
腾讯云开发者社区
墨问西东
Tencent Cloud
腾讯技术创作特训营
第二季
写作
每个人都是天生的作家
是最好的自我投资
AI

腾讯云开发者社区 | 墨问西东

Tencent Cloud

腾讯技术创作特训营
第二季

写作是最好的自我投资

每个人都是天生的作家

怎么才能把技术文章写好？

刘　超

资深研发专家、极客时间专栏
《趣谈网络协议》作者

腾讯云开发者社区 | 墨问西东

怎么才能把技术文章写好?

主讲人：刘 超

自己会是一回事儿，能讲给别人是另一回事儿

写好技术文章，是我职业生涯长期收益的事情

- 自我回顾需要：不要高估自己的记忆力，以自己的思路写出来是重新拾起一个技术的最好方式，尤其是对于一个职业生涯较长的架构师
- 个人影响力需要：公司内部影响力，行业影响力，交到朋友，遇到贵人，多一些机会，写技术文章是内向的人特别好的沟通方式
- 向上汇报需要，客户沟通需要：将技术成果以别人想听的故事或者思路讲给他人
- 面试跳槽需要：随着职业生涯的增长，你没有时间全面的在一个小时表现所有，且无法像上一条一样事先知道他人思路
- 向下培训需要：如何让他人快速了解某个事情，到比较快速上手的程度，而非从头学习

写技术文章的常见误区

- 大段大段的贴代码，不如直接看代码
- 从书籍或者他人博客中摘抄下来的，过一段时间就不认识他了，所以必须要用自己的思路写
- 不忍心拉下任何一个细节，容易错失重点，或者让人望而却步
- 为了浅显和吸引力，减少信息传递，让读者没有收获，不如去刷抖音

为什么要学会用比喻?

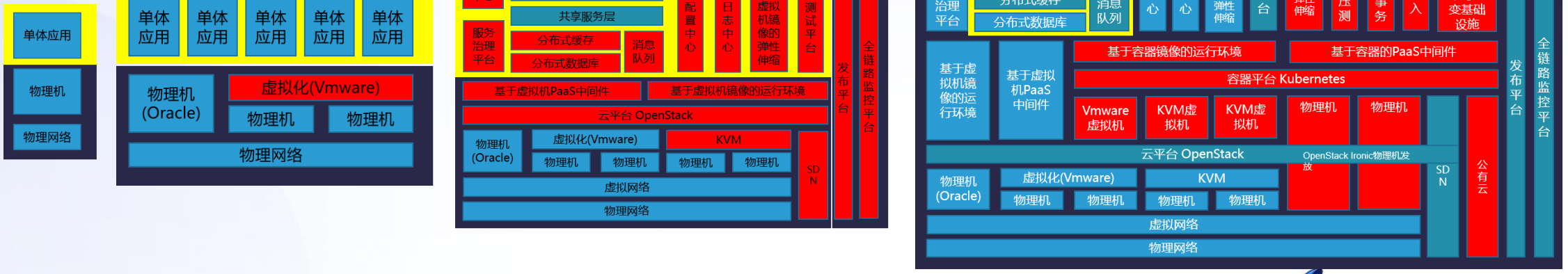
- 看到比喻不要杠比喻的不恰当之处，体会比喻的相似之处，夸张之处
- 相似是为了记住原理，夸张是为了记住关键差异
- 比喻是很久之后索引到知识大概原理及关键特性的工具，足够在学习新知识的时候，快速跨过一些门槛

为什么要学会讲述发展史?

- 某项技术的应用场景源于设计的原始初衷，而原始初衷是从历史上的痛点发展过程中来的
- 了解发展历史有利于将技术用于最该使用的场景，虽然从技术角度，也能干别的，就是有些别扭，还容易出事儿
- 复杂的技术体系是很难讲清楚的，但是无论多么复杂的技术体系，都是从简单场景逐渐演化过来的，从发展历史角度来看，脉络就会比较清晰

为什么要学会使用图像?

- 图像是体系化总结复杂知识的良好工具，可以表达知识体系的逻辑关系和关键点
- 回忆一个知识往往只需要一个图就能回忆大部分
- 前后图形之间的差别可以明显表现出发展变化



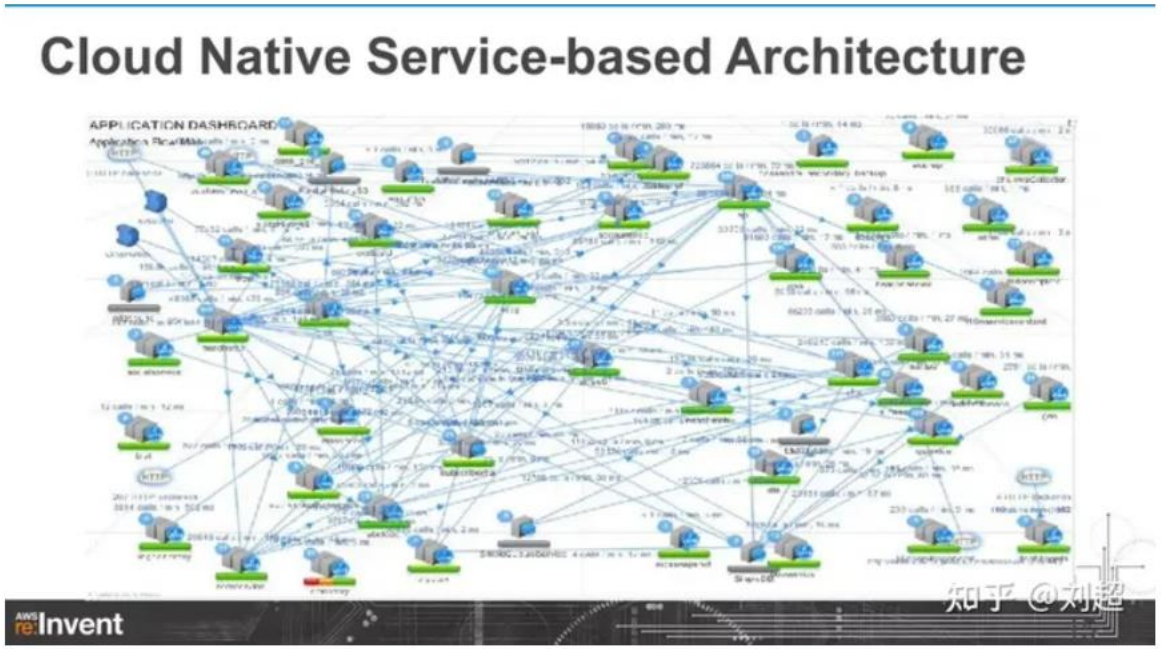
复杂技术的写作技巧

- 复杂技术的写作难点
  - 循环依赖，无论从那个入口写，都有前序知识没有讲清楚
  - 写着写着，随便打开一扇门，又是一个新的庞杂体系，容易偏离主题
  - 技术难度大，学了后面的，忘了前面的

复杂技术的写作技巧

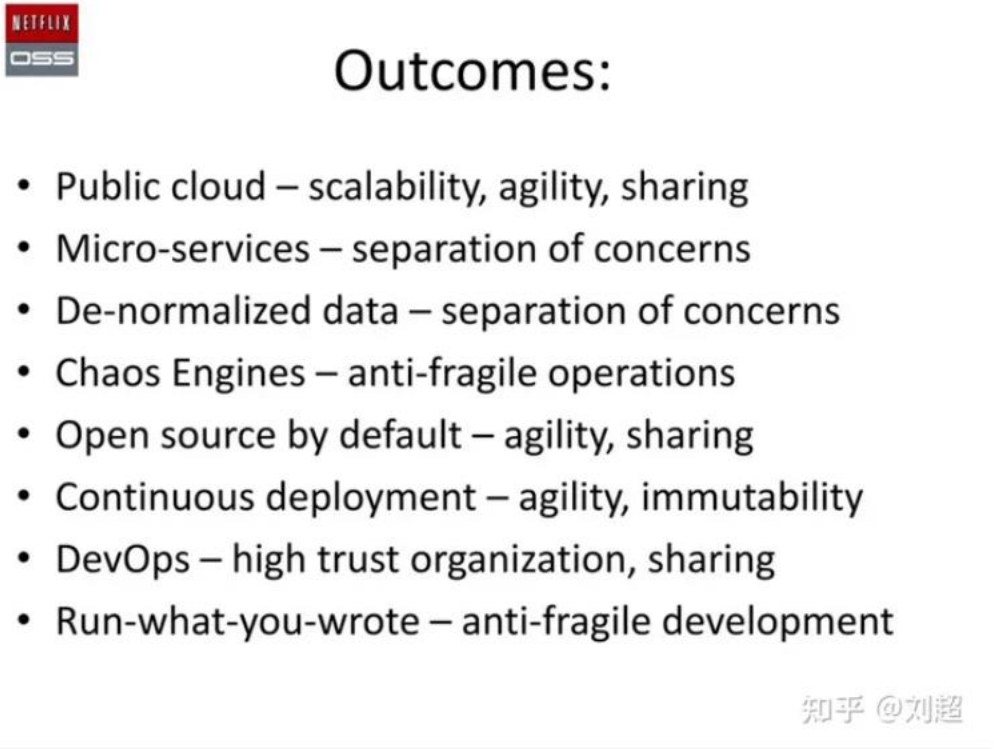
- 按历史脉络书写是一个相对简单的方法
  - 循环依赖：历史是没有循环依赖的，再某个时间点一定能够讲清楚之前的状态
  - 容易偏离主题：历史会在最恰当的节点，实践中最痛的地方打开一扇门，不会偏离主题
  - 前后呼应问题：注意历史上的前后呼应，往往后来出现的技术，在很早之前就有影子，最后风行起来还是被淘汰，都会在历史上埋下伏笔

第二张图看起来比较头晕，密密麻麻纷繁复杂的的服务之间的调用，不过你也不用管他。当时云原生和微服务还没有火，也谈不上追风口，Netflix是真的因为业务的演进事实性的架构变成这样的，完全处于业务的需要。



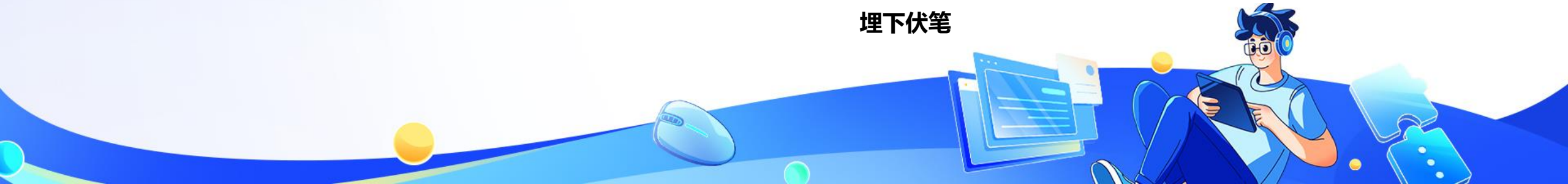
打开窗口不展开

我们回到那次演讲的PPT，在下面一页，Netflix表明了他为什么要这样做。



虽然当时云原生的概念还处在初级阶段，但是Netflix在这里的表述已经从实践角度非常全面的描述了未来云原生的方方面面，以至于多少年后，当我们的业务真的发展到某个程度，回过头来再看这页《Outcomes》PPT，才理解其中真意。

埋下伏笔



可能你会问，是不是所有的业务场景都能熔断和降级呢？当然不是了，如果要支付，但是无法获得准确的金额，那肯定不行。如果要下单，使用优惠券，发现获取不了优惠券的扣减信息，也是不行的。因而我们把服务之间的依赖分为强依赖和弱依赖，牛肉面馆没有牛肉和面，今天生意就做不下去，这是强依赖，不可以熔断和降级，但是如果牛肉面馆没有凉菜，生意还是可以做下去的，这是弱依赖。这就需要我们能够分析出强弱依赖，其实服务之间的依赖关系看起来纷繁复杂，但是真实一个业务中，完全强依赖的整条链路还是其中很少的一部分，这部分我们长成为核心交易链路，或者核心调用链路，这条链路是一个环节出问题整个业务就运行不下去的，是整个团队要重点关注的部分，好在这部分非常少，比如电商里面，只有下单，支付的链路是比较核心的，哪怕浏览商品这些，都可以降级。

做个比喻

这个时候你发现，和传统的应用中，一个技术总监或者运维总监掌控全局的时代过去了，已经没有一个人可以掌控系统的全部了，对于服务的治理需要从集权到民主，是不是理解Netflix的说法了，我们回到《Outcomes》那一页，要信任员工和团队。每个服务都会有一个小团队进行维护，一方面满足快速迭代，一方面保障质量，服务的Owner像牛肉面馆老板一样思考，从上游服务进货，为下游提供服务。每个层次的服务根据依赖的服务和中间件的约定来预估自己对于下游客户的SLA约定，包括高可用性，QPS(每秒查询数)，TPS(每秒交易数)，MRT(平均返回时间)。每个服务梳理强弱依赖关系，并采取相应的策略。

比如上游E服务约定的QPS和TPS的值应该默认认为他能达到，如果不满足，E服务会自己扩容，和你无关，但是如果你超过了约定的值，E服务是有权力自己进行限流的，你要根据你依赖的上游算一个自己的QPS和TPS，并且对于超过的部分也进行限流。

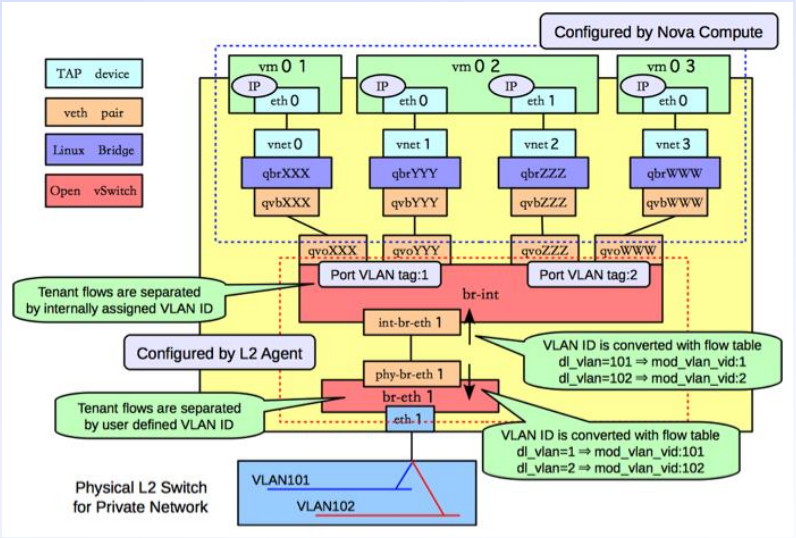
对于上游E服务约定的MRT，是你设定调用他超时时间的参考，如果真超时了，就说明E出问题了。如果E没有超过约定的MRT值，但是你仍然感觉太慢了，无法满足你对上游的MRT承诺，这个时候，你就需要通过异步调用或者消息队列的方式，让这个慢的部分异步的完成。

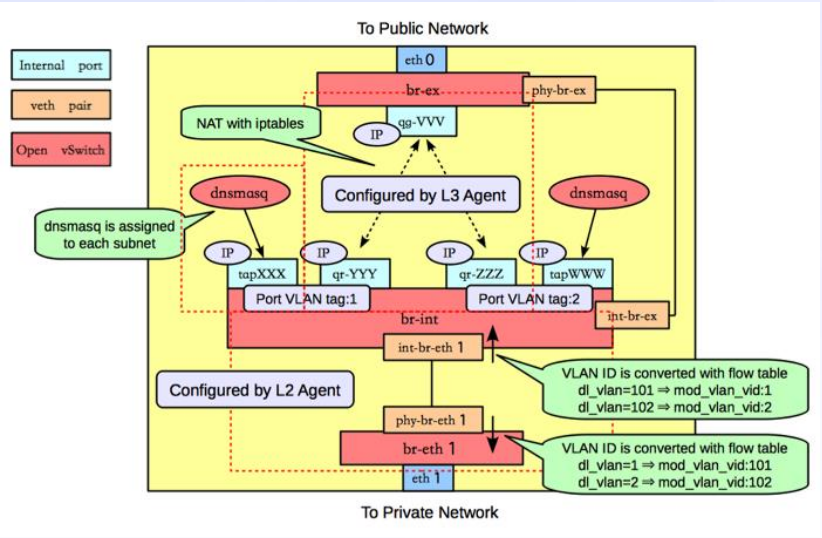
对于弱依赖，你要想好如果挂了怎么办，太慢了怎么办各种场景。对于强依赖，仍然要考虑后面挂了或者慢了，虽然不能正确返回了，但是如何不被弄挂。

照应开头

复杂技术的写作技巧

- 由近及远，从身边的简单场景写到复杂的场景
  - 循环依赖：比喻的作用，需要找出身边场景和复杂场景的相似之处，简单场景循环依赖少，但是可以自闭环，讲清楚后，可以作为后面复杂场景的前序依赖。例如在家庭网络里面讲IP，网段等，到了数据中心看总体架构图就有基础
  - 容易偏离主题：如果打开一扇门，发现里面知识过多，及时关闭，用一个比喻描述关键特性，再后面再另外展开
  - 前后呼应问题：图像逐渐展开，就像打游戏中地图逐渐展开，可以让读者清晰感受前后知识的关系。并且如果后面学习的时候，忘记了前序知识，无需重新学习，扫一眼图就足够后续学习的基础。最后整体图展现的时候，给读者一种打游戏通关的畅快之感

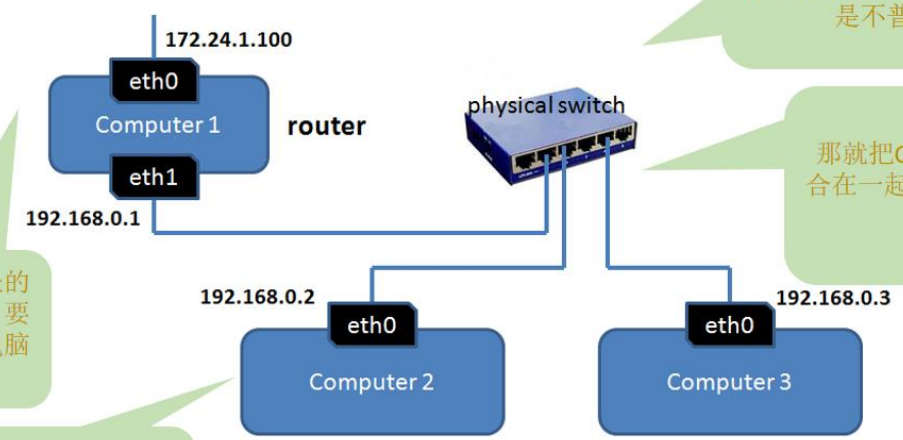




复杂的知识，从哪里开讲都不对

怎么理解？

- 回到大学寝室，或者你的家里

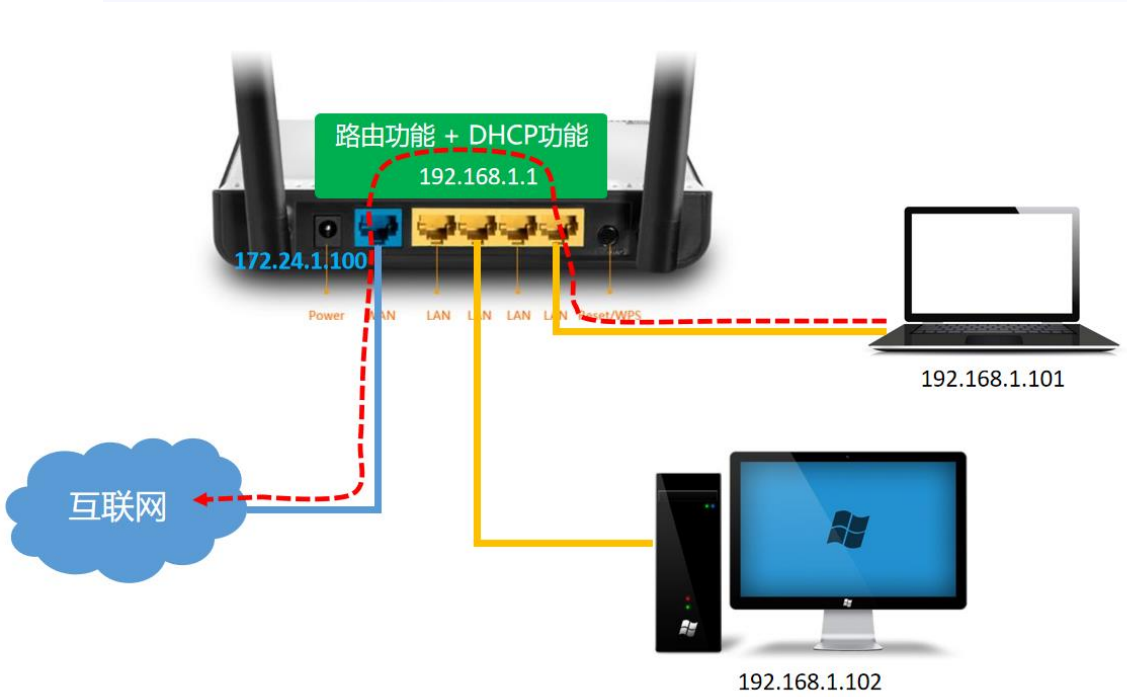


什么？你们寝室直接买路由器？好吧，暴露年龄了，2000年初的时候，路由器还是不普遍的

那就把Computer1和switch合在一起，想象成你们家的路由器

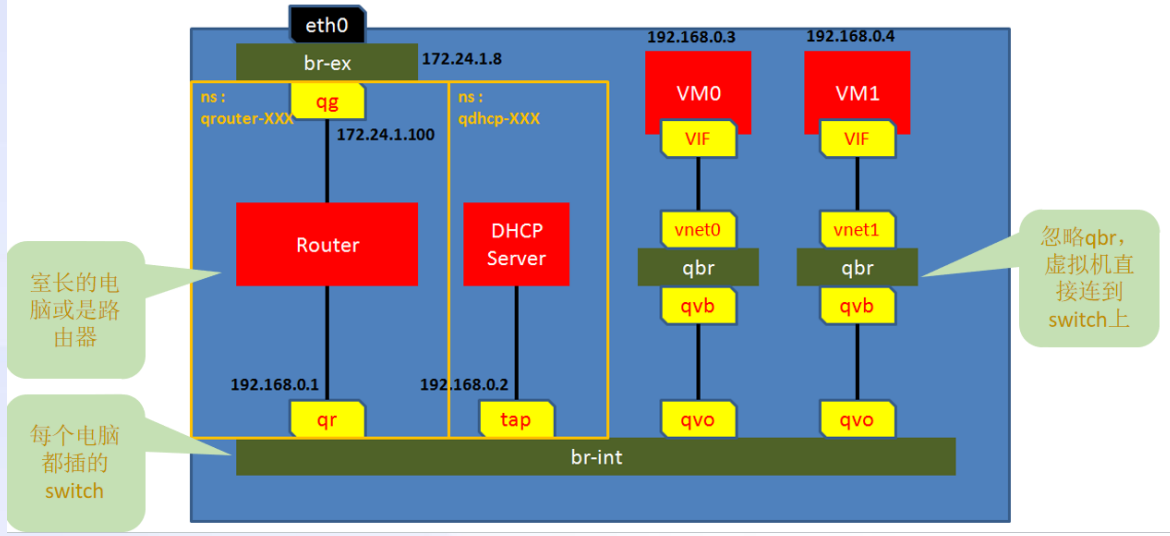
是不是只有寝室长的电脑插两块网卡，要上网，寝室长的电脑必须开着

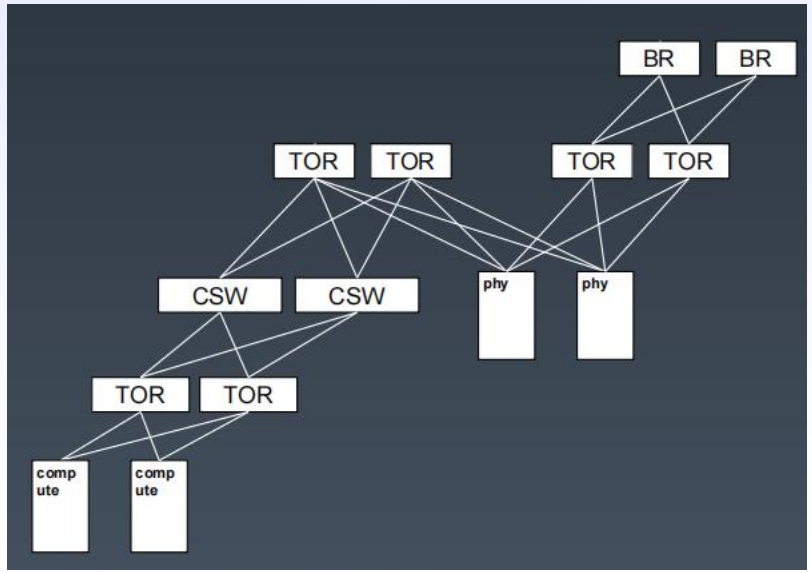
其他寝室同学只需要一张网卡

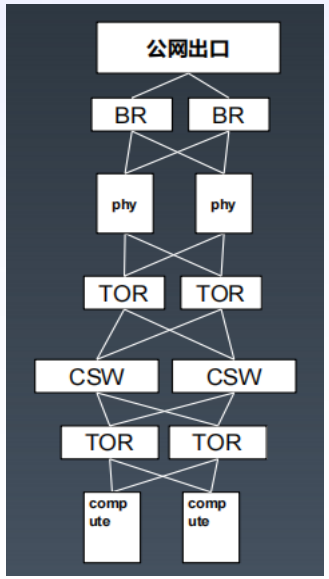


怎么理解？

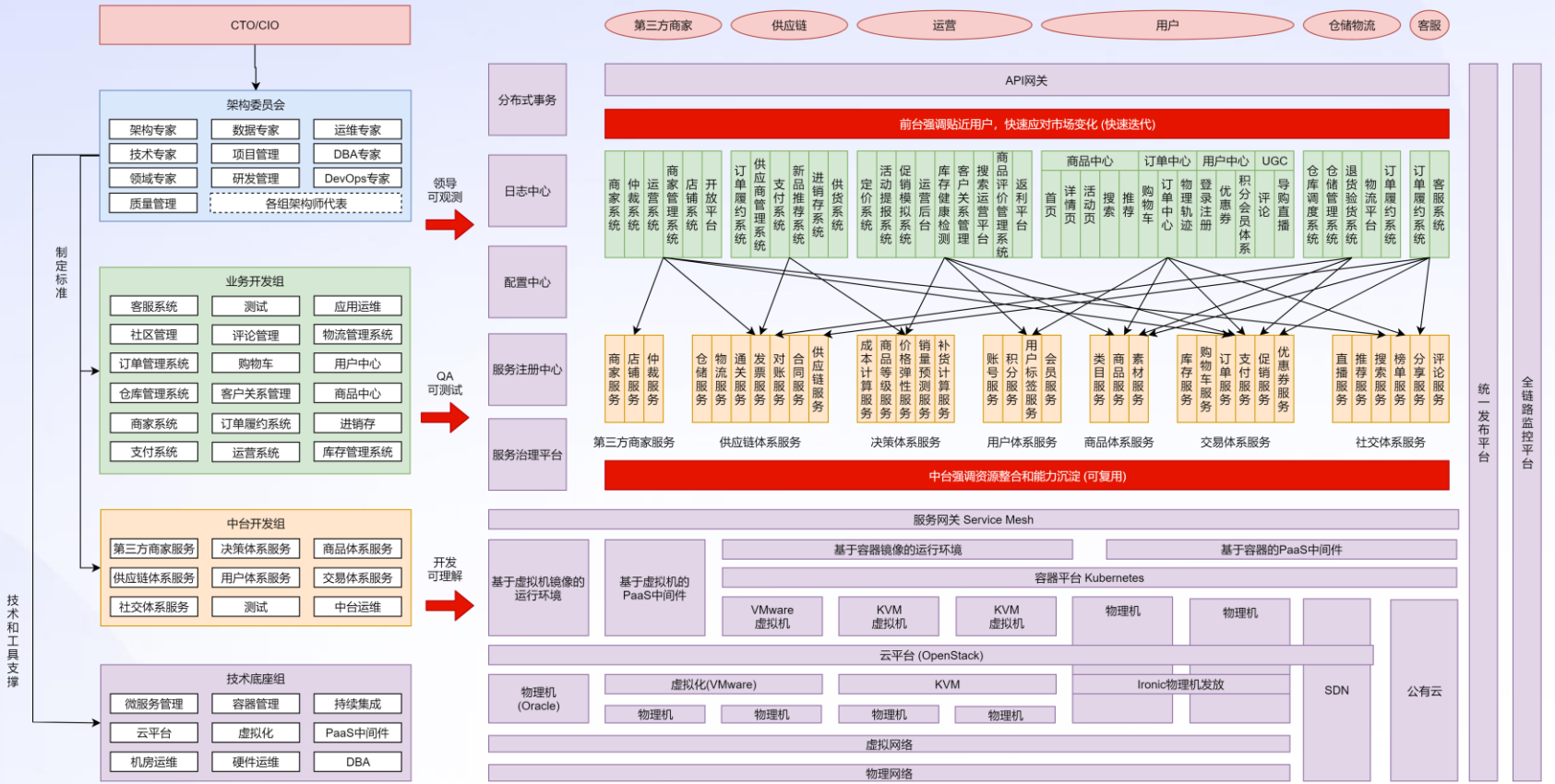
- 把寝室想象成物理机，电脑就变成了虚拟机







逐渐扩展到数据中心


CTO/CIO
架构委员会
架构专家
数据专家
运维专家
技术专家
项目管理
DBA专家
领域专家
研发管理
DevOps专家
质量管理
各组架构师代表
领导
可观测
制定标准
业务开发组
客服系统
测试
应用运维
社区管理
评论管理
物流管理系统
订单管理系统
购物车
用户中心
仓库管理系统
客户关系管理
商品中心
商家系统
订单履约系统
进销存
支付系统
运营系统
库存管理系统
QA
可测试
中台开发组
第三方商家服务
决策体系服务
商品体系服务
供应链体系服务
用户体系服务
交易体系服务
社交体系服务
测试
中台运维
开发
可理解
技术和工具支撑
技术底座组
微服务管理
容器管理
持续集成
云平台
虚拟化
PaaS中间件
机房运维
硬件运维
DBA
第三方商家
供应链
运营
用户
仓储物流
客服
分布式事务
日志中心
配置中心
服务注册中心
服务治理平台
API网关
前台强调贴近用户，快速应对市场变化 (快速迭代)
商家系统
仲裁系统
运营系统
商家管理系统
店铺系统
开放平台
订单履约系统
供应商管理系统
支付系统
新品推荐系统
进销存系统
供货系统
定价系统
活动提报系统
促销模拟系统
运营后台
库存健康检测
客户关系管理
搜索运营平台
商品评价管理系统
返利平台
商品中心
首页
详情页
活动页
搜索
推荐
订单中心
购物车
物理轨迹
用户中心
登录注册
优惠券
积分会员体系
UGC
评论
导购直播
仓库调度系统
仓储管理系统
退货验货系统
物流平台
订单履约系统
订单履约系统
客服系统
商家服务
店铺服务
仲裁服务
仓储服务
物流服务
通关服务
发票服务
对账服务
合同服务
供应链服务
成本计算服务
商品等级服务
价格弹性服务
销量预测服务
补货计算服务
账号服务
积分服务
用户标签服务
会员服务
类目服务
商品服务
素材服务
库存服务
购物车服务
订单服务
支付服务
促销服务
优惠券服务
直播服务
推荐服务
搜索服务
榜单服务
分享服务
评论服务
第三方商家服务
供应链体系服务
决策体系服务
用户体系服务
商品体系服务
交易体系服务
社交体系服务
中台强调资源整合和能力沉淀 (可复用)
统一发布平台
全链路监控平台
服务网关 Service Mesh
基于虚拟机镜像的运行环境
基于虚拟机的PaaS中间件
基于容器镜像的运行环境
基于容器的PaaS中间件
容器平台 Kubernetes
VMware 虚拟机
KVM 虚拟机
KVM 虚拟机
物理机
物理机
云平台 (OpenStack)
物理机 (Oracle)
虚拟化(VMware)
KVM
Ironic物理机发放
SDN
公有云
物理机
物理机
物理机
物理机
虚拟网络
物理网络

组织架构

技术架构

腾讯云开发者社区 | 墨问西东

THANKS

谢谢观看