# 遗留问题

## 1. 动态添加location限制

- proxy_pass 只支持预先定义的upstream。 不支持 proxy_pass http://www.xxx.com 以及unix socket。
- location 中变量使用未set 定义的自定义变量时，默认是给空值。 大量使用会有内存问题（静态配置location 会检测，并报错）。

### 1.1 不支持在内部配置access_log指令

### 1.2 仅仅支持proxy_pass到upstream

### 1.3 Location 内存问题1：

目前动态增加location后，需要执行njt_http_merge_locations 操作，与所在的的server，做变量合并，正常情况下，都是server 层向location 做向下合并，但如果某模块或第三方模块，做了反向合并，并且使用了动态location 的pool内存，当该location 被删除后，server 结构中就会出现“野”指针。 但如果是server 的pool 内存，又可能存在内存泄漏。

#### 目前做法：

按照动态设计原则，使用的是动态location 的pool 内存。

- 如果发现有其他模块或第三方模块，使用了反向合并，并且申请了pool 内存的情况，需要做代码修改。
- 需要遵循动态location 的规范。
- 后续考虑新的解决方法

### 1.4 Location proxy_pass问题2：

动态location 中的proxy_pass 后的upstream 目前必须是配置文件中，提前定义的。如果动态添加的location 包含的proxy_pass 是http://www.baidu.com 等这种静态的url 地址，或unix socket 文件句柄，在njet 实现都是做为特殊的upstream 的形式存在，并且属于server 级别的，并且是多location 共享的，目前这种upstream 的内存建立在server 的级别，需要增加多location 引用的计数器，删除最后引用该upstream location 时需要删除upstream，另外upstream 列表目前使用的是数组，内存无法释放只能重用。 目前存在内存释放的问题，留到下版本解决。api 对未提前定义的upstream 在添加时，进行检测过滤。

目前做法：

对proxy_pass 进行检测。

## 2. 主动健康检查

### 2.1 不支持双向认证，不校验服务端证书

### 2.2 不支持udp，tcp

## 3. sidecar

## 4. SSL一致性

第三方模块如果使用ssl，需使用NJet的ssl版本（NJet1.0为tassl，后续移植为铜锁）

## 5. 国密

在njet中配置国密证书，因为是利用同样的指令load，签名证书必须先配置，后配置加密证书。

如果前后有几处配置对同一个upstream进行反向代理，注意在是否使用国密的问题上要保持前后有一致。

## 6. 动态黑白名单

不支持ipv6设置

## 7. 动态配置

配置错误信息无法返回到调用方，仅仅记录到日志文件中

动态配置中使用（新创建的）变量目前无法清除

## 8. copilot：ctrl

配置文件中需要指定不同于主配置的error文件，便于调试分析