17 | 部署管理：低风险的部署发布策略

2019-11-19 石雪峰

讲述：石雪峰
时长 17:57 大小 16.44M

你好，我是石雪峰，今天我来跟你聊聊部署管理。

在 DevOps 年度状态报告中，有四个核心的结果指标，其中仅“部署”这一项就占了两个关键指标，分别是**部署频率**和**部署失败率**。顺便提一下，另外两个指标是**前置时长**和**平均故障修复时长**。

对 DevOps 来说，部署活动就相当于软件交付最后一公里的最后一百米冲刺。只有通过部署发布，软件真正交付到最终用户手中的时候，前面走过的路才真正创造了价值。

部署和发布这两个概念，经常会被混用，但严格来说，部署和发布代表两种不同的实践。**部署是一组技术实践，表示通过技术手段，将本次开发测试完成的功能实体**（比如代码、二进

制包、配置文件、数据库等）**应用到指定环境的过程**，包括开发环境、预发布环境、生产环境等。部署的结果是对服务器进行变更，但是这个变更结果不一定对外可见。

发布，也就是 Release，更偏向一种业务实践，也就是**将部署完成的功能正式生效，对用户可见和提供服务的过程**。发布的时机往往同业务需求密切相关。很多时候，部署和发布并不是同步进行的，比如，对于电商业务来说，要在 0 点上线新的活动，那么如果部署和发布不分离，就意味着要在 0 点的前 1 秒，完成所有服务器的变更，这显然是不现实的。

那么，我想请你思考这样一个问题：所谓的低风险发布，是不是要在发布之前确保本次变更的功能万无一失了，才会真正地执行发布动作呢？

事实上，即使没这么说，很多公司也都是这样做的。传统软件工程在流程设计的时候，也是希望通过层层的质量手段，来尽可能全面地验证交付产品的质量。典型的应用就是测试的 V 模型，从单元测试、集成测试、系统测试，到用户验收，还有各类专项测试，其实都是为了在发布之前发现更多的问题，以此来保障产品的质量。

那么，在 DevOps 模式下，是否也倡导同样的质量思想呢？我觉得这是一个有待商榷的问题。

实际上，随着发布频率的加速，留给测试活动的时间越来越有限了。与此同时，现在业务的复杂度，也比十年前高了不知道多少个等级。每次发布涉及 PC 端、移动端，还有小程序、H5 等多种形态，更别提成百上千的终端设备了。要在有限的时间里，完成所有的测试活动，本来就是件很有挑战的事情。而且，各个公司都在衡量测试开发比，更是限制了测试人力投入的增长，甚至还要不断下降。

你当然可以通过自动化手段来提升测试活动的效率，但穷尽测试本来就是个伪命题。那么，明明说了 DevOps 可以又快又好，难道是骗人的吗？

当然不是。这里的核心就在于 DevOps 模式下，质量思想发生了转变。简单概括就是：**要在保障一定的质量水平的前提下，尽量加快发布节奏，并通过低风险发布手段，以及线上测试和监控能力，尽早地发现问题，并以一种最简单的手段来快速恢复。**

这里面有几个关键词：**一定的质量水平**，**低风险发布手段**，**线上测试和监控**，以及**快速恢复**。我分别来给你解释一下。

一定的质量水平

这个“一定”要怎么理解呢？对于不同形态的软件来说，质量标准的高低自然是不相同的。比如，我有一个制造卫星的同学，他们对于软件质量的要求就是要做到几年磨一剑，甚至是不计成本的。但对于互联网这种快速迭代的业务来说，大家都习惯了默认会出问题，所以在圈定测试范围和测试覆盖的基础上，只要完成严重问题的修复即可发布，低级别的问题可以在后续的众测和灰度的环节继续处理。

所以，与定义一个发布质量标准相比，更重要的随着 DevOps 的推广，扭转团队的质量观念。**质量不再是测试团队自身的事情，而是整个交付团队的事情**。如果出现了线上问题，团队要一起来定位和修复，并且反思如何避免类似的问题再次发生，从失败中学习。

而测试能力的向前、向后延伸，一方面，提供了工具和平台以帮助开发更容易地进行自测；另一方面，加强针对线上监控埋点等类型的测试，可以保证线上问题可以快速暴露，正常获取辅助分析用户行为的数据，这会全面提升整体的发布质量。

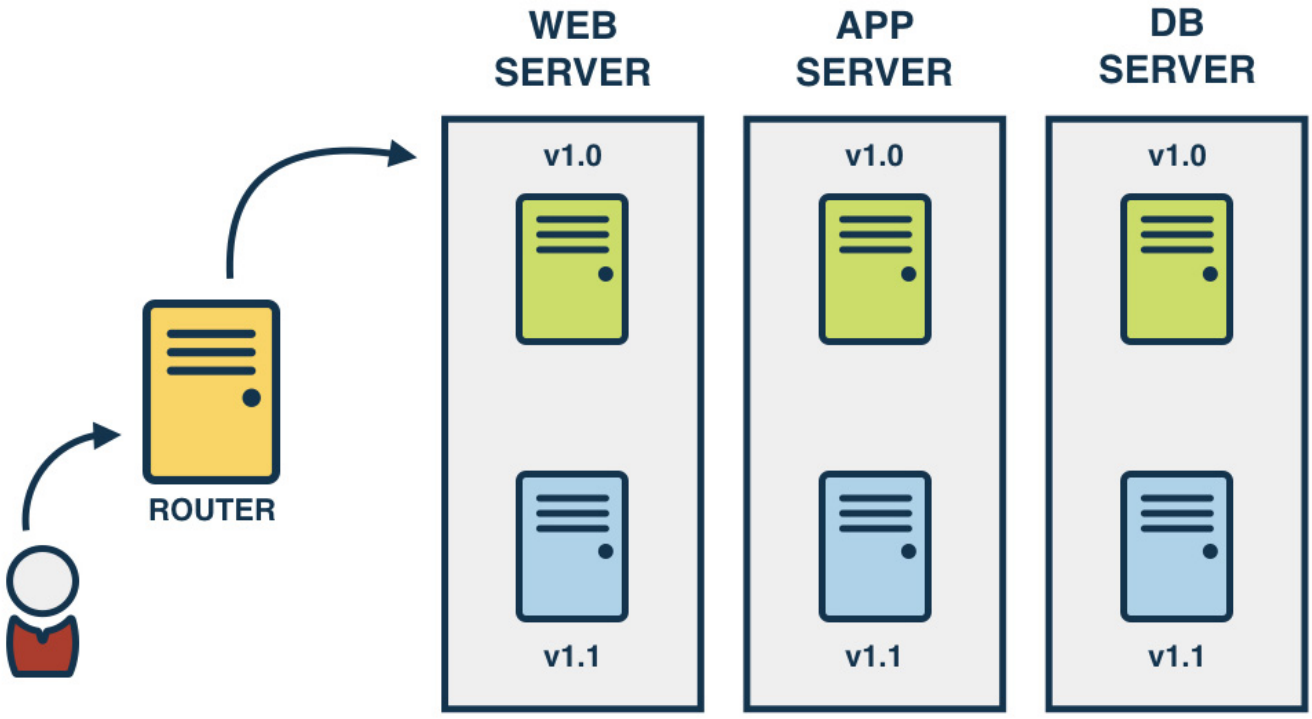
低风险的发布手段

既然发布是一件不可回避的高风险事情，那么，为了降低发布活动的风险，就需要有一些手段了。典型的包括以下几种：蓝绿部署，灰度发布和暗部署。

1. 蓝绿部署

蓝绿部署就是为应用准备两套一模一样的环境，一套是蓝环境，一套是绿环境，每次只有一套环境提供线上服务。这里的蓝和绿，只是用于区分两套环境的标志而已。在新版本上线时，先将新版本的应用部署到没有提供线上服务的环境中，进行上线前验证，验证通过后就达到了准备就绪的状态。在发布时间点，只要将原本指向线上环境的路由切换成另外一套环境，整个发布过程就完成了。

一般来说，**这种方式的实现成本比较高**。因为有两套一模一样的环境，只有一套用于真正地提供线上服务。为了减少资源浪费，在实际操作中，另外一套环境可以当作预发布环境使用，用来在上线之前验证新功能。另外，在这种模式下，数据库普遍还是采用同一套实例，通过向下兼容的方式支持多个版本的应用。



> 图片来源：https://www.gocd.org/2017/07/25/blue-green-deployments.html

2. 灰度发布

灰度发布，也叫金丝雀发布。与蓝绿部署相比，灰度发布更加灵活，成本也更低，所以，在企业中是一种更为普遍的低风险发布方式。

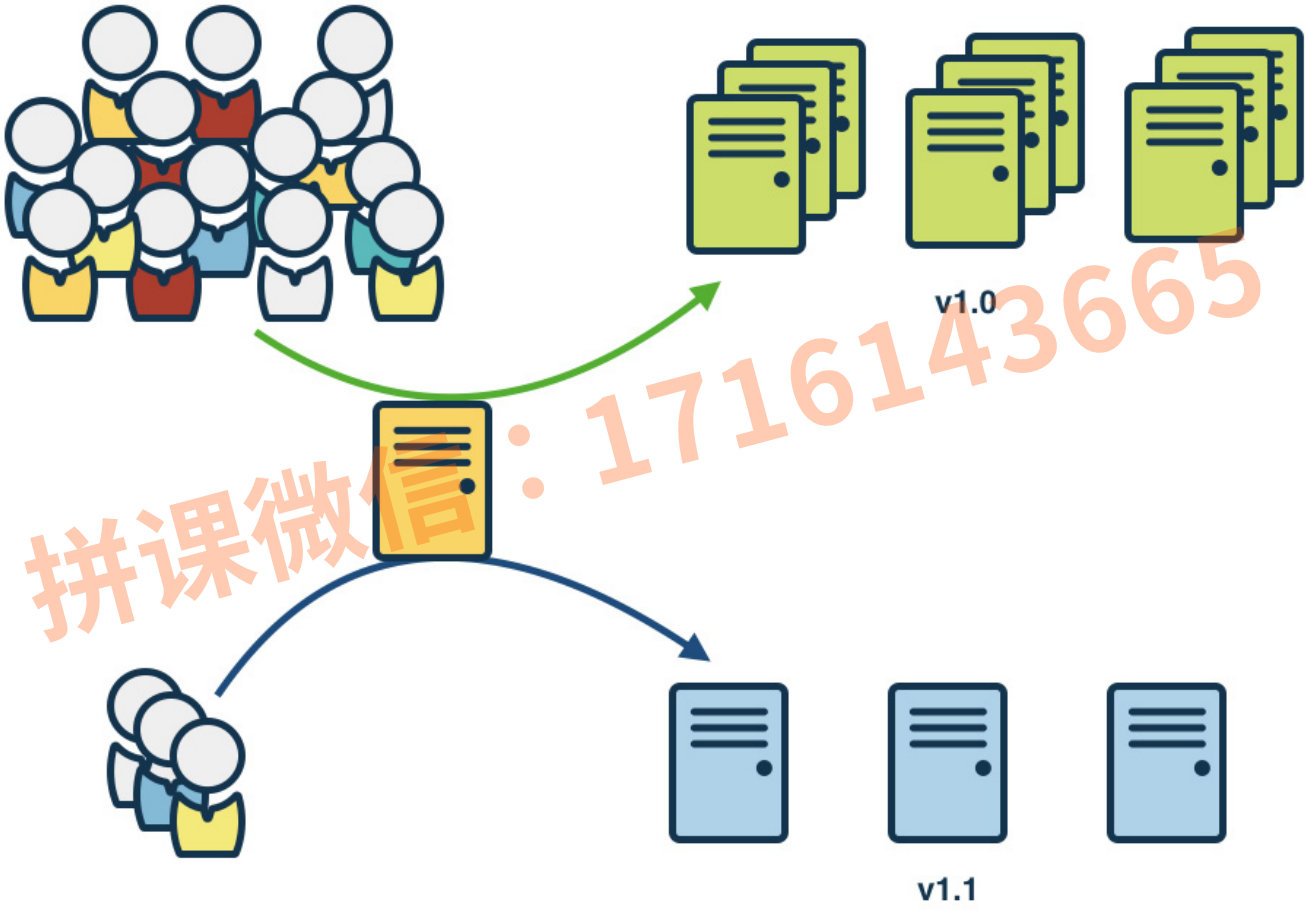
灰度发布有很多种实现机制，最典型的就是采用一种渐进式的滚动升级来完成整个应用的发布过程。当发布新版本应用时，根据事先设计好的灰度计划，将新应用部署到一定比例的节点上。当用户流量打到这部分节点的时候，就可以使用新的功能了。

值得注意的是，要保证同一个用户的行为一致性，不能时而看到新功能，时而看到老功能。当然，解决办法也有很多，比如通过用户 ID 或者 cookie 的方式来识别用户，并划分不同的组来保证。

新版本应用在部分节点验证通过后，再逐步放量，部署更多的节点，依次循环，最终完成所有节点的部署，将所有应用都升级到新版本。分批部署只是实现灰度发布的方法之一，利用配置中心和特性开关，同样可以实现指向性更强的灰度策略。比如，针对不同的用户、地域、设备类型进行灰度。

对于移动端应用来说，灰度发布的过程也是必不可少的。我以 iOS 平台应用为例，带你梳理下发布的步骤。首先，公司的内部用户可以自行下载安装企业包，进行新版本验证和试用。试用 OK 后，再通过官方的 Testflight 平台对外开启灰度，这样只有一部分用户可以收到新版本通知，并且在 Testflight 中安装新版本。灰度指标符合预期后，再开启全量用户升级。

现在很多应用都采用了动态下发页面的方法，同样可以使用特性开关，来控制不同用户看到不同的功能。



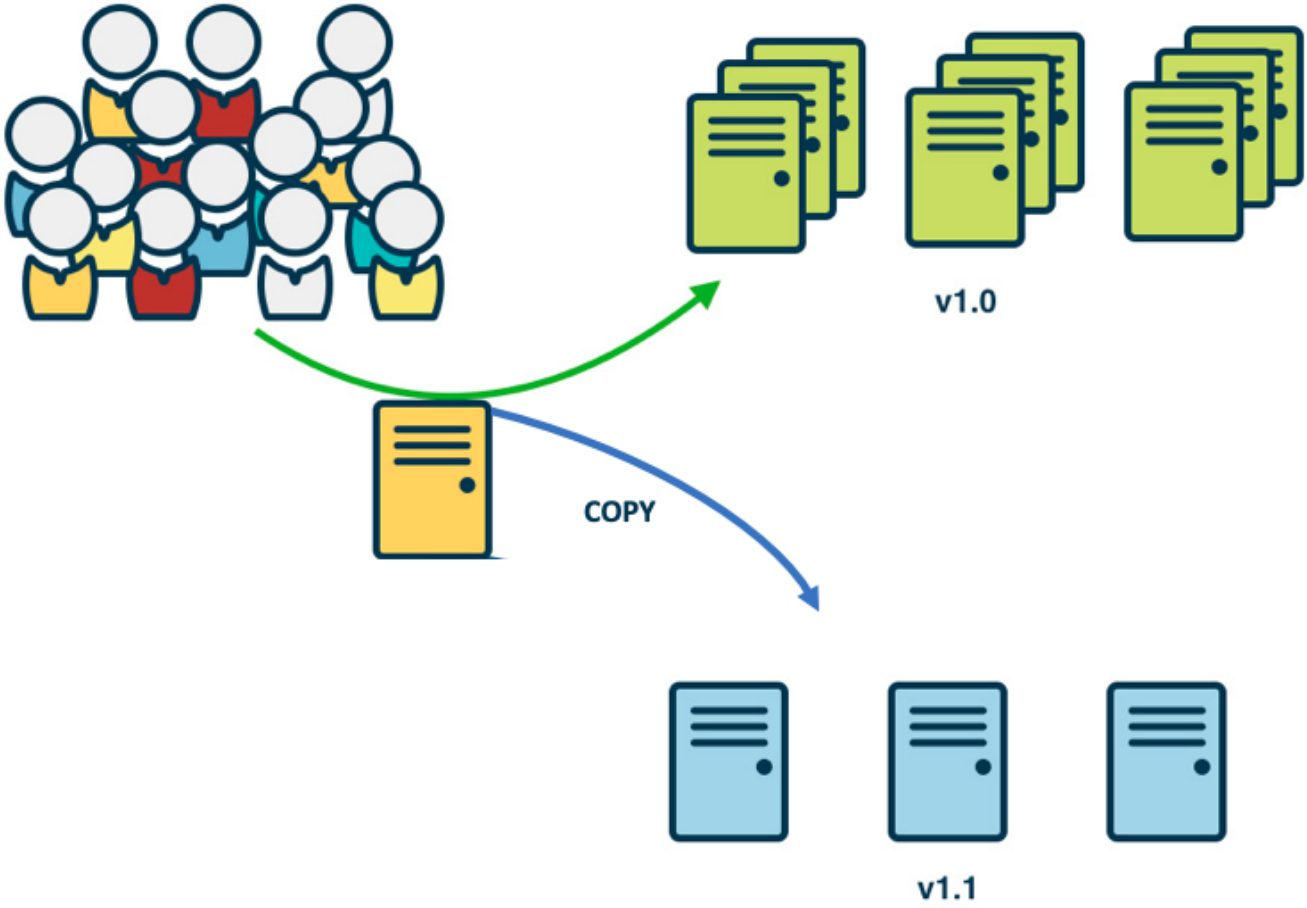
> 图片来源：https://www.gocd.org/2017/07/25/blue-green-deployments.html

3. 暗部署

随着 A/B 测试的兴起，暗部署的方式也逐渐流行起来。**所谓暗部署，就是在用户不知道的情况下进行线上验证的一种方法**。比如后端先行的部署方式，把一个包含新功能的接口发布上线，这个时候，由于没有前端导向这个接口，用户并不会真实地调用到这个接口。当用户

进行了某些操作后，系统会将用户的流量在后台复制一份并打到新部署的接口上，以验证接口的返回结果和性能是否符合预期。

比如，对于电商业务场景来说，当用户搜索了一个关键字后，后台有两种算法，会给出两种返回结果，然后可以根据用户的实际操作，来验证哪种算法的命中率更高，从而实现了在线的功能验证。



> 图片来源：https://www.gocd.org/2017/07/25/blue-green-deployments.html

以上这三种低风险发布手段，如果应用规模整体不大，蓝绿部署是提升系统可用性的最好手段，比如各类 Hot-standby 的解决方案，其实就是蓝绿部署的典型应用。而对于大规模系统来说，考虑到成本和收益，灰度发布显然就成了性价比最高的做法。如果想要跑一些线上的测试收集真实用户反馈，那么，暗部署是一种不错的选择。

线上测试和监控

那么，如何验证多种发布模式是正常的呢？核心就在于线上测试和监控了。实际上，在 DevOps 中有一种全新的理念，那就是：**监控就是一种全量的测试。**

你可能会问，为什么要在线上进行测试？这岂不是非常不安全的行为吗？如果按照以往的做法，你应该做的就是花费大量精力来建立一个全仿真的预发布环境，尽可能地模拟线上环境的内容，以达到验证功能可用性的目标。但只要做过测试的团队就知道，测试环境永远不能替代生产环境，即便在测试环境做再多的回归，到了生产环境，依旧还是会有各种各样的问题。

关于测试环境和生产环境，有一个特别有趣的比喻：测试环境就像动物园，你能在里面看到各种野生动物，它们都活得都挺好的；生产环境就像大自然，你永远无法想象动物园里的动物回到大自然之后会有什么样的行为，它们面临的就是一个完全未知的世界。产生这种差异的原因有很多，比如环境设备的差异、用户行为和流量的差异、依赖服务的差异等，每一个变量都会影响组合的结果。

那么，既然无法事先模拟发布后会遇到的所有场景，该如何做线上验证呢？比较常见的，有三种手段。

1. 采用灰度发布、用户众测等方式，逐步观察用户行为并收集用户数据，以验证新版本的可用性是否符合预期。

这里的主要实践之一就是**埋点功能**。在互联网产品中，埋点是一种最常用的产品分析和数据采集方法，也是数据驱动决策的主要依据之一。它的价值就在于，根据预先设计的收集和监控数据的方法，采集用户的行为、产品质量、运营数据等多维度的数据。

大型公司一般都实现了自己的埋点 SDK，根据产品设计需求，可以自动化地采集数据，并配置采集粒度；对于小公司来说，像**友盟**这种第三方统计工具，就可以满足绝大多数情况的需求了。

2. 用户反馈。

除了自动化的采集数据之外，用户主动的反馈也是获取产品信息的第一手资料。而用户反馈的渠道有很多，公司里面一般都有**用户运营和舆情监控系统**，用于按照“关键字”等自动爬取各个主流渠道的产品信息。一旦发现负面的反馈，就第一时间进行止损。

3. 使用线上流量测试。

这一点在讲暗部署时我也提到过，最典型的实践就是**流量镜像**。除了做线上的 A/B 测试，最常用的就是将线上真实的用户流量复制下来，以实时或者离线的方式回放到预发布环境中用于功能测试。

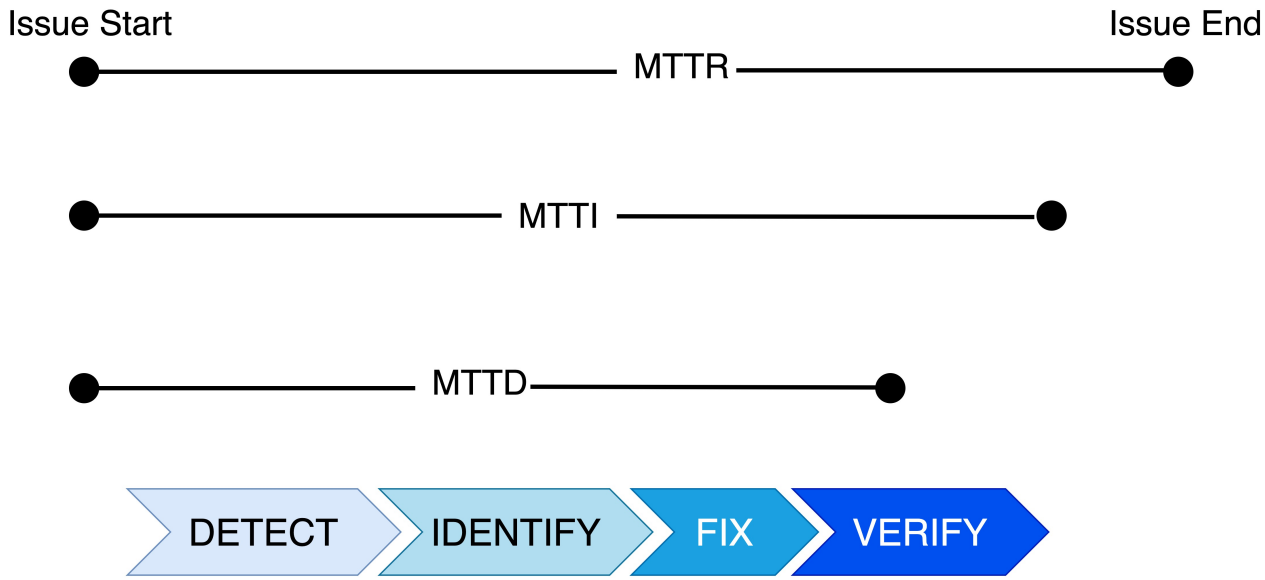
除此之外，流量镜像还有很多高级的玩法。像是根据需求选择性地过滤一些信息，比如使用只读的查询内容来验证搜索接口。另外，还可以按照倍数放大和缩小流量，以达到服务压测的目的。还有，可以自动比对线上服务和预发布服务的返回结果，以验证相同的流量过来时，两个版本之间系统的行为是否一致。另外，流量镜像的数据可以离线保存，这对于一些偶发的、难以复现的用户问题，提供了非常难得的数据积累，可以帮助研发团队进一步分析，以避免此类问题的再次发生。

在工具层面，我推荐你使用开源的 **GoReplay 工具**。它基于 Go 语言实现，作用于 HTTP 层，不需要对系统进行大量改造，并且能很好地支持我刚才提到的功能。

快速恢复

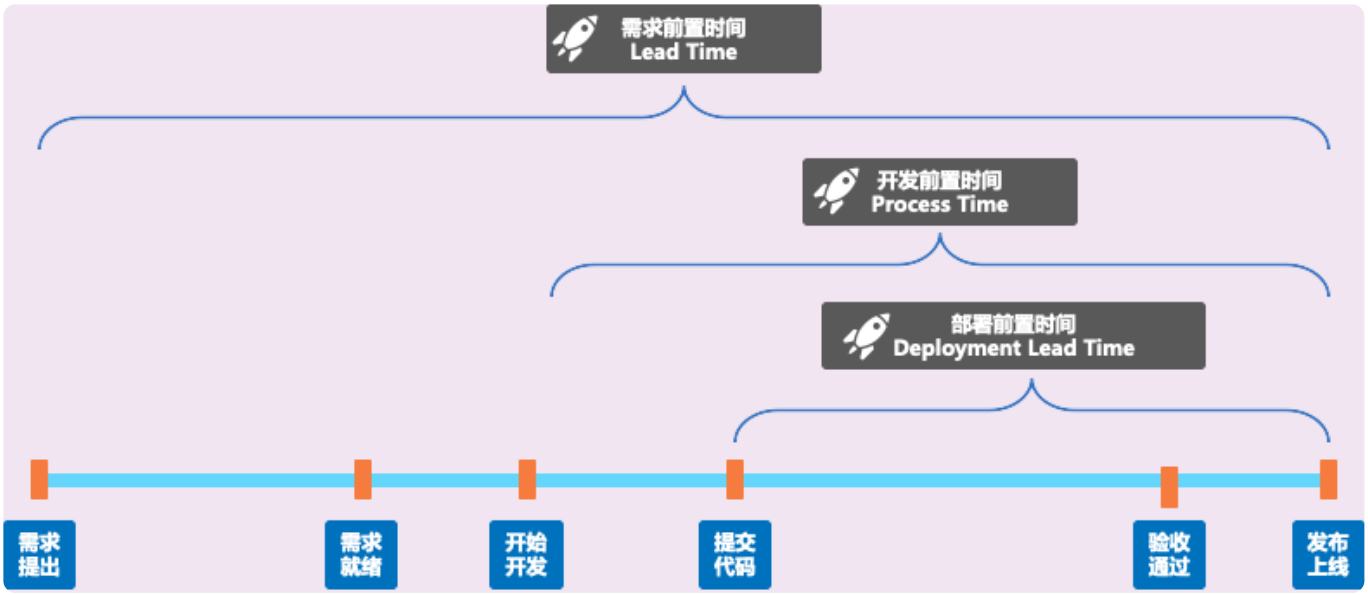
一旦发现新版本发布后不符合预期，或者有严重的缺陷，最重要的就是尽快控制局面，解决故障。**平均故障修复时长**（MTTR）是 DevOps 的四个核心指标之一，DevOps 的质量信心不仅来源于层层的质量门禁和自动化验证，出现问题可以快速定位和修复，也是不可忽视的核心能力之一。

平均故障修复时长可以进一步拆解为平均故障检测时长（MTTD）、平均故障识别诊断时长（MTTI），以及平均故障修复时长（MTTR）。在故障发生后，根据服务可用性指标 SLA，对问题进行初步分析定位，明确解决方案。在这个领域，一款好用的线上诊断工具，可以大大地帮助你缓解燃眉之急。比如**阿里的开源工具 Arthas**，就可以实时监控堆栈信息、JVM 信息，调用参数，查看返回结果，跟踪节点耗时等，甚至还能查看内存占用、反编译源码等，堪称问题诊断利器。



初步对问题进行分析定位后，你可以有两种选择：**向前修复和向后回滚**。

向前修复就是快速修改代码并发布一个新版本上线，向后回滚就是将系统部署的应用版本回滚到前一个稳定版本。无论选择哪一种，考验的都是自动化的部署流水线和自动化的回滚能力，这也是团队发布能力的最佳体现。而在 DevOps 的结果指标中，部署前置时长描述的恰恰就是这段时长。当然，最佳实践就是自动化的流水线。往往在这个时候，你就会希望流水线更快一些，更自动化一些。



最后，再提一点，你可能在很多大会上听过“故障自愈”，也就是出现问题系统可以自动修复。这听起来有点神奇，但实际上，故障自愈的第一步，就要做好**服务降级和兜底策略**。这两个听起来很专业的词是啥意思呢？别着急，我给你举个例子，你就明白了。

我给你截了两张某购物 App 的图片，你可以对比看下有什么不同。

如果你仔细看的话，你会发现，单这一个页面就有大大小小 8 个差异。所以，**服务降级就是指，在流量高峰的时候，将非主路径上的功能进行临时下线，保证业务的可用性**。典型的做法就是通过**功能开关的方式**来手动或自动地屏蔽一些功能。

而兜底策略是指，当极端情况发生时，比如服务不响应、网络连接中断，或者调用服务出现异常的时候，也不会出现崩溃。常见的做法就是缓存和兜底页面，以及前端比较流行的骨架屏等。

总结

在这一讲中，我给你介绍了 DevOps 模式下质量思想的转变，那就是要在保障一定的质量水平的前提下，尽量加快发布节奏，并通过低风险发布手段，以及线上测试和监控能力，尽早地发现问题，并以一种最简单的手段来快速恢复。

质量活动是有成本的，为了保证快速迭代发布，一定程度的问题发生并不是末日，更重要的是通过质量活动向前向后延伸，并在生产环境加强监控和测试。同时，三种典型的低风险发布方式可以满足不同业务场景的需求。当问题发生时，不仅要做到快速识别，快速修复，还要提前通过服务降级、兜底策略等机制保证系统服务的连续性。

思考题

你所在的企业采用了哪些手段来保障部署活动是安全可靠的呢？

欢迎在留言区写下你的思考和答案，我们一起讨论，共同学习进步。如果你觉得这篇文章对你有所帮助，欢迎你把文章分享给你的朋友。

上一篇　16 | 环境管理：一切皆代码是一种什么样的体验？

下一篇　18 | 混沌工程：软件领域的反脆弱

精选留言 (2)

写留言

leslie
2019-11-19

其实老师解释的A/B测试部署并非传统的A/B测试部署：故而文中用暗部署，其实一定特性上结合了"滚动（Ramped）部署“。其实可能目前大多数中小企业用灰度发布更多-主要是迅速，毕竟对于中小企业时间成本太宝贵了，能快点就不慢点。

监控是最好的风险管理和问题的验证：可能不同的企业有不同种部署方式；同一套软件A企业部署的极其顺利的可能到了B企业就有问题，故而老师的"监控就是一种全量的测...

展开

作者回复: 赞一个，这个总结是对文章很好的补充，工作这些年，我也越来越觉得依靠质量工作来保证质量只能守住下限，甚至有时候连下限都守不住，真正在未知世界能依赖的，就是监控能力，和快速发布回滚的能力

1

2019-11-19

可以介绍下一些常用的企业级监控方案么？感觉很多时候监控不太全面。

作者回复: 你好，立体化监控是个大问题，我给你分享一个体系模型，可以参考一下。

链接:https://pan.baidu.com/s/1-4Nth3h_kaLywHfoD0URhQ 密码:rb7o