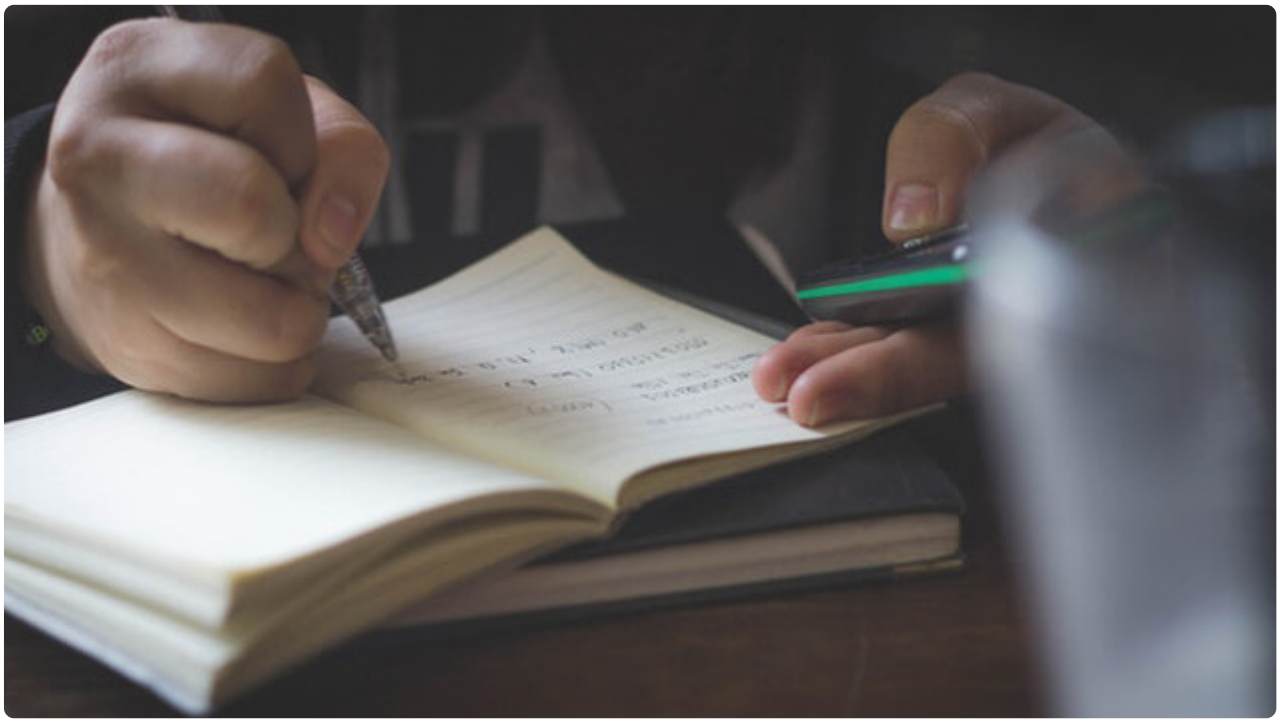
14 保留证据：能玩出花来的 Nginx 日志

更新时间：2020-01-09 09:50:37



> 与有肝胆人共事，从无字句处读书。——周恩来

前言

对于一个大型的应用来说，日志有着举足轻重的作用。通过查找日志，我们可以迅速定位到问题发生的原因，在第一时间 fix bug 。
一个优秀的软件必然有着一个出色的日志系统， Nginx 也不例外，容我一一为你道来~

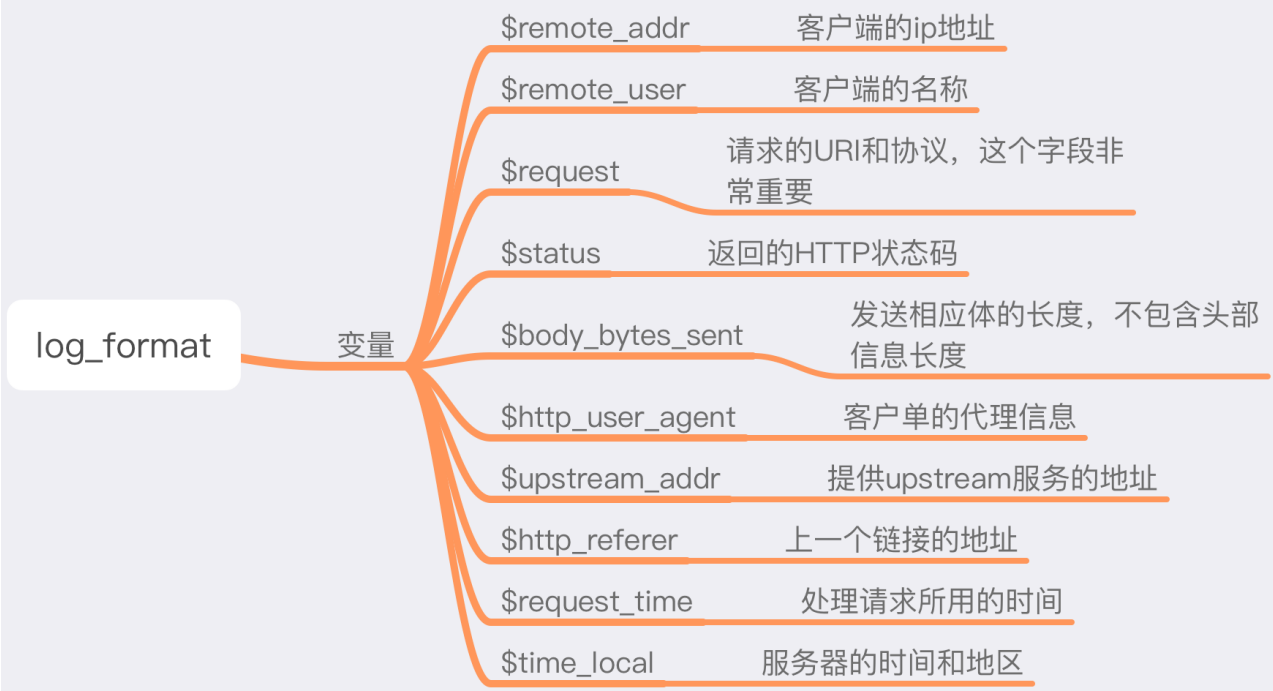
> 可能你不相信，我刚开始从事互联网行业的时候，根本不知道什么是日志，也不知道日志有什么作用，线上出问题的时候完全不知道应该怎么排查问题，超级尴尬~

日志

Nginx 日志分为很多种，比如访问日志(access_log)，错误日志(error_log)，以及我们前面介绍过的 Rewrite 日志。其实在工作中，我们经常使用到的就是 access_log 和 error_log 。

日志格式

我们在记录日志的时候，肯定要告诉Nginx自己感兴趣的内容，要不然Nginx也不知道要记录哪些东东。此时我们可以通过log_format指令来完成这个功能。这个功能和Nginx的变量机制有很强的关联，我们可以使用Nginx提供的一些变量。最常用的变量如下：



比如，许多服务器都会配置如下的日志格式：

```
log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                  '$status $body_bytes_sent "$http_referer" '
                  '"$http_user_agent" "$http_x_forwarded_for"'
                   '$upstream_addr $upstream_response_time $request_time ';
```

上面我们定义了一个名称为main的日志格式，可以在后面的access_log中使用。

访问日志

访问日志又叫做access_log，它应该记录当前服务器的所有请求，通过access_log指令来配置，该指令有几个特殊的参数，我们看一下：

```
access_log path [format [buffer=size]
```

这里面的path就是我们的访问日志要保存的路径。
format就是每条日志记录的格式，也就是上面的log_format定义的格式。
buffer定义了一个缓冲区的大小，只有当内存中的日志体积大于buffer的时候才会持久化到磁盘，这样可以减少磁盘读写次数，提高效率。

下面是一个配置示例:

```
access_log  "/usr/local/nginx/logs/access_log"  main;
```

通过扫描access_log日志，配合awk等工具可以计算每个接口的成功率，耗时等各个性能参数。

错误日志

错误日志通过error_log来定义，如下：

```
error_log file [level];
```

其实error_log的配置很简单，它的参数也是简单明了的，这里面有一个level，它表示日志等级。其实几乎所有的软件在记录日志的时候都会有等级这个概念，在不同的场景下使用不同的等级。比如我们在调试软件的时候可能希望尽可能的记录更多的日志，方便我们调试。但是在正式的生产环境，我们可能不会输出特别多的日志以节省磁盘空间，这时候我们会选择一个其他的等级。
Nginx提供了debug，info，notice，warn，error，crit，alert，emerg八个等级，它们从左至右等级一次变高。一般线上我们都会选择warn或者error等级。比如下面的配置就使用了error级别：

```
error_log  "/usr/local/nginx/logs/error_log"  error;
```

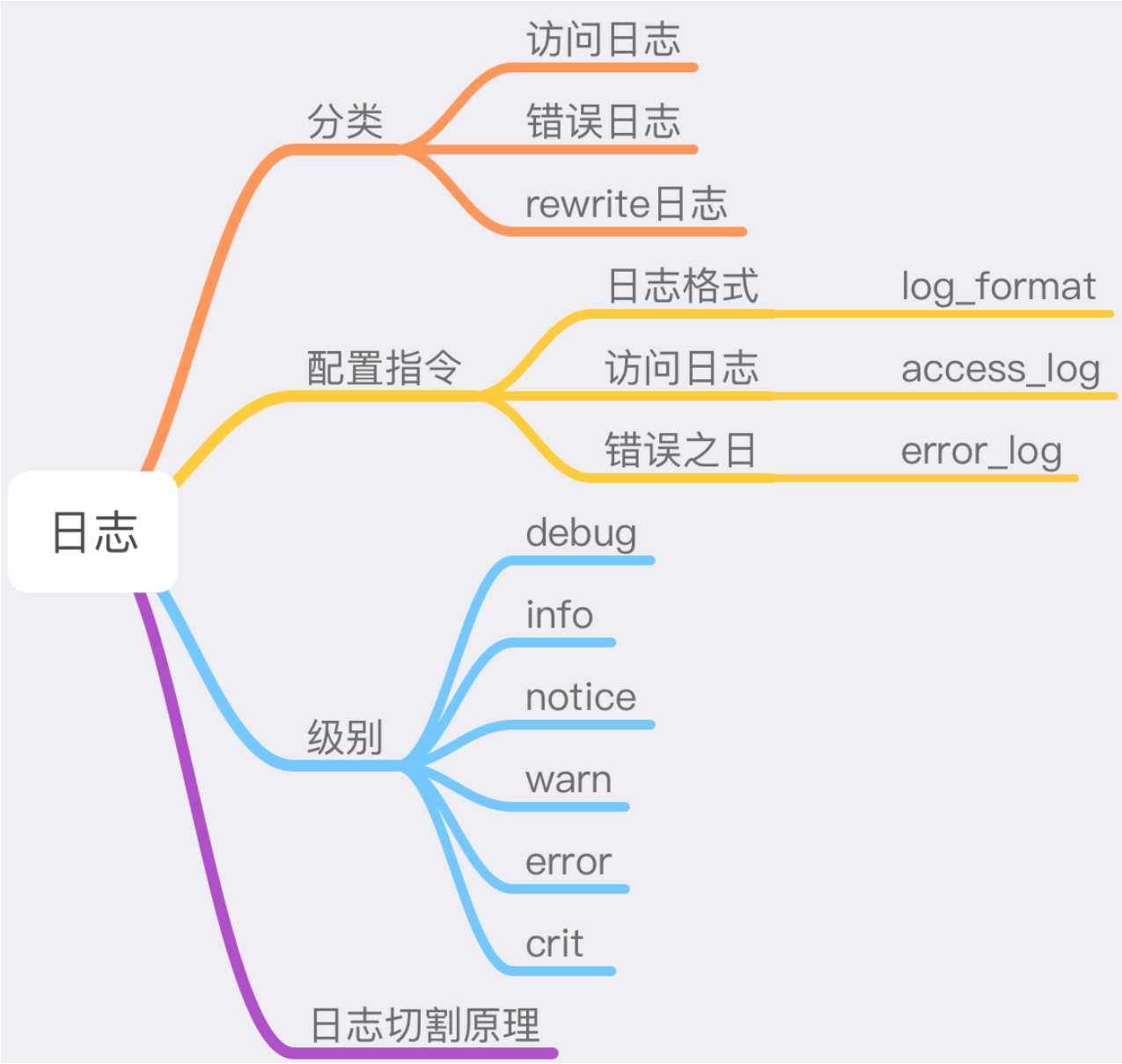
日志分割

随着服务的运行，我们的日志文件可能变得越来越大，这样对于排查问题非常的不变，这个时候我们可能就需要对日志文件进行切割了。一般情况下，我们会按照小时对日志进行切割，当然了，如果日志量比较小的话，也可以按照天进行切割。
我们在最开始介绍Nginx常用操作的时候提到过，当Nginx接收到USER1的时候，会重新打开日志文件，日志切割正式利用了Nginx的这一个特性。
我们可以通过一个shell脚本来完成脚本切割功能:

```
#!/bin/bash
pidPath="/usr/local/nginx/logs/nginx.pid"
oldAccessLog="/usr/local/nginx/logs/access_log"
echo $oldAccessLog "\n"
# 按照小时生成一个新的文件
newAccessLog=$oldAccessLog`date "+%Y%m%d%H"`
echo $newAccessLog "\n"
mv $oldAccessLog $newAccessLog
# 通知nginx重新打开日志文件
kill  -USR1 `cat ${pidPath}`
```

将上面的脚本放到crontab中，每小时执行一次就可以实现日志分割了。

总结



}

13 找到你的Mr Right：Location 查找原则

15 加密你的通话记录：从 HTTP 到 HTTPS