18 | 如何做好持续交付中的多环境配置管理?

2018-01-28 赵成

赵成的运维体系管理课 进入课程 >

讲述：黄洲君
时长 07:46 大小 5.34M

上一篇内容中，我们讲到软件配置中的代码配置和应用配置，这两种配置之间最大的区别就是看跟环境是否相关。由此，就引出了持续交付过程中最为复杂的环境配置管理这个问题，准确地说，应该是不同环境下的应用配置管理。

今天我就结合自己的经验和你聊一聊环境管理的解决方案。

多环境问题

上篇内容我们介绍了应用配置的三种情况，今天我们稍微聚焦一下，以上篇文章中提到的前两种应用配置场景为主进行介绍，也就是平台类的业务。我们一起来看同一套软件在持续交付过程中和交付上线后，多环境情况下的配置管理问题。

我们先从生命周期的角度，对环境做个简单说明，主要包括：

开发环境，主要是在应用或软件开发过程中或完成后，开发人员对自己实现的代码进行单元测试、联调和基本的业务功能验证；

集成环境，开发人员完成代码开发并自验证通过后，将应用软件发布部署到这个环境，测试人员再确保软件业务功能可用，整个业务流程是可以走通的；

预发环境，在真实的生产数据环境下进行验证，但是不会接入线上流量，这也是上线前比较重要的一个验证环节；

Beta 环境，也就是灰度环境或者叫金丝雀发布模式。为了整个系统的稳定性，对于核心应用，通常会再经历一个 Beta 环境，引入线上万分之一，或千分之一的用户流量到这个环境中；

线上环境，经历了前面几个阶段的业务功能和流程验证，我们就可以放心地进行版本发布了，这个时候就会将应用软件包正式发布到线上 。

以上是一个持续交付体系中必须要有的几个环境。环境建设，又是一个比较大的话题，我们后面会专门来讲，今天就聚焦在环境配置管理上。

不同环境下的应用配置管理

我们前面提到，环境配置管理，解释得更准确一点，应该是**不同环境下的应用配置管理**，所以这里的核心是**应用配置管理**。因为有多个环境，所以增加了其管理的复杂性。持续交付过程中涉及到应用配置管理的属性和关系如下：

应用属性信息，比如代码属性、部署属性、脚本信息等，你可以参考之前我们对这块的细分，这里就不细讲了；

应用对基础组件的依赖关系。这个也不难理解，应用对 DB、缓存、消息以及存储有依赖，不同的环境会有不同的访问地址、端口、用户名和密码。另外，在分布式架构中，一个应用必然要依赖一个环境中的其它应用，这时对于应用的服务注册、服务发现也要求必须在本环境中完成。举一个最简单的例子，我们肯定不允许一个线上应用服务注册到线下环境中去，让线下业务测试的服务调用影响到线上服务运行，要不然就会导致线上的业务故障了。

讲到这里，你应该会发现，对于我们假设的平台类业务场景，应用的基础属性信息是不会随着环境的变化而发生变化的，但是应用依赖的基础设施和依赖这个关系是会随环境不同而不

同的。所以，我们可以再进一步理解，**环境配置管理主要是针对应用对基础设施和基础服务依赖关系的配置管理。**

如果是针对不同的客户进行私有化部署的软件，那么应用的基础属性信息可能也会发生变化，但是这样的场景就更会更加复杂一些，但是配置管理上的解决思路上是一样的，所以这里我们还是简化场景。

环境配置管理解决方案

上面详细讲解了环境和应用配置之间的管理，下面就结合我自己团队和业界的一些方案，来看看这个问题应该怎样解决。

我们的示例尽量简化场景，重点是讲清楚思路。所以我们假设现在有三个环境：

开发环境

预发环境

线上环境

继续假设某应用有配置文件：config.properties，里面存储了跟环境相关的配置，简化配置如下：

缓存地址：cache.app.url

消息地址：message.app.url

数据库地址：db.app.url

支付调用地址：pay.url

支付调用 Token：pay.app.token

很明显，不同的环境，配置是不完全相同的。我们看以下几个解决思路。

方案一，多个配置文件，构建时替换。

这是一个比较简单且直接有效的方式，就是不同环境会分别定义一个配置文件，比如：

开发环境 dev_config.properties

预发环境 pre_config.properties

线上环境 online_config.properties

这几个配置文件中的配置项保持相同，但是配置的值根据环境不同是不一样的，不过都是固定的实际信息。比如开发环境配置文件中的缓存地址：

cache.app.url=10.88.77.66

而线上环境配置文件中：

cache.app.url=10.22.33.44

然后在构建时，我们会根据当前所选定的环境进行替换。比如，我们现在构建开发环境的软件包，这时就会选择 dev_config.properties 作为配置文件，并将其文件名替换为 config.properties 打包到整个软件包中。

我们看下这种方案的优缺点：

优点，就是简单直接。在环境相对固定且配置项变化不大的情况下，是最为简便的一种环境配置管理方式。

缺点，也比较明显。首先是在实际的场景中，我们的环境可能会更多，且交付上线后可能还会有线上多环境。这时每多出一个环境，就要多一个配置文件，这时配置项的同步就会成为大问题，极有可能会出现配置项不同步，配置错误这些问题。特别是如果配置项也不断地增加和变化，管理上会变得非常繁琐。再就是，这里需要针对不同环境进行单独的构建过程，也就是要多次打包，这一点是跟持续发布的指导建议相悖的。

方案二，占位符（PlaceHolder）模板模式。

这种方案在 Maven 这样的构建工具中就可以很好地支持，直接示例如下：

cache.app.url=${cache.app.url}

我们可以看到，这种模式下，配置项的值用变量来替代了，具体的值我们可以设置到另外一个文件中，比如 antx.properits（这个文件后面在 autoconfig 方案中我们还会介绍），这里面保存的才是真正的实际值。

这时我们只需要保留一个 config.properties 文件即可，没必要把值写死到每个不同环境的配置文件中，而是在构建时直接进行值的替换，而不是文件替换。这个事情，Maven 就可以帮我们做，而不再需要自己写脚本或逻辑进行处理。

不过，这个方案仍然不能很好地解决上面第一种方案提到的问题，配置文件是可以保留一个了，但是取值的 antx.properties 文件是不是要保留多个？同时，对于配置文件中配置项增加后，antx.propertis 文件中是否同步增加了配置，或者配置项名称是否完全匹配，这一点 Maven 是不会帮我们去检查的，只能在软件运行时才能验证配置是否正确。

最后，这个方案还是没有解决只打包一次的问题，因为 Maven 一旦帮我们构建完成软件包之后，它并没有提供直接针对软件包变更的方式。所以，针对不同的环境，我们仍然要打包多次。

方案三，AutoConfig 方案。

AutoConfig 是阿里开源的 Webx 框架中的一个工具包，在阿里的整个持续交付体系中被广泛应用，它继承了 Maven 的配置管理方式，同时还可以作为插件直接与 Maven 配合工作，针对我们上面提到的部分问题，它也针对性地进行了加强和改进，比如：

配置校验问题。AutoConfig 仍然是以上述第二种方案的模板模式为基础，最终通过 antx.properties 文件中的配置值来替换，但是它会做严格校验；同时也可以自定义校验规则，来检查配置项是否与模板中的设定严格匹配，如果不匹配，就会在构建时报错，这样就可以让我们提前发现问题，而不是软件包交付到环境中运行时才发现。

只打包一次的问题。AutoConfig 不需要重新构建就可以对软件包，比如 war 包或 jar 包的配置文件进行变更，所以它很好地解决了针对不同环境需要重复构建的问题。但是，比较遗憾的是，它的 Maven 插件模式并没有解决这个问题，还需要与 AutoConfig 工具模式配合才可以。

讲到这里，我们可以看到 AutoConfig 的方案已经相对完善了，也可以满足我们的大部分需求，但是它仍然没有解决多环境配置值管理的问题，我们是通过多个 antx.properties 文件来管理，还是有其它方式？

这里，我们**就需要基于 AutoConfig 做一下二次开发了**，也就是我们要把这些配置项做到一个管理平台中，针对不同环境进行不同值的管理，然后根据 AutoConfig 的规则，在变

更后生成对应不同环境的配置文件，然后再结合 AutoConfig 针对配置管理文件的能力，这样就可以很方便地做多环境的软件包构建了。

这样的配置项管理平台，AutoConfig 自己也没有做，所以需要我们自己开发。同时，对于比较敏感的配置信息，特别是涉及用户名、Token、核心 DB 地址等信息，还是不要放到配置文件中，最好是放到管理平台中，进行受控管理。同时，这里也要特别强调，密码信息一定不允许放在配置文件中出现，更不允许以明文的方式出现，这一点是需要开发、运维和安全共同来保障的。

总结

今天我们针对多环境的配置管理进行了分享，这里更多的还是思路和方案上的引导。如果你对 Maven 和 AutoConfig 不熟悉的话，建议自行查询资料进行学习了解，甚至是自己动手尝试一下。

另外，对于文章中的方案，我是尽量简化了场景来分享的，虽然思路上是相通的，但是实际情况下各种细节问题会更繁琐，要具体问题具体分析。

你在这个过程中遇到过什么问题？有什么好的解决方案？或者还有什么具体的疑问？欢迎你留言与我一起讨论。

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！

上一篇　17 | 持续交付的第一关键点：配置管理

下一篇　19 | 开发和测试争抢环境？是时候进行多环境建设了

精选留言 (8)

写留言

贺明
2018-05-27
2

集中用携程开源的apollo这类配置中心来管理呢？定个配置型的命名规范和集中管理中心，是不是解决之道

王德宝@me
2018-03-15
1

我们是在第一种基础上做的改进，不同环境有不同的配置文件，但不是打包替换，是不同的环境有系统的环境变量来区分，应用自然就知道加载哪和配置文件了，请老师评价这个方案

展开

作者回复: 运行时的参数这样做是没问题的，主要是启动时依赖的参数，这个必须是启动前也就是构建后的软件包中，就必须确定好的，所以这部分就要根据不同的环境构建不同的包

Tom
2018-03-05

1

淘宝的配置中心还有一个叫diamond的，webx和diamond都是基于JAVA的；现在的业务系统有多种语言，如果是其他语言比如php，python，nodejs和go，这种分布式配置中心可以只使用一个？还是说要根据每个编程语言各自做一个？

作者回复: 一个就够了，服务端提供不同的语言的client或者http接口就好

张青
2018-02-06

1

AutoConfig这种思路的静态配置中心，是目前见到最好的处理多环境配置的方案。

作者回复: 有实践经验吗？可以简单分享下，这块业界分享的确实不多。

阳生
2018-01-29

1

我最近在折腾配置中心，比较纠结服务怎么做自动注册，还是说手动配置再进行注册

作者回复: 可以先思考下你要解决的问题是什么。

无争就是y...
2018-06-14

在开源的分布式配置中心上面二次开发管理界面应该可以满足不同环境的配置问题，比如disconf

王德宝@me

2018-03-26

接我之前的留言，是应用启动时会根据不同的环境变量来加载指定的配置文件。包是同一个。

Spring有这个机制，spring.profiles.active

展开

> 作者回复: 针对spring的配置可以这样做，但是有很多配置我们并不是通过spring管理的，这套机制就没法用起来，不过这个思路还是值得借鉴的。

manatee

2018-01-30

想请问下老师，如果在同一个环境下有多个版本在进行同步开发那这个多分支的配置应该如何处理呢

> 作者回复: 每个分支，单独一套配置喽。