

17 | 持续交付的第一关键点：配置管理

2018-01-26 赵成

赵成的运维体系管理课

[进入课程 >](#)



讲述：黄洲君

时长 09:23 大小 5.37M



今天我们来看持续交付的第一个关键点：**配置管理**。按照持续交付的理念，这里所说的配置管理范围会更广，主要有以下几个部分。

版本控制

依赖配置

软件配置

环境配置

讲持续交付，一上来就先讲配置管理，主要还是想强调：**配置管理是基础，是关键**。我们后面将要讲的每一个持续交付环节，都对配置管理有很强的依赖。这个基础工作做不好，也就

谈不上的持续交付的成功。**勿在浮沙筑高台，我们做工具平台或系统，一定要重视基础的建设。**

同时，这里还有一个前提，就是**一定要做到代码和配置的分离**。不要让配置写死在代码里，需要依靠严格的规范和约束。同时，对于那些因历史原因遗留在代码中的配置，要多花些时间和精力把配置剥离出来，做这项工作没有什么好的方法或经验，只能多上心，多投入些精力。

配置管理中，对于版本控制和依赖配置目前都有比较成熟的工具体系支持，也有丰富的实践经验供我们参考学习，下面我会做一个简要的介绍。

对于软件配置和环境配置管理，这两项配置跟我们自身的业务软件特性强相关，是整个持续交付过程的关键，我会结合我们自身的实践经验进行重点介绍和分享。

版本控制

版本控制的主要作用是保证团队在交付软件的过程中能够高效协作，版本控制提供了一种保障机制。具体来说，就是团队在协作开发代码的情况下，记录下代码的每一次变更情况。

说到这里，你是不是想到了 SVN 和 Git 这样的版本管理工具？对，其实我们每天都在接触，每天都在不停地做这个事情，所以目前看来这是一件很平常的事情。

关于这一部分我在后面的文章里会介绍关于提交阶段的实践经验。这里我们只要知道，**版本控制及其工具是必不可少的，因为这是开发团队协作最基础的工具**。现在应该很少有团队不采用版本控制的管理机制吧？

依赖管理

这里以 Java 为例，我们使用 Java 进行开发，必然会依赖各种第三方的开源软件包。同时，内部还会有不同组件的二方包。这些三方包和二方包就是一个应用编译和运行时所依赖的部分。

有过开发经验的同学肯定都知道，即使运行一个非常简单的 Web 应用，都会有大量的 jar 包依赖。如果人工去管理这些依赖，基本上是不可能的，所以就需要有依赖管理的工具。

对于 Java 来说，我们熟知的依赖管理工具有 Ant、Maven 和 Gradle。当然这些工具不仅仅提供依赖管理这样单一的能力，一般都具备以下几个能力：

二方包、三方包的仓库（Repository）管理；

依赖管理；

构建打包。

下面介绍下我自己的实践经验。因为我们的经验基础都在 Maven 上，再加上 Maven 周边有一些优秀插件和业界经验可以借鉴，比如后面将要介绍到的 AutoConfig，所以我们选取了 Maven 作为主力构建工具。

大致用法是建立一个本地 Maven 源，构建时会优先从本地源中获取依赖包，本地源中没有对应的依赖时，会从公网上下载，同时缓存到本地。这也是业界绝大多数公司采用的一种通用方案。具体如何构建打包呢？这个内容会在构建阶段进行分享。

软件配置

这里我把软件配置细化为两类：**一类是代码配置，一类是应用配置。**

1. 代码配置

我们可以这样理解，**代码配置是跟代码运行时的业务逻辑相关的**。比如应用的服务接口、并发线程数、超时时间等这些运行时参数；还有类似于业务或技术上的开关，比如商品评论是否开放、优惠时间段设置等等。

2. 应用配置

还记得我们在标准化文章中提到的应用吗？**应用配置就是应用这个对象的属性和关系信息**。我们把应用配置放到持续交付这个场景中进行分析，对于这个配置可以细分为：

应用构建时配置，比如它的编程语言、Git 地址以及构建方式等；

应用的部署配置，源代码目录、应用日志目录、Web 日志目录、临时目录、脚本目录等；

应用的运行配置，应用启停、服务上下线方式、健康监测方式等；

应用运行时与基础组件的关联关系，比如其依赖的 DB、缓存、消息以及存储的 IP 地址、域名、端口、用户名或 Token 等。

从上面这种分类方式中，应该可以体会到，我们对于配置的分类，也是基于应用生命周期的不同阶段进行分解和分析的。所以，标准化的过程也是一个持续迭代的过程。不同的场景下，一个应用可能会具备不同的属性。这个时候，如果我们无法在一开始就把这些属性梳理得清清楚楚，具备标准化的意识和思路就显得更为重要。这样，当我们遇到新场景的时候，随时可以对它做标准化分析和建模。

3. 代码配置和应用配置的区别

从上面的分析中，你有没有找出两者的区别？这里建议你暂停一下，花一分钟时间自己先想想代码配置和应用配置有什么区别，再往下看。

从区别上讲，我们可以认为代码配置是跟业务或代码逻辑相关的，动一下就会改变系统执行状态，是运行时的配置，但不依赖周边环境。而应用配置，是跟业务和代码逻辑无关的，不管你怎么动，业务逻辑是不会改变的，但是它跟环境相关。

与环境相关，按阶段分又大致可以分为两个阶段、三种情况。

第一种，软件在交付过程中，环境会不一样。比如我们正式发布软件前，会历经开发测试环境、预发环境和生产环境等等。那开发测试环境访问的 DB，跟线上访问的 DB 就不能是同一套。同时这个环境中的应用，依赖的大多是本环境内的基础组件和应用，但不是必然，原因我们后面会讲到。还有日志级别也可能不同，比如测试环境可以开 Debug 级别，但是线上是绝对不允许开 Debug 的。

第二种，软件交付上线后，线上可能会存在多机房环境，特别是有海外业务的公司，一个站点可能会在中国、北美、欧洲以及东南亚等不同区域建立当地访问的分站点；或者大型网站做了单元化，在国内也会分多机房部署，这个时候每个机房的环境配置必然不同。

第三种，软件交付后，一套软件可能交付给不同的客户，分别独立运行，比如类似 ERP、CRM 这样的软件，或者私有部署的 SaaS 服务等。不同客户的基础环境是不一样的，有的可能是 Linux，有的是 Unix，还有的可能是 Windows，这时应用配置中的各种目录、用户名等信息可能也是不一样的，软件的交付模式就取决于最终的客户环境。

对于平台类的产品，遇到第一、二种情况的可能性更大，这两种情况更多的是对周边依赖的配置不同，比如不同的服务注册中心、DB、缓存或消息等等。对于一些针对不同客户进行

私有部署的产品，可能更多的是第三种情况，这种情况就是应用的基础配置比如目录、用户名以及基础软件版本等会有不同。

我们回到代码配置和应用配置之间的区别这个问题上来。

对于代码配置，我们一般会通过 Config Server 这样专门的配置管理服务进行动态管理，在软件运行过程中可以对其进行动态调整。之所以增加这些配置，主要是让开发能够以更灵活的方式处理业务逻辑。当然，也有一些为了稳定性保障而增加的配置，比如限流降级、预案开关等等。对于前者运维不必关注太多，而后者是运维关注的重点，这个内容我们后面讲到稳定性部分会重点分享。

对于应用配置，是我们在构建软件包时就需要面对的问题。这个配置取决于环境，所以就延伸出持续交付过程中非常重要的一个配置管理：**环境配置管理**。解释一下就是，**不同环境中的应用配置管理**。

环境配置是我们在持续交付过程中要关注的重中之重，也是最为复杂的一部分。我们自己的团队在做多环境发布和管理的时候，遇到最头疼的问题就是环境配置管理，我们下一期就着重来聊聊环境的配置管理。

今天我和你分享了做持续交付的第一步：配置管理，主要包括版本控制、依赖配置、软件配置和环境配置四个部分。关于今天分享的内容，你有怎样的思考或疑问，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 16 | 持续交付知易行难，想做成这事你要理解这几个关键点

下一篇 18 | 如何做好持续交付中的多环境配置管理？

精选留言 (6)

 写留言



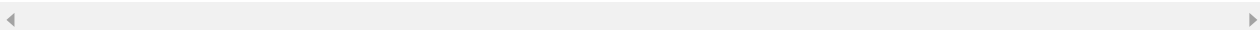
喜剧。

2018-01-29

 7

看赵老师的文章学到很多，但是还是有个小期望，希望后续能出一份运维体系的思维拓扑图。这样我们的思路会更清晰。

作者回复: 这个可以有，抽空搞一个





付盼星

2018-03-06

 1

看到这里下一步要干的事情就是把代码配置和环境配置分离，以前代码配置和环境配置都

下放给开发了，现在看来把环境配置分给运维控制更合适，比如docker启动时的参数设置，尤其是jvm

展开 ▾

作者回复: 反过来会不会更好，开发要能够自助维护，尽量不依赖运维的人，运维可以制定标准和规范

◀ ▶



haormj

2018-01-27

👍 1

代码配置和环境配置混到一起确实存在问题，不同机房的同一个应用配置都不同，而且对于注册中心，数据库，缓存的依赖都直接作为代码配置，所以当修改应用的一个参数，需要修改好多次。

展开 ▾

作者回复: 还是要区分开，因为管理方式会有不同

◀ ▶



haormj

2018-01-27

👍 1

文章写的太好了，干货~就如作者说的，版本管理已经习惯了，依赖管理确实很关键，否则没办法对代码进行进行构建，不过我们这里都是编译成二进制，之后也可以直接构建为镜像，对于代码配置和环境配置，现在我们都是没有区分，都作为代码配置了，期待作者下一篇文章。

作者回复: 对你有所帮助，很开心😊

◀ ▶



Zane

2019-01-08

👍

之前工作时频繁修改配置，隐约发觉应该区分环境的配置和业务逻辑的配置。感谢老师这篇文章提供了理论基础，坚定了我的想法。

请问老师有没有相关的课外资料供参考？想做一些扩展阅读，但没有搜索到相关信息。

◀ ▶





马天奇

2018-06-23



将配置从运行代码中剥离

展开 ▾