链滴

运维高手第 07 课： Jenkins 的持续集成架构及常见问题

作者：chenteng

原文链接：https://ld246.com/article/1636468932985

来源网站：链滴

Jenkins 的持续集成架构及常见问题

@[toc]

学前提示与基础概念

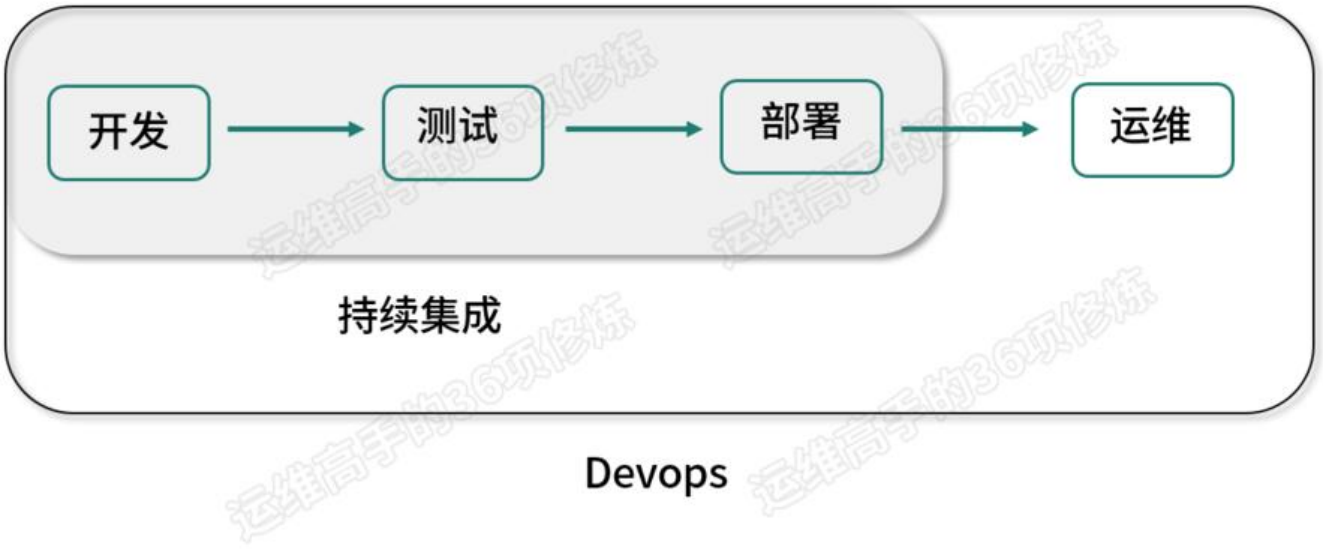
一、Jenkins 简介

首先，说到 Jenkins，你肯定知道它是一个用于搭建持续集成平台的开源工具，它的官网地址是：http://jenkins.io/。那什么是持续集成呢，持续集成是近些年来企业管理、软件开发、软件上线过程中的个概念。我的理解是它主要强调的是：

开发人员在提交了新的代码以后，立即进行构建、（单元）测试等相关流程；自动化部署。等。

二、持续集成和 Devops 的差异

持续集成这个概念想必我们早已耳熟能详了，近些年新产生的一个概念叫作 Devops，那持续集成和 evops 有什么差异呢？你可以看一下我这里画的一张图示：



我这里画了最外圈的一个方框，代表的是 Devops 一层，其里面包含了持续集成的流程部分。我们看，里面的小方框中是持续集成覆盖的范围，包含**三个部分：开发、测试和部署**，这三个环节对企业开来说是必备的，是必须要经过的流程，持续集成作到了把三个过程很好的相互串联起来。

所以持续集成覆盖的主要是开发、测试和部署三个环节，但是持续集成忽略了一个重要的步骤，就是线以后的运维环节。运维的成本对于企业来说也是消耗比较大的，而 Devops 就集成了整个持续集成节，并且重点融入了运维环节来考虑。

三、持续集成和 Devops 共同的技术要求

总结下，持续集成和 Devops 共同的技术要求是：

1. 自动化，我们要做到部署自动化、测试自动化等。
2. 各项流程需要无缝打通，如开发到测试，测试到部署，这些都要求无缝地把各个管理环节和流程打。
3. 交给统一平台进行可视化管理。

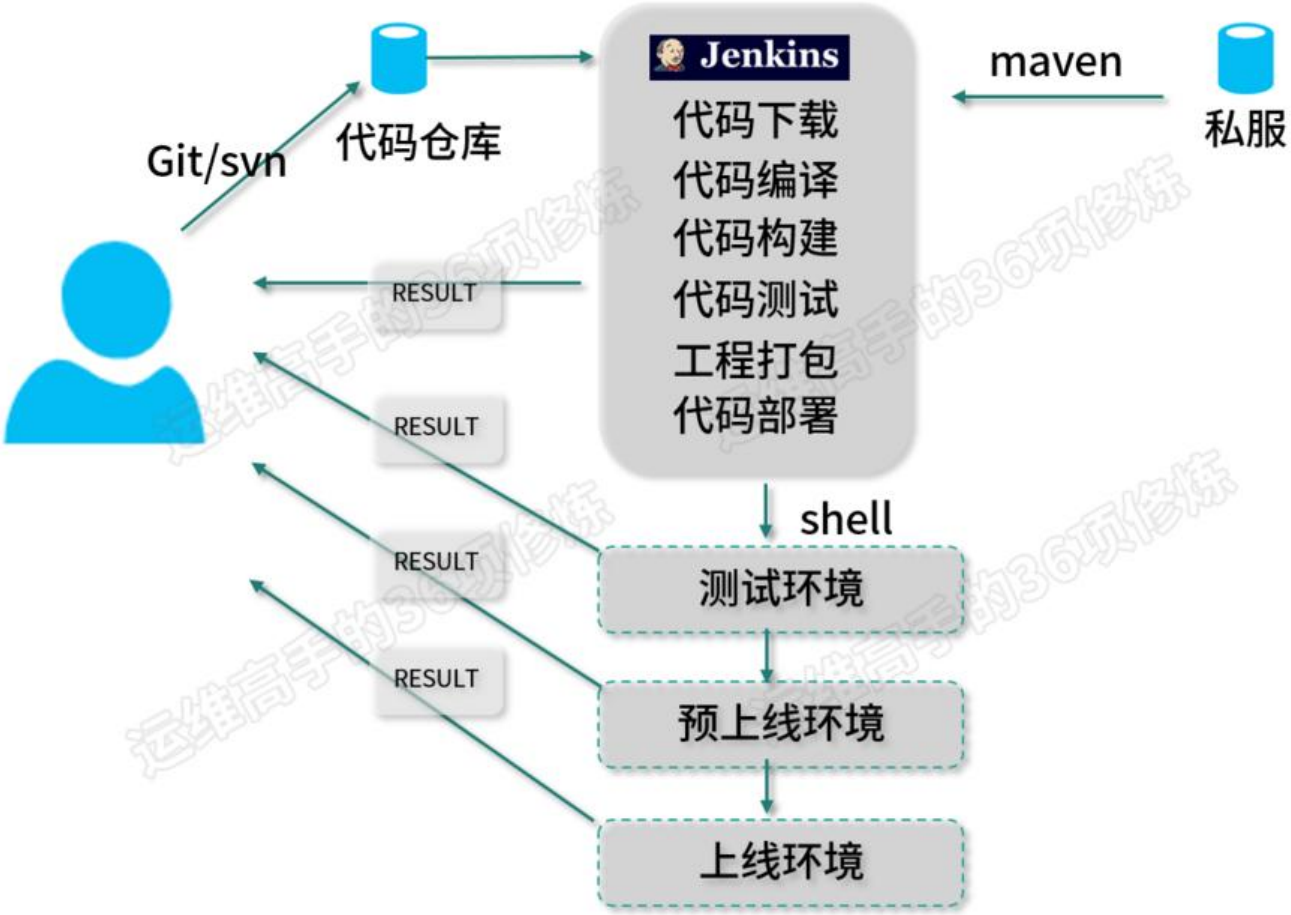
这就是持续集成和 Devops 所具备的共性要求。随着课程后续的进展，我们不仅会慢慢感受到持续集的作用，更会学习到 Devops 的详细实现方式。

四、Jenkins 的持续集成方案

接下来我们来介绍第一部分的内容：基于 Jenkins 的持续集成方案。

4.1 小型集成架构

以我对 Jenkins 的了解，首先第一个持续集成的方案，就是小型的持续集成架构。我们会看到这样一图：



整体上 Jenkins 把整套流程进行了串通，从研发上传到代码仓库，通过代码仓库自动触发代码的下载代码的编译及构建，接下来做代码的测试，并且完成工程的打包，最后进行代码的部署。

我们看到，代码的部署可以通过 Shell 脚本来实现，自动发布到预上线环境和上线等环境，多套环线打通也是在 Jenkins 中完成的。

这个架构包含这几个部分：

（1）版本库管

用 SVN 或者 Git 来做代码版本的管理，Git 和 SVN 的区别是，Git 是一个多分支的代码管理系统，常现在大部分的企业已将 SVN 改为 Git 来做代码管理。

（2）代码构建工具

Jenkins 调用的代码构建工具，如果是 Java 的相关服务，就需要调用到 maven 或者 ant 等工具来实 Java 代码的编译和构建。

（3）私服系统

企业会搭建一个自己的私服，可以代理远程仓库和部署自己或第三方构件。

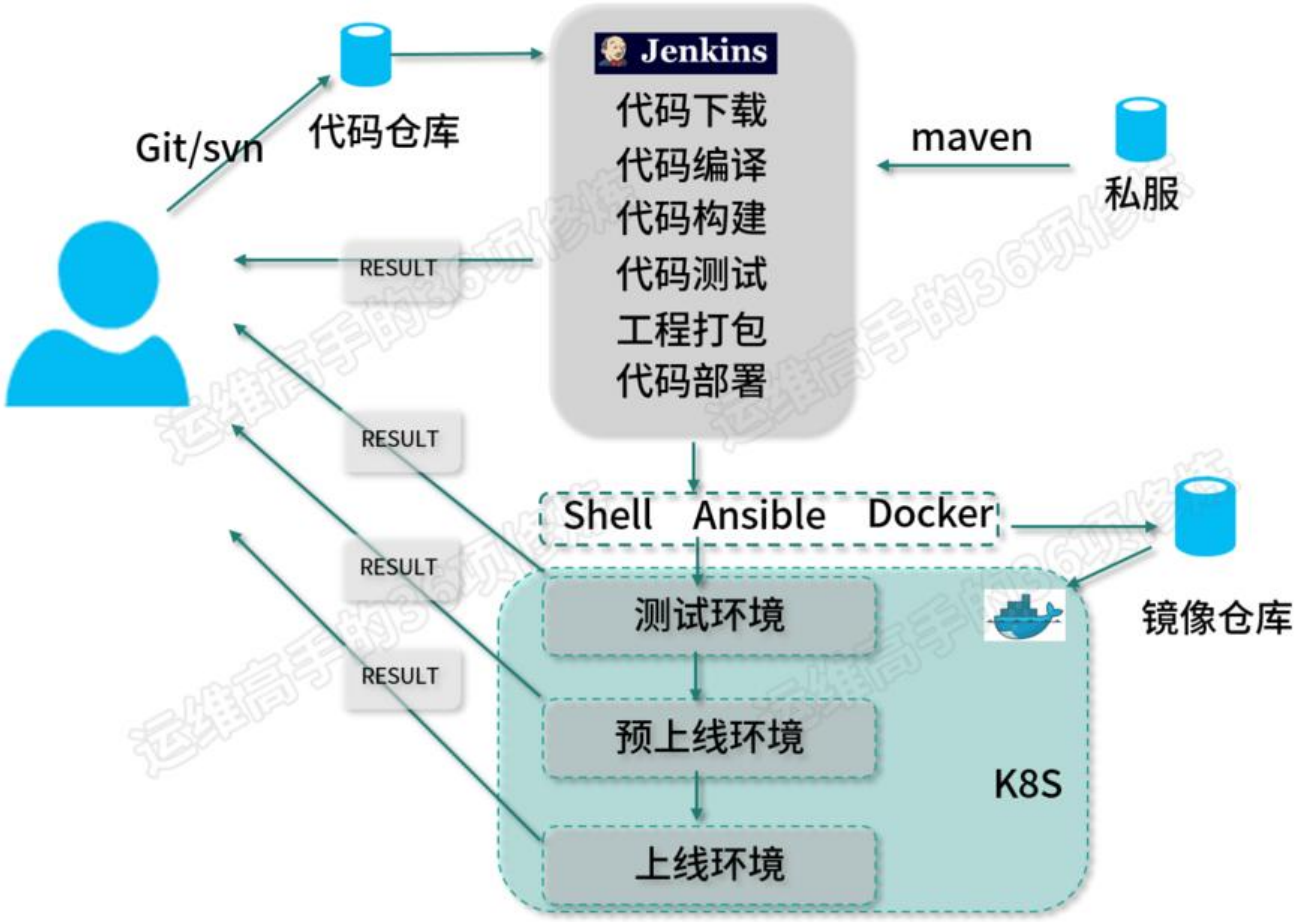
（4）Junit 和 TestLink 通常可以用为单元测试。

（5）Shell 脚本逻辑

主要为了串联发布流程各个步骤，把测试好的代码工程包通过执行 Shell 脚本发布到各个部署环境。

4.2 微服集成架构

接下来我们再看一下所谓的微服务集成架构。



微服务是当前的一个主流架构，Jenkins 很好的支持企业微服务部署环节。K8s 完美地解决了调度，载均衡，集群管理、有状态数据的管理等微服务面临的问题，成为企业微服务容器化的首选解决方案 Jenkins 支持打包部署通过 Docker 来承载应用，我们会把应用封装进镜像，然后通过 Jenkins 打包个完整的镜像并上传到镜像仓库。

另外，我们会看到在小型的集成方案里面，Shell 是有局限性的，管理起海量的服务系统，不如 Ansib

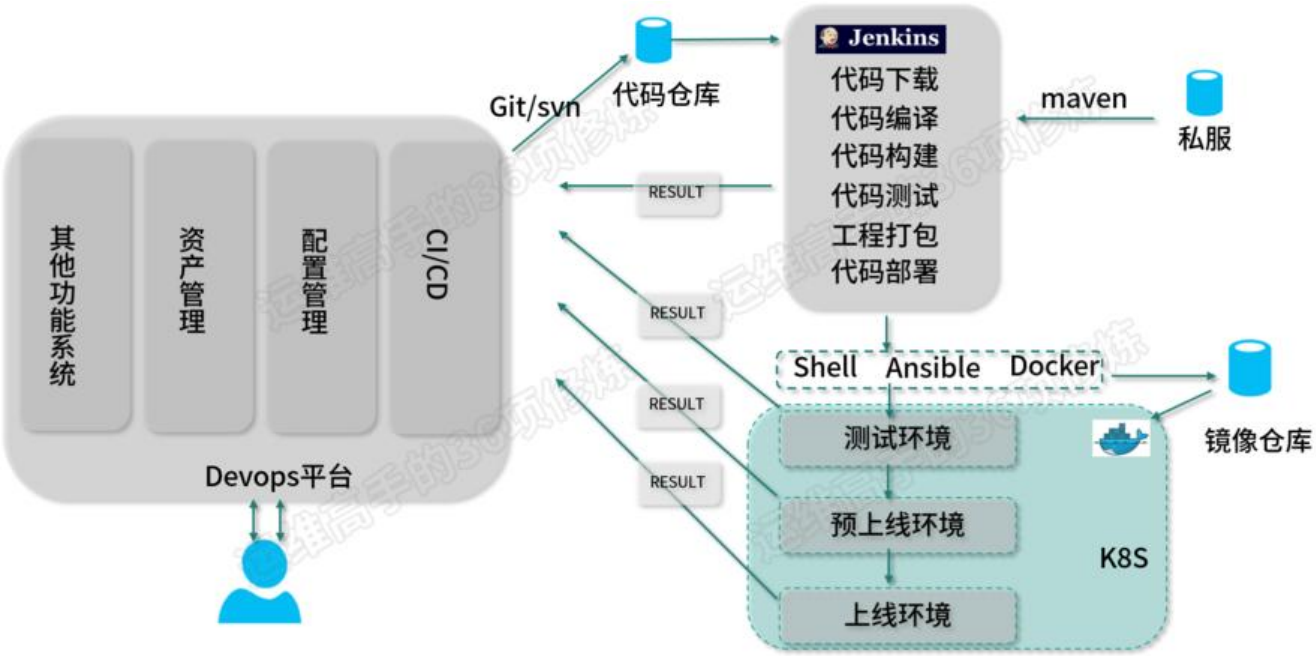
e 或者自动化执行的管理模块方便，自动化运维能力也没有那么强大。这个时候我们则可以用 Ansible 来做自动化任务。

我们会看到镜像仓库，它在这套环境中的代码发布管理流程中至关重要。我们需要把每一个镜像，按它的版本号和发布时间和主题描述来进行版本的管理，方便于发布记录及回滚，所以仓库管理也成了服务集成架构里面非常重要的一个组件。

Jenkins 在微服务的集成架构里面起着一个核心的持续集成环境平台的作用。

4.3 Devops 集成架构

最后的一张图示，就是 Devops 集成架构了：



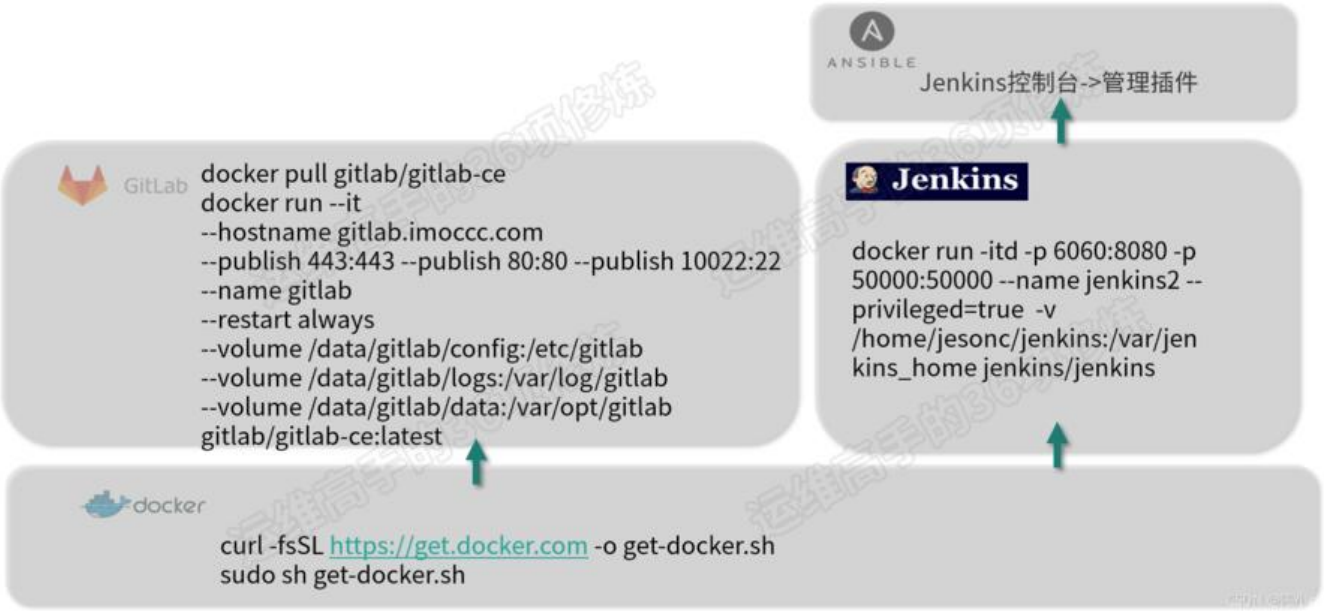
这个 Devops 平台不仅处理代码发布和自动化持续测试，更多的把运维的相关工作融入到 Devops 平中。比如企业的资产管理、配置管理、监控等相关的运维任务平台都融合到整体的一个叫 Devops 的台。

这样 Devops 优势会更加好体现到持续集成，如：它要发布的应用名称、主机信息等，都可以通过 Deops 平台直接获取，又如：持续集成平台需要调用指令也可以封装到 Devops 中。

五、关注重点问题

给你介绍了 Jenkins、Devops、持续集成相关概念，接下来我们来讲一讲 Jenkins 需要你关注的几问题。

问题一、推荐快速搭建方式



第一个问题，就是 Jenkins 本身怎么来搭建，这里先给你介绍一套通用的搭建方案，随着本章节(部模块)的内容往后进行，我们会了解到更多的 Jenkins 部署上的架构。这个通用的架构是什么样子的？这里凸现出两点：第一点就是尽可能的把 Jenkins 相关的组件来做容器化；第二点就是相关自动化任务执行尽量交给 Ansible 来处理。这个是一个比较通用的结构。

如何来快速搭建这样的一套结构呢？这里我列了一些主要用到的命令给你介绍，你可以网上查一下相的文章。

首先，我们会看到 Jenkins 整个通用的部署架构里面，GitLab 和 Jenkins 是两个开源的平台，Jenkins 是一个持续集成平台，而 GitLab 是一个基于 Git 的管理项目，它在 Git 管理工具的基础上，来做 We 的平台化的代码管理。 另外，GitLab 建议用容器的搭建方式上，且已有一些通用的镜像可以供下载所以，Jenkins 和 GitLab 我们都可以通过官方的相关镜像来直接生成容器。

我们来做部署的话，需要先安装 Docker 的环境，你可以参考官网的方式去安装 Docker 的相关环境启用 Docker 服务。而 Jenkins 和 GitLab 通过下载相应的镜像，把镜像生成为对应的容器，从而启服务。

另外，在 Jenkins 启动正常以后，我们就可以通过控制台来进行插件下载\管理并对工作流进行配置所以这是搭建一个通用平台的搭建步骤。

问题二、Jenkins 日志文件体积过大问题

你使用 Jenkins 来做持续集成需要关注的第二个问题，就是 Jenkins 产生的日志体积过大的问题。

我们知道 Jenkins 会默认将自己的运行日志都输出到 jenkins.log 中，尤其是在非容器的部署模式下常常由于日志输出过多、工作流任务过多，导致磁盘出现膨胀问题，磁盘空间被占满也会导致你的任流无法稳定运行。我们往往需要去服务器上进行日志的删除，但是这里就会遇到两个痛点：

1. 第一个痛点，就是产生的日志集中，且没有一个可切割的工具，它会导致磁盘占用率过高，需要有个方式来进行定时的清理，运维人员不可能频繁的去机器上做日志清理。

2. 第二个痛点，如果通过 rm 命令删除日志可能会遇到一个问题，就是 Jenkins 本身占用这个进程，们删除日志文件以后，因为进程依然是在运行状态的，所以它的空间并不会马上释放，需要重启 Jenkis 进程。

Jenkins 日志文件体积过大问题-解决方式

日志的切割工具 logrotate

这个时候怎么做呢？我推荐你使用一套日志的切割工具 logrotate 来做 Jenkins 的日志切割。logrotate 又是什么呢？它是 Linux 平台下的日志管理工具，用于分割日志，也就是删除旧的日志文件并且创新的日志文件，可以起到一个日志轮转的作用。

我们配置 logrotate 需要按照配置文件规范来进行操作。假设我这里有一个 jenkins.log 文件，那么这里列了一个事例来介绍一下 logrotate 的配置文件：

```
/var/log/jenkins/jenkins.log {

hourly    //日志切割频率

copytruncate  //输出的日志拷(copy)一份出来，再清空(trucate)原来的日志

missingok //包容文件没有找到的错误

rotate 8 //轮转次数及保留的文件数

compress  //是否通过gzip压缩转储以后ß的日志文件

delaycompress  //延迟压缩

size 5G  //目标文件需要满足大于指定大小（最高优先）

}
```

在配置文件里面，第一项配置的 hourly，表示切割的频率，按小时来进行切割。第二个配置(copytrucate)就是输出日志拷贝，再清空原有日志，也就是说我会把原来的日志拷贝出来一份，然后再清空这日志文件，这样的好处就是不会直接删除日志，相当于先做一个备份。

另外一个配置 missingok，可以包容文件找不到的错误，也就是说，如果我们日文件不存在，它也不因为错误而导致中断。rotate 8 表示保留的文件份数，也就是底下会切割多少份副本做保留。后面的ompress 就是进行 gzip 压缩，我们可以把日志通过压缩的方式减少磁盘的空间占用。

后面的 delaycompress 是延迟压缩，避免 CPU 性能损耗过大可以做到延迟的压缩。size 5G 是另外条切割的规则，也就是说，我虽然没有到一个小时，但是我的目标文件已经大于 5 个 G，所以我要优满足这条规则来进行切割。

在配置好配置文件以后，我们就可以在控制台执行 **logrotate --force /etc/logrotate.d/jenkins** 令，通常我们可以把logrotate 加入到定时任务里面去执行这条命令，这样的话就可以做到定时的日轮转了。这就是日志体积过大问题我推荐的解决方式。

问题三、持续集成规范性标准定义

最后一个需要关注的问题就是需要优先定义持续集成规范，自动化实现的基础就是需要建立规范性，可能的在标准的基础上来作持续集成，这是十分重要的。那 Jenkins 在持续集成中需要定义哪些规范？这里给你罗列如下：

应用名称

1、应用名称：{应用类型}-{端口}-{应用名称} or {类型}-{应用名称}

例如：api-8080-gateway,api-gateway

首先第一个就是对于单个应用的名称，推荐给你的方式是可以按照这样的格式来定义自己的规范性。前面是 {应用类型}-{端口}-{应用名称} 或者使用 {类型}-{应用名称}，你可以用这种方式来定义每一个用的名称。

主机名称

2、主机名称：{地区}-{机房}-{开发语言}-{项目名称}-{应用类型}-{序列号}

例如：bj-yz-{java}-imoocc-api--01

第二个就是对本地所需要管理的主机资源的名称(主机名)。对于主机名的管理是非常重要的，所以在机名的定义上建议你可以使用 {地区}-{机房}-{开发语言}-{项目名称}-{应用类型}-{序列号} 来对主机定，例如可以按我这种方式来定义这一台主机的名称。

日志路径

3、日志路径：/opt/logs/{应用类型}/{应用名称}-{端口}

例如：/opt/logs/web/gateway-8080

第三个就是日志的路径，日志路径管理也是非常重要的，我们需要查看相关的日志，所以要定义好规性。如统一放到了 /opt/logs/ 目录下，每一个 {应用类型} 是根据什么来进行分门别类，然后 {应用称}-{端口} 来做这一级目录。所以对于应用日志这一级的路径，你可以参考这样的一个规范来进行定。

代码路径

4、代码路径：/opt/apps/{应用名称}

例如：/opt/apps/api-8080-gateway

还有就是代码的路径，代码的路径存放也要遵循整体的规范性定义，这里假设我把所有的应用都放到了 /opt/apps/ 这个目录下。这个时候就可以按 {应用名称} 为每个应用进行归纳，所以我们可以参考这的一种方式来规范代码路径。

tomcat 实例路径

5、tomcat 实例路径：/opt/tomcats/{应用名称}

假设你是用 Java 语言代码，并且用的是 tomcat 来做的 servlet/JSP容器，如果你在主机上面有多个 tmcat 的话，那么需要你定义好它的实例名称及路径，这里也给你推荐一个定义方式。

Jenkins job命名

6、Jenkins job命名: {环境}{项目名称}{应用名称}

如：TEST_IMOOCC1_api-8080-gateway

还有就是 Jenkins job，同样我们也需要给它定义好相关的名称，方便我们来进行管理，这里推荐你采用环{环境}{项目名称}{应用名称}，如这样的一种方式来定义 job 名称。

这样定义好以后，会使持续集成的整套开发过程和管理过程更加的清晰，所以标准的定义是在整套持集成里面我们需要去做的至关重要的一点。