> 点赞再看，养成习惯，微信搜索【**三太子敖丙**】关注这个互联网苟且偷生的工具人。
>
> 本文 **GitHub** https://github.com/JavaFamily 已收录，有一线大厂面试完整考点、资料以及我的系列文章。

前言

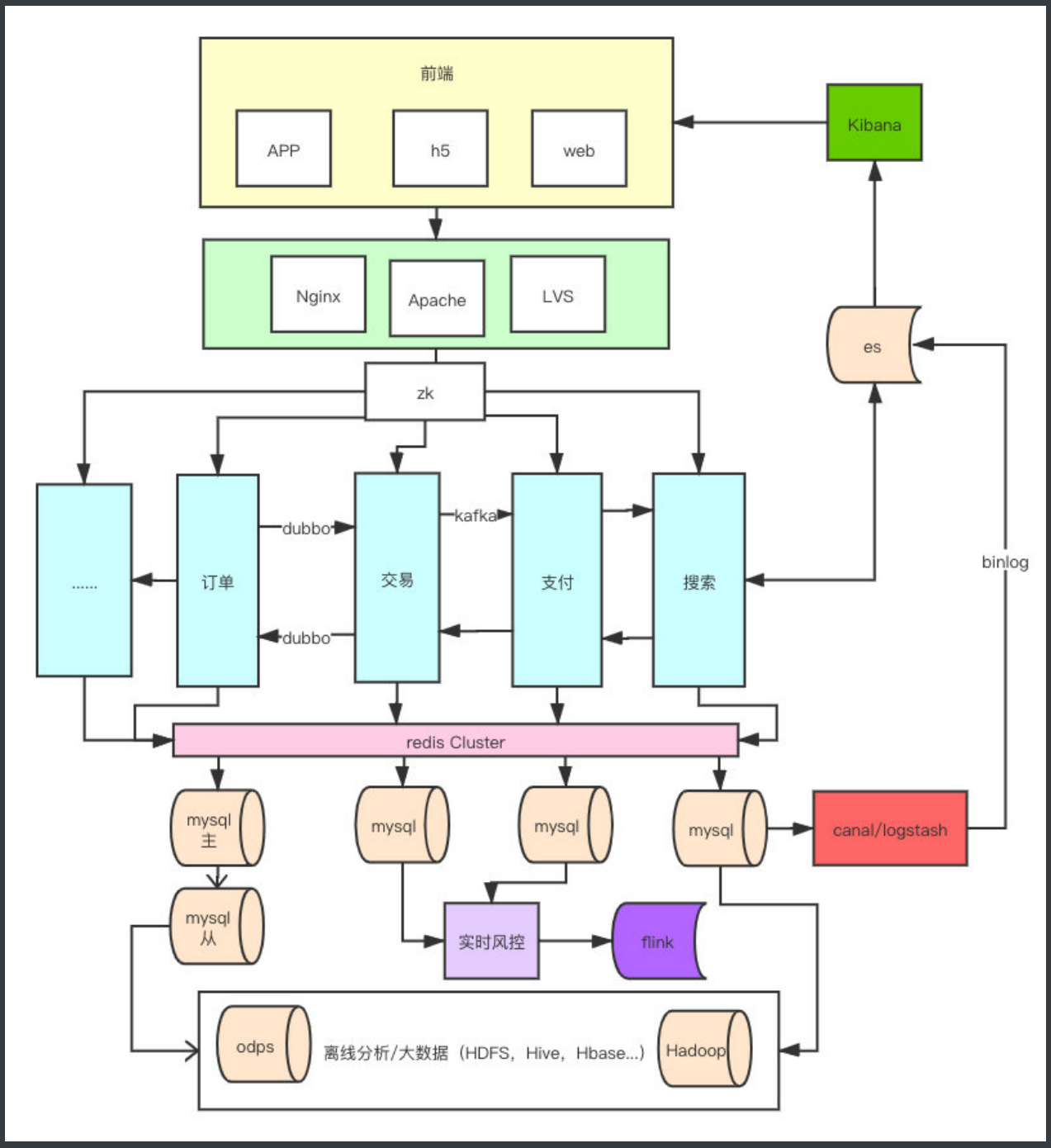
这期我想写很久了，但是因为时间的原因一直拖到了现在，我以为一两天就写完了，结果从构思到整理资料，再到写出来用了差不多一周的时间吧。

你们也知道丙丙一直都是创作鬼才来的，所以我肯定不会一本正经的写，我想了好几个切入点，最后决定用一个**完整的电商系统**作为切入点，带着大家看看，我们需要学些啥，我甚至还收集配套视频和资料，**暖男**石锤啊，这期是呕心沥血之作，**不要白嫖**了。

正文

在写这个文章之前，我花了点时间，自己臆想了一个电商系统，基本上算是麻雀虽小五脏俱全，我今天就用它开刀，一步步剖析，我会讲一下我们可能会接触的技术栈可能不全，但是够用，最后给个学习路线。

Tip：请多欣赏一会，每个点看一下，看看什么地方是你接触过的，什么技术栈是你不太熟悉的，我觉得还算是比较全的，有什么建议也可以留言给我。



不知道大家都看了一下没，现在我们就要庖丁解牛了，我从上到下依次分析。

前端

你可能会会好奇，你不是讲后端学习路线嘛，为啥还有前端的部分，我只能告诉你，**傻瓜**，肤浅。

我们可**不能闭门造车**，谁告诉你后端就不学点前端了？

前端现在很多也了解后端的技术栈的，你想我们去一个网站，最先接触的，最先看到的是啥？

没错就是前端，在大学你要是找不到专门的前端同学，去做系统肯定也要自己顶一下前端的，那我觉得最基本的技术栈得熟悉和了解吧，丙丙现在也是偶尔会开发一下我们的管理系统主要是**VUE**和**React**。

在这里我列举了我目前觉得比较简单和我们后端可以了解的技术栈，都是比较基础的。

作为一名后端了解部分前端知识还是很有必要的，在以后开发的时候，公司有前端那能帮助你前后端联调更顺畅，如果没前端你自己也能顶一下简单的页面。

HTML、**CSS**、**JS**、**Ajax**我觉得是必须掌握的点，看着简单其实深究或者去操作的话还是有很多东西的，其他作为扩展有兴趣可以了解，反正入门简单，只是精通很难很难。

在这一层不光有这些还有**Http协议**和Servlet，**request**、**response**、**cookie**、**session**这些也会伴随你整个技术生涯，理解他们对后面的你肯定有不少好处。

Tip：我这里最后删除了**JSP**相关的技术，我个人觉得没必要学了，很多公司除了老项目之外，新项目都不会使用那些技术了。

前端在我看来比后端难，技术迭代比较快，知识好像也没特定的体系，所以面试大厂的前端很多朋友都说难，不是技术多难，而是知识多且复杂，找不到一个完整的体系，相比之下后端明朗很多，我后面就开始讲后端了。

网关层：

互联网发展到现在，涌现了很多互联网公司，技术更新迭代了很多个版本，从早期的单机时代，到现在超大规模的互联网时代，几亿人参与的春运，几千亿成交规模的双十一，无数互联网前辈的造就了现在互联网的辉煌。

微服务，**分布式**，**负载均衡**等我们经常提到的这些名词都是这些技术在场景背后支撑。

单机顶不住，我们就多找点服务器，但是怎么将流量均匀的打到这些服务器上呢?

负载均衡，LVS

我们机器都是IP访问的，那怎么通过我们申请的域名去请求到服务器呢?

DNS

大家刷的抖音，B站，快手等等视频服务商，是怎么保证同时为全国的用户提供快速的体验?

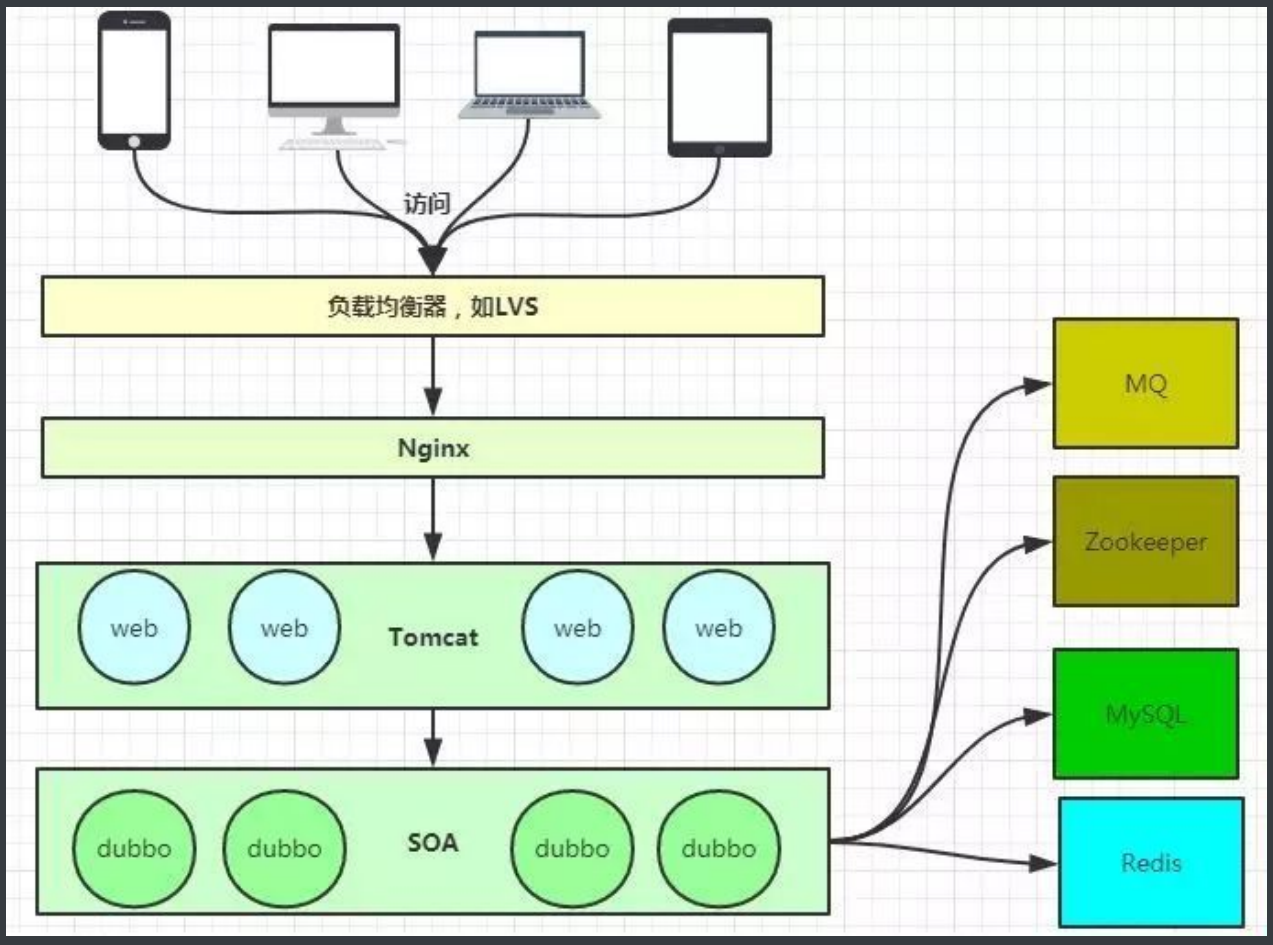
CDN

我们这么多系统和服务，还有这么多中间件的调度怎么去管理调度等等?

zk

这么多的服务器，怎么对外统一访问呢，就可能需要知道反向代理的服务器。

Nginx



这一层做了**反向负载、服务路由、服务治理、流量管理、安全隔离、服务容错**等等都做了，大家公司的**内外网隔离**也是这一层做的。

我之前还接触过一些比较有意思的项目，所有对外的接口都是加密的，几十个服务会经过网关解密，找到真的路由再去请求。

这一层的知识点其实也不少，你往后面学会发现**分布式事务，分布式锁**，还有很多中间件都离不开**zk**这一层，我们继续往下看。

服务层：

这一层有点东西了，算是整个框架的核心，如果你跟我帅丙一样以后都是从事后端开发的话，我们基本上整个技术生涯，大部分时间都在跟这一层的技术栈打交道了，各种琳琅满目的中间件，计算机基础知识，Linux操作，算法数据结构，架构框架，研发工具等等。

我想在看这个文章的各位，计算机基础肯定都是学过的吧，如果大学的时候没好好学，我觉得还是有必要再看看的。

为什么我们网页能保证安全可靠的传输，你可能会了解到**HTTP，TCP协议**，什么三次握手，四次挥手。

还有**进程、线程、协程，什么内存屏障，指令乱序，分支预测，CPU亲和性**等等，在之后的编程生涯，如果你能掌握这些东西，会让你在遇到很多问题的时候瞬间get到点，而不是像个无头苍蝇一样乱撞（然而丙丙还做得不够）。

了解这些计算机知识后，你就需要接触编程语言了，大学的**C语言**基础会让你学什么语言入门都会快点，我选择了面向对象的**JAVA**，但是也不知道为啥现在还没对象。

JAVA的基础也一样重要，**面向对象**（包括类、对象、方法、继承、封装、抽象、 多态、消息解析等），常见API，数据结构，**集合框架**，**设计模式**（包括创建型、结构型、行为型），**多线程和并发**，**I/O流，Stream，网络编程**你都需要了解。

代码会写了，你就要开始学习一些能帮助你把系统变得更加规范的框架，SSM可以会让你的开发更加便捷，结构层次更加分明。

写代码的时候你会发现你大学用的**Eclipse**在公司看不到了，你跟大家一样去用了**IDEA**，第一天这是什么玩意，一周后，真香，但是这玩意收费有点贵，那免费的**VSCode**真的就是不错的选择了。

代码写的时候你会接触代码的仓库管理工具**maven**、**Gradle**，提交代码的时候会去写项目版本管理工具**Git**。

代码提交之后，发布之后你会发现很多东西需要自己去服务器亲自排查，那**Linux**的知识点就可以在里面灵活运用了，查看进程，查看文件，各种**Vim**操作等等。

系统的优化很多地方没优化的空间了，你可能会尝试从**算法**，或者优化**数据结构**去优化，你看到了HashMap的源码，想去了解红黑树，然后在算法网上看到了二叉树搜索树和各种常见的算法问题，刷多了，你也能总结出精华所在，什么**贪心，分治，动态规划**等。

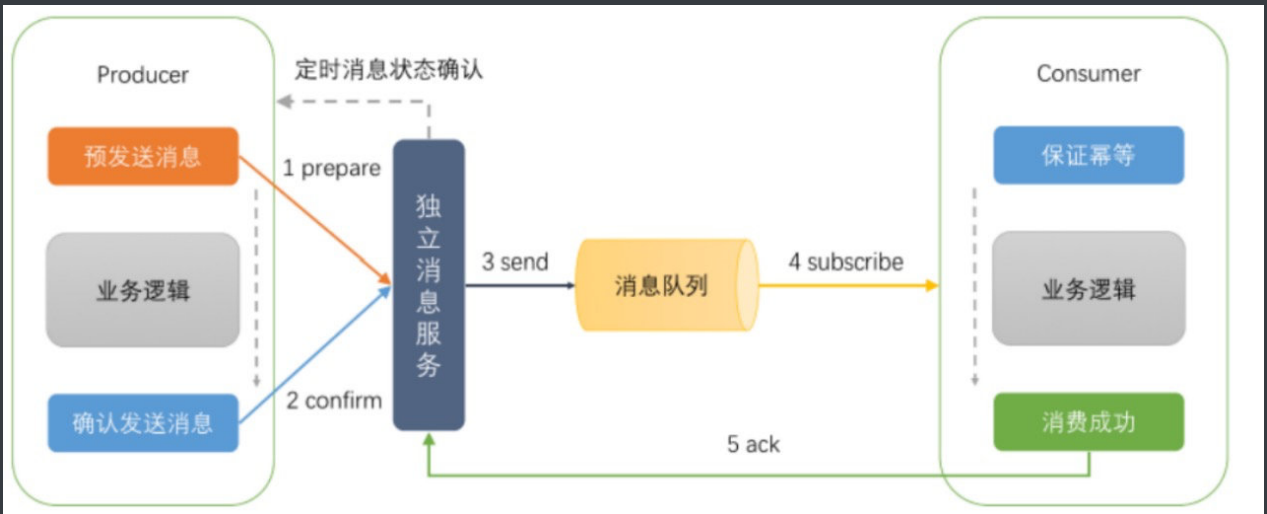
这么多个服务，你发现**HTTP**请求已经开始有点不满足你的需求了，你想开发更便捷，像访问本地服务一样访问远程服务，所以我们去了解了**Dubbo，Spring cloud**。

了解Dubbo的过程中，你发现了RPC的精华所在，所以你去接触到了高性能的**NIO**框架，**Netty**。

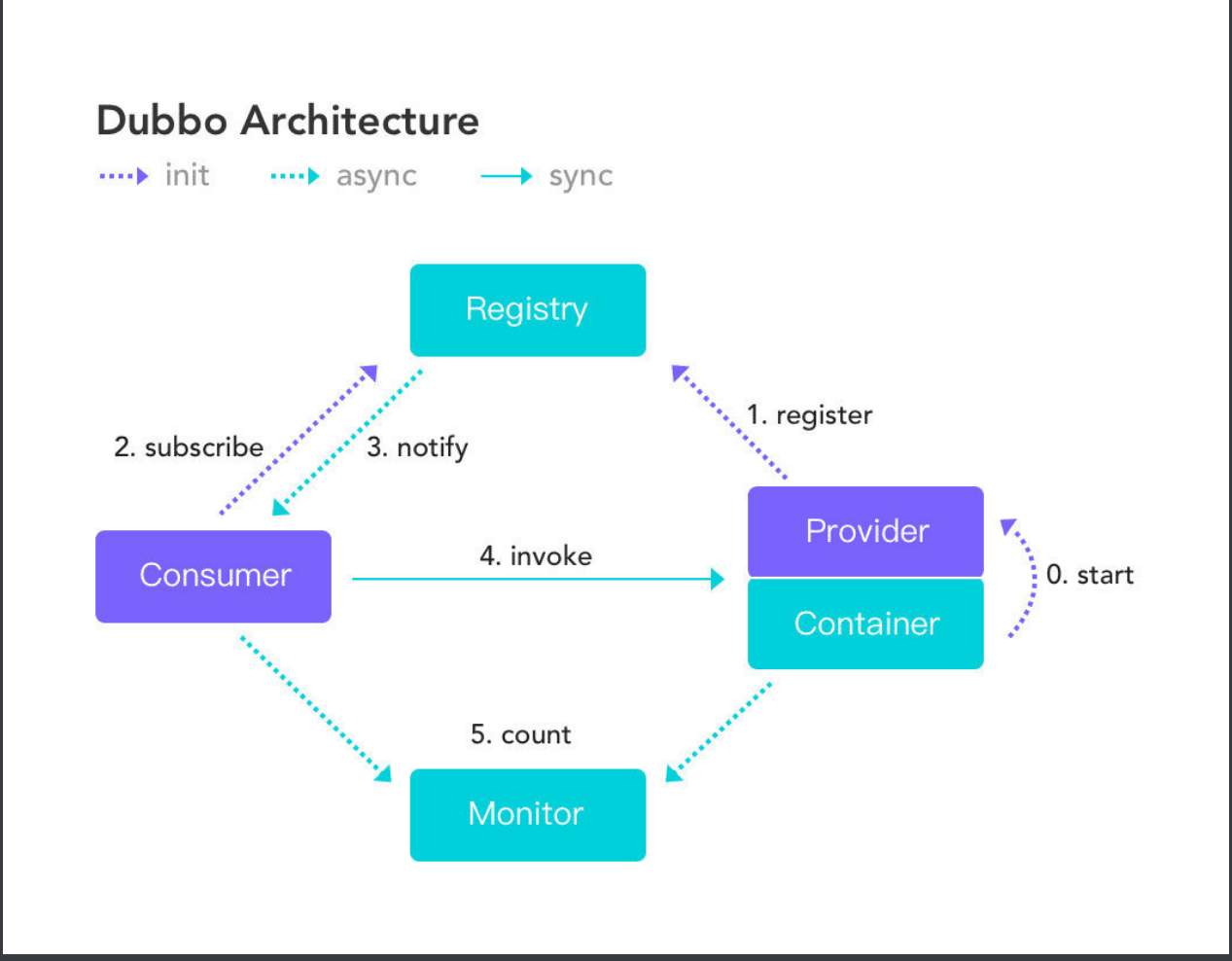
代码写好了，服务也能通信了，但是你发现你的代码链路好长，都耦合在一起了，所以你接触了**消息队列**，这种异步的处理方式，真香。

他还可以帮你在突发流量的时候用队列做缓冲，但是你发现分布式的情况，事务就不好管理了，你就了解到了分布式事务，什么**两段式，三段式，TCC，XA，阿里云的全局事务服务GTS**等等。

分布式事务的时候你会想去了解**RocketMQ**，因为他自带了分布式事务的解决方案，大数据的场景你又看到了**Kafka**。



我上面提到过**zk**，像**Dubbo**、**Kafka**等中间件都是用它做注册中心的，所以很多技术栈最后都组成了一个知识体系，你先了解了体系中的每一员，你才能把它们联系起来。



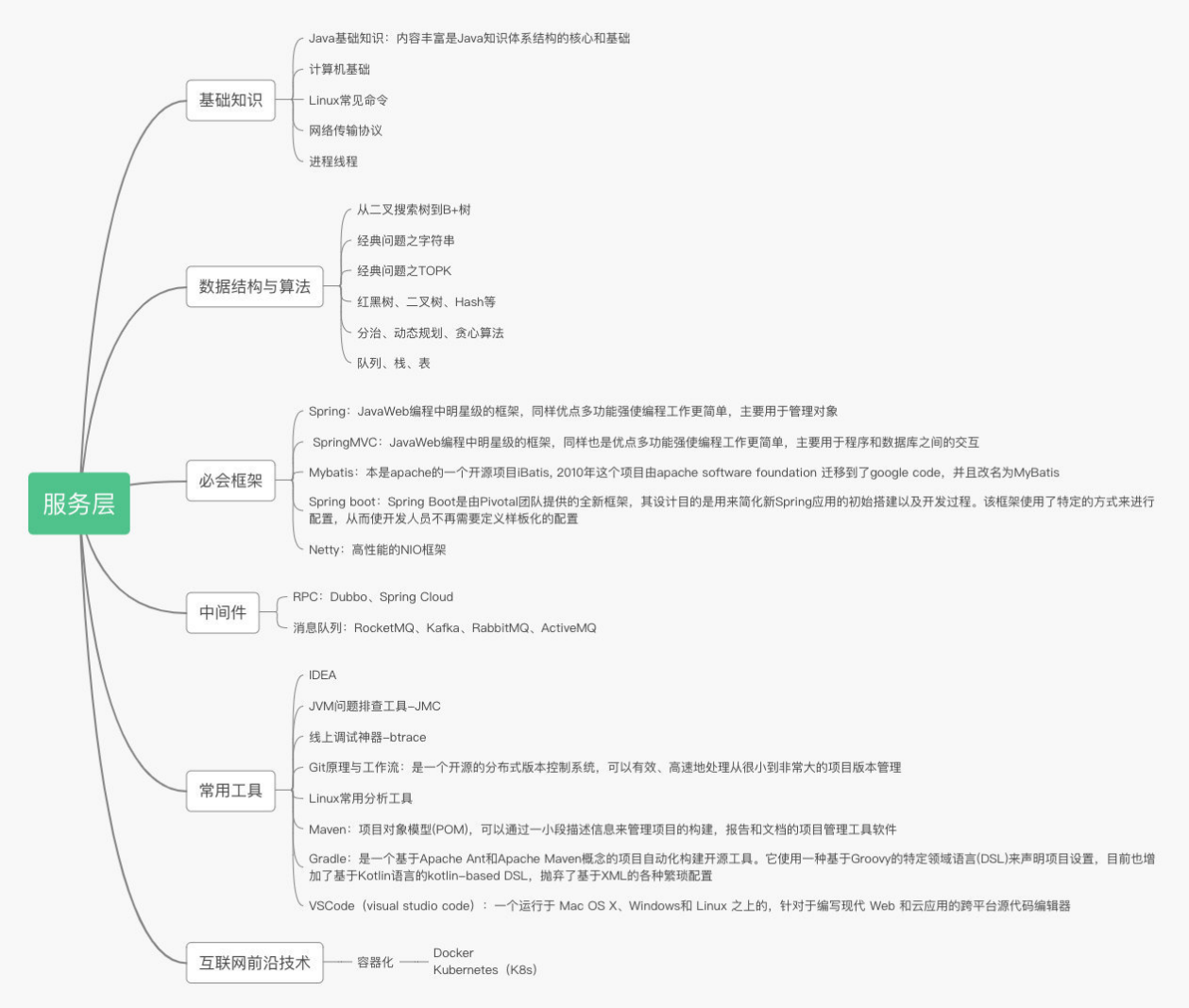
服务的交互都从进程内通信变成了远程通信，所以性能必然会受到一些影响。

此外由于很多不确定性的因素，例如网络拥塞、Server 端服务器宕机、挖掘机铲断机房光纤等等，需要许多额外的功能和措施才能保证微服务流畅稳定的工作。

Spring Cloud 中就有 **Hystrix 熔断器、Ribbon客户端负载均衡器、Eureka注册中心**等等都是用来解决这些问题的微服务组件。

你感觉学习得差不多了，你发现各大论坛博客出现了一些前沿技术，比如容器化，你可能就会去了解容器化的知识，像**Docker，Kubernetes（K8s）**等。

微服务之所以能够快速发展，很重要的一个原因就是：容器化技术的发展和容器管理系统的成熟。



这一层的东西呢其实远远不止这些的，我不过多赘述，写多了像个劝退师一样，但是大家也不用慌，大部分的技术都是慢慢接触了，工作中慢慢去了解，去深入的。

好啦我们继续沿着图往下看，那再往下是啥呢？

数据层：

数据库可能是整个系统中最值钱的部分了，在我码文字的前一天，刚好发生了微盟程序员删库跑路的操作，删库跑路其实是我们在网上最常用的笑话，没想到还是照进了现实。

这里也提一点点吧，36小时的故障，其实在互联网公司应该是个笑话了吧，权限控制没做好类似**rm -rf**、**fdisk**、**drop**等等这样的高危命令是可以实时拦截掉的，**备份，全量备份，增量备份，延迟备份，异地容灾**全部都考虑一下应该也不至于这样，一家上市公司还是有点点不应该。

数据库基本的**事务隔离级别**，**索引**，**SQL**，**主被同步**，**读写分离**等都可能是你学的时候要了解到的。

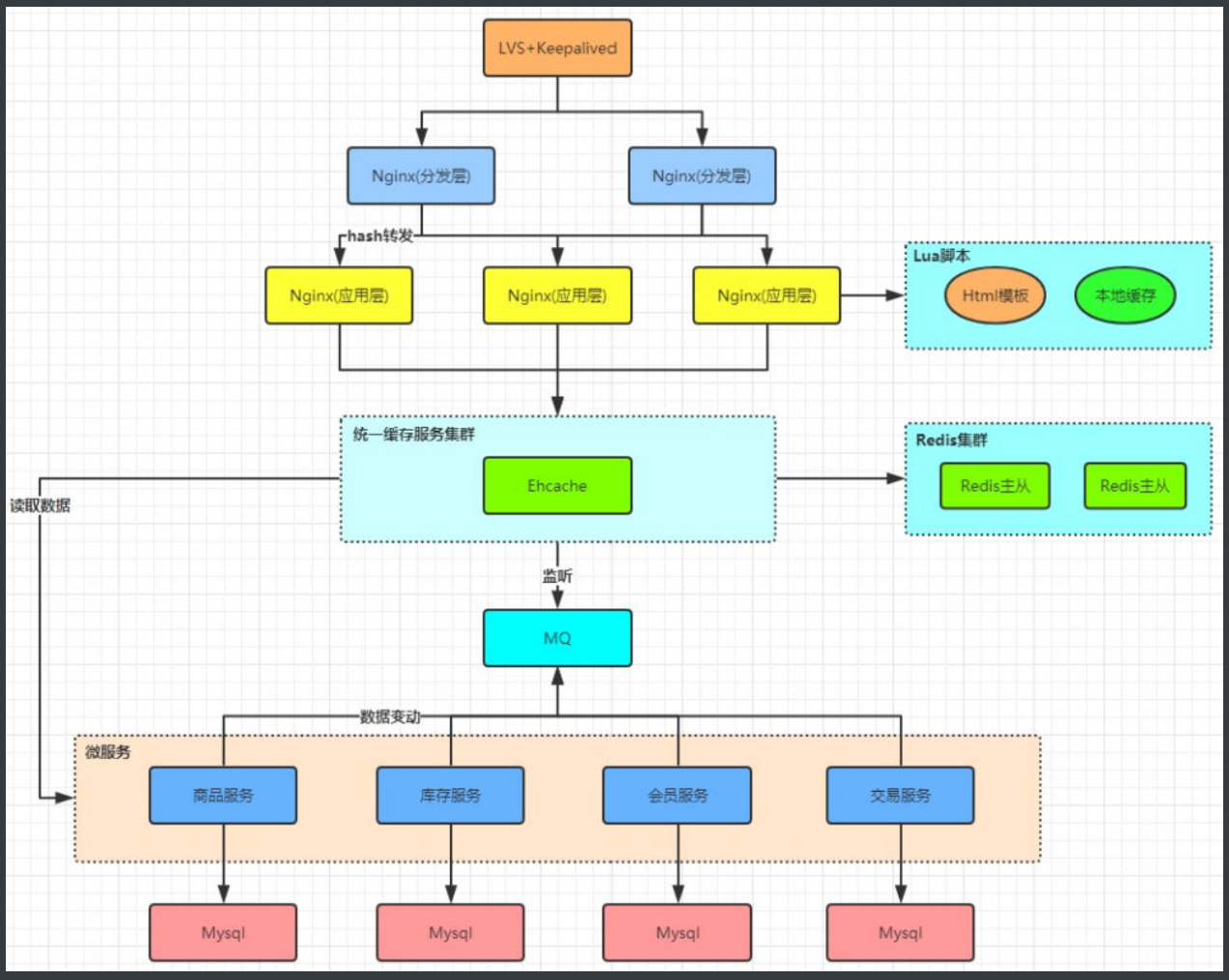
上面我们提到了安全，不要把鸡蛋放一个篮子的道理大家应该都知道，那**分库**的意义就很明显了，然后你会发现时间久了表的数据大了，就会想到去接触分表，什么**TDDL**、**Sharding-JDBC**、**DRDS**这些插件都会接触到。

你发现流量大的时候，或者热点数据打到数据库还是有点顶不住，压力太大了，那非关系型数据库就进场了，**Redis**当然是首选，但是**MongoDB**、**memcache**也有各自的应用场景。

Redis使用后，真香，真快，但是你会开始担心最开始提到的安全问题，这玩意快是因为在内存中操作，那断点了数据丢了怎么办？你就开始阅读官方文档，了解**RDB**，**AOF**这些持久化机制，线上用的时候还会遇到**缓存雪崩击穿**、**穿透**等等问题。

单机不满足你就用了，他的集群模式，用了集群可能也担心集群的健康状态，所以就得去了解**哨兵**，他的**主从同步**，时间久了Key多了，就得了解**内存淘汰机制**.....

他的大容量存储有问题，你可能需要去了解**Pika**....



其实远远没完，每个的点我都点到为止，但是其实要深究每个点都要学很久，我们接着往下看。

实时/离线/大数据

等你把几种关系型非关系型数据库的知识点，整理清楚后，你会发现数据还是大啊，而且数据的场景越来越多多样化了，那大数据的各种中间件你就得了解了。

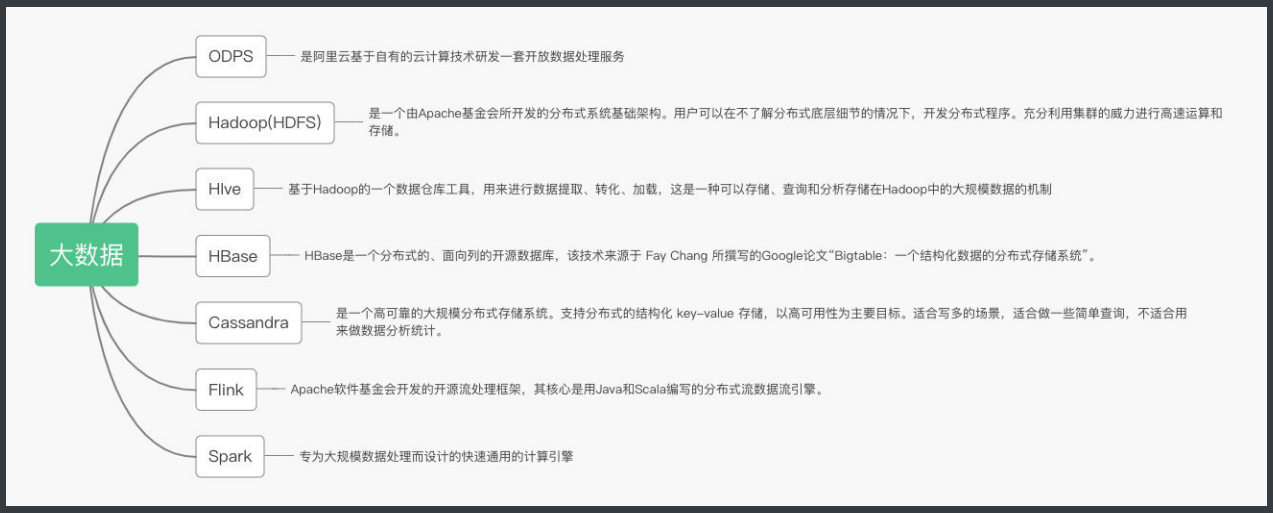
你会发现很多场景，不需要实时的数据，比如你查你的支付宝去年的，上个月的账单，这些都是不会变化的数据，没必要实时，那你可能会接触像**ODPS**这样的中间件去做数据的离线分析。

然后你可能会接触Hadoop系列相关的东西，比如于**Hadoop（HDFS）**的一个数据仓库工具**Hive**，是建立在 Hadoop 文件系统之上的分布式面向列的数据库**HBase** 。

写多的场景，适合做一些简单查询，用他们又有点大材小用，那**Cassandra**就再合适不过了。

离线的数据分析没办法满足一些实时的常见，类似风控，那**Flink**你也得略知一二，他的窗口思想还是很有意思。

数据接触完了，计算引擎**Spark**你是不是也不能放过.....



搜索引擎：

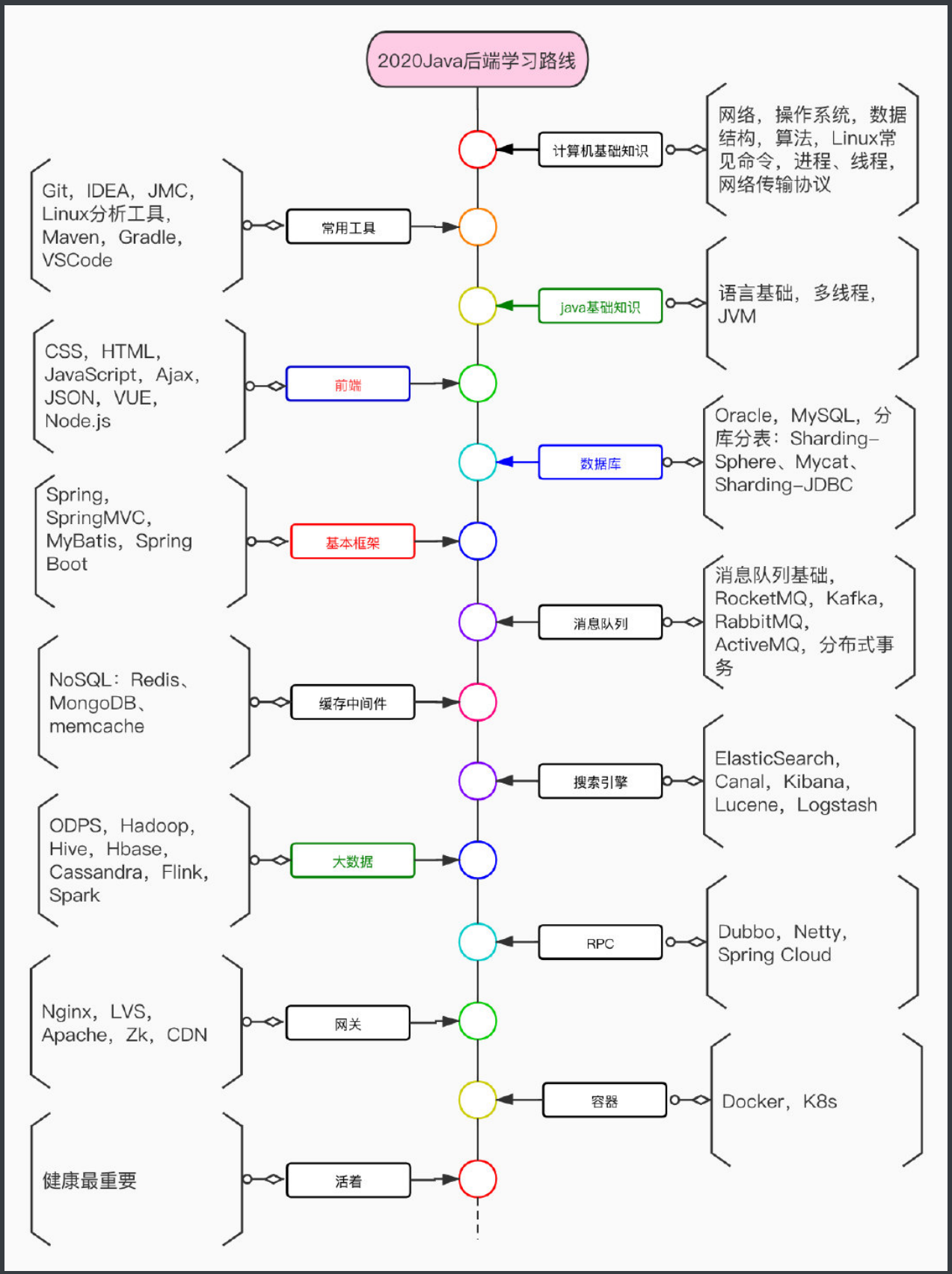
传统关系型数据库和NoSQL非关系型数据都没办法解决一些问题，比如我们在百度，淘宝搜索东西的时候，往往都是几个关键字在一起一起搜索东西的，在数据库除非把几次的结果做交集，不然很难去实现。

那全文检索引擎就诞生了，解决了搜索的问题，你得思考怎么把数据库的东西实时同步到**ES**中去，那你可能会思考到**logstash**去定时跑脚本同步，又或者去接触伪装成一台**MySQL**从服务的**Canal**，他会去订阅MySQL主服务的**binlog**，然后自己解析了去操作Es中的数据。

这些都搞定了，那可视化的后台查询又怎么解决呢？**Kibana**，他他是一个可视化的平台，甚至对Es集群的健康管理都做了可视化，很多公司的日志查询系统都是用它做的。

学习路线

看了这么久你是不是发现，帅丙只是一直在介绍每个层级的技术栈，并没说到具体的一个路线，那是因为我想让大家先有个认知或者说是扫盲吧，我一样用脑图的方式汇总一下吧，如果图片被平台二压了，可以去公众号回复**【路线】**。



资料/学习网站

JavaFamily：由一个在互联网苟且偷生的男人维护的GitHub

CodeGym：一个在线Java编程课程，80%的内容是练习，适合一窍不通的入门者。

Wibit Online Java Courses： 一个非常有趣的编程学习网站，各种生动的动画形象能让人忘记学习的枯燥。在线视频学习，非常适合零基础。

stanford CS106A: Programming Methodology： 斯坦福经典课程系列，完全没有编程经验，想学Java语言的，可以看看这个课程。

Bloombenc： 一个在线交互式学习平台，老师可以根据你的学习能力和节奏修改他们的教学方法，还可以在平台上编码。

Imooc：慕课网，我大学的C语言就是在这里看的

CodeAcademy： 比较实用的Java在线课程，注重的是在找工作时非常有用的技术能力。

PLURALSIGHT：整合了很多Java的视频课程，部分免费，部分付费，可以根据自己的需要挑选。

Lynda Online Java Training Videos：Java进阶课程，包括如何使用JDBC来集成MySQL数据库，Reflection API，管理文件和目录等。

九章基础算法班（Java）：中文在线互动课，随时开始学习。

BeginnersBook：Java初学者免费教程，有稍微一些编程基础之后，可以跟着文档里的代码练习。

docs.oracle.com/javase/tutorial：官方Java指南，对了解几乎所有的java技术特性都非常有帮助。

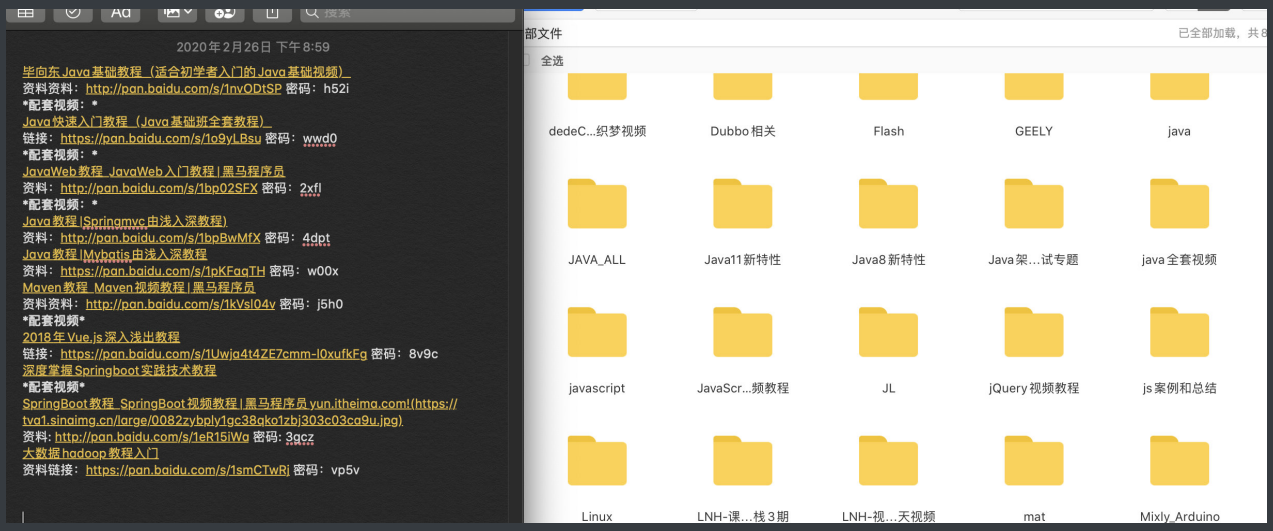
JournalDev：Java相关教程及问答

JavaWorld：最早的一个Java站点，每周更新Java技术文章。

developer.com/java： 由http://Gamelan.com 维护的Java技术文章网站。

IBM Developerworks技术网站：IBM的Develperworks技术网站，这是其中的Java技术主页

Tip：本来这一栏有很多我准备的资料的，但是都是外链，或者不合适的分享方式，博客的运营小姐姐提醒了我，所以大家去公众号回复【**路线**】好了。

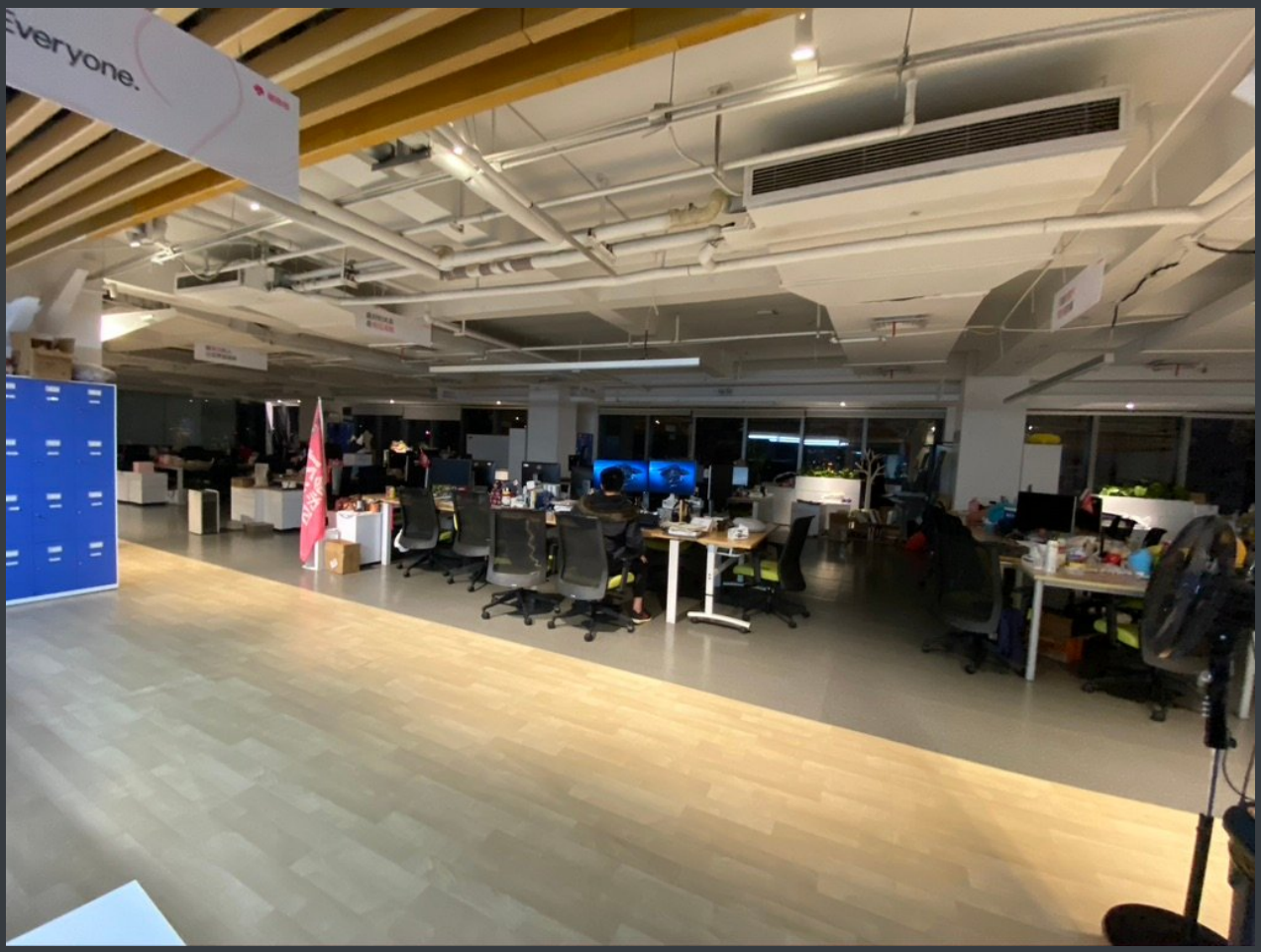


絮叨

如果你想去一家不错的公司，但是目前的硬实力又不到，我觉得还是有必要去努力一下的，技术能力的高低能决定你走多远，平台的高低，能决定你的高度。

如果你通过努力成功进入到了心仪的公司，一定不要懈怠放松，职场成长和新技术学习一样，不进则退。

丙丙发现在工作中发现我身边的人真的就是实力越强的越努力，**最高级的自律，享受孤独**（周末的歪哥）。



总结

我提到的技术栈你想全部了解，我觉得初步了解可能几个月就够了，这里的了解仅限于你知道它，知道他是干嘛的，知道怎么去使用它，并不是说深入了解他的底层原理，了解他的常见问题，熟悉问题的解决方案等等。

你想做到后者，基本上只能靠时间上的日积月累，或者不断的去尝试积累经验，也没什么速成的东西，欲速则不达大家也是知道的。

技术这条路，说实话很枯燥，很辛苦，但是待遇也会高于其他一些基础岗位。

所实话我大学学这个就是为了兴趣，我从小对电子，对计算机都比较热爱，但是现在打磨得，现在**就是为了钱**吧，是不是很现实？若家境殷实，谁愿颠沛流离。

但是至少丙丙因为做软件，改变了家庭的窘境，自己日子也向小康一步步迈过去。

说做程序员改变了我和我家人的一生可能夸张了，但是我总有一种下班辈子会因为我选择走这条路而改变的错觉。

我是敖丙，一个在互联网苟且偷生的工具人。

创作不易，本期硬核，不想被白嫖，各位的**「三连」**就是丙丙创作的最大动力，我们下次见！

> 文章持续更新，可以微信搜索「 **三太子敖丙** 」第一时间阅读，回复【**资料**】有我准备的一线大厂面试资料和简历模板，本文 **GitHub** https://github.com/JavaFamily 已经收录，有大厂面试完整考点，欢迎Star。


三太子教育
微信搜索