**NJS介绍**

**1,什么是NJS**

njs是nginScript的简称，是为了 NGINX 和 NGINX Plus 而开发的 JavaScript 实现，它被设计用于在服务器端处理请求。它通过融入 JavaScript 代码对 NGINX 的配置语法进行扩展，以便实现复杂的配置。

**2,NJS能做什么**

nginScript 同时支持 HTTP 和 TCP/UDP 两种协议，所以它的应用场景很广。

生成自定义的日志格式， 日志里可以包含普通 NGINX 变量无法表示的值。

实现新的负载均衡算法。

通过解析 TCP/UDP 协议，实现应用层的粘性会话。

检查和修改 HTTP 请求消息和响应消息的 body（已经支持 TCP/UDP）。

从 nginScript 代码里发起 HTTP 子请求。

编写 HTTP 认证处理器（已经支持 TCP/UDP）。

读写文件。

除此之外还不断有新特性再逐渐加入进来

**3,NJS 对决 Lua**

多年来，NGINX社区创建了很多编程式扩展，Lua是其中最受欢迎的一个。Lua被作为NGINX的一个模块，典型的就是OpenResty，基于nginx做了很多插件。

但lua语言并不是一种特别流行的开发语言，比较小众。而我们为了解决一个问题去专门学习一门语言，门槛相对过高。所以nginx的官方团队推出了NJS。

但目前在成熟度上，njs和lua的差距还是蛮大的，比如访问redis，njs目前无法支持。

**4,NJS 容易被误解**

NJS != JavaScript 更不是nodejs

njs是JavaScript/ECMAscript的子集 。它实现了大部分的JavaScript语言的能力，没有完全遵从ECMAScript标准，同时抛弃了JavaScript比较难懂的部分。

njs不是通过V8引擎实现的 。而是通过一个更小、能耗更低、更符合nginx应用场景的小虚拟机（VM）来实现。可以理解为nginx为其实现了一套自己的词法解析。

njs是跑在nginx的配置文件里 。 比如：nginx.conf文件里。所以njs可以完成传统配置文件所能处理的所有事情，同时可以让配置管理动态化。这也是njs出现的最重要的原因。

Web端的JavaScript中的很多语法都不适用与njs中

简而言之，相比较lua，给大家提供了一个热度更高的语言实现方式。

**5,NJS的设计原则**

为了满足服务器端代码的执行需求，并与 NGINX 的请求处理架构保持兼容。njs的设计遵循了如下原则。

**根据请求来创建和销毁运行时环境**

nginScript 使用单线程执行字节码，可以实现快速的初始化和销毁。系统为每个请求创建一个新的运行时环境，因为不需要初始化复杂的状态和辅助组件，所以启动速度非常快。在处理请求期间分配一个内存池，请求处理完毕之后销毁内存池。这种内存使用模式避免了对象的跟踪和释放工作，也不需要使用垃圾回收器。

**非阻塞的代码执行**

nginScript 的运行时环境是通过 NGINX 和 NGINX Plus 的事件驱动模型来执行的。当某个 nginScript 规则在执行阻塞操作时（比如读取网络数据或向外部发起子请求），NGINX 和 NGINX Plus 会将该虚拟机挂起，直到阻塞操作事件结束。也就是说，规则的编写会变得很简单，NGINX 和 NGINX Plus 会在内部把它们变成非阻塞操作。

**只实现必要的语言特性**

ECMAScript定义了 JavaScript 的规范。nginScript 实现了ECMAScript 5.1，以及ECMAScript 6中与数学运算相关的规范。实现自己的 JavaScript 运行时给了我们充分的自由，我们可以优先实现服务器端需要的特性，忽略掉不需要的部分。

**与请求处理阶段的深度集成**

NGINX 和 NGINX Plus 把请求的处理分为不同的阶段。配置指令一般会在一个特定的阶段执行，本地 NGINX 模块经常利用这个特性检查和修改请求消息的内容。nginScript 通过配置指令把某些处理阶段暴露出来，从而能够控制 JavaScript 代码的执行。这种方式保证了本地 NGINX 模块的强大和灵活，同时保持 JavaScript 代码的简单性。

**6，和其他语言插件相比的优势和劣势**

优势：NGINX官方团队出品，2015 年 9 月启动以来，有很多特性和核心语言支持不断被添加进来。

劣势：发展时间尚短，功能上还有所欠缺。

**7,在特来电中的应用**

业务场景：saas化

对于特来电的web运行平台，我们有很多saas化客户。假如平台站点是x.teld.cn

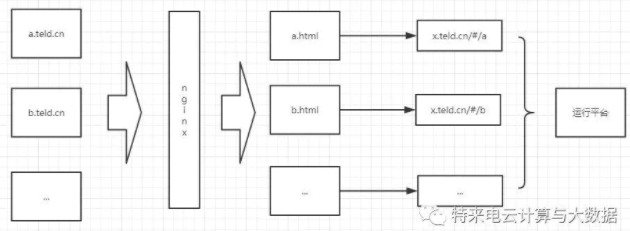
而saas用户域名为：

a.teld.cn

b.teld.cn

c.teld.cn

运行结构如图：



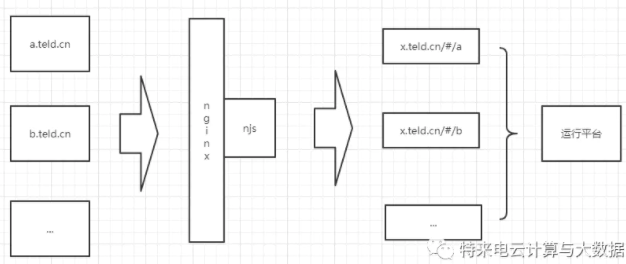
由于使用的是hash路由，转向url中携带#，所以没法通过配置实现动态化

这种结构有个最大的问题，就是每增加一个saas用户都要部署一个html以及配置

对大家来说，这是一种很糟糕的体验

我们开始尝试寻找解决方案，开始选的是lua，但总感觉对于这样简单的场景，引入lua容器，有点过重，有点杀鸡用牛刀的感觉。通过对比最终采用njs来实现。

改造后如图：



过njs脚本来实现请求的变化。

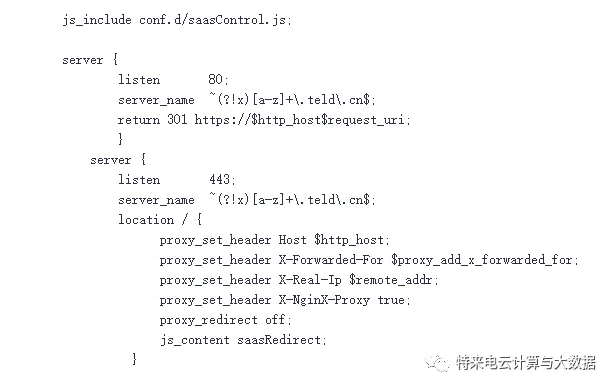
首先在conf文件中对server\_name采用正则表达式拦截

进入njs脚本，根据不同客户动态输出不同的内容到客户端。

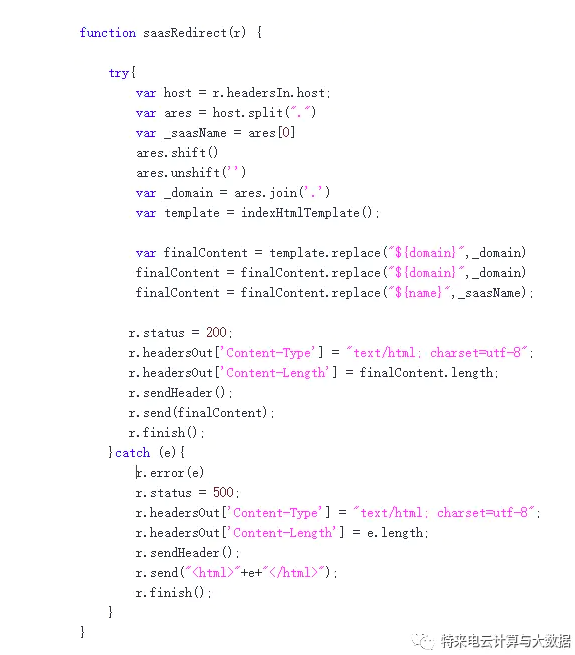
经过调整，saas客户的增加不会对部署程序造成影响，目前上线半年，体验良好。

配置及部分代码

在配置文件中：



在saasControl.js中



**8,总结**

整体来讲，njs还是处于发育期，和lua在功能上比还是比较弱。

如果用nginx根据映射表来反向代理内网IP+端口，目前来说lua更合适。

但由于njs是nginx官方团队出品，长远来看，无论从质量还是功能上，都很值得期待。