

22 | 持续交付流水线软件构建难吗？有哪些关键问题？

2018-02-07 赵成

赵成的运维体系管理课

[进入课程 >](#)



讲述：黄洲君

时长 08:39 大小 4.95M



上期文章我们介绍了需求分解与应用对应的管理方式，以及提交环节的开发协作模式，今天我们详细介绍一下提交阶段的构建环节，也就是我们经常提到的代码的编译打包。

构建环节

由于静态语言从过程上要比动态语言复杂一些，代码提交后，对于 Java 和 C++ 这样的静态语言，我们要进行代码编译和打包。而对于 PHP 和 Python 这样的动态语言，就不需要编译，直接打包即可。

同时，编译过程就开始要依赖多环境以及多环境下的配置管理，并根据不同的环境获取不同的配置，然后打包到最终的软件发布包中。

下面我就结合自己的实践经验，以 Java 为例，对构建环节做下介绍。

构建过程中我们要用到以下**4 种工具**：

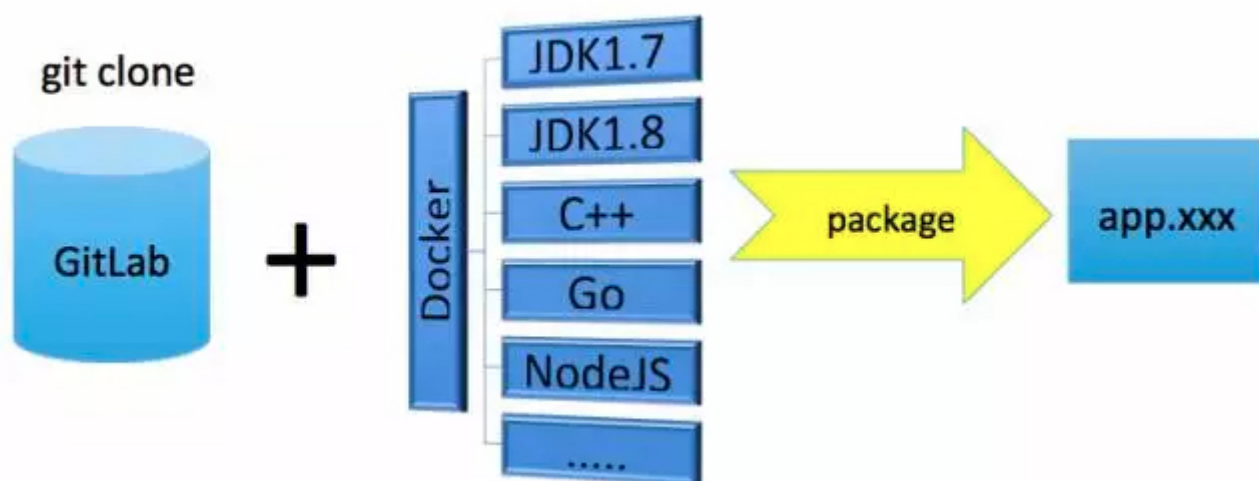
Gitlab，代码管理工具，也是版本管理工具；

Maven，依赖管理和自动化构建工具，业界同类型的工具还有 Gradle 等；

Docker，用来提供一个干净独立的编译环境；

自动化脚本和平台，自动化构建的任务我们使用 Python 脚本来实现代码获取、编译执行、软件包生成等。

具体整个构建过程图示如下：



我们以 Java 为例描述如下。

1. 首先准备好 JDK 的编译镜像，这个镜像环境与线上运行环境保持一致，比如 OS 版本、内核参数以及 JDK 版本等基础环境。当需要启动一个构建任务时，就创建一个对应的 Docker 实例，作为独立的编译环境。

2. 构建任务会根据应用配置管理中的 Git 地址，将代码克隆下来放到指定的编译目录。Docker 实例启动后，将编译目录挂载到 Docker 实例中。

3. 执行 `mvn package` 命令进行编译打包，最终会生成一个可发布 war 的软件包。同样的，对于 C++、Go、Node.js，也会准备好类似的编译镜像。不同的是，打包时，对于

C++ 中的 cmake 和 make, Go 中的 go install 等等, 最终也会生成一个可发布的软件包。

4. 构建完成后, 生成软件包放到指定构件库目录, 或者直接发布到 maven 的构件库中管理, 然后将 Docker 实例销毁。

上述就是一个完整的构建过程。在这里, 你一定会有一些疑问, 那么, 我先回答几个比较常见的问题, 欢迎你留言和我继续讨论。

几个关键问题

1. 配置文件如何打包?

这个问题, 我们在前面持续交付的多环境配置管理文章中, 已经详细介绍过。这里我们结合构建过程, 再介绍一下。

在上述第 3 个步骤中, 我们要进行代码编译。按照持续交付理念, 软件只需打包一次就可以各处运行, 这对于代码编译是没有问题的, 但是对于一些跟环境相关的配置就无法满足。

比如, 我们前面讲到, 不同的环境会涉及到不同的配置, 如 DB、缓存。而且, 其他公共基础服务在不同环境中也会有不同的地址、域名或其他参数配置。

所以, 我们就需要建立环境与配置之间的对应关系, 并保存在配置管理平台中, 至于如何做, 大家可以参考前面多环境配置管理的文章。

这里我们回到打包过程上来。

在做构建时, 我们是可以确认这个软件包是要发布到哪个环境的。比如, 按照流程, 当前处于线下集成测试环境这个流程环节上, 这时只要根据集成测试环境对应的配置项, 生成配置文件, 然后构建进软件包即可。如果是处于预发环境, 那就生成预发环境对应的配置文件。

在我们的实际场景中, 多个环境需要多次打包, 这与我们持续交付中只构建一次的理念相悖。这并不是有意违背, 而是对于 Java 构建出的交付件, 最终无论生成的是 war 包, 还是 jar 包, 上述提到的跟环境相关的配置文件, 是要在构建时就打入软件包中的。

而且在后续启动和运行阶段，我们是无法修改已经构建进软件包里的文件及其内容的。这样一来，配置文件无法独立发布，那么就必须跟软件包一起发布。所以，在实际场景下，我们要针对不同环境多次打包。

那么，我们如何确保多次打包的效果能够和“只构建一次”理念的效果相一致呢？

这就还是要依赖我们前面介绍的各个环节的建设过程，主要有以下 3 个方面：

代码提交。通过分支提交管理模式，每次构建都以 master 为基线，确保合入的代码是以线上运行代码为基础的。且前面的发布分支代码未上线之前，后续分支不允许进入线上发布环节，确保发布分支在多环境下是同一套代码。

编译环境统一。上述过程已经介绍，编译环境通过全新的 Docker 容器环境来保证。

配置管理。前面介绍到的多环境配置管理手段，通过模板和 auto-config 的配置管理能力，确保多环境配置项和配置值统一管理。

至此，一个完整的软件构建过程就完成了。可以看到，如果充分完善前期的准备工作，在做后期的方案时就会顺畅很多。

2. 为什么用 Docker 做编译环境的工具？

Docker 容器很大的一个优势在于其创建和销毁的效率非常高，而且每次新拉起的实例都是全新的，消除了环境共用带来的交叉影响。而且对于并发打包的情况，Docker 可以快速创建出多个并行的实例来提供编译环境，所以无论在效率上还是环境隔离上，都有非常好的支持。

你可以尝试一下我的这个建议，确实会非常方便。

3. 为什么不直接生成 Docker 镜像做发布？

在使用 Docker 容器做编译的过程中，我们最终取得的交付件模式是一个 war 包，或者是一个 jar 包，这个也是我们后续发布的对象。

可能有读者会问：为什么不直接生成 Docker 镜像，后续直接发布镜像？

这确实是一个好问题。如果单纯从发布的维度来看，直接发布镜像会更方便，更高效。不过，在现实场景下，我们应该更全面地看问题。

早期我们曾有一段时间使用 OpenStack+Docker 的模式进行物理机的虚拟化，以提升资源利用率。这实际上是将容器虚拟机化。

也就是说，虽然 Docker 是一个容器，但是我们的使用方式仍然是虚拟机模式，要给它配置 IP 地址，要增加很多常用命令比如 top、sar 等等，定位问题需要 ssh 到容器内。

这里一方面是因为基于 Docker 的运维工具和手段没有跟上，当时也缺少 Kubernetes 这样优秀的编排工具；另一方面，我们之前所有的运维体系都是基于 IP 模式建设的，比如监控、发布、稳定性以及服务发现等等，完全容器化的模式是没有办法一步到位的。

所以，这里我们走了个小弯路：容器虚拟机化。那为什么我们不直接使用虚拟机，还能帮我们省去很多为了完善容器功能而做的开发工作？所以一段时间之后，我们还是回归到了 KVM 虚拟机使用方式上来。

这样也就有了上述我们基于虚拟机，或者更准确地说，是基于 IP 管理模式下的持续交付体系。

经过这样一个完整的持续交付体系过程后，我们总结出一个规律：

容器也好，虚拟机也罢，这些都是工具，只不过最终交付模式不一样。但是前面我们所讲的不管是标准化、多环境、配置管理等等这些基础工作，无论用不用容器都要去做。而且，容器的高效使用，一定是建立在更加完善和高度标准化的体系之上，否则工具只会是越用越乱。

关于持续交付流水线软件构建方面的内容，我们今天先分享到这里，欢迎你留言与我讨论。

如果今天的内容对你有帮助，也欢迎你分享给身边的朋友，我们下期见！


赵成的运维体系管理课

带你直击运维的本质

赵成

美丽联合集团技术
服务经理



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 21 | 人多力量大vs.两个披萨原则，聊聊持续交付中的流水线模式

下一篇 23 | 持续交付中流水线构建完成后就大功告成了吗？别忘了质量保障

精选留言 (9)

写留言



喜剧。

2018-02-10

2

如果多套环境多套编译感觉在效率上有点低。针对java程序，我们能否统一打成jar包，然后利用启动参数来控制你是在使用什么样的配置文件呢？如果使用的是微服务架构，我们是否最好将配置文件剥离出来，使用配置中心来管理？

展开

作者回复: 有很多配置是启动时就会依赖的，特别是跟环境相关的这些，比如数据库连接不上可能就直接启动异常了，进程无法正常启动，那读取配置中心的配置也就无从谈起了。



赵丹

2018-05-15



1

我们公司用的svn进行代码管理，jenkins，ant工具实现的自动化构建部署，我们使用的工具部署方式需要调整吗，怎么调整会更好些，希望老师给点建议。

展开 ▾



海格

2018-04-04



1

我们就是使用的配置中心，编译完打包成docker镜像，然后调用接口进行部署，启动容器，在启动容器的时候会传入几个参数，会先拉取配置文件，然后在运行jar包。

展开 ▾



Adam

2019-03-26



我是这样做的。pod里面initcontainer配置一个拉取config的程序（程序得开发写好封装成镜像），初始化容器运行完成后才会开始运行业务容器。initcontainer传入不同的环境变量以及应用名就可以拉取对应的配置。

展开 ▾



松花皮蛋me

2019-02-17



我们是只打包一次，然后发布时根据环境将配置文件更换，配置文件和启动脚本和库都是分文件夹存放的。另外docker镜像发布是趋势，不需要先申请机器，快速扩容

展开 ▾



快乐就好

2018-08-18



现在kubernetes 可以说已经足够完善了，对于应用的发布，我们发布是直接发布应用打包到容器的容器吗？是否有什么缺陷？



森鑫鑫

2018-07-15



可能有读者会问：为什么不直接生成 Docker 镜像，后续直接发布镜像？

这里我能想到一个场景就是客户端的持续发布，这个是无法用镜像代替的。

展开 ▾



天使

2018-05-24



通过环境变量配置docker，应用读取环境变量获得应用配置；通过config server为应用提供代码配置；docker 的好处在于开发机器上可以迅速启动进行调试；依赖太多的话，启动docker的时候将依赖连到线上dev或qa环境

展开 ▾



江龙

2018-03-01



实现一套配置中心服务，每个环境独立的一套配置中心，服务起来时会相应的去对应的配置中心中动态请求配置。这样就能实现只一次构建，且代码与配置分离

展开 ▾

作者回复: 有些配置是启动时就依赖的，这种配置只能在构建时打入软件包

