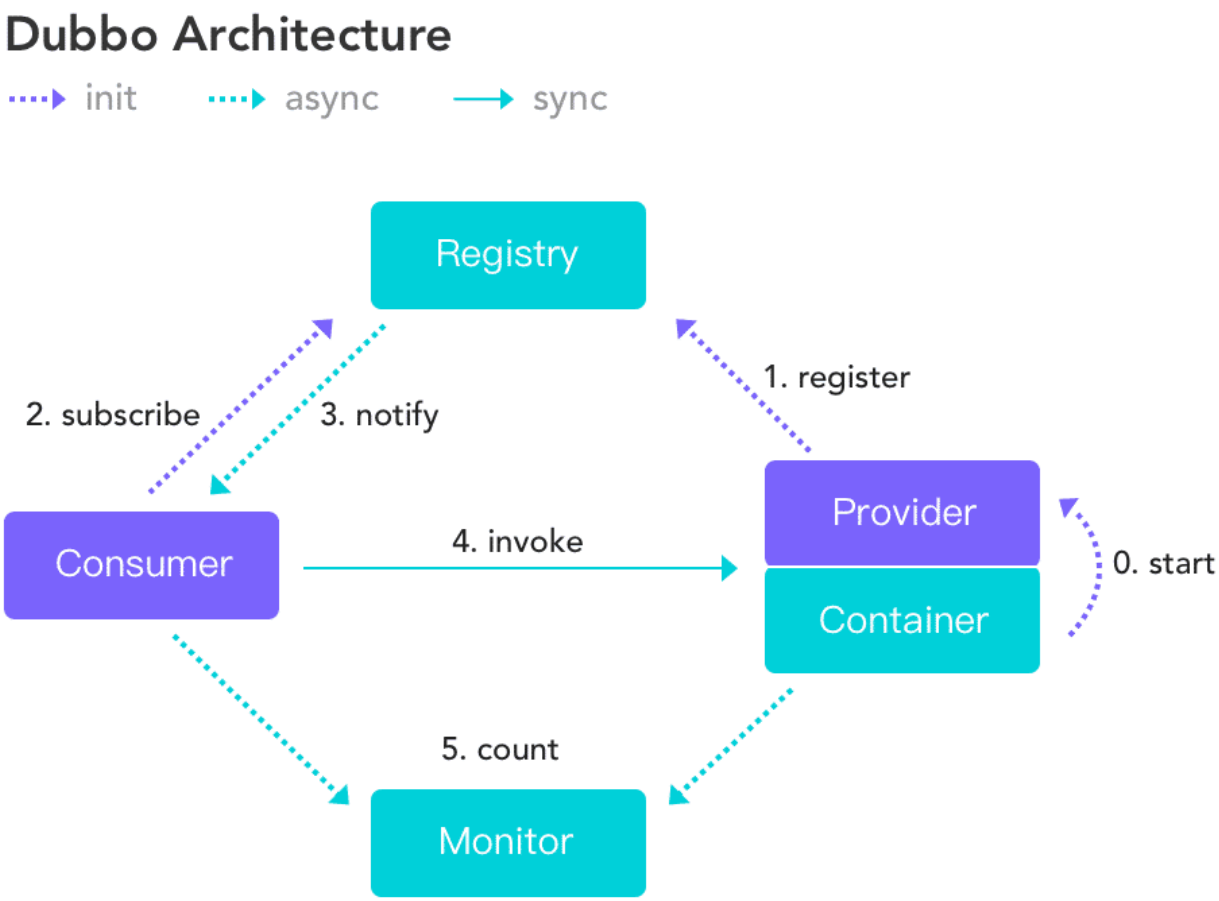
Diving-in-Dubbo

> Apache Dubbo™ *是一款高性能、轻量级的开源Java RPC框架，它提供了三大核心能力：面向接口的远程方法调用，智能容错和负载均衡，以及服务自动注册和发现。*

翻开Dubbo的官方文档的首页，首先映入眼帘的是如上一段醒目的文字，还有右边如下关于Dubbo架构的表意图示：



Dubbo已经是 Apache 基金会的顶级项目，甚至一些后起的框架也在汲取其精华、借鉴它的设计。在 Java 开发社区，它有着大量的拥趸，甚至吸引了很大一个研读源码的群体。在开发者眼中，它被当做一个微服务框架对待，这早已超过 RPC框架 的范畴。它的成功离不开如下本身丰富的特性支持，更离不开阿里这个千亿规模的企业对开源文化的拥抱和贡献。

面向接口代理的高性能RPC调用

提供高性能的基于代理的远程调用能力，服务以接口为粒度，为开发者屏蔽远程调用底层细节。

智能负载均衡

内置多种负载均衡策略，智能感知下游节点健康状况，显著减少调用延迟，提高系统吞吐量。

服务自动注册与发现

支持多种注册中心服务，服务实例上下线实时感知。

高度可扩展能力

遵循微内核+插件的设计原则，所有核心能力如Protocol、Transport、Serialization被设计为扩展点，平等对待内置实现和第三方实现。

运行期流量调度

内置条件、脚本等路由策略，通过配置不同的路由规则，轻松实现灰度发布，同机房优先等功能。

可视化的服务治理与运维

提供丰富服务治理、运维工具：随时查询服务元数据、服务健康状态及调用统计，实时下发路由策略、调整配置参数。

使用 Java 做企业级开发离不开大量优秀开源框架的支持，然而早年间这些框架大部分是中国本土以外的开发者贡献的，国人鲜有贡献。然而随着阿里在商业上的成功，广袤的中国市场、海量的网络请求，传统开源件也渐渐变得难负重任。阿里去IOE的进程中，投入重金网罗并培养了大量优秀人才，他们博采众长，吸纳了大量国际开源社区关于分布式开发的最佳实践，开发了不少优秀的分布式基础框架或中间件。它们经历了阿里这个生态环境的捶打，在实战中被烤炼，经受了高负荷的考验，阿里秉持开放心态将其中一些开源了，Dubbo就是其中之一。阿里开源的举动为一些开发者带来了福音，同时也带动了国内其他一些互联网公司对开源的拥抱。自此，越来越多的国人开发者涌现在开源社区，早年不贡献只索取的态势也在渐渐改变，而这也促进了国内IT方面的工程水平和能力，也直接促进了这几年中国移动互联网的高速发展。

Why ?

《深潜Dubbo》于本人而言，是一本大部头的技术书籍。类似剖析Dubbo源码的网文汗牛充栋，为什么我坚持要写?

其实在写此书之前，我有过几年的分布式和微服务的开发经验。然而就像绝大部分开发者一样，重心更多放在业务逻辑开发上，对框架的实现原理却鲜有了解，大部分时候能够用得很灵活，但遇到问题后，却没法快速定位出根源因素。加上国内IT界风行的加班文化，几乎很难挤出时间去深度了解一门技术的底层实现原理，更别说通读源码，数次的尝试，数次的不了了之。开发框架就像一把双刃剑，给业务开发人员提供了极大便利的同时，其底层运行机制或实现原理却像幽灵一般，拒我们于千里之外。技术人本身的言语木讷，加上本人对刷题的天生抗拒，没法讲透的机制原理，所有这些让自己陷入一种技术人的 穷人困境 。

于喜爱编程的人而言，能够成为技术人是一件非常幸福的事情，一个人，一台电脑，他就能享受着创造的乐趣。我想，女娲所在的世界虽然孤苦清冷，但是能够捏泥为人，她必定是幸福的。早些年，个人认为技术的存在意义是用于解决业务问题的，因而将重心偏于应用开发，讲究代码的可扩展性和适配能力，享受着编码所带来的乐趣，然而却天真的想依靠参与产品开发完成 高级技能 的实践与内化，忽略了深度的理论积累，职业上因此没法及时获得突破，时常怀疑自己是否“叶公好龙”。

Java 是一门面向对象的编程语言，换言之，它是很讲究编码结构的，比如业界所提倡的面向对象编程和设计的 S.O.L.I.D原则 (https://learnku.com/articles/4160/solid-notes-on-object-oriented-design-and-programming-oodoop)，其实设计模式正是遵守这些原则所总结出来的最佳实践。优良的编码结构是创建一个易于维护和扩展的软件系统的基础，即使系统在不断的迭代中，也不会因此而变得混乱。曾经见过一些业务开发人员，包括来自某些大厂的，所编业务代码相当凌乱，他们甚至不知道什么是重构。后续接力的开发人员功底再扎实、能力再强，在这种凌乱面前适配变化也会显得无所适从，即便拥有好的调试方案，也会如履薄冰，生怕锅从天上来。

一门技术或者一个框架都是为了解决某类通用问题而存在的，为了适配问题变化的多样性，有着各种各样的技能点。然而通常，这些技能点，于大多数只能接触到增删改查操作的业务开发人员而言，只有不到20%会在日常编码中被我们用到，书本上学到的其它技能点很多是没法亲自去实践一遍的，比方说 Spring 框架多处用到的代理机制。

How ?

一个优秀的框架，它必定是遵守 S.O.L.I.D原则 的，有着优良的代码结构和极强的可扩展能力，同时为了框架的强大也会应用到一些 高级技能 。Dubbo作为一个 RPC 框架， 微内核+插件 的设计原则保证它了灵活的扩展性，还应用了诸多分布式的技术，经历过大厂生产环境的严苛验证，代码简短精悍，诸如此等，让我有了对它探索的欲望。

框架源码很大程度上更像一部剧本，是由很多零散的细节有机组合“拼凑”而成的，单就某个细节，我们很难理解它的存在，只有把它放大到一个更为宏大的系统层面，纵向需要知道它的前世今生，横向需要察觉它与周遭的联系，只有这样我们才能通晓其存在的形式与意义。而这一纵一横，却急速加重了人脑的认知负荷。这种负荷普通人是难以承受的，但是厉害的编剧，即便撰写过多部剧本，它们依然可以部部精彩，没有重样。这其中的奥秘就在于，尽管细节浩如烟海，但细节的成因或者细节间的组合关系却逃脱不了有限的几个套路。换言之，一个厉害的编剧或者导演，必定是一个套路梳理和总结的高手。

有人说如果要想要了解一片海域的生态情况，最好的方式是去研究它的珊瑚礁，它会带领你了解到所有的秘密。然而，假如想要了解该海域状况的是一只鱼，那么势必会困难重重，因为据说鱼儿的记忆只有区区5分钟。其实于我们大多数人而言，阅读源码是一件非常痛苦的事情，我们的处境就像前面所说的鱼儿一样，并不会好太多。

然而，这个过程却又是十分必要的，它是我们熟练掌握某种技术的必经之路，一种技能或者设计手段，只有内化了，才足矣自诩掌握了，就像学会游泳一样，此后，无论间隔多久，跳进水里，你依然会游，只是技巧上可能有些生疏了。因此就像编剧一样，我们阅读源码的目的不是为了知道，而是为了让自己浸润在那个语境中，熟知或者总结出一个个能够成为剧作的套路，让这些渐渐内化的套路为己所用。

有人会说，写得好的书籍那么多，何必大费周章去研读源码了？也许你和我有过一样的感觉，曾经接触过很多介绍技术的书籍，会把实现的机制或者原理描述得很生动易懂，但是当我们放下书本去思考到底是怎么实现的时，却往往没法想出个所以然出来，真是拿起时“学富五车”，放下后却“江郎才尽”，难免

会陷入智不如人的自我否定中。深入原因是组成一个系统的所有细节或技能点之间有着纵横交错的关系，这些书籍着重讲了单个背后实现的机制，却往往忽略了它和周边的关系，也忽视了不同读者理解的差异性。

曾经多次阅读源码的尝试，也让我意识到读懂不等于通透，内化最好的方式是尝试去尝试剖析所以然，然后将其写下来，于是这进一步促成我去写《深潜Dubbo》一书。遗憾的是，日常自然语言中同样一句话，不同的人会有不同的表达，理解的意思也不尽相同，不像程序源码，能够精确地表达某个过程或者意图，几乎不存在二义性，因此请原谅我没法给您不含大量源码的《深潜Dubbo》。另外，于后进生而言，学生时代老师眼中所谓“显然”，往往是他们所跨不过去的一道坎，一丁点的懵懵懂懂却常常酿成大面积的知识短路，因此本书中关于机制的剖析中，我除了详细阐述实现过程外，还会尽量在该过程不被扰乱的前提下，将相关源码打散重组，剔除不相干部分，然后再呈现给读者您。

Then ?

有朋友曾建议我将构成《深潜Dubbo》的序列文章以公众号的形式分享出来，在仔细思考之后，个人觉得不妥，理由是阅读源码是一个思绪逐渐展开需要慢慢咀嚼的过程，并不适合在移动场景中进行。另外，定期不定期的放出一些 标题党 式的文章来，无形中会给大家带来某种焦虑感，IT人所在的行业，本身就节奏快压力大，我应诚意对待自己读者，尽量少地打扰您们。

通读框架源码，尝试过的读者会知道，这个过程会比较辛苦，会因为各种原因没法坚持下去。不过此刻，我却为您感到开心，既然您能读到这里，大概率说明您也遇到过和我同样的困境，或者不希望走上和我同样的弯路，也在积极寻求突破，期待自己能够获得真知，而不仅仅是想停留在“知道”这个层面。《深潜Dubbo》一书的创作过程中，我一直秉持着作品心态，期待能够帮助到更多的读者，带着他们还有您，一起去克服通读源码时遇到的重重困难。

最后，非常期待您的反馈和支持，它将鼓励我继续创作，也会让此书变得更加完善。如果您觉得本书能帮上自己，希望同我一起坚持完成对Dubbo框架源码的探索，那么我鼓励您尽量多地给本身赞赏，可能它本身并不值那个价值，但坚持无价，赞赏给了您还有我坚持下去的勇气和决心。况且您不必像我一样辞掉工作专心研读源码，只需挤出下班之后的一点业余时间随我梳理的思路慢慢品味，相信您会不虚此行。

静以储势·Shuke 的赞赏码

如果此书能为您辉煌的职业生涯贡献一点点力量的话，那将是我人生莫大的荣幸。