运维 85 条军规

1) 承载能力优先 ——随后再进行优化 —— 不遵守这条规则必定带来故障停机时间。不要在故障停机时间的压力下进行优化——要先集中精力提高承载能力。

2) 以 Postgres 为例，一定要确保你的每一个网络都能匹配得上你的 WAL 文件、Slony 复制、快照技术以及基于磁盘的 DB 版本化（快照的衍生品）

3) 不要把问题‘优化’到你的架构之中。为了解决问题而新加进来的一些东西往往后来都会变成运维沉重的负担。 要确保在运维工程化中开发出来的工具交接完整。过后再回头进行进一步的开发往往不灵。更重要的是，变更请求可能会破坏已经安排好的工程计划。

4） 保持简单。保持简单，因为你很聪明 别把事搞的太复杂 因为你行的。

(译者:KISS 原则 Keep It Simple, Stupid)

5）应该非常谨慎地使用 缓存 ，为了保护资源一致性，它很难进行水平缩放。

如果你作的是一个可以横向扩展的东西，

明智或审慎的做法是不要添加的缓存层。

如果非要使用，它应该是为最终用户获得性能，

不是为了赢得一个网站的容量;

6） 不要所有代码都自己写; 不要所有东西都外包; 在合适的时间使用合适的工具,完成你的工作.

(译者: 不要重复造轮子)

7)协商-真正有效的谈判唯一方式是先作一些调研,制定一些可行的性方案.这样你可以挑选你的首席开发商,如果你真的需要. 别虚张声势.

8）一直保持 N+1。如果 N=1，无论任何情况下不要轻易使用+1，这个 1 只用于当 N down 机情况下。当使用冗余服务器来承载负载时候，不要让你的系统超过 49%的负荷。当有机会能只用 N+2 的架构时候，使用它。

9）数据丢失不是任何一个公司所能承担的风险--这是举世所知的真理。数据丢失造成的损失远远大于保持数据不丢失所花的成本。

10）无论何时何地尽可能并行化。这是复路考虑最重要的手段。比如，如果利用 MogileFS 来做位置感知，并且需要实时的复制数据，一个可行的方法是每一台 MogileFS 服务器可以复制它的数据去 MogileFS 的负载均衡中心。尽可能多的启用多的平行。

11）阅读手册。至今，我还是坚持要先通读 RAID 卡的手册，以确认是否有什么细微的差别。恶魔都隐藏在细节里。做足功课吧！

12）知道瓶颈所在，并知道怎么去定位它，一层层排查，查找是不是硬盘、内存或者 cpu 的阻塞了。通常这个很简单。

13)定期做系统容量管理程序。积极一点。如果没有容量数据的曲线，你很难知道你系统的薄弱之处。

14）不要促成失败，不要害怕改变。

15）别挖陷阱给自己跳。不要认为你的工作成果将能作为未来的工作的动力。

16 ）运维人员写的代码应该是运维工具，而不是应用软件。

17）在运维团队中，不要低估了项目管理、文档撰写以及财务分析人员的价值。他们比给予工资更有价值。

18）监控一切。报警异常问题。其他部分记录数据用来做趋势分析信息。

19）定期的流程查看各个地方的趋势数据。

20）不要把监控弄的很乱，不然他就没有啥意义了。

21）要确保监控系统简单到让公司的每个人都能上手。你可能会很吃惊监控数据指标转换成为业务指标、市场指标和销售等等指标有多频繁。

22 ） 只在可以做出相应改善的地方做检查。 否则就不要浪费时间了。

23）公开你的检查报告，并附上相关数据，以便他人可以容易的阅读到关键点，并能关联到响应的数据。

24）在每一个技术点都分配人手。

25）要给重要人员配备后备人手。

26）要不断的招聘。甚至是当你没有名额，也要一直招聘。

27）要严于律己。无论你有多聪明或者你认为你多 NB，你也要不断的提高自己。

28 ）要尽可能多的把自己和其他公司做比较。眼光要往外看。

29) 挑选一个展会或回忆，只有一个，一年一次，去参加。如果展会一年举办多次，之参加一次。

30) 购买你需要的，而非你想要的。永远都不要摘掉企业这顶帽子带上“什么对我是最简单最安全的”这顶帽子。

31)永远只做最生意有益的事情，即使这意味着你需要离开。

32)正式问责机制——记录承诺，标记它们，揭示为兑现的承诺。

33) 你不应失败两次以上。恐惧感有点好处。但要知道长期犯错和无意犯错的区别。

34 无情一些——你的对手如此。

35)视工作为在你完成时需要签上你的名字。这也意味着完成你的工作。

36)变得对别人有用。

37)与初创公司合作——提供你的专业技术和规模，你将获得免费的产品作为回报，有时甚或一生。

38）容量是一个业务/产品问题。这意外着每个页面、每个请求、每次登录的网络成本等等在做正确的业务/产品决策时候都必须是都透明

39)一直打破预算。运维团队通常是最大的花费者。通常没有收入没有达到预算，但是运维团队可以有很多方法来延期采购。

40）过去能正常的做的事，不见得现在或者未来都会正常。所以做的时候，先用工具测试一下。

41）文档化。把所有东西都写成文档。要让新人挨个去问怎么做事。

42）用一张大尺寸的图绘制你数据中心的网络拓扑。

43 ）用一张图描绘出你每一个产品的业务流程图。

44) Faq-O-Matic, Wiki, 在这里人们可以很容易的发布“这是如何修复这个”的文章，并让它容易被找到的地方。这里是技术编写者能派上用场的地方，但是，最重要的是使文档更

容易，即使是非正式的。

45) 确保所有人，任何人都可以被替换。

46)多数人在家里做的比在办公室里更多，有些人则不然。

47)捆绑下单——你可以要求更多折扣，更好的条款等等。当你成批购进硬件时。你可以要求一切——最低价格，备件包，租赁期限，只要他们还没有得到订单。

48)同你的供货商保持长期关系——确保你在下份工作时依然能够联系他们。

49）给运维团队的每一个人配置可以用来远程工作的超级装备：掌上电脑、无线上网卡、24 英寸液晶显示器等等。雇佣大拿得到回报要远大于远程雇佣的本地人员。记住运维工程师都是用电达人，能充分的利用屏幕上的每一个像素点。

50)完全陷入 IT 标准的泥潭。直到 Mac 运行 office 2007 和 outlook，你必须运行 windows。间断的。除非全用 mac 本，否则这会破坏会议日程表、联系人或者邮件列表的办公效率。如果有个员工愿意工作的在 xp 环境。这是非常少。这条法则，现在是陈旧的/未公认的，陷入泥潭不一定是最好的方法。这个列表非常的 07 化。

51 ）有个合理的采购流程。知道你的预算，确信能管好它。从财务得到实际的金额。在技术驱动的预算/报告和财务驱动的预算/报告直接往往有一定的差距。作为一个搞的运维管理者要能形成模型把这些差距计入销售总成本中。一个理解这些事情的 CFO 有助于推动业务决策。

52）周会一定要持续进行。对上次会议的事件逐一落实结果和追责。

53）建立一个分离的逐级升级系统，用以消除由于开发者代码的问题对线上系统的不良影响。这主要是运维问题、代码问题都存在的情况下在开发跟踪系统或者运维跟踪系统中往往会丢掉，最后无人理睬。建立一个独立的跟踪系统来解决这些问题可以使得问题简单清晰。

54 ）产品开发从设计开始的每个阶段都要和运维相结合。这样，扩展性，监控和可靠性都融入到产品里。这样同时也可以确保运维负责的硬件采购、监控系统按时到位，运维手册得到及时更新，最后产品按照预计时间上线运行并且都符合运维标准。

55）在公司真正的实践——Sarbanes，WebTrust 安全审计认证，SAS 70 审计标准，Visa 和银行等等。如果你真的成功了，这些都是你不得不打交道的。早点开始这些准备其实很简单，不需要太多的知识。部署一个工单/任务跟踪工具，使用它。把变更控制和变更管理纳入同样的系统里，使用它。其他信息也可以放进来。系统就可以帮助我们找出像“上周变更了什么”这类信息。

56）简化给冗余留和多点登录的流程。一开始或许很难，但是一个没有真正的扩展性和可靠性的系统，才会真正耽误你获得成功的时间。

57）Oracle 标准版(SQL Server 标准版)是值得购买的。如果你可以限制住自己不超过标准版的需求，那就绝对值得买，哪怕你刚刚开始创业还不需要他。

58）Postgres 和 MySQL 是一个免费的考虑。如果你不是特别在意事务完整性，MySQL 是很好的选择。直到"真空"和 Postgres 单词的强制链被打断，Postgres 代表一个不可预知的，通常消极诡异的数据库。

59）容量设计应该按照每日峰值再上抛 20% ~ 30% 的冗余。除非你是个迁移技术热衷

者。

60）尽量多读一些经济杂志。它们通常是免费的，只需你填写一些调查问卷就可以免费获得。新闻的价值是巨大的。让他们投递到你家里，工作的时候读杂志的机会趋近于零。

61）保障安全。开发人员不应该有线上环境的权限，应该做代码复核。这是和运维之间的职责分离。运维团队中应该有人控制其他运维人员权限的权限。制定员工手册，告知违反安全条例所带来的严重后果。从开始就要从物理的、逻辑的、功能的各个方面来保护客户的数据安全和隐私。万一有客户要和你对峙起来，你发现起来发现自己只是靠勇气和勤奋来保护客户数据，那你就傻了。

62） 控制好访问入口。首先要保证大家可以正常完成工作；其次要确保你知道他们是从哪里登录的。启用双因素身份验证方法。

63）对于人们访问生产环境必经之路的壁垒和网关宿主，击键记录很重要。对于 Windows 可能稍微有点难度，不过有些网关可以提供自动截屏功能。

64）如果有状况的情况下，确保有冗余登录点连线到生产环境。不要期望公司的 VPN 在网络中断的时候还能连上生产环境。直接把 VPN 架设在线上环境里。

65）使用 LDAP 认证，哪怕你只有 10 台机器，通过复制 passwd 和 shadow 文件的方式来管理，你也需要 LDAP 认证。

66）不要低估在 UNIX 环境中一台 Windows Server 2003（2008）设备的作用。如果只是因为不懂 Windows，那么去学，而不是排斥它。

67）不要在无效的无线方案上浪费大家的时间。人们都机动的，他们希望在沙发上，会议室里，门口，到处都要上网。一定要保证无线 ad 的可靠。

68） 总有人把额外的精力和时间都投入到工作上——直接通过他们的请假单好了。而另一

些人恰恰相反只把注意力放在怎么通过自己的请假单。在个人时间安排上，运维人员总是做出巨大的牺牲，他们随时准备凌晨 3 点爬起床快速响应排障需求。

69）通过集中式的关系数据库管理你所有的产品成果。然后通过数据复制分发到资产，人员，网络，合同等所有数据到异地。没错，要的是一个在线的实时可用的复制，而不是每天晚上备份到磁带。

70）尽可使用自动化流程以确保安全，包括操作系统或者产品的上线，文件的分发，日志的分析等。

71）自动化操作通过运维数据库获得配置（真理来源）。

72）服务器通常有三种状态——离线，在线，产品态。在线就是说正在通过 cfengine、rsync 或者其他你在使用的工具完成配置。产品态表示已经走流量了。同时还需要一个状态，这个状态下的设备可以在不提供生产服务的情况下收集或者测试数据。

73）注重日志数据。在设备下线或者重建之前，一定要先导出日志。

74）如果规模发展太快以至于没有太多时间来做优化，那就尽力锁定一切——流程还能进行即可，就不要改变它，直到后来有了绝对必要的理由。总之，锁定默认值，等待成长到必要时再审视。

75）你永远无法避免运维工程师在你基础设施最关键的地方犯错——比如在哪台机器上不小心执行 rm -rf / 命令。

76）为团队保持好玩和有趣的气氛——如果他们不再享受他们的工作，他们就会找别的事情来消遣。要让团队有主人翁意识，运维不是哪个经理的个人任务。

77）提供 99.999% 可用性的真正价值在于让我们有能力保持灵活。这意味着当你需要的时候可以充分利用冗余。这会让物理变更、设备登入点变化、代码修改和回退等等都游刃有

余。这个对于公司本身价值巨大，甚至比对客户还大。

78）如果你能做到 99.999%，给客户 100% 的服务承诺。

79） 不要丢掉按流程发布软件的能力。应该丢掉的是你自己回滚或者转移到旧版本代码的能力。压根就不应该“处理”这种徒劳的失败转移。当事情变得不 如人意的时候，你更应该做的是找个大玩意儿来挡住你的肥屁股。CYA = 保持敏捷 = 成功的公司。

80） 在脑子里要清楚知道为什么以及这样做的为了达到的目的，为客户构建产品每一个具体步骤。不管你部署给最终用户的是什么，把这些放在最先考虑，即你所有（基础设施、流程和人员）的设计都是为了提供最好的服务和产品。

81）第一次就要做对了。很少有机会让你回去在做一遍的。重做是对公司资源的巨大浪费。要提高命中率，一次就要成功。

82）接触业内人士、盟友和类似的企业，看看他们的运维是怎么做的。很可能他们碰到了跟你一样的挑战，而解决的方法更好。不要害怕分享自己的经验和处理过程，因为别人也会回馈的。他山之石，可以攻玉！

83）招人要招那些好到可以让你担心位子不保的那样的人，招你欣赏和可以学习的榜样，招那些你愿意和他一起工作的。这感觉甚至超过你招聘一个工作考评为 A 的员工。

84） IT 和运维是完全不同的两个概念。一个不错的运维经理应该可以管理好企业 IT，但是一个传统的 IT 工程师很难有能力处理互联网运维任务。

85）当你开始一份新工作或者在每年的起始，都应该去争取预算。这不是说推车那老破车往前走，而应该是基 于历史数据做出最佳推荐方案。如果你正在评估一份新工作，请确认你完完全全的知道预算以及预算的来源。同时，还应该有完善这份预算的权利。