

18 | 蓝绿红黑灰度发布：这些五颜六色的发布到底怎么用？

2019-10-02 葛俊

研发效率破局之道

[进入课程 >](#)



讲述：葛俊

时长 16:09 大小 14.80M



你好，我是葛俊。今天，我来和你聊聊最近流行的一些部署、发布方法，以及测试右移。

最近几年，我见到了很多跟颜色相关的部署、发布方法，比如蓝绿部署、红黑部署、灰度发布等。今天，我会首先与你分享它们的基本定义和要解决的根本问题；然后，与你一起深入看一看高效应用这些方法的基本原则，以及一些具体的实践。

各种部署方式的定义

我们先来看看蓝绿部署（Blue-green Deployment）、红黑部署（Red-black Deployment）和灰度发布（Gray Release，或 Dark Launch）的定义和流程吧。

蓝绿部署

蓝绿部署，是采用两个分开的集群对软件版本进行升级的一种方式。它的部署模型中包括一个蓝色集群 A 和一个绿色集群 B，在没有新版本上线的情况下，两个集群上运行的版本是一致的，同时对外提供服务。

系统升级时，蓝绿部署的流程是：

首先，从负载均衡器列表中删除集群 A，让集群 B 单独提供服务。

然后，在集群 A 上部署新版本。

接下来，集群 A 升级完毕后，把负载均衡列表全部指向 A，并删除集群 B，由 A 单独提供服务。

在集群 B 上部署完新版本后，再把它添加回负载均衡列表中。

这样，我们就完成了两个集群上所有机器的版本升级。

红黑部署

与蓝绿部署类似，红黑部署也是通过两个集群完成软件版本的升级。

当前提供服务的所有机器都运行在红色集群 A 中，当需要发布新版本的时候，具体流程是这样的：

先在云上申请一个黑色集群 B，在 B 上部署新版本的服务；

等到 B 升级完成后，我们一次性地把负载均衡全部指向 B；

把 A 集群从负载均衡列表中删除，并释放集群 A 中所有机器。

这样就完成了一个版本的升级。

可以看到，**与蓝绿部署相比，红黑部署只不过是充分利用了云计算的弹性伸缩优势**，从而获得了两个收益：一是，简化了流程；二是，避免了在升级的过程中，由于只有一半的服务器提供服务，而可能导致的系统过载问题。

至于这两种部署方式名字中的“蓝绿”“红黑”，只是为了方便讨论，给不同的集群取的名字而已，通过不同颜色表明它们会在系统升级时运行不同的版本。

灰度发布

灰度发布，也被叫作金丝雀发布。与蓝绿部署、红黑部署不同的是，**灰度发布属于增量发布方法**。也就是说，服务升级的过程中，新旧版本会同时为用户提供服务。

灰度发布的具体流程是这样的：在集群的一小部分机器上部署新版本，给一部分用户使用，以测试新版本的功能和性能；确认没有问题之后，再对整个集群进行升级。简单地说，灰度发布就是把部署好的服务分批次、逐步暴露给越来越多的用户，直到最终完全上线。

之所以叫作灰度发布，是因为它介于黑与白之间，并不是版本之间的直接切换，而是一个平滑过渡的过程。

之所以又被叫作金丝雀发布，是因为金丝雀对瓦斯极其敏感，17 世纪时英国矿井工人会携带金丝雀下井，以便及时发现危险。这就与灰色发布过程中，先发布给一部分用户来测试相似，因而得名。

好了，以上就是几种有颜色的部署发布方式了。如果你还有哪些地方理解得不够透彻，可以去网络上搜索相关文章，或者直接给我留言吧。接下来，我将继续按照黄金圈法则，来帮助你深入了解这些部署、发布方式的 Why、How 和 What。

蓝绿、红黑部署和灰度发布的 Why

究其根本，这些部署、发布方法，是为了解决频繁发布的生产模式带来的两个问题：

减少发布过程中新旧服务切换造成的服务停止时间。蓝绿部署和红黑部署都能实现无宕机时间部署（0 downtime deployment）。

控制新版本发布因为质量问题带来的风险。灰度发布就是一个例子。

蓝绿红黑灰度发布的 How

实现这两个目标的基本原则，是把服务上线过程拆分为部署、发布和发布后 3 个阶段，并充分利用这 3 个阶段的特点来提高服务上线的效率、质量和安全性。

这 3 个阶段的详细定义和特点分别是：

部署（deploy），指的是我们把一个代码包拷贝到服务器上运行，但并不把它暴露给用户，也就是并不给用户提供服务。这个阶段比较耗时，但因为还没有面向用户，所以风险很小。

发布（release），是把部署好的服务暴露给用户的过程，也就是开始真正上线服务用户了。这个过程可以通过负载均衡的切换很快实现，但风险很大，一旦出现问题损失就会比较大。

发布后（post-release），指的是服务完全上线以后的阶段。因为产品已经完全上线，我们的主要工作不再是预防，而是变成了监控和降低损失。

以红黑部署为例，从开始在新生成的集群 B 上部署新的版本，到线上的流量通过负载均衡指向 B 之前，是处于部署阶段；而负载均衡从 A 指向 B 的过程，就是发布阶段；等到负载均衡完全指向 B 之后，就进入了发布后阶段。

部署、发布、上线这几个名词，其实区分不太明显，我们平时在讨论服务部署上线时，也经常会混用。在这里，我之所以要和你明确区分这几个阶段，是因为我们可以针对每个阶段的特点来实现两个目标：

提高上线产品的效率，也就是减少发布过程中新旧服务切换造成的服务停止时间。

提高上线产品的安全性，也就是控制新版本引入的质量问题。

蓝绿、红黑部署和灰度发布 What

关于提高上线产品的效率，实践主要有两个：一是利用负载均衡切换线上流量，二是使用功能开关切换线上流量。这两种方法都比较简单。

而提高上线产品的安全性，相对来说就比较复杂了，但又很重要。因为在敏捷、持续交付等开发模式愈发流行的今天，产品的研发节奏越来越快，我们必须在上线过程中，在生产环境上进行更多的测试，以保证产品质量。

讲到这里，你可能一下就想到了，这正是我们在上一篇文章中提到的测试右移要做的工作。接下来，我就与你分别介绍如何在部署、发布、发布后这 3 个阶段提高上线产品的安全性，也就是测试右移的实践。

部署阶段的实践

在部署阶段，因为服务还没有真正面对用户，所以比较安全。在这一步，我们可以尽量运行比较多的检验。但**一定要注意的是**，我们在运行检验的时候，不能产生副作用，也就是不能影响到正在给用户提供服务的系统。具体来说，我们可以运行集成测试、流量镜像（shadowing，也叫作 Dark Traffic Testing or Mirroring）、压测和配置方面的测试这 4 种检验。

第一种检验是，集成测试。

集成测试，指的是对模块之间的接口，以及模块组成的子系统进行的测试，介于单元测试和系统测试之间。

传统的集成测试是在测试环境或类生产环境上进行的。这种方式的问题在于，测试运行的环境和生产环境差别较大，不容易发现生产环境可能会出现的问题。一个最典型的原因是，在这些非生产环境上，只有测试用例在运行，没有在处理任何真实的用户请求，所以在生产环境中运行集成测试，才可能发现在非生产环境上难以发现的问题。

在具体进行集成测试的时候，如果所做的操作没有产生数据，也就是不会产生副作用，会比较安全。如果产生了数据，我们一般有两种处理方法：

第 1 种方法是，对测试产生的数据添加一个“测试”标签。同时代码的逻辑，对有测试标签的数据都进行特殊处理，比如说完全忽略。

第 2 种方法是，对测试用例产生的请求，就直接不写数据。具体实现方法是，在业务里直接添加这个特殊处理的逻辑。如果你使用的是服务网格（Service Mesh），则可以使用服务的代理（比如 Sidecar Proxy）来进行处理。

第二种检验是，流量镜像。

流量镜像，指的是对线上流量的全部或者一部分进行复制，并把复制的流量定向到还没有面向用户的服务实例上，从而达到使用线上流量进行测试的效果。

关于引流实现，通常是使用代理，比如 Envoy Proxy 和 Istio 配合使用。如果你想深入了解引流的实现方式，可以参考 [“使用 Envoy 做镜像引流”](#) 这篇文章。

需要注意的是，使用引流进行测试时，不能给生产环境带来副作用。具体办法与集成测试的处理方法类似，我们也可以给引流产生的数据打标签，在流量复制的时候，对复制的请求统

一添加一个特殊字段（比如 shadow），从而让接收到请求的服务可以对其进行特殊处理。

使用流量镜像，除了普通的检测之外，还有一个比较有用的实践就是，对测试流量与实时服务流量的运行结果进行对比，来检查新服务的运行是否符合预期。Twitter 在 2015 年开源了一款这样的代理工具 [Diffy](#)，它可以在镜像流量的同时调用线上服务和新服务，并对结果进行对比。

第三种检验是，压测。

压测，也是在部署阶段比较有价值的一种测试方法。比如，我们可以把新服务部署到一个比较小的集群上，然后把线上环境的流量全部复制并指向这个新集群，以相对客观地了解最新服务的抗压能力。

第四种检验是，配置方面的测试。

系统配置方面的变更，一旦出现问题，往往会给业务带来重大损失。部署阶段，因为不直接面向用户，所以是测试配置变更的好时机。

如果你想深入了解这部分内容的细节，可以参考 [Facebook 关于可靠性的见解](#) 这篇文章。

发布阶段的实践

在发布阶段，我们可以使用金丝雀发布和监控两种方法，来及早发现错误，并减少错误带来的损失。

第一个方法是，金丝雀发布。

金丝雀发布，是发布阶段最基本、最常见的实践。这里，我两个小贴士：

让金丝雀服务先面向内部用户，也就是 Dogfooding，来降低出现问题时造成的损失。

最近几年出现的一些部署工具和平台，比如 Spinnaker，已经对金丝雀发布有了 [比较好的支持](#)。你可以考虑直接使用，降低引入成本。

第二个方法是，监控。

监控，是安全发布必不可少的关键环节，其重要性不言自明。在发布过程中，我们应该注意监测用户请求失败率、用户请求处理时长和异常出现数量这几个信息，以保证快速发现问题并及时回滚。

发布后的实践

产品成功发布之后，我们的主要工作就是监控和补救，具体实践包括三个：监控、A/B 测试和混沌工程（Chaos Engineering）。

第一个实践是，监控。

服务上线后，我们需要提供有效的监控，来了解服务的质量。关于监控的内容，我推荐参考可观察性（Observability）的三大支柱，即日志、度量和分布式追踪。如果你想深入了解这部分内容，推荐你看一下[这篇文章](#)。

第二个实践是，A/B 测试。

系统上线之后发现问题，有一个快速的补救办法是，继续使用旧的服务代码。对于这一点，我们可以通过 A/B 测试的方法来实现。

也就是说，添加风险比较大的新功能时，使用 A/B 测试让新旧功能并存，通过配置或者功能开关决定使用哪一个版本服务用户。如果发现新功能实现有重大问题，可以马上更改配置（而不需要重新部署服务），就能重新启用旧版本。

第三个实践是，混沌工程。

混沌工程，指的是主动地在生产环境中引入错误，来测试系统的可靠性的工程方法。最早为人熟知的混沌工程，是网飞（Netflix）公司的 Chaos Monkey。这种方法可以引入的错误主要包括：

- 杀死系统中的节点，比如关闭服务器；

- 引入网络阻塞的情况；

- 切断某些网路链接。

不过，一般是在公司达到了很好的稳定性之后，对稳定性有更上一层楼的需求时，或者是对稳定性要求特别高的公司，混沌工程的价值才比较大。

小结

我首先与你介绍了一些常用的部署、发布方式，包括蓝绿部署、红黑部署和灰度发布。这些方法的目的是为了解决频繁发布的生产模式带来的挑战。而解决这些挑战最基本的原则是，把服务上线的过程拆分为部署、发布和发布后 3 个阶段，并分别进行优化。

在部署阶段，我们要充分利用服务还没有暴露给用户的特点，尽量进行集成测试、压测、配置测试等检测；在发布的阶段，我们主要是采取灰度发布的方式并配合使用监控，在出现问题时，马上进行回滚；而在发布后阶段，则有监控、A/B 测试以及混沌工程等实践。

在我看来，快速发布模式没有给测试留下足够的时间，我们的确需要在部署上线的过程中，在提高产品质量上多下功夫。Spinnaker 这种原生支持灰度发布的工具的出现和流行，也正表明了这一趋势。在这种在生产环境上进行测试的方式，最关键的是要做好风险控制。

另外，这种模式给测试团队带来了非常大的挑战。我觉得，在不久的将来，传统测试方式会越来越不流行。测试团队需要尽快转型，来适应这种新的开发模式。

思考题

你觉得金丝雀发布可以用在移动端应用或者桌面应用上吗？如果可以的话，大概要怎么实现呢？

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再见！

研发效率破局之道

Facebook 研发效率工作法

葛俊

前 Facebook 内部工具团队 Tech Lead



新版升级：点击「👤 请朋友读」，20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 17 | 测试左移：测试如何应对新的开发模式？

下一篇 特别放送 | 每个开发人员都应该学一些VIM

精选留言 (1)

写留言



日拱一卒

2019-10-02

你觉得金丝雀发布可以用在移动端应用或者桌面应用上吗？如果可以的话，大概要怎么实现呢？

首先，我们目前一般都在采用红黑部署的模式，对于不重要的更新，偶尔会尝试灰度部署。灰度部署可能对于互联网类型的大流量高并发的应用更有意义，因为它们会涉及到...
展开

作者回复: @日拱一卒 同学，每一次的回答都很到位。先赞一个！

这里稍微补充一点。在移动端进行金丝雀发布还是比较普遍的。因为很多移动端APP。都是强依赖一个后端服务。可以比较方便的通过后端API来进行控制。如果是桌面版的程序，要实现金丝雀发布，前提也是需要有一个和后端服务沟通的渠道。

这里的一个技巧，是使用功能开关来控制版本的回归，而不用要求客户重新安装旧版本。

