49 | 发布、升级与版本管理

2019-10-15 许式伟

许式伟的架构课 进入课程 >

讲述：姚迪迈
时长 08:08　大小 7.46M

你好，我是七牛云许式伟。

今天我们探讨服务治理的第一个环节：发布与升级。

在应用开发工程师完成一个版本的迭代后，他们交付的是软件新版本的源代码，这些代码存储在源代码仓库中。

一次正常的发布过程，大体分为这样几个典型的步骤：

构建：从源代码仓库检出源代码，编译出对应的目标文件，也就是我们新版本的软件。

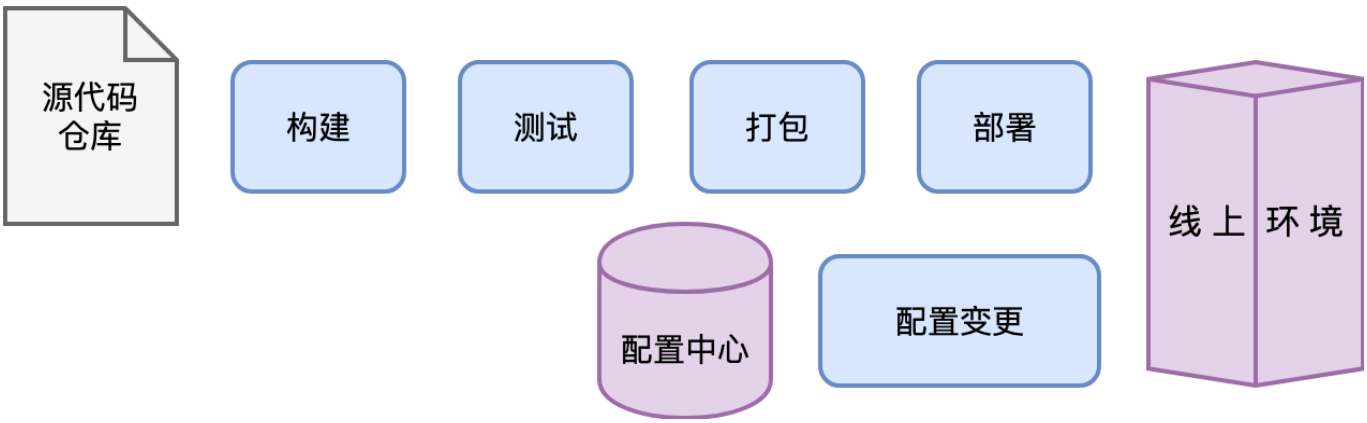
测试：对新版本的软件进行测试，以确认软件的质量符合期望。

打包：将新版本的软件及其执行所需的相关文件，比如配置文件，一起打包并记录相应的版本号。

部署：将打包好的新版本更新到线上环境。为了保证线上环境的质量，更新过程往往需要灰度，而不是一步到位直接全面切换到新版本。

当然，并不是所有的升级都是发布新版本的软件。有时候我们仅仅只是进行配置变更，也就是修改线上的配置参数。配置参数可能存在于软件配套的配置文件中，也可能存在于线上的某个配置数据库。

整个发布与升级的过程，大体可以用下图来表示。



从上面我们可以看出，发布是一个具备很强的事务特征的工作，过程很复杂。不仅如此，发布工作的心智负担也很大。所有 SRE 都应该牢牢记住以下这句七字箴言：

> 变更是故障之源。

我们应该怎么做，才能彻底解决发布与升级的问题？

让我们从 “工程师思维” 的角度，用系统化、产品化的思维来考虑这样一个复杂事务。

我们第一个要回答的问题是：我们的发布哲学是什么？

密闭性与可重复性

为保障服务可靠运行需要可靠的发布流程，我们首先要保证的是发布过程的密闭性与可重复性。

可重复性是我们的核心目标。相同的版本可以反复发布，不应该由此产生什么副作用。只有做到可重复，我们才可以安全地进行升级，或者在发现问题时安全地回滚。

要做到可重复性，就需要保证密闭性。

所谓密闭性（Hermetic），简单说就是环境的完整性。

比如，软件的源代码必须是密闭的，每次通过特定的版本号，检出内容必须是完整的，一致的且可重复的。编译的时候不需要再去任何第三方额外检出外部依赖的源代码。

再比如，从构建过程来说，同样必须确保一致性和可重复性。让两个工程师在两台不同的机器上基于同一个源代码版本构建同一个产品，构建结果应该是相同的。这意味着它不应该受构建机器上安装的第三方类库或者其他软件工具所影响。构建过程需要指定版本的构建工具，包括编译器，同时使用指定版本的依赖库（第三方类库）。编译过程是自包含的，不依赖于编译环境之外的任何其他服务。

从自动化到自服务

发布过程一方面是如此复杂，另一方面却又频繁地被执行。所以单单将发布事务做到单次发布的自动化是远远不够的。

为了应对大规模扩张，每个团队必须能够自给自足。故此，很多公司会成立工程效率团队。工程效率工程师将负责开发工程效率平台，包括发布相关的工具，制定发布的最佳实践。

这样，产品研发团队可以自己掌控和执行自己的发布流程。

每一个团队都可以决定多久或者什么时候来发布产品的新版本。发布过程可以自动化到“基本不需要工程效率工程师干预”的程度。很多项目都是利用自动构建工具和部署工具平台来自动构建、自动发布的。发布过程是真正自动化的，工程师仅仅在发生问题时才会进行干预。

这就是自服务的思想。

在这种配合模式下，团队之间配合有着清晰的边界。工程效率团队为发布平台的效率负责，产品研发团队为产品负责。用工程师土话来说，这叫 “吃自己的狗粮” 。

追求速度

以什么样的频率来发布新版本比较好?

我们认为在质量保障，能力满足的前提下，越频繁越好。

可以从两个角度来看版本发布的频率。

其一是市场竞争。产品迭代速度可以看作市场竞争力的体现。尤其是面向用户的软件，发布频率往往需要非常频繁。甚至有的团队会采用一种 “测试通过即发布（Push On Green）” 的发布方式，也就是说，发布所有通过测试的版本。

其二是工程质量。我们认为，频繁的发布可以使得每个版本之间的变更减少。这种方式使得测试、出错的调试和定位工作变得更简单。

所以，无论是从市场竞争还是工程质量管理的角度，我们都鼓励这样的版本发布哲学：

> 少量发布、频繁发布。

从数据驱动的角度，我们需要监测各种数据，尤其是我们关注的核心指标。例如，我们需要监测发布速度，也就是从代码修改提交到部署，再到生产环境一共需要多长时间。

重视质量，尊重流程

在发布流程中，有很多需要进行质量保障的环节。包括：

代码评审（Code Review），批准源代码改动；

批准创建新的发布版本，基于源代码仓库的某个版本，以及可能的少量 Bug 修改；

批准实际去部署某个发布版本；

批准配置修改。

要确保在发布过程中，只有指定的人才能执行指定的操作，而不能随随便便跳过必要的环节进行发布。另外，SRE 需要非常了解某个新发布中包含的所有具体改动，以便在发布出现问题时可以更快地进行在线调试。

这意味着，自动化发布系统需要能够整合并提供每个发布中包含的所有改动的报告，包括但不限于源代码修改的记录、Bug Issue、配置修改等等。

配置管理

配置管理在发布过程中看起来很小，但是它其实是线上不稳定性的重要来源。

配置管理随时间在不停地发展。七牛云早期通过代码仓库来管理线上环节的所有配置。这有非常大的好处，所有的配置变更就如同源代码变更一样，可以被跟踪，也可以进行严格的代码评审。

但随着集群规模的增加，这种方式的弊端也越来越突出。

最大的问题是，配置变更并不完全来源于版本发布。线上故障也会引发配置变更，比如 A 机器由于某种原因要下线，可能需要把服务迁移到 B 机器，这也会引发配置变更。

随着机器数量的增加，线上配置变更就会变得相当频繁。

基于代码仓库做配置变更管理，在应对硬件故障时显得很拙劣。在理想情况下，硬件故障的响应应该是免操作的，不需要 SRE 进行任何操作。

有两个方式可以解决这个问题。

方式一是引入配置中心，把有些高频的配置变更支持做到应用逻辑中去。服务治理中有一个子课题叫 “服务发现”，就是基于这样的思想。

方式二是将配置管理与物理硬件环境彻底进行解耦，这也是数据中心操作系统（DCOS）在做的事情。本质上，你也把它理解成是将高频的配置变更支持做到应用逻辑中，只不过这由一个基础平台来实现罢了。

结语

今天我们探讨服务治理的第一个环节：发布与升级。它包括了以下这些子过程：

构建；

测试；

打包；

部署；

配置变更。

我们并没有探讨具体的发布与升级系统怎么做，虽然业界针对发布的各个环节其实都有蛮多的实作案例。如果你正在评估应该采纳什么样的系统，可以结合我们今天探讨的发布哲学来进行评估。

发布系统非常复杂，有很大的事务工作量。要做到高效的发布能力，工程师思维是关键性的支撑，我们需要坚持以系统化的思维来彻底解决发布问题。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。下一讲我们聊聊“日志、监控与报警”。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

上一篇　48 | 事务与工程：什么是工程师思维？

精选留言 (6)

写留言

葫芦娃
2019-10-15

“SRE 需要非常了解某个新发布中包含的所有具体改动，以便在发布出现问题时可以更快地进行在线调试”——发布出现问题还是应该先回退版本，恢复服务吧，调试定位问题感觉应该业务开发来做，SRE通常也无能为力，如果是devops就没什么可推脱了

展开

作者回复: 是这样，出问题首先第一要务应该先恢复服务，但是有可能的话还应该尽可能保留现场，所以把流量切走是更好的做法。

2

Aaron Cheung
2019-10-15

七牛云项目发布是sre还是软件开发工程师自己发布呢

作者回复: sre

1

Fs
2019-10-19

这篇比较简单，事务性介绍

诗泽
2019-10-18

看了上一节和这一节内容对于“事务性工作”还是不太理解，老师可以详解一下吗？

曹龙
2019-10-15

收获满满

风清扬

2019-10-15

老师，发布升级版本管理后面会有详细讲解吗？光讲解原理，没实际操作，很难有具体的收获。