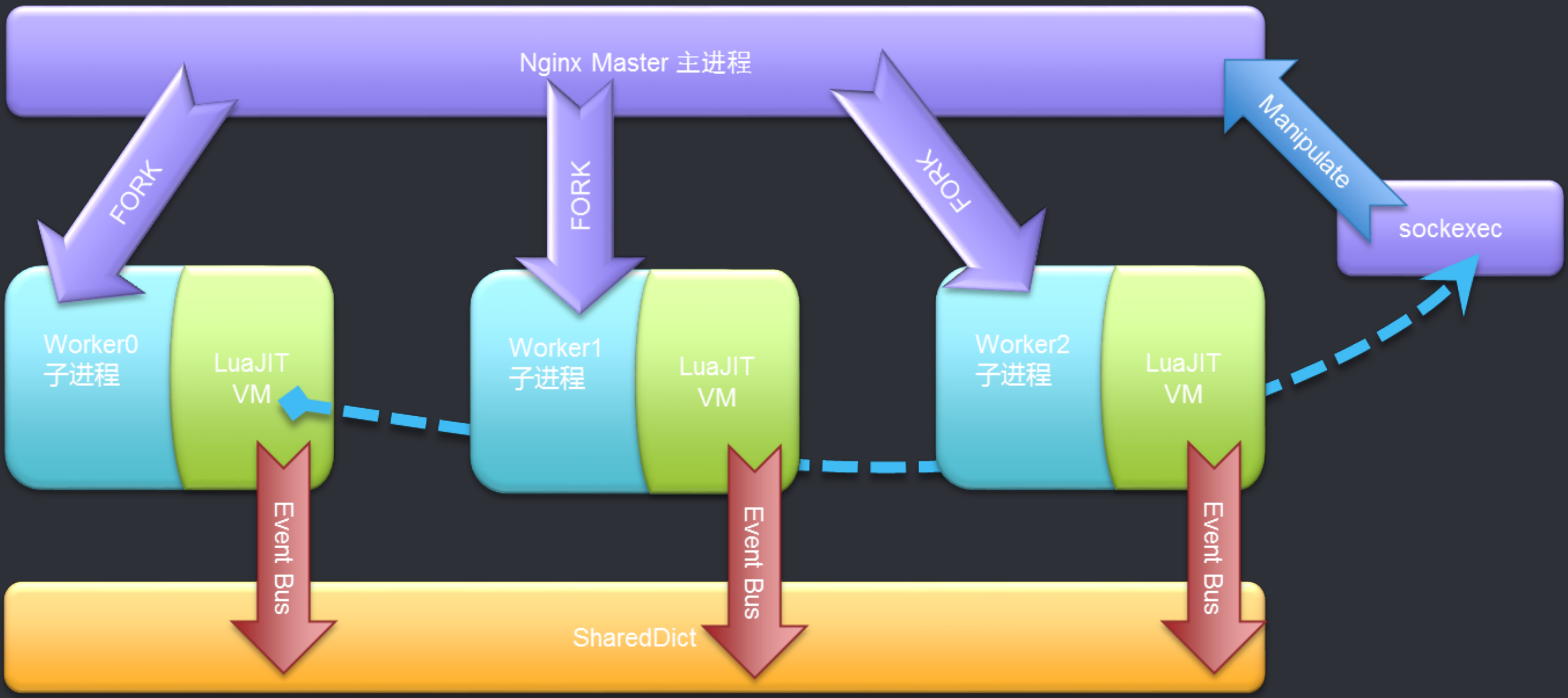
当然,这里同样非常感激 Sumory 不光提供了优秀的 web框架：https://github.com/sumory/lor

同时贡献的 orange 网关: https://github.com/sumory/orange
带给了社群广阔的思路

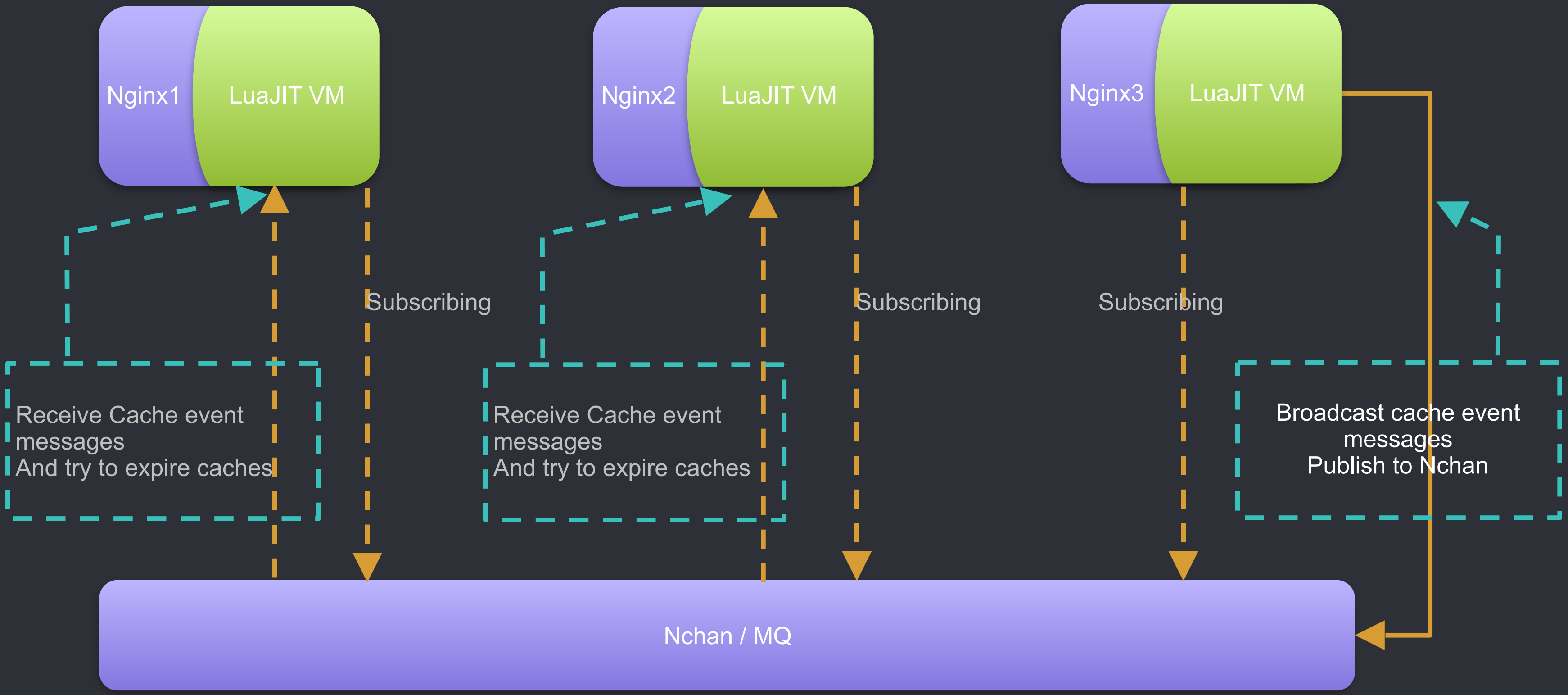
Event bus / Message bus / Cache bus

跨Worker子进程通讯， 跨Nginx 实例通讯，跨机房通讯

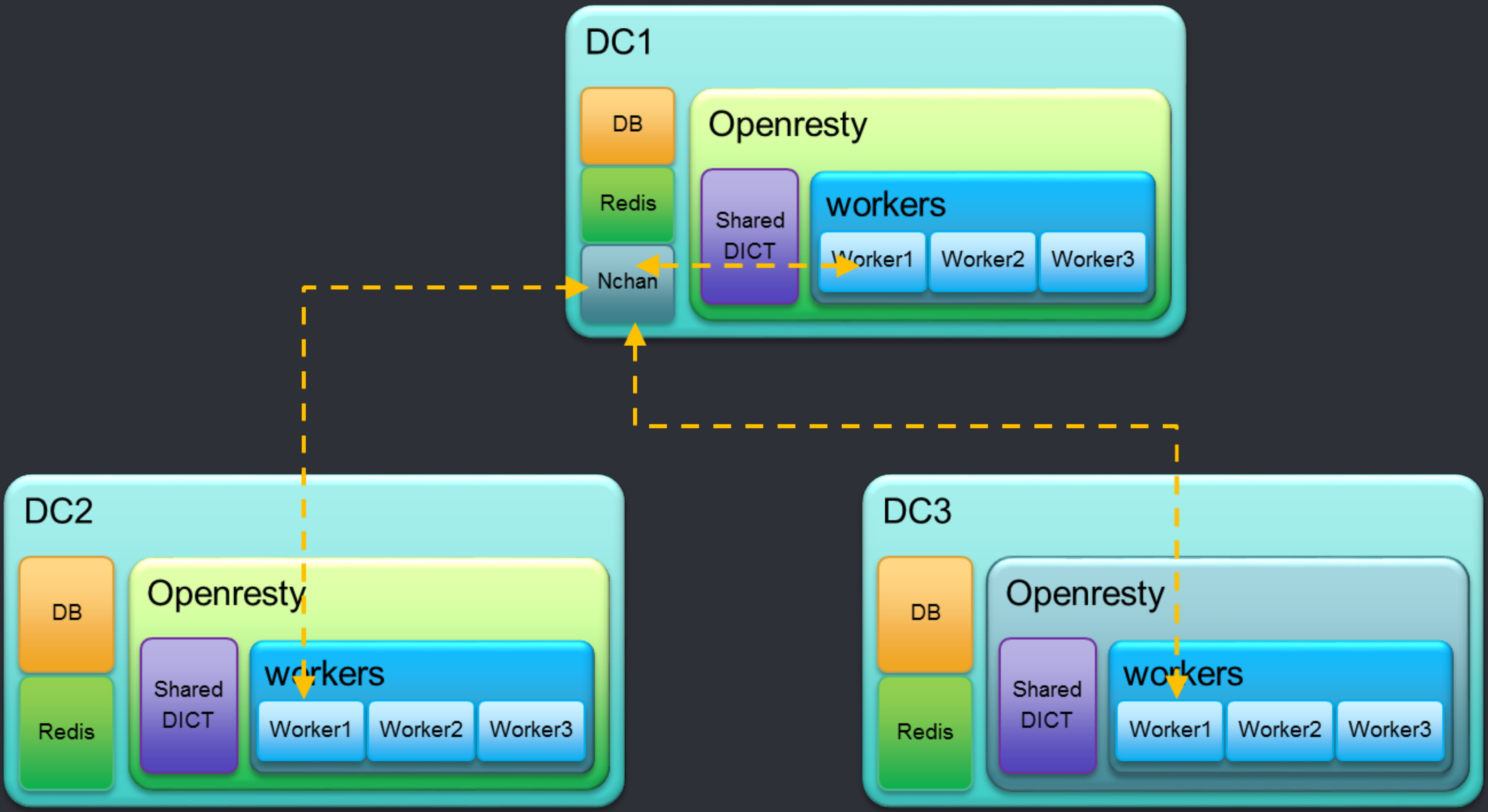
Event Bus works with sockexec and Cache Bus



Message Bus and Cache Bus work with Nchan


Nginx1
LuaJIT VM
Nginx2
LuaJIT VM
Nginx3
LuaJIT VM
Subscribing
Subscribing
Subscribing
Receive Cache event messages And try to expire caches
Receive Cache event messages And try to expire caches
Broadcast cache event messages Publish to Nchan
Nchan / MQ

Cross Data Center sync



Pros	Cons	Alternitives
Low latency High concurrency Easy Integrate Authenticate Very little resource required	Unreliable Persistency Memory Consuming	nats.io NSQ lua-resty-nsq LDAP/DB sync RabitMQ Kafka

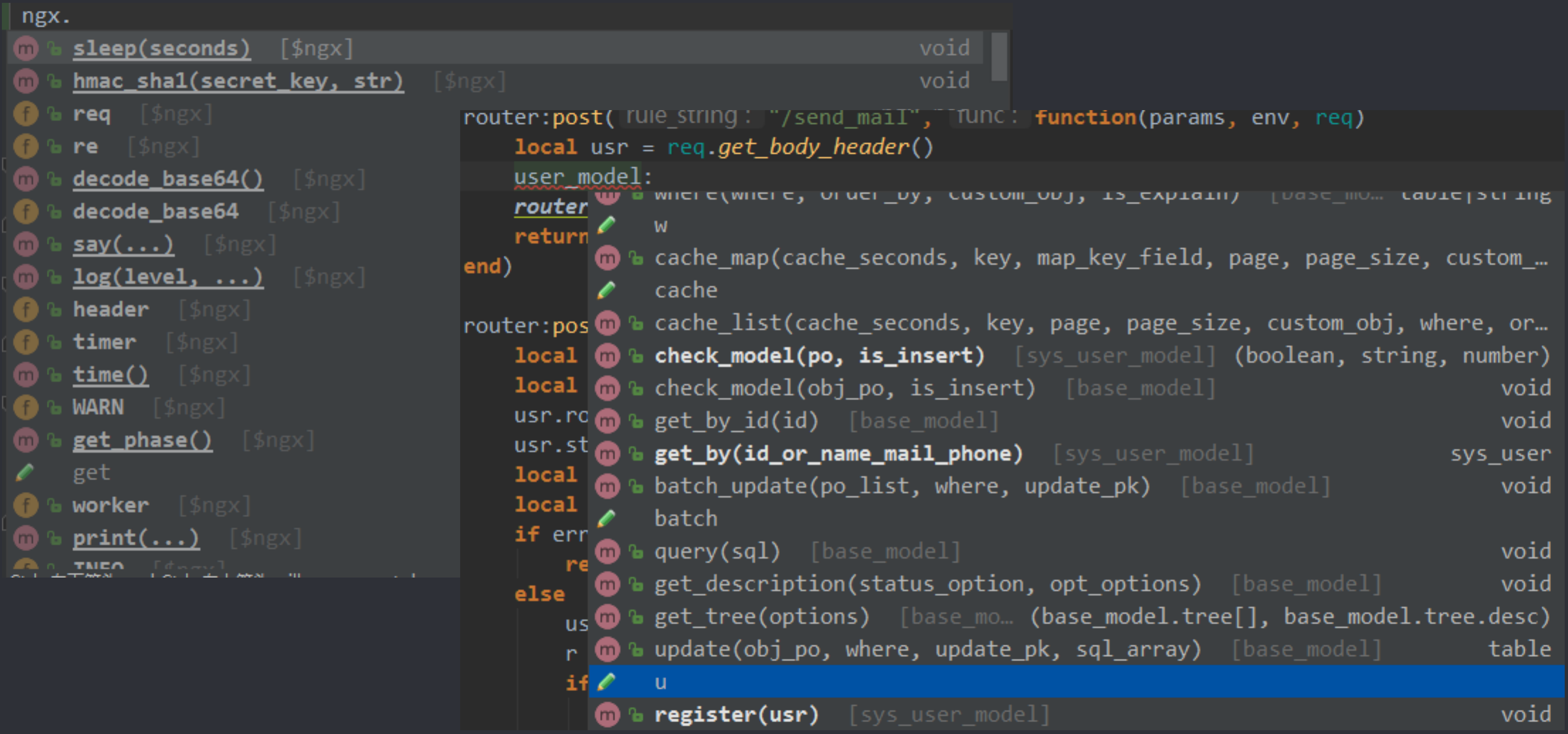
基于EmmyLua Doc 的系统文档生成器

-- 如果没有EmmyLua 我们不可能这么快完成这套业务系统

-- 感谢 阿唐 github.com/EmmyLua, 大幅降低了开发难度, 使得开发Lua能和 Java /C# 开发一样的体验成为可能.

Demos: github.com/EmmyLua/IntelliJ-EmmyLua

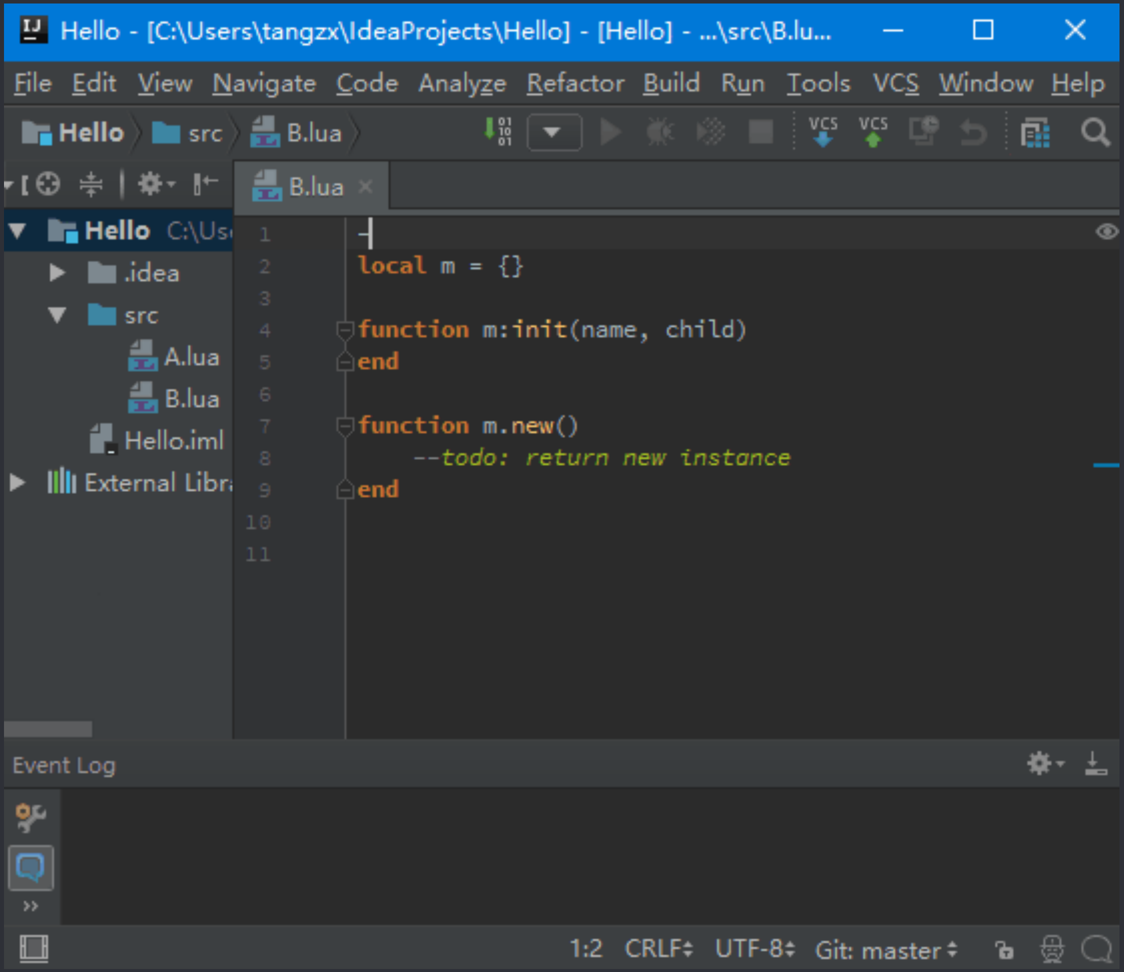
参考 https://emmylua.github.io/ 发现更多惊喜



```
ngx.
sleep(seconds)   [$ngx]                                void
hmac_sha1(secret_key, str)   [$ngx]                    void
req   [$ngx]
re   [$ngx]
decode_base64()   [$ngx]
decode_base64   [$ngx]
say(...)   [$ngx]
log(level, ...)   [$ngx]
header   [$ngx]
timer   [$ngx]
time()   [$ngx]
WARN   [$ngx]
get_phase()   [$ngx]
get
worker   [$ngx]
print(...)   [$ngx]
INFO
```

```
router:post( rule_string: "/send_mail", func: function(params, env, req)
    local usr = req.get_body_header()
    user_model:
    router
    return
end)

router:pos
    local
    local
    usr.ro
    usr.st
    local
    local
    if err
        re
    else
        us
        r
        if
```

```
where(where, order_by, custom_obj, is_explain)   [base_mo…  table|string
w
cache_map(cache_seconds, key, map_key_field, page, page_size, custom_…
cache
cache_list(cache_seconds, key, page, page_size, custom_obj, where, or…
check_model(po, is_insert)   [sys_user_model]   (boolean, string, number)
check_model(obj_po, is_insert)   [base_model]                        void
get_by_id(id)   [base_model]                                         void
get_by(id_or_name_mail_phone)   [sys_user_model]                 sys_user
batch_update(po_list, where, update_pk)   [base_model]               void
batch
query(sql)   [base_model]                                            void
get_description(status_option, opt_options)   [base_model]           void
get_tree(options)   [base_mo…   (base_model.tree[], base_model.tree.desc)
update(obj_po, where, update_pk, sql_array)   [base_model]          table
u
register(usr)   [sys_user_model]                                     void
```



Hello - [C:\Users\tangzx\IdeaProjects\Hello] - [Hello] - ...\src\B.lu...

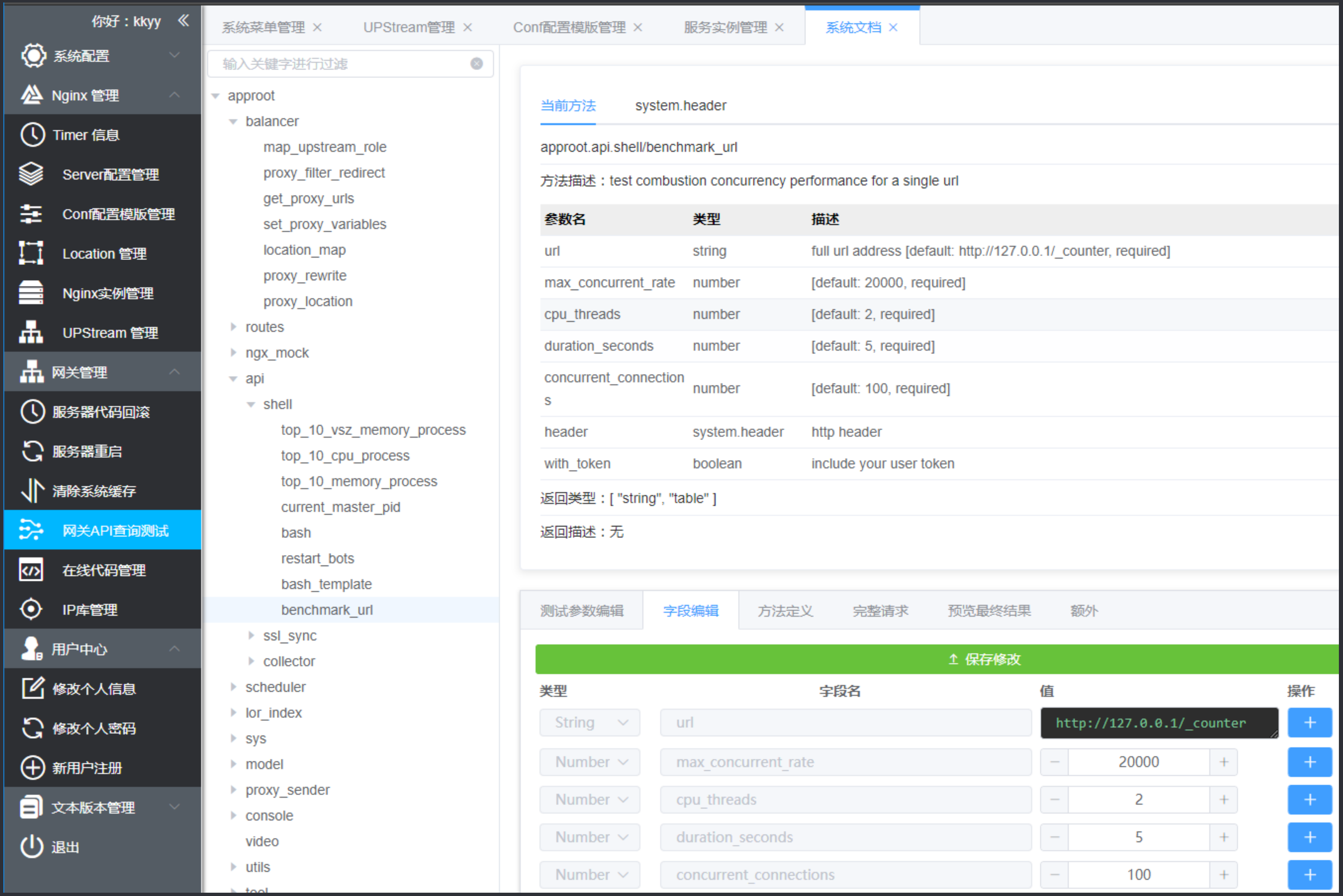
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Hello
src
B.lua
.idea
A.lua
Hello.iml
External Libr
Event Log

```lua

local m = {}

function m:init(name, child)
end

function m.new()
    --todo: return new instance
end
```

1:2 CRLF UTF-8 Git: master

EmmyLua document samples

```lua
---benchmark_url test combustion concurrency performance for a single url
---@param url string @full url address [default: http://127.0.0.1/_counter, required]
---@param max_concurrent_rate number @ [default: 20000, required]
---@param cpu_threads number @[default: 2, required]
---@param duration_seconds number @[default: 5, required]
---@param concurrent_connections number @[default: 100, required]
---@param header system.header @http header
---@param with_token boolean @include your user token
---@return string, table
function _M.benchmark_url(url, max_concurrent_rate, cpu_threads, duration_seconds,
concurrent_connections, header, with_token)
    local bash = table.array(15)
    ins(bash, 'wrk -L ')
    local gtoken = 'wfg-token'
    if header and type(header) == 'table' then
        if (with_token and not header[gtoken]) or (header[gtoken] and #header[gtoken] < 20) then
            header[gtoken] = uc.get_current_req_token()
        end
        for key, val in pairs(header) do
            ins(bash, '-H "' .. key .. ': ' .. val .. '"')
        end
    end
    ins(bash, '-R' .. max_concurrent_rate)
    ins(bash, '-t' .. cpu_threads)
    ins(bash, '-d' .. duration_seconds)
    ins(bash, '-c' .. concurrent_connections)
    ins(bash, url)
    --Convert `abc_efg` into `abc-efg`
    local sh = concat(bash, ' ')
    local res, err = _M.bash(sh)
    return res, sh
end
```



你好：kkyy

系统配置
Nginx 管理
Timer 信息
Server配置管理
Conf配置模版管理
Location 管理
Nginx实例管理
UPStream 管理
网关管理
服务器代码回滚
服务器重启
清除系统缓存
网关API查询测试
在线代码管理
IP库管理
用户中心
修改个人信息
修改个人密码
新用户注册
文本版本管理
退出

系统菜单管理 × | UPStream管理 × | Conf配置模版管理 × | 服务实例管理 × | 系统文档 ×

输入关键字进行过滤

- approot
  - balancer
    - map_upstream_role
    - proxy_filter_redirect
    - get_proxy_urls
    - set_proxy_variables
    - location_map
    - proxy_rewrite
    - proxy_location
  - routes
  - ngx_mock
  - api
    - shell
      - top_10_vsz_memory_process
      - top_10_cpu_process
      - top_10_memory_process
      - current_master_pid
      - bash
      - restart_bots
      - bash_template
      - benchmark_url
    - ssl_sync
    - collector
  - scheduler
  - lor_index
  - sys
  - model
  - proxy_sender
  - console
  - video
  - utils
  - tool

当前方法 | system.header

approot.api.shell/benchmark_url

方法描述：test combustion concurrency performance for a single url

参数名	类型	描述
url	string	full url address [default: http://127.0.0.1/_counter, required]
max_concurrent_rate	number	[default: 20000, required]
cpu_threads	number	[default: 2, required]
duration_seconds	number	[default: 5, required]
concurrent_connections	number	[default: 100, required]
header	system.header	http header
with_token	boolean	include your user token

返回类型：["string", "table"]

返回描述：无

测试参数编辑 | 字段编辑 | 方法定义 | 完整请求 | 预览最终结果 | 额外

保存修改

类型	字段名	值	操作
String	url	http://127.0.0.1/_counter	+
Number	max_concurrent_rate	20000	+
Number	cpu_threads	2	+
Number	duration_seconds	5	+
Number	concurrent_connections	100	+

OpenResty™ IT大咖说 知识共享平台

Thanks!

ANY QUESTIONS?

You can find me at whyork@gmail.com

Github.com/yorkane