升级是个麻烦事

屈春河

创建日期：2020-06-17

最近遇到了几个与升级相关的问题，特此写下此文。在本文中的升级特指的是系统所依赖的运行环境的升级，包括操作系统（例如Linux）升级、系统服务（例如Nginx、MariaDB、PHP、Redis、JDK和Kafka等）升级和第三方软件包/库升级等。

在系统的开发和运维过程中，都不可避免地遇到升级问题。之所以要升级，概括起来可能是如下几方面的原因

- 更好的安全保障。随着时间，系统会不断地暴露出各种安全漏洞和安全隐患，及时或者定期更新，能够弥补这些漏洞和隐患，从而有效抵挡恶意的攻击。
- 修复已发现的bug。系统，尤其是大型系统，在发布之后会陆续地发现一些bug，因此需要及时的更新，修复这些bug，以避免触发bug，造成系统问题或者故障。
- 实现更高的性能。一般而言，新版本会带来新的优化，实现不同程度的性能提升。这种情况对于操作系统和系统软件比较常见。
- 提供更强的功能。在新的版本中新增了诸多新功能，比如支持新的协议和提供了新的接口等。

升级可不是简单地用新版替换老版就完事了。相对于被替换的老版本，新版本或多或少地引入了一些变化，包括

- 依赖环境的变化，例如新的配置或者新的第三方依赖。
- 调用接口的变化，例如修改了接口名称或者参数格式。
- 输出结果的变化，例如调用返回的结果发生了改变等。

上述的各种变化在各种软件升级中都会存在，但是在开源软件中特别明显，从而造成新的版本不能完全向下兼容的问题。这些变化带来了升级的复杂性，使得升级往往需要配套的代码更改和系统测试，稍有不慎或者疏忽就可能带来一系列不可预料地问题，轻者运行异常，重者会使得整个系统不可用，甚至进一步影响相关系统，造成大面积的服务瘫痪。

无论是开发人员，还是运维人员，一定要认真对待升级问题。对于升级，一定要按照如下流程按部就班，一步一步操作，切不可疏忽大意，否则升级往往不会带来惊喜，而会带来惊悚。

- 在升级前，慎重地决策是否需要升级。需要认真阅读新版本的相关文档，包括Release Notes、Change Log或者README等，然后估计升级所需要的成本和所带来的风险，以确定是有升级的必要性。如无必要，尽可能不要升级。
- 在上线前，在测试系统上充分的测试，即在测试过程中需要尽可能地覆盖所有与升级系统相关的功能和模块。
- 在上线时：确保平滑稳定的服务割接。一方面要做好升级失败进行回退的预案，另一方面可以针对不同系统对于可靠性要求的不同采用不同的升级上线策略，例如对于可靠性要求

不高的系统，可以采用暂停服务来升级上线，而对于那些需要保证高可靠性的系统，最好采用灰度更新。

- 在上线后：过渡期间增加巡检和监控。更新上线后有两个重要的时间段，即第一天和第一周。在上线的 24 小时内，最容易爆发问题，因此需要专人值守，监控日志和性能。如果在上线 24 小时之内没有出现任何问题，则可以转入定时巡检，但是巡检的频率要高于日常监控。如果在上线一周内都没有出现任何问题，那么说明本次升级基本成功，可以转让常态化监控。

升级的复杂性与很多因素有关，除了系统的可靠性要求外，IT 实施的规模也是重要的一个因素。服务器越多、所部署的服务和系统越多，升级的难度也越大，升级的风险也越高。此外，升级的复杂性还与企业的性质类型和所属领域密切相关，也就是说对于不同类型的企业，升级的复杂程度和成本开销是迥然不同的。

曾经有人在社交媒体上吐槽说在与华为设备和系统集成时，发现其所用的开源软件版本都非常非常低。之所以如此，是因为如下电信设备制造商所固有的特点决定的

- 产品的部署地域广。设备和系统部署在世界各点，归属不同的电信运营商所有。
- 产品的生命周期长。很多电信设备和系统的生命周期超过十几年，甚至几十年，大量老旧设备和系统还一直在线提供服务。
- 没有系统管理权限。这些设备和系统部署在运营商的网络内，并由运营商来管理和维护，如果没有所属运营商的授权，华为这样的提供商是没有权限访问这些设备和系统。
- 涉及第三方的系统。电信设备和系统不是孤立的提供服务，往往需要被集成到第三方系统和设备上。
- 服务可靠性要求高。电信设备和系统的可靠性需要满足 5 个 9，即 99.999%，也就是说一年中不能正常提供服务的时间不能超过 5.26 分钟。

上述情况使得电信设备和系统很难大规模的升级。如果在新设备和新系统中采用新的运行环境，则会面临众多版本所带来的复杂性，例如需要重新测试，甚至开发，并且一旦需要安装补丁或更新功能，则需要面临各种不同版本的系统运行环境，造成巨大的复杂性。因此，对于华为这样的企业而言，维护一个一致的开发和运行环境，能够显著低降低开发、测试、运维和升级的成本。

与华为不同，阿里这样的互联网服务提供商，虽然拥有的 IT 设施也异常庞大并且对于系统可靠性要求也非常高，但是对于系统的升级较为积极。这是因为这些 IT 设施集中部署在数个到数十个不等的数据中心内，并且系统由自己维护和自己运营，从而使得升级过程可控和可管，可以有效避免升级所带来的复杂性和风险性。

对于中小公司而言，升级的主要原因是安全性，如果有重大的安全升级，则必须及时更新系统，避免遭受攻击。除了重大的安全升级，我个人建议如有可能，尽量根据情况定期地升级更新：

- 如果一个系统还在不停的更新演进，那么建议该系统所依赖的第三方库或包，尽量一年左右升级一次。
- 对于 Linux 以及核心的系统服务，建议尽量两年左右升级一次，可以采用分阶段升级，从非核心系统开始升级，如果升级之后的两周时间内没有出现升级相关的问题，则认为升级成功，则开始升级下一个系统。

为了保证向下兼容性，可以仅仅升级小版本。

即使小版本升级，也会遇到问题。比如最近我们将 mysql-connector-java 从 5.1.40 升级到 5.1.49 后，在执行“LOAD DATA CONCURRENT LOCAL INFILE ....”这类数据导入 SQL 时，会抛出错误“com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: The used command is not allowed with this MariaDB version”，如图-1 所示。初步排查，跟关键字 LOCAL，但是这个关键字又不能完全去掉，例如 JDBC 程序与 MariaDB 服务不在一个服务器上或者虽然在一个服务器上，但 MariDB 服务没有读权限时，必须使用这个关键字。又继续排查，发现是因为之前的版本默认支持 INFILE，而在新版本中默认不支持，需要配置 client 端的连接属性，显示配置支持 INFILE，即 connectionProperties=...;allowLoadLocalInfile=true。此外，如果是初次上线，还需要查看 MariaDB 服务器端的配置 local_infile 是否为 true

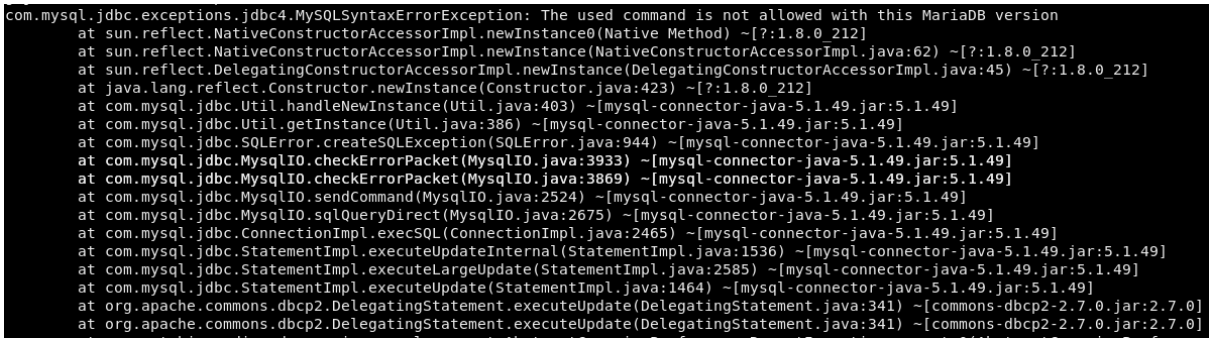

图 1: mysql-connector-java 升级错误日志