網易 NETEASE
www·163·com

开发-运维-开发，一路走来的收获与感悟

杭研后台——何登成

自我简介

- 何登成
- 研究生期间：5年数据库研发；
- B2B期间：2年DBA，数据库运维；
- 网易期间：重转研发；

- 一定的研发和运维经验；

Outline

- 从开发到运维，我所学到的知识；
- 从运维回到开发，我的思维转变；

从开发到运维，我所学到的

- 对系统线上运行状况了如指掌
- 线上操作无小事
- 遵守流程
- 风险意识，一切操作均可能失败
- 注意日常操作的整理与收集
- 老大是用来背黑锅的
- 与产品相关的所有人员打好关系
- 如非必要，不要在节假日前做上线操作
- 运维要懂得说不
- 尽可能准备应急预案
- 注重总结：你会在同一个水沟前跌倒两次吗？
- 及时反馈、指导开发

I：对系统线上运行状况了如指掌

- 运维第一步
  - 在做运维前，必须对线上的系统运行情况了如指掌；
  - 系统部署架构；
  - 系统服务的产品；
  - 系统运行现状；
  - 系统的监控与报警；
  - ...
- 画外音
  - 你敢娶(嫁)一个你不了解的女人(男人)吗？

II：线上操作无小事

- 运维必须要牢记的原则
  - 任何一个你认为微不足道的操作，都可能导致系统出现问题；
  - 举例说明：触发系统Bug；耗尽资源 ...
- 我的经验
  - 1个月：无线上登录权限；
  - 3个月：可登录线上，只有只读权限；(同样可能触发风险：做一个全表Select，耗尽I/O资源，替换热点内存...)
  - 开始：老大操作我学习；后来：我操作老大监督；

III、遵守流程

- 所有的线上操作，必须要有成熟的流程；
- 所有的线上操作，必须按照流程进行；

- 流程是前辈踩了无数坑之后，总结下来的。除非你想重新踩这些坑，否则一切按照流程来做；
  - 如果对流程好奇，可以请教制定者，这么做的原因是什么？不这么做有什么风险？

IV、风险意识，一切操作均可能失败

- 运维人员，必须有强烈的风险意识。如何带着风险意识做事？
  - 所有线上变更，必须先整理命令与脚本；
  - 所有的脚本，必须经过线下测试，验证通过；
  - 所有的操作，必须考虑失败后回滚情况，准备回滚脚本；

- 我的经验
  - 所有上线操作，提前一天准备好脚本，测试通过；准备好回滚脚本，测试通过；线上操作，基本不敲新的命令；

V、注意日常操作的整理与收集

- 好记性不如烂笔头，放之四海而皆准的道理；
  - 提高工作效率；
  - 降低记错而导致误操作的可能性；
  - ……
- 我的经验
  - 做DBA期间，我整理了近120+条，1000+行的常用Oracle命令，基本上能够处理大部分操作；

VI、老大是用来背黑锅的

- 嘿嘿！是不是觉得这一条很不可思议！！
  - 潜台词：老大不让我背黑锅就不错了 ☹
  - 但是：你必须牢记这句话！！
- 怎么做才能让老大来背黑锅？
  - 你碰到一个问题，流程上没说怎么做，你也拿捏不准怎么做。此时该怎么办？
  - 别犹豫，第一时间告知老大，让老大来拿主意；
  - 老大做决策，你来做实施。做对了，老大的功劳，应该的；做错了，老大的决策出错，老大都不知道怎么做，你更不了解了☹ 黑锅已给老大背上

VII、与产品相关的所有人员打好关系

- 做事，说到底还是做人！
- 你能相信，几波完全不相识的人，能合作把一个产品/项目做好，运维好吗？
  - 产品新功能上线，该通知谁？
  - 平时有了疑问，该咨询谁？
  - 线上出了问题，该联系谁？
  - ……

VIII、如非必要，不要在节假日前做上线操作

- 这应该算是运维领域一个共识，并且也经过多次的应验；
  - 节假日前做操作，导致问题的可能性更大；
  - 节假日前不做操作，线上都有可能抽风；
- 为什么会这样？
  - 节假日前，心已经飞走了...
  - 节假日前做上线操作，节假日出了问题，找谁来处理？
  - 节假日本身的访问模式、压力，与平时有较大差异；

IX、运维要懂得说不

- 作为一个运维，有线上的操作权限，有时是相当吃香的！这时，要经得住诱惑，要懂得说不！

- 你有可能收到哪些请求？
  - Hi，哥们！帮我线上导些数据？帮我把线上的配置修改下？帮我做一些数据订正？帮我悄悄的上线一个功能？...
  - 对于这些请求，最好的做法是：拿流程做挡箭牌，懂得对哥们说不！

X：尽可能准备应急预案

- 系统出问题了，你老大，老大的老大，甚至是大BOSS都被惊动了，正虎视眈眈的站在你的背后，指手画脚的要你在5分钟内把问题给解决掉！！！此时的你：
  - 背脊发凉！
  - 脑袋发昏！！
  - 两手发抖！！！
  - 别说定位/解决问题，正常思考的能力都丢失了...
- 该怎么办？
  - 如果针对此问题，你刚好有一个应急预案，不慌不忙的拿出来，三下五除二，立马解决；你还愁着不升官发财吗？？到时别忘了请我吃顿饭！

XI：注重总结：你会在同一个水沟前跌倒两次吗？

- 第一次跌进一个水沟
  - 可谓不知；
- 第二次跌进同一水沟
  - 所谓不智；
- 第一次犯的错，只要遵守了流程，谓“不知者不罪”；但，同样的错误，接二连三的犯，你就要问问自己，是不是自身存在问题了！

XII、及时反馈、指导开发

- 上线运维/日常运维/例行巡检 过程中，所有的不爽之处，别客气，提出来，向开发反馈！
  - 首先，这是开发应该做的职责；
  - 其次，开发其实也正愁着没事可干呢！

从开发到运维，我所学到的

- 对系统线上运行状况了如指掌
- 线上操作无小事
- 遵守流程
- 风险意识，一切操作均可能失败
- 注意日常操作的整理与收集
- 老大是用来背黑锅的
- 与产品相关的所有人员打好关系
- 如非必要，不要在节假日前做上线操作
- 运维要懂得说不
- 尽可能准备应急预案
- 注重总结：你会在同一个水沟前跌倒两次吗？
- 及时反馈、指导开发

从运维回到开发，我的思维转变

- 活着，好好活着
- 木桶效应
- 出门找别人之前，把钥匙留下
- 没有秘密可言：有问题直接告诉我，别让我猜
- 完美是很难的，懂得取舍
- 你这么牛逼，你同事知道吗？
- 机器能自动搞定的，为什么要人来做？
- 专业人做专业事：你为什么帮运维做决策？
- 它山之石，可以攻玉
- 那么重视线上环境，为何却对测试环境弃之如履？

I、活着，好好活着

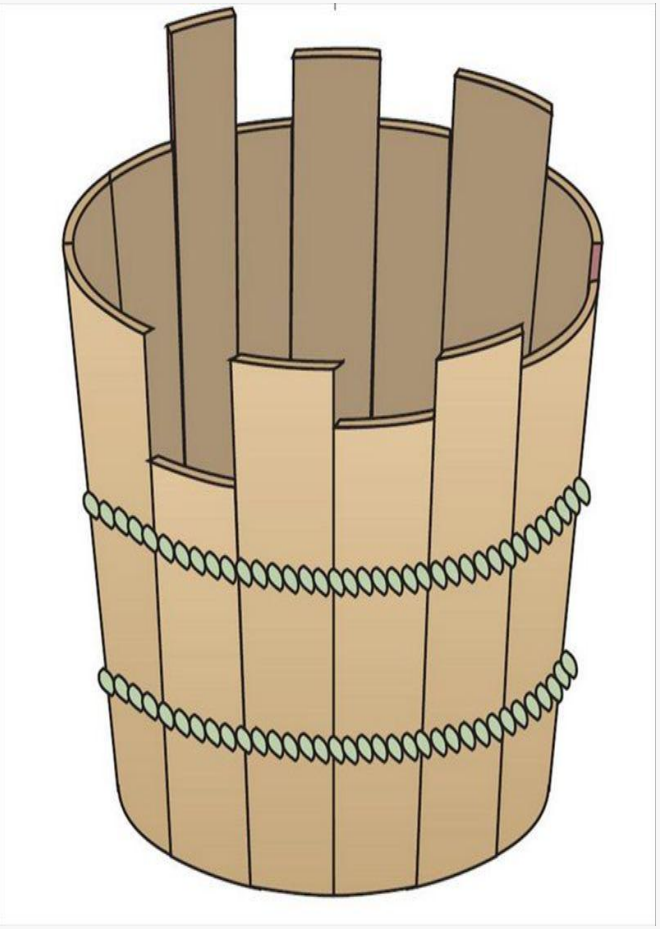
- 考察一个系统，说到底就是考察系统的可用性！

- 活着
  - 设计/研发系统，针对各种异常情况，针对各种未考虑周全的地方，要做到充分的容错；
  - 碰到异常，可以报警，可以监控，但是不能简单的就把系统宕掉：线上慎用assert断言！

- 好好活着
  - 不仅要活着，还要好好活着，不能赖活着；
  - 压力激增，连接暴涨，系统能不能处理各种突发情况？保证系统的平稳运行？限流，控流，监控，报警… …

II、木桶效应

- 系统能够提供的性能指标，是他能支持的最高性能？是他的平均性能？
  - 都不是！系统的最低性能，才是系统真正的性能指标；
- 系统运行过程中，足够稳定吗？是否存在大幅波动？大幅波动的原因是什么？
  - 定位到这些问题，并予以消除；



III、出门找别人前，把钥匙留下

- 大家应该都遇到过这种情况：
  - 门锁着，家人带着钥匙出去了，说是很快回来，谁知一等就是一下午☹
- 对于系统实现，意味着什么？
  - 有没有持有资源去做I/O？
  - 有没有持有资源，去掉用外部接口？
  - ……
  - 你不能假设外部的操作一定很快完成，你不能出门前，把钥匙也带走了；
  - 或者，外部调用使用超时机制，保证调用的最长时间，如果你长时间不回来，别怪我把门给撬了啊；

IV、没有秘密可言

- 小两口热恋中
  - 你猜我为什么不开心？
  - 你猜我想吃什么？
  - ……
  - 你抓狂不？
- 系统上线之后，就是运维人员的女朋友，遇到问题如果也要靠猜，运维人员抓狂不？
  - 你的系统，要暴露出足够的信息，方便运维人员在运维过程中，碰到问题能较快定位到原因所在；

V、完美是很难的，懂得取舍

- 开发人员的性格中，有追求完美的因子。但是：
  - 完美是很难的，要懂得取舍，有舍才有得；
- 哪些情况下，需要考虑取舍？
  - 这样实现简单，方便后期维护，但不是最优的；就这么做。
  - 这样做容错处理，简直太恶心了；系统必须容错。
  - 系统中打了好多采样点，影响了系统性能；方便后期运维；
  - ……

VI、你这么牛逼，你同事知道吗？

- 除非系统是你一个人做的，除非系统开发出来后，后续不需要进行改进与升级。否则：
  - 必须考虑整个团队的技术水平；
  - 这么写代码，同事看的懂吗？能进行审核吗？
  - 当前的代码，后续维护/接手的难度大吗？
  - 非特殊需求与场景下，代码以简洁易懂为标准，杜绝奇淫巧计；

VII、机器能自动搞定的，为什么要人来做？

- 运维，追求的是运维自动化；开发，更应该将大部分操作，都自动化起来
  - 系统的单元测试，每天还需要手动来运行吗？
  - 复杂点的测试系统，还需要手动一步步搭建吗？
  - 性能分析采集到的数据，还需要一行行去分析吗？(何不画个图呢？)
  - ……

VIII、专业人做专业事：你为什么帮运维做决策？

- 术业有专攻，开发与运维都有自己擅长的领域。开发做好自己份内之事，其余的交由运维搞定。

- 哪些是开发不擅长，而运维擅长的？
  - 系统运行需要多少内存？
  - 系统需要消耗多少IOPS？
  - 系统加锁/回调超时应该设置为多少？
  - ……
  - 所有这些，因产品而异，既然在开发过程中无法确认，何不暴露出来，由运维人员来进行配置？

IX、它山之石，可以攻玉

- 你是否听过这句话：
  - 当你觉得自己的想法很创新时，十有八九是知识面太窄☹
- 我做运维时，管理的是世界上最好的数据库系统——Oracle；我重新做研发后，做的是基于MySQL的存储引擎，从Oracle中借鉴良多；
- 不要蒙头做自己的系统，有时多抬抬头，看看外面的世界，别人的做法，能给你很大的启发；
  - 性能优化——Brendan Gregg；并发编程——Jeff Preshing；... ...

X：那么重视线上环境，为何却对测试环境弃之如履？

- 线上环境
  - 追求3个9，4个9的可用性；
- 测试环境
  - 配置差；可用性几乎没有；
- 二者这么鲜明的对比，是应该的吗？
  - 重视测试，才能够保证系统上线的稳定性；
  - 重视测试环境，才能够保障测试的有效性；

从运维回到开发，我的思维转变

- 活着，好好活着
- 木桶效应
- 出门找别人之前，把钥匙留下
- 没有秘密可言
- 完美是很难的，懂得取舍
- 你这么牛逼，你同事知道吗？
- 机器能自动搞定的，为什么要人来做？
- 专业人做专业事：你为什么帮运维做决策？
- 它山之石，可以攻玉
- 那么重视线上环境，为何却对测试环境弃之如履？

Question ?