目录

01、为什么我们要做这个架构师课程

我从 2015 年开始，以“中华石杉”作为网名跟某网站合作，开始出线上课程（两个 Spark 大数据相关的课程），后来一直到 2017 年，又跟另外一个网站合作，也推出了两个线上课程（亿级流量电商网站的商品详情页系统架构实战、Elasticsearch 顶尖高手）。一直到如今的 2018 年，实际上一晃我利用业余时间在线上视频课程这块，也已经做了 3 年了。

3 年内我积累了大量的学员，承蒙部分学员的厚爱，确实得到了不少同学对我的授课风格以及课程质量的认可。**但是在指导大量的学员就业的同时，我发现了很多同学职业发展的问题**。很多同学都是积极好学，自己看了很多的书，网上买了很多的视频，也参加过一些培训课程，但是发现自己的技术始终会在某个瓶颈处徘徊，觉得技术始终没法达到一个很高的位置。

为什么呢？**这里我援引大量学员给我的反馈**：我是看了很多书，我是看了很多视频，我是参加了一些大几千块钱、甚至上万块钱的培训。但是我学到的仅仅是一些理论知识，写一些 HelloWorld，仅仅是明白了一些技术的原理，然后平时参加线上学习，学到的所谓“项目”，其实都是十几个小时，或者几十个小时，充其量就几千行代码的小 Demo，跟大公司里复杂生产环境下的大型项目，简直相差了十万八千里。根本不知道复杂的大系统在线上几百台甚

至上千台服务器的环境下，如何开发、测试、部署、回滚、监控和报警。也根本不知道在一个业务极度复杂的系统中，缓存到底该怎么用？MQ 到底该怎么用？怎么分库分表？怎么读写分离？不同的机器用什么样的配置可以抗住多高的并发量？虚，心里很虚！

上面就是很多我的老学员给我的真实反馈，在这个过了 40 岁码农就很难混的年代，每个人内心都充满了焦虑。**很多学员告诉我**，他们去外面面试，自己也通过看书，学习视频课程，参加线上培训，准备了不少东西，但是去面试了就傻眼了，因为发现人家要的是你有过真真实实复杂而且大型的项目经验，这个项目最好是能支撑上亿用户，高并发、高可用、分布式的大型复杂系统。结果发现，自己准备的大量理论知识，人家问了问，还是不会给你所谓的架构师等高阶的职位。

于是实际上，**互联网行业内出现了这样的一个阶层固化：**

很多学历在大专~普通本科，常年在中小型公司工作的同学，一直重复做着技术含量很低的工作，公司就是没什么用户量，就是没什么技术挑战，你又能做出什么花样来呢？然后自己看书、看微信公众号推送文章、看各种视频课程、参加线上培训，却发现学习到的始终还是各种理论知识、HelloWorld 程序以及 Demo 项目。出去面试，始终迈不过去，没有高并发、高可用、分布式的大型系统架构经验的这道门槛。我见过大量这样的同学，可能工作也 5 年，8 年，甚至 10 来年的都有，但是技术能力就停留在高级 Java 工程师的水准，薪资也是在 20 多 k 到 30k 之间徘徊。虽然不是说每个普通出身的同学都会这样，也有人出身普通，但是凭借着自己的努力做的非常好的，但是我身边却是大量的普通出身的学员都是上面的这个进退两难的情况。

然而很多大公司的同学，都是 211、985 的名校硕士背景出身，毕业就去各大互联网公司拿 15k 甚至 20k（这两年校招薪资猛涨）的 offer，接着工作 2 年后，带着支撑过几千万用户量的项目经验跳槽一下，就是另外一个大公司的高级 Java 工程师，薪资 20 多 k 的样子；接着再干两三年内，带着支撑过上亿用户量的大型系统的经验再跳槽一下，就可以出去带个团队，拿着技术专家的 title，薪资可能都 30 多 k 了；再干个两三年，带着有团队管理经验+几亿用户量的大型系统经验，出去跳个槽，也也许就是某小公司的技术经理、技术总监的级别，薪资四五十 k，甚至五六十 k，手握创业公司的大量期权，买房买车，人生大赢家。虽然不是说每个名校硕士出身的人都可以这样一帆风顺，但是确实这样的人我身边很多，比比皆是。

这就是所谓的我们这个行业内的阶层固化，因为我其实一直都是第二种情况出身，所以在做线上课程以前，根本也不太了解第一种同学的情况，但是这 3 年的线上授课经验+与大量的学员的沟通交流，**我现在也深刻了解了这些中小型公司的同学在职业发展道路上的困难**。

在了解大量学员的职业发展困难，以及行业现状之后，我对每个学员都指出了一个至关重要的实现人生逆袭的要点：看书、看视频只是一个辅助作用，起不了决定性因素，你们真正需要的，是能够在一个大型、真实而且业务极度复杂的系统中，经历一遍从 0 到 1，从 1 到 10，从 10 到 100 的全过程。从 0 到 1 的时候，如何带领一个团队快速迭代完成一个 10 万行代码的系统的交付，同时还能保证系统的流程规范性以及代码质量？从 1 到 10 的时候，如何带领一个团队，以微服务架构的方式开发来交付一个几十万行代码的复杂系统，快速迭代大量的业务逻辑，同时完成良好的系统架构设计，保证系统能承载每秒钟几千的并发量？从 10 到 100 的时候，如何在业务极度错综复杂、代码量高达几十万行的业务系统中，引入

各种各样的技术，实现一套高并发、高可用、高稳定性、高性能的系统架构？你有没有能力自己为公司写一个研发框架，将公司的开发效率提升 10 倍以上？你有没有能力自己为公司研发一个复杂的分布式中间件系统，在没有开源项目合适的情况下来支撑公司业务的快速发展？

完整经历上面的真实大公司工作场景，才是一个普通背景出身的同学，要逆袭的关键点，在这个过程中，你学习的所有的技术和知识，都是为了解决一个几十万行代码的复杂业务系统中的各种真实棘手的问题，学习技术不是用来写 HelloWorld 和 Demo 的，学习技术是要用来解决线上系统的真实问题的。

这个过程，只有用至少 1~2 年的时间真实的经历了一遍，你才能脱胎换骨，在技术上达到一个全新的层次，接着你再出去面试的时候，才再也不会虚，不会被人质疑你就会看书、看视频、会一些理论、没有真实的大型系统架构经验！

而我们，在 2017 年年底~2018 年年初的时候，下定决心要用自己的品牌，用 1 年多的时间打造一个真正能帮助这些中小型公司的同学积累真实大型系统经验的 Java 架构师课程！而不是仅仅教给同学们一些理论知识、HelloWorld 以及 Demo 项目！

下面就是我们的互联网 Java 架构师训练营的完整内容介绍！说明一点，这个大纲仅仅是展现出来我们对整个 Java 架构师训练营的课程体系的介绍，而不会到每个课时里面讲什么东西的细节，因为细节是我们的核心竞争力，属于保密内容，仅仅对已经报名的学员可见！

（备注：在开课之前招收第一批学员的时候，我们出过一个大纲，但是实际上我们真实开始授课之后，实际内容比最早的那个大纲内容多几倍！这次的大纲是根据我们实际开始授课之后的情况定出来的未来 1 年多的课程规划）

02、我们的架构师课程的特点

上面已经说过，我们的这个架构师课程是想要解决行业中哪些同学的哪些职业发展的问题，因此这里在介绍课程的具体内容之前，我们先给大家说下我们的架构师课程的定位和特点：

（1）坚持用 50 万行+代码的大型、真实而且极度复杂的电商系统作为贯穿一年多的项目实战：很多不明所以的同学会觉得，如果一个培训课程，可以让他做七个、八个、甚至十几个项目，那是最好不过的。然而大家想一下，你的项目经历，要的是精，而不是多。任何一个大型的系统，起码要做 1 年以上的时间，才会初步成形成规模，如果几个月时间内，我带你做七八个项目，那么不用想，每个项目都是个 Demo 罢了。做 100 个 Demo，不如踏踏实实用 1 年多的时间做一个几十万行代码的大型项目。

（2）100%完整的 Java 技术体系和架构知识：很多同学会问为，老师你讲不讲这个，老师你讲不讲那个。实际上你们根本不用问，在一个如此复杂的几十万行代码的大型系统中，Java 领域几乎所有的技术都有用武之地，难道不是吗？任何一个技术的产生，就是为了解决实际生产项目中的某个通用的问题，于是乎开源后，大家才会去用那个技术。我们之所以用

如此复杂的大型系统作为项目实战，实际上就是以为在这里会遇到大量的技术挑战，进而可以让我们完美的将 Java 领域几乎所有的技术和架构知识都运用到这套复杂的大型系统中去，所有技术都是在复杂系统中实战。

（3）绝对不干讲理论和 Demo！真实项目驱动深入细致地讲解每个技术：对于我们项目里涉及到的每个技术，我们不会跟传统的模式那样，功能 1+HelloWorld，功能 2+HelloWorld，功能 3+HelloWorld，这样子来讲。因为这样讲是没用的，我们全部都是项目驱动，每个技术，都是先细致的讲一下核心的技术和功能，接着就是直接投入大型的生产项目中去使用和开发，接着在项目中我们会发现，咦？用了这个技术之后，好像有一些技术问题啊，比如用了 MQ 以后，消息可能会丢？那么我是不是要用 MQ 的事务支持特性啊！比如用了缓存之后，针对某个复杂的业务场景，我们想要在缓存中执行某个复杂的原子操作，那么是不是可以用 Redis 的 Lua 脚本支持啊？就是这样，我们彻底摒弃掉传统的一个理论接一个理论，一个 HelloWorld 接一个 HelloWorld，最后来一个 Demo 项目串一遍的模式。我们从 0 开始，就是真实而且复杂的大型系统驱动，所有技术都是落地到项目里去解决你遇到的技术问题。这样，我们会对每个技术都进行深入而且细致的讲解，同时每个技术点，你都不是写 HelloWorld，而是在项目中实战解决项目中的问题。我们认为这样的学习效果是最好的！

（4）1000 台+服务器的线上真实生产环境演示：有同学问，老师，我跟着你写了那么多的代码，学了那么多的技术，结果，我根本不知道这套东西到底能不能支撑高并发啊？能支撑多少并发啊？需要多少台服务器啊？不用担心，我们定的规划是，未来会在线上服务器这块投入巨额资金，用阿里云的服务器，需要演示的时候按需租用，然后将我们每个阶段做好的系统部署到线上真实的生产环境上去，从几十台、到几百台、再到上千台，让你看到整套系统是如何在线上生产环境部署和抗住每秒几千几万甚至几十万的并发压力的，线上参数如何配置的。虽然这套环境只能我来演示，大家只能在视频里看到我的全过程，因为环境太过于昂贵，不可能每个人都去这种环境操作的。但是其实能看到这个过程，能拿到我发放的完整线上操作手册，其实已经很有价值了，大家心里有数，学到的一整套系统架构，如何部署的，是真的可以抗住线上高并发压力的。

（5）20+个开源项目的源码深度剖析：如果要有深厚的技术内容，那么必须阅读大量开源项目的源码，而我们的架构班课程，除了用真实的大型系统、真实的生产环境、完整的技术体系以外，同时非常注重底层内功的积累，因此会带着大家来剖析 redis、zookeeper、kafka、dubbo、spring cloud、spring boot、spring mvc、spring、spring security、mybatis、sharding-jdbc、rocketmq、等常见知名的开源技术的源码。

（6）现场手工画图+现场手写代码+大白话讲解：看过我别的课程的同学，应该知道我这个人讲课，最讨厌用 ppt，ppt 做的再精美也没用，最好的授课效果，就是上课的时候，现场一点一点画图，一边画一边用大白话讲；然后现场写代码，一点一点写，一边写一边用大白话讲。只有这样才能把复杂的技术讲明白。

（7）自己手写公司需要的基础框架：很多同学都说现在会用一些比较优秀的开源框架，比如 TKmybatis，Guns 等等。但是实际上，你要明白，如果你就是用用这些框架非常简单，而对于一个大公司而言，很多时候，这些开源框架都是无法满足我们的需求的，我们需要自己基于开源的框架基础之上，自己来研发这些框架。所以我们会带着大家自己手写完全可以投入生产环境使用的开发框架，让你公司的开发效率提升 2~3 倍。

（8）自己手写公司需要的底层中间件和分布式系统：在有些领域里，可能开源的技术是无法满足我们的，比如说，我们想要 redis 除了可以做缓存，还要可以做存储，可以基于磁盘+内存的模式实现大规模的 kv 存储系统；或者是公司里需要一个可以支持 SQL 的流式计算系统，而开源的 Storm 不支持。那么我们就得自己来研发这些中间件系统，或者是底层的基础架构层面的分布式系统，来支撑亿级流量的大规模请求调用。

（9）SaaS 云平台架构：在大公司里，经常会出现说，我们自己的技术可以很好支持自己公司的复杂业务场景，接着因为有很好的技术积累，所以我们想要将自己的技术和产品通过 SaaS 云的模式暴露出去，让别人按需付费来租用和使用。我们也会带着大家来设计和开发 SaaS 云平台架构。

（10）2000 小时课程 + 1.5 年的学习周期：我们的完整课程体系达到 2000 小时的课程，从 2018 年 3 月开始全力录制，将会一直持续到 2019 年 9 月，全部课程结束，整个课程学习周期是 1.5 年左右！而且这个过程中，课程大纲不是一成不变的，随着行业趋势发展，但凡有任何最新的技术和架构，我们都会第一时间加入课程里！而且在 2019 年 9 月课程结束之后，我们还会继续每年都将最新的技术版本，以及最新的技术、架构和项目，都更新到课程里去，对老学员全部是永久性免费升级！对于我们的学员来说，都是利用上班的业余时间来学习的，因此需要做好一个准备，非常努力的每周学习至少 20 小时+，用自己大量的晚上下班时间+周末休息时间来努力学习，做好高三时候的拼搏劲头！

03、项目阶段一：从 0 到 1！10 万行代码的电商系统 v1.0（240 小时）

这个阶段，我们面对的是 0 用户，0 并发，唯一要做的事情，就是将一个大型的项目，从 0 到 1 给启动起来，开发出来！先做出来一个版本，让所有人看一看！

核心要点：

（1）从 0 开始动手写 10 万行代码的包含 14 个中心的大型电商系统 v1.0 版本
（2）Spring Boot 核心技术在电商系统中全程开发实战
（3）包含 15 个环节的系统开发流程规范在电商系统中全程落地实践
（4）实用型的项目管理流程规范在电商系统中全程落地实践
（5）23 种设计模式在电商系统中，结合复杂的业务逻辑全程开发实战
（6）Maven 和 Git 的深入分析以及实战使用

在这个阶段中，我们的第一个重点，就是会开发出来一个大型电商系统的 v1.0 版本，包含了权限中心、商品中心、采购中心、WMS 中心、会员中心、支付中心、订单中心、物流中心、评论中心、促销中心、客服中心、财务中心、调度中心、库存中心，**共计 14 个中心，共计 10 万行代码**。

重点之二，这个阶段的技术架构，主要是基于 Spring Boot + Spring MVC + Spring + MyBatis 的架构来开发，技术架构非常简单，其实国内很多大型互联网公司的核心系统在刚开始从 0 到 1 的时候，都是这种非常简单的技术架构。因为这个时期，你要的不是技术多牛，而是开发速度必须要快。因为此时你并没有任何用户量，在互联网行业，速度是第一位的，如果你做个非常复杂的技术架构，一做做一年，一年以后系统才上线，那么黄花菜都凉了。

在这个阶段，我们的重点之三，就是在有一定复杂度的业务系统开发过程中，先站在架构师的角度设计了软件工程中的一整套系统开发流程和规范，包含了需求分析、概要设计、详细设计、工程初始化、Git 工作流、编码开发、单元测试、冒烟测试、静态代码扫描、Code Review、集成测试、系统测试、验收测试、系统上线、线上运维，接着在电商系统 v1.0 的开发过程中全程落地实践。

我们的重点之四，就是站在架构师的角度为公司制定了一整套的项目管理流程和规范，接着在电商系统 v1.0 的开发过程中全程落地实践。

我们的重点之五，就是将 23 种设计模式全部放在真实而且复杂的电商业务逻辑中全程实践，只有在复杂业务逻辑中才可以掌握设计模式到底是怎么用的。

在这个阶段，实际上作为架构师，就是要用最简单的技术架构用最快的速度开发出来一套系统，同时保证所有人都是按照标准的开发流程规范以及项目管理流程规范来走，为后续团队快速扩张以及规范化打好基础，同时严格在详细设计环节引入设计模式来设计你的代码，确保编写出来的代码都是高质量的。

下面给大家看一下我们项目阶段一的课程目录截图：

136_完成角色管理模块的代码编写
137_完成评论管理模块的部分功能开发
138_完成类目管理模块的代码编写（一）
139_完成类目管理模块的代码编写（二）
140_完成查看购物车功能的单元测试和冒烟测试
141_完成库存管理模块的单元测试和冒烟测试（一）
142_完成库存管理模块的单元测试和冒烟测试（二）
143_完成库存管理模块的单元测试和冒烟测试（三）
144_完成角色管理模块的单元测试和冒烟测试
145_完成评论管理模块剩余功能的开发
146_开发类目管理模块以及对原型模式完善深度克隆功能
147_完成操作购物车功能的开发以及测试
148_完成调度中心的调度采购入库功能的开发
149_完成账号管理模块的代码编写
150_完成评论管理模块的单元测试和冒烟测试
151_完成编辑类目功能的代码编写
152_基于优化后的组合模式与访问者模式完成删除类目功能
153_完成类目管理模块的单元测试和冒烟测试（一）
154_完成类目管理模块的单元测试和冒烟测试（二）

234_完成物流中心的运费模板管理模块的测试
235_完成财务中心的采购结算单管理模块的开发
236_完成wms中心的销售出库单管理模块50%功能的开发
237_完成订单中心剩余功能的开发（二）
238_完成采购中心的采购单管理模块50%功能的开发
239_基于策略模式完成物流中心的运费计算模块的开发
240_完成财务中心的供应商结算模块的开发
241_完成wms中心的销售出库单管理模块剩余50%功能的开发
242_完成订单中心剩余功能的开发（三）
243_结合模板方法模式对订单中心的状态模式进行优化
244_完成采购中心的采购单管理模块剩余50%功能的开发
245_完成物流中心的运费计算模块的测试
246_完成财务中心的采购结算单管理模块的测试
247_完成wms中心的销售出库单管理模块的测试
248_完成订单中心最后几个接口的开发
249_完成采购中心的采购单管理模块的测试
250_基于解释器模式与组合模式重构规则解析代码
251_基于装饰模式重构订单状态管理器和订单操作日志

04、项目阶段二：微服务架构+100 台机器支撑业务快速迭代（240 小时）

这个阶段，我们要面对的是比如 500 万的用户量，如果网站搞个活动，比如秒杀啥的，最高也许会有个几千 QPS，我们的系统架构要能够支撑住！

这个阶段处于系统从 1 到 10 的阶段，团队会快速从几个人扩张到二三十人，你作为一个技术团队的 leader，需要能够带着这个中小型的团队，应付多个 PM 铺天盖地而来的需求。因为此时系统的 v1.0 版本已经出来了，经过初步的验证之后，接下来的目标就是两个：

第一，在几个月内快速将一个完整而且复杂的业务系统给开发出来，会快速迭代出来 v1.1、v1.2、v1.18 等等 N 个版本，可能你要做到每隔几天就发一个版本，甚至多个版本并行开发、

并行测试、依次上线；

第二，能够让快速扩张中的几十人的技术团队协调配合起来，高效率的开发一个复杂的业务系统

所以针对这个阶段的目标，你作为一个技术 leader，此时应该做的第一件事情，就是先将系统架构整体改造为微服务的架构，为业务的快速迭代打下坚实的架构基础。

（1）领域驱动设计：你面对一个复杂的单块业务系统，该如何将这套系统改造为微服务的架构？第一步，你就要解决如何进行服务建模的问题，你需要梳理清楚复杂的业务逻辑关系，同时设计好你的微服务架构中到底要包含哪些服务？拆的太粗了不行，拆的太细了也不行，我们会带着大家学习如何用领域驱动模型设计的思想，面对项目阶段一开发好的 10 万行代码的电商系统 v1.0，来学习如何进行服务建模。

（2）Spring Coud：在这个阶段，我们会将 Spring Cloud 作为微服务的基础技术架构，因此在这里我们会快速学习 Spring Cloud 各个组件的核心技术，接着立刻就将 Spring Cloud 应用到项目中去。既然已经设计好了一个单块应用要拆分为哪些服务，下一步我们就是要基于一整套的微服务技术来进行单块系统的拆分，将 10 万行代码的电商系统用 Spring Cloud 技术栈来彻底重构，重构成微服务的架构。

（3）敏捷开发：用 Spring Cloud 作为微服务基础技术架构之后，其实团队里每个人都会负责一个或者几个小服务，此时每个服务都完全自治，都可以独立的开发、测试和上线，灵活性大大增强，那么接着要做的事情，就是为了让团队基于微服务架构，更加快速的支持接踵而来的业务需求。会从传统的系统开发模式，转变为 Scrum 模式的敏捷开发流程与规范！

（4）敏捷项目管理：当你的开发模式从传统的瀑布式开发，转变了敏捷开发模式之后，整个项目管理也需要从传统的模式转变为敏捷项目管理，让微服务架构+敏捷开发+敏捷项目管理，搭配起来，将你的团队开发效率提升到极致，最大限度的用有限的人力资源，快速的支持大量需求的迭代！

（5）Docker + Kurbernetes（k8s）：当你基于微服务架构+敏捷开发模式+敏捷项目管理，快速搞定一个版本之后，如何将大量的微服务在线上进行部署以及管理？目前业内最流行最好的办法，就是基于 Docker + kurbernetes 来部署、运维和管理大量的微服务！

（6）DevOps+持续集成+持续交付：好，现在咱们知道如何使用微服务架构开发业务系统，如何用敏捷模式把控一个微服务架构项目的开发，同时也知道后续会部署在 Docker + kurbernetes 上，但是有个问题，如果要将开发效率发挥到极致，难道还是按照以前的传统模式来测试、集成、部署？当然不是！我们需要使用 DevOps 思想，基于 Jenkins 引入持续集成、自动化测试、自动化部署、灰度发布等各环节，无情自动化！

（7）微服务运维支撑平台：好，那你们我们现在已经将微服务架构的一套电商系统改造好，开发好了，多达数十个服务，然后也基于持续集成+自动化测试+自动化部署的思想，在 Docker + kurbernetes 部署好了，也可以方便的进行服务的运维和管理了。接着问题来了，如此庞大规模的服务在线上运行，你怎么进行监控？你怎么进行报警？对于数十个服务你如何快速定

位异常日志？怎么对线上的各种异常情况进行排查和定位？我们会基于**Open Falcon 来进行分布式系统的监控和报警**，同时基于 **ELK 来采集分布式系统的日志**，提供统一日志中心来方便我们排查和定位各种异常。

（8）微服务治理平台：好，现在我们一套复杂的微服务系统在线上运行着，那么接下来大量的问题来了，我想看看复杂微服务系统的调用链路？我想自动统计每个服务的 QPS 和可用性？我想对服务进行强制分层避免循环依赖？我想对调用链路级别进行监控和报警？我想有一套系统调用链路异常之后的自动化排查机制？我想进行服务调用的鉴权限制谁来调用我？我想有一套敏感信息加密的配置系统？这些问题，都是微服务架构中最常见的一些问题，目前几乎没有哪个开源的项目可以解决上述问题，在大公司里，我们通常选择自己研发，因此在这个环节，我们会自研一套大型完全可以投入生产环境的微服务治理平台！

（9）分布式架构：各位同学，应该注意到一点，微服务后的系统架构，实际上已经从一个单块应用演变为一个分布式系统了，一个完整的大系统，实际上是由大量的小服务组成的子系统，多个子系统组成的一个大系统。任何一个用户请求，都要从上游开始，历经 N 个服务的调用，才能完成一个请求的处理，这不就是典型的分布式系统么？多个系统协作来完成一个功能。所以我们的系统一定会遇到大量的分布式技术问题：**分布式事务、分布式协调、分布式锁、分布式会话（Spring Session）、单点登录（CAS）、分布式调度（Elastic-Job）**，等等。在这里，我们会带着大家在复杂的业务系统中来全程实践所有的分布式技术问题。

（10）ZooKeeper：深入讲解 zk 的核心技术以及高阶技术，同时深入剖析 zk 底层的机制，原理以及算法。针对上面的各种分布式技术问题，我们可能会大量的用到 ZooKeeper 这块技术。

（11）100 台+服务器的真实线上生产环境的演示：这个阶段，我们计划采用 100 台+的服务器来演示整套微服务系统如何在线上生产环境来部署，我们会通过压测来给大家看到，这套架构支撑每秒几千的访问量是完全没问题的。

总结一下，上面的所有微服务架构后引出来的东西：领域驱动设计、Spring Cloud、敏捷开发、敏捷项目管理、Docker + kurbernetes、DevOps+持续集成+持续交付、微服务运维支撑平台、微服务治理平台、分布式技术问题，每个环节，我们都会对涉及到的技术进行细致深入的讲解，同时最重要的是，我们会全程对所有东西基于项目阶段一的那套 10 万行代码的电商系统来实战。记住一点，干讲 API，干讲 HelloWorld，干讲 Demo，很简单。但是一旦在一套业务复杂的系统中来做和踩坑，那么对技术的磨炼、掌握以及实战经验的积累，是成几倍的提升的，也这有这样，大家才能学到真功夫，硬本领。

在这个阶段，我们课程的总体代码量将会达到 15 万行+。

05、项目阶段三：开源框架源码剖析+开发公司基础框架（180 小时）

这个阶段，我们的目标是，支撑一个 20 人的团队，每个月完成多个小版本的快速迭代！

在这个阶段，我们实际上还是处于系统从 1 到 10 的阶段，因为我们刚刚投入人力，耗费时间完成了整套微服务技术架构的改造和升级，这套架构足以支撑几十个人对一个复杂的业务系统，分而治之，快速开发快速迭代。

然而此时限制我们开发效率的一个问题来了，如果我们还是要大量手写各种基础的 CRUD 业务代码，如果我们还是要在各个服务里自己实现一些通用的功能模块，那么还是会导致我们的团队无法聚焦于业务，会分散精力，降低我们的效率。

因此在项目阶段三，我们会投入精力分析常见的流行开源框架的源码，彻底剖析这些开源框架的设计思想以及其中的开发技巧，在精读了多个开源框架的源码之后，我们要尝试基于这些开源框架，来封装一套适合我们自己公司的基础研发框架出来！目标只有一个：无情的自动化，让框架自动完成大量的基础 CRUD 工作和基础模块功能，将业务开发的效率提升 2~3 倍！

（1）常见开源框架的源码剖析：我们会深入剖析 Spring Cloud、Spring Boot、Spring Web MVC、Spring、Spring Security、MyBatis、Activiti、Lucene 等这些流行的开源框架的源码

（2）基于开源框架来研发公司的基础框架：我们会综合基于 Spring Cloud、Spring Boot、Spring Web MVC、Sping、Spring Security、MyBatis、Activiti、Lucene 这些开源框架，来自己研发出一套转为公司内部使用的基础框架，包含：权限管理、单点登录、授权认证、用户管理、通用存储层/业务层/控制层、配置管理、日志管理、监控管理、服务治理、工作流管理、全文检索，等等。

如果不为公司写一套统一的开发框架，那么不同的服务，不同的项目，都会一次又一次的对一些基础性的功能，自己进行封装，导致相同的功能，不同的服务中的代码都是不一样的，增加了开发成本。同时对于大量的 CRUD 功能，我们会采用约定大于配置的思想，通过基础框架，让 RD 们加几个注解，所有 CRUD 代码都是在系统运行的时候自己动态生成。通过这套基础开发框架，可以让我们的项目开发效率提升至少 2~3 倍！

在这个阶段，我们的课程总体代码量将会达到 20 万行+。

06、项目阶段四：30 万行代码的大型分布式电商系统（120 小时）

这个阶段的目标，就是利用我们的微服务架构和基础框架，在几个月内，快速追评竞争对手，完善我们的电商平台的几乎所有核心功能！

好，各位同学，目前我们还是处于系统 1 到 10 的阶段，但是我们作为公司的核心技术团队，已经完成了电商系统 v1.0 的微服务架构改造，同时为公司研发了一款统一的开发框架，基于微服务架构+统一开发框架，我们的业务迭代的整体效率将会提升至少三倍！就是说，原本既有项目阶段一的那套技术架构、开发流程以及项目管理流程，一个几十人的技术团队搞定 N 个业务需求可能需要 3 个月，但是现在只要 1 个月就可以搞定！

万事俱备，只欠东风！

在这个阶段，我们将会基于一套优秀的技术架构和基础框架，快速迭代至少 20 个版本，将电商系统 v1.0 一直迭代到 v1.20，深入复杂的电商业务需求中，将这套分布式的大型电商系统的复杂度提升 10 倍！

（1）权限中心：权限中心在这个阶段将会增加大量的复杂功能逻辑，包括引入多层级部门支持、引入多层级角色支持、在各个业务模块中引入上下级关系、引入操作日志、敏感操作监控，等等，业务上将会复杂好几倍。

（2）商品中心：商品中心在这个阶段会增加大量的复杂功能，引入前台类目和后台类目的分离、引入更多的商品类型（比如虚拟商品）、引入非常复杂的商品搜索功能、引入非常复杂的品类筛选功能、引入复杂的商品审批流程，等等，业务上会复杂好几倍。商品中心是电商系统中的重点，因此我们这块会将业务功能迭代到跟大型 B2C 电商平台一样的标准。

（3）采购中心：采购中心在这个阶段会引入采购退货支持、供应商绩效考评、采购计划管理、采购竞价、采购询价、合同发票管理，等等，采购中心是个重要的模块，因为大家做点电商系统，必须要知道商品是从哪儿来的？这点很重要，这个直接是跟库存那块关联的，所以这块的业务很重要，业务会复杂好几倍。

（4）WMS 中心：也就是仓储中心，这块会引入全国异地多仓库支持、将商品入库流程全面信息化（到货确认、验货打码、商品上架）、将商品出库流程全面信息化（打单、拣货、复核、包装、交接发货）、引入库存调拨功能、引入库存盘点功能、引入库存统计功能。WMS 中心实际上是电商系统的重点，涉及到了物流这块，因为所有的商品都需要采购之后入库，销售之后出库，所以电商业务逻辑必须跟仓储打交道。这一块的业务起码复杂 10 倍。

（5）会员中心：这块会参照大型 B2C 电商平台的标准，引入更多的会员相关功能，包括会员充值、会员积分消费或兑换、会员等级特权、付费 VIP 会员特权、更多的会员积分规则、更多的会员成长规则、邀请新会员返利，业务会复杂好几倍。

（6）支付中心：这是本次升级绝对重点中的重点，支付中心，是电商系统里至关重要的命门级模块，所以我们这次可能会将支付中心的业务逻辑升级到复杂 20 倍！包括了支付清算、账务会计、核算中心、运营管理、核心管控、收银台、交易中心、收费管理、安全管理、工作台、银行网关、渠道管理、资金管理、支付 BI，等等。

（7）订单中心：订单中心同样是电商系统中的核心之中的核心，本次订单系统的业务逻辑将会复杂至少 10 倍！我们将会引入父子订单功能、订单拆单功能、优惠分摊功能、换货功能、线下服务订单、订单统计分析、订单处理引入工作流，等等。

（8）物流中心：电商其实就是 3 个流，资金流、信息流和物流。而用户浏览商品，靠信息流，购买商品走资金流，商品发货之后走物流。所以物流中心至关重要，关系到了电商系统的商品是如何发到用户手中的。这里会引入同城配送、门店自提、更复杂的运费模板、对接多个第三方物流商的支持，等等。

（9）评论中心：在评论中心里，我们也会引入更多的功能，包括更多的评论标签、更复杂的评论奖励政策、追评功能支持、支持晒单视频上传、自动识别恶意刷好评、自动识别恶意刷差评，等等。

（10）促销中心：促销中心是电商中非常重要的一个模块，支持更多的促销活动类型、支持拼团、预售、秒杀等促销活动、支持更多类型的优惠券、支持更加复杂促销使用规则，这块我们会完全参照大型 B2C 电商平台支持的促销活动来做。

（11）客服中心：客服中心是处理售前售后的非常重要的一个环节，在售前，客服中心要支持用户的售前咨询，在售后客服中心要处理用户的投诉、退货、换货等操作。我们在这里会基于 WebSocket 技术实现一套网页版的客服系统，同时会完善售后客服工单支持，加入投诉工单、建议工单、咨询工单、换货工单，等等。

（12）财务中心：财务中心是电商系统中同样一个非常重要的地方，因为资金流会走支付系统出去和进来，但是财务中心是对整个电商系统的资金流进行盘点的地方。我们在这里将会引入：交易流水、统计报表、财务对账、发票核销、会计报表、退货报表，等等功能。

（13）调度中心：调度中心是电商运营管理系统和仓储管理系统交互的核心点，主要是负责对全国各地多个仓库的商品进行调度入库和调度出库，非常核心。这里我们将会引入自动调度入库分配功能、多仓库调度支持、更复杂的调度出库算法，等等功能。

（14）库存中心：库存中心同样是电商中的核心，我们会在这里引入自动库存监控、虚拟库存、活动库存、预售库存、自动定期补货、自动定量补货、库存统计报表，等等功能，业务上同样会复杂好几倍。

（15）CMS 中心：这块将会是我们新增加的一块中心，简单来说，就是通过 CMS，内容管理的思想，对电商网站中的首页图片和文本、商品、商品排行榜、导航栏等部位进行动态内容管理。

（16）移动电商：这块我们会通过 PC 端的商城系统，构建一整套的支持移动电商 App 的后台接口和系统，也是我们新增加的一块内容，这样我们的商城系统也是可以支持移动端 App 的请求的。

看着内容很多是不是？但是其实基于我们之前架构好的一整套微服务技术体系+基础框架，可以专注于业务逻辑开发，各个中心和服务同时并行快速迭代，充分发挥几十人技术团队的协作优势来！最重要的是，通过个阶段的业务迭代开发，我们手上将会拥有一套业务功能完备，打通商品采购、仓储管理、商品展示、用户下单购买、商品出库、售后退货换货的全流程，绝对真实复杂而且可以投入生产环境使用的电商系统，不是那种几千行代码的所谓电商 Demo 项目可以比拟的！

本阶段的代码量将会达到 30 万行+，有了这套复杂的电商系统，我们再继续进入后续的高并发、高可用等架构的环节，你将有机会基于一套 30 万行代码的复杂电商系统来实战所有的技术和架构，这个里面对每个人的磨炼和成长，绝对不是说写 HelloWorld、干学理论可以比拟的！

07、项目阶段五：亿级流量+高并发+可伸缩+海量数据+1000 台机器的系统架构（300 小时）

这个阶段，我们要面对的是上亿的用户，每天亿级的流量，随便搞个促销活动也许 QPS 会每秒上万！如果是双 11 的瞬间抢购狂潮，也许 QPS 会达到上十万！而且我们的数据库里可能每天单表都要新增几百万的数据量，单表也许要达到几亿的数据量级！我们的系统架构要能支撑住！

到了这个阶段开始，我们就要开始经历系统从 10 到 100 的阶段了，系统从 1 到 10 的时候，大概就几万~几十万个用户，属于快速迭代业务，小步快跑，小规模试点，用户量不大，就算搞活动，撑死最高并发几千，根本用不上什么高大上的技术。在系统从 1 到 10 的阶段，最主要的是开发效率，快速迭代业务，把业务都做出来。

但是在从 10 到 100 的阶段，那就不是这么回事儿了，这个时期，产品开始大推，用户数量快速上涨，可能快速的就突破了几千万用户，甚至是上亿用户，每秒并发量，很可能会在几万/s 到几十万/s，都可能。所以这个阶段，我们就是要引入大量的技术，将系统架构升级的非常复杂，构建出一整套的高并发系统架构来！

在这个阶段，我们会深入讲解但是不限于以下技术：

（1）Redis：核心的缓存技术， 必须深入讲解！我们在 redis 这块的讲解，会非常的深入，包括从零开始讲解， redis 各种高阶用法，高阶知识，以及集群，高可用，等等，同时会讲

解如何进行企业级的集群部署、监控以及管理。

（2）MongoDB：最经典的 NoSQL 数据库， redis 定位主要是缓存，而 MongoDB 主要是 NoSQL 数据库！功能更加强大，非常适合用于一些高并发，高性能，数据要持久化，而且功能要求比较复杂的场景。这块会深入讲解 MongoDB 的核心以及高阶技术，包括生产环境的部署架构，运维和管理方案。

（3）RabbitMQ：ActiveMQ 曾经作为作为 java 领域最常用的 MQ 中间件，但是现在用的公司已经很少了。目前国内互联网公司用的最多的是 RabbitMQ，这次一定是重点讲解的，深入讲解 RabbitMQ 的各种核心以及高阶技术点，包括生产集群的运维部署方案。

（4）基于 MySQL+Sharding-JDBC 的 分库分表与读写分离：一旦访问量变大，势必要对数据库进行水平拆分，垂直拆分，以及分表，支撑更高的并发访问量，更大的数据量，更强的访问性能。此处会深入讲解各种分库分表的技术方案，并且应用到系统架构中去，同时会深入讲解 MySQL 和 Sharding-JDBC 相关的技术细节。

（5）Nginx：绝对深入系统的讲解 nginx 这块的知识，nginx 是非常重要的一块知识点，Java 架构师必备，包含了 Nginx 初中高级所有知识点。 用 nginx 加入系统架构中，完全可以做到高并发访问的支撑。

（6）Tomcat：tomcat 作为 java 架构师来说，必须深入的掌握和理解，因此会深入讲解 tomcat 的各种中高阶技术，同时深入剖析 tomcat 的内核，一直到底层， 让大家掌握强悍的技术内功。 Tomcat 的优化，同样可以保证系统架构的高并发访问能力。

（7）LVS+KeepAlived：高可用的负载均衡层，将高并发的请求往后端多台 nginx 服务器进行负载均衡。这块的讲解，不是市面上那种简单部署和搭建，会深入讲解 LVS+KeepAlived 的各种核心以及高阶技术，做到全网独家。 这块讲解之后，就会将 LVS+KeepAlived 应用到系统架构中去，同样支撑高并发的访问。

（8）Elasticsearch：用最精炼的语言深入讲解 es 分布式搜索引擎技术，同时采用 es 来构建电商系统的搜索功能，基于 es 分布式的特点，构建高性能，高并发的搜索模块。

（9）FastDFS：Java 领域最常用的分布式文件存储系统，不同于市面上的简单教你 FastDFS 的搭建和部署，我会深入讲解 FastDFS 的核心以及高阶技术点。这块讲解之后，会将 FastDFS 分布式文件系统应用到我们的系统架构里去。

（10）大型网站系统架构：页面静态化架构、动静分离架构、图片服务器分离架构，讲解一个大型电商网站中的页面静态化架构，如何一步一步来设计，以及如何一步一步达到最佳的静态化架构，绝对不是简单的 freemarker 模板静态化那么简单的讲解。 此外，还会讲解如何采用 nginx 负责静的部分， tomcat 负责动的部分，动静分离开来的整套架构。 这个架构讲完之后，就会立即改造电商系统架构。采用页面静态化+动静分离架构之后，系统的性能和高并发能力会大大增强。同时随着访问量变大，必须将图片单独独立出来，放在独立的服务器上，因为图片是非常耗费资源的，如果和其他服务器混合在一起，访问性能不会达到最佳状态。这块讲完之后，也会应用到我们的系统架构里去。

备注一下，上面只不过是列举出了一些技术而已，但是实际我们到时候要讲的技术还不止上面这些！

在上面的那些技术本身学习完之后，我们会基于这些基础的技术，将整个大型电商系统逐步迭代成能够支撑上亿注册用户、每天亿级流量访问、每秒 QPS 达到几万甚至上十万的高并发电商系统架构！

（1）支撑亿级流量+高并发+海量数据的搜索引擎系统：电商网站的流量，除了首页以外，一般就是搜索的流量是最大，而且需要搜索的数据量可能会非常大，有几百万到几千万的数据量级！此时发展到了上亿流量的网站之后，就需要开发一套能支撑超高并发高性能的电商搜索引擎系统，此时就需要综合使用 Elasticsearch、 Hadoop、 Flume、 Kafka、大型搜索引擎的架构设计，包括搜索引擎算法的优化，等等，来升级一套亿级流量的电商搜索引擎系统。

（2）支撑亿级流量+高并发+海量数据的商品详情页系统：电商网站最核心的系统之一就是商品详情页系统，因为商品详情页会在搜索之后，承载大部分的读流量，因此需要站在亿级访问量的基础之上，开发一套亿级流量的商品详情页系统架构。在这里我们结合极度复杂的业务场景，深度实践一整套的复杂缓存架构！而且这里我们的商品数据量会非常大，可能是至少几百万甚至是几千万的量级！

（3）支撑亿级流量+高并发+海量数据的订单系统：订单系统同样是非常的核心， 商品详情页是高并发读系统，而订单系统就是高并发写系统，在这里会针对亿级的流量，开发一套高并发的亿级流量电商订单系统架构，架构非常之复杂。在这里我们结合极度复杂的业务场景，深度实践一整套的复杂消息队列架构！而且虽然用户量飙升，可能每天都是几十万个订单，甚至是上百万个订单，都有可能！

（4）支撑亿级流量+高并发+海量数据的购物车系统：实际上在用户量上来之后，购物车这块的压力也是非常大的，因为现在大多电商都是走的离线购物车，也就是说，不是简单的将每个用户的购物车数据放在 session 里，而是离线存储的。那么在大量用户高并发大流量的请求下，购物车系统也需要设计与演进为高并发的系统架构，而这个架构，也是非常复杂的。同时随着大量的用户进来，可能购物车这块的数据量级也会达到亿级别！

（5）支撑亿级流量+高并发+海量数据的支付系统：支付系统也是非常核心的系统之一，而且架构上非常的复杂，我们同样会开发一套亿级流量的高并发支付交易系统，支付系统本身就是非常复杂的，再加上需要让这样复杂的系统支撑亿级流量高并发请求，这块的技术挑战可以说是相当的大。在这个用户量级下，每天都可能发生数十万笔的支付交易流水，数据量很快就会破亿级别！

（6）支撑亿级流量+高并发+海量数据的会员系统：会员系统是电商中非常容易面对高并发的系统，因为会员相关的数据在电商的各个环节都会被大量的访问和取用，所以在这里，我们会将整个会员系统升级到高并发的系统架构。我们的定位是上亿用户，那么我们的会员系统中的数据，必然会有亿级别的数据量！

（7）支撑亿级流量+高并发+海量数据的秒杀系统：在双 11 等大促，秒杀的场景下，都需要一套独特的系统架构来支撑， 这就是秒杀系统！我们会开发一套超高并发的秒杀系统架构来，这套架构，同样会非常的复杂。。

（8）支撑亿级流量+高并发+海量数据的评论系统：如果是一个用户量上亿的大型电商网站，日常发表评论以及查看评论，都是非常大的量，那么评论系统如何支撑高并发的请求呢？我们同样要将这个系统升级到高并发的系统架构！大家想，这个量级下的电商系统，每天同样可能有数十万条商品评论进库，很快就会到亿级的数据量！

（9）支撑亿级流量+高并发+海量数据的库存系统：库存系统，是一个电商系统中的核心，因为如果库存错误，或者有问题，会导致整个电商流程无法运转下去！所以我们这里会将库存系统升级为非常复杂的可以抗高并发的系统架构！在商品数据可能都数千万的场景下，每个商品都有关联的库存数据，库存数据量很容易就上亿级的规模！

（10）支撑亿级流量+高并发+海量数据的移动电商 App 后台系统：现在相当大比例的用户都是走移动 App 端浏览和购物的，那么移动电商 App 的后台系统架构，该如何升级为复杂的高并发系统架构呢？这里我们会做出一套非常复杂的高并发移动电商 App 系统架构！

（11）支撑亿级流量+高并发+海量数据的电商峰值系统：为了应对什么双 11、618、双 12 等各种购物节，对于这种活动时候的高峰流量，如何开发一套统一的电商峰值系统来应对呢？我们在这里会开发一个大型的电商峰值系统！

（12）可伸缩的系统架构：总结整套系统如何通过分布式的方式，来确保任何一个环节都支持伸缩，进而在需要的时候快速扩容来支撑更高的并发量，因为如果高并发系统架构做好之后，理论上，其实支撑几万 QPS，几十万 QPS， 都是一个架构，但是只不过在需要的时候需要尽可能的扩容！

（13）基于线上真实的 1000 台+服务器的生产环境演示：在我们基于各种技术和复杂的方案，做出来一整套的复杂系统架构之后，我们会基于 1000 台+的真实服务器，来演示完整的线上生产环境的部署、优化以及如何抗住每秒几万甚至上十万的 QPS！

这边可以着重说明一下，为什么我们要在之前花费大量精力做出一个如此复杂的 30 万行+代码量的电商系统？就是为了能够在这个环节，基于真实、大型而且复杂的系统，极度复杂的业务，将各种技术融入进去，加上针对复杂场景设计的各种复杂技术方案，真正让大家学习到如何把控一个复杂系统的高并发架构！这种方式学习到的技术、架构以及经验，绝对不是写一堆 Demo 和 HelloWorld 可以比拟的，学习的效果至少提升 10 倍！

在这个阶段，项目整体代码量将会达到 35 万行+！

08、项目阶段六：高性能系统架构（120 小时）

这个阶段的目标，我们是要能够针对亿级流量的大规模请求，让系统的各个环节的性能保持稳定和高效，让我们的系统整体响应延时控制在 200ms 以内！

在我们彻底实现亿级流量高并发系统架构的同时，实际上我们也同样会面临大量的性能问题，因为你的整套系统的各个环节，如果有任何一个地方性能太差，那么都会导致系统的整体吞吐量下来，影响用户体验。

所以在这个阶段，我们将从前到后对系统进行各个环节的性能优化，实现以一整套的高性能系统架构！

这里面包括但是不限于以下内容：

（1）架构性能优化：在架构层面，总结如何采用缓存+异步+集群来进行性能的优化

（2）JVM 性能优化：深入讲解 java 虚拟机的内部原理，同时深入讲解 jvm 如何进行性能优化，如何监控线上 jvm 的性能情况，如何用各种工具定位 jvm 的性能问题，如何对各种场景下的问题，对 jvm 进行性能优化以及故障的解决。

（3）MySQL 性能优化：深入讲解 MySQL 数据库的 InnoDB 存储引擎内部原理，同时对 MySQL 的高阶的技术，包括事务，锁， 等技术进行深入讲解，同时深入讲解如何对 MySQL 进行性能优化，包括 SQL 的调优

（4）整个系统各个环节的性能优化：RPC 接口调用性能优化、Java 代码层面的性能优化、并发编程性能优化、IO 性能优化、集合使用性能优化、连接池性能优化，等等

这个阶段，我们将会基于我们开发好的一整套几十万行代码的复杂系统，对所有的性能问题，全部模拟演示线上的性能问题，接着针对复杂场景下的各种性能问题，手把手带着大家实操演练，去解决我们的复杂业务系统中存在的各类性能问题！

09、项目阶段七：高可用+安全性+稳定性的系统架构（120 小时）

这个阶段，我们的目标，就是要让我们的系统无论数据库挂了，还是 MQ 挂了，还是缓存挂了，任何一个地方挂了，都可以保持 99.99%的可用性！

业务已经发展到了一定的阶段，用户数上亿，并发这块是撑住了。但是，系统中总是有各种各样的故障，这么多的用户，系统的可用性非常重要，一旦系统崩溃，那么系统就不可用了。比如 MQ 挂了怎么办？某个服务挂了怎么办？数据库挂了怎么办？缓存挂了怎么办？啥啥挂了怎么办？

高可用架构的目标只有一个：一套大型复杂的系统中，任何一个地方挂了，都不能导致系统出现致命性的全盘崩溃！ 如何让系统保持高可用呢？这就是要将整套电商系统架构升级为高可用的架构了。

（1）高可用工程流程：总结已经做好的微服务架构下的一整套自动化测试以及全流程测试，包括自动化部署，灰度发布对于高可用的意义。

（2）高可用隔离架构：涵盖包括线程隔离、进程隔离、集群隔离、机房隔离、 热点隔离、资源隔离，等等各种复杂的高可用隔离架构方案，并且应用到我们的系统架构中去。

（3）高可用限流架构：讲解限流算法、限流机制、分布式限流、接入层限流等各种复杂的高可用限流技术方案，并且应用到我们的系统架构中去。

（3）高可用降级架构：讲解自动降级、人工降级、读降级、写降级、多级降级等各种复杂的高可用讲解技术方案，并且应用到我们的系统架构中去。

（4）超时与重试+回滚+压测：讲解大公司中如何设计超时与重试机制，如何设计回滚方案，如何设计压测方案，来确保系统的高可用性。

（5）硬件高可用+软件高可用+DNS 防劫持+CDN 高可用：讲解这套方案如何应用到电商系统的高可用性架构中去。

（6）接入层+业务层+数据层的高可用方案：讲解这套方案如何应用到电商系统的高可用性架构中去。

（7）稳定性架构：同理，既然高可用保障了以后，那么稳定性如何保障？系统可能到处都有 bug，或者是网络不稳定，数据库性能不稳定，那该怎么办？如何快速的识别出来稳定性问题，然后迅速自动化的排查和解决？进而保持系统的稳定性？我们这里将会实现一整套的保证系统稳定运行的架构！

（8）安全性架构：那么如果保证了系统的高可用性、高稳定性之后，另外一个非常重要的一点，就是保证系统的安全性！比如常见的 XSS 攻击、注入攻击、CSRF 攻击、防火墙、安全漏洞扫描、加密解密技术，等等，这里不一一列举了。我们将会实现一整套的保证系统安全的架构！

这边可以着重说明一下，为什么我们要在之前花费大量精力做出一个如此复杂的 30 万行+代码量的电商系统？就是为了能够在这个环节，基于真实、大型而且复杂的系统，极度复杂的业务，将各种技术融入进去，加上针对复杂场景设计的各种复杂技术方案，真正让大家学习到如何把控一个复杂系统的高可用架构！这种方式学习到的技术、架构以及经验，

绝对不是写一堆 Demo 和 HelloWorld 可以比拟的，学习的效果至少提升 10 倍！

同时，我们会带着大家在线上真实的生产环境做复杂的容灾演练，对各种线上事故都模拟后演练系统是如何 cover 住所有的线上生产事故的！

在这个阶段，项目整体代码量将会达到 40 万行+！

同时备注一下，上面的所有高可用相关技术方案，都是结合复杂电商业务系统来做的，所以不要看一个方案的名字很简单，其实做起来相当之复杂！任何技术或者方案，用 Demo 来演示都很简单，但是放一个复杂业务系统里，那这个方案可能会变得相当之复杂！而且我们上面只不过是列举了部分高可用相关技术而已，更多的方案没写在这里！

10、项目阶段八：自己研发中间件以及分布式系统（240 小时）

这个阶段，就是要针对公司里的某些业务场景，没有合适的开源项目可以用的时候，我们自己研发一些底层的分布式中间件或者分布式系统，来支撑亿级流量的大规模请求调用！

好了，各位同学，如果你能够经历上面如此之多的技术挑战，一路披荆斩棘走到这里，那么说明你的技术实力已经有非常强悍的积累了！下一步，就是挑战架构师最底层的技术功底，就是自己来研发底层的中间件系统以及分布式系统！

为了能够研发底层的中间件系统和分布式系统，你首先需要积累大量的底层基础技术，包括但不限于以下的内容：并发编程、NIO、Netty、网络通信、CPU、磁盘、内存、操作系统、数据结构和算法、存储、序列化，等等。

同时还会对分布式系统的底层知识进行深入的学习：包括分布式系统的异步通信、心跳感知、主备切换、流量和数据调度、服务调度、重试机制，等等。

接着我们将会手把手带着你来自己动手开发以下中间件系统和分布式系统：

（1）类似 Dubbo 的分布式 RPC 框架
（2）类似 Storm 的分布式流式计算系统，但是我们要支持流式 SQL 支持和流 join 等复杂操作
（3）类似 Hadoop（HDFS）的分布式存储系统
（4）类似 Spark 的分布式计算系统
（5）类似 HBase 的高性能高并发的分布式 NoSQL 数据库

当然你要明白一点，上面那些分布式系统，都很复杂，我们不可能做的每个都跟开源的一样，但是起码我们可以把每个中间件和分布式系统的核心架构都做出来，仅仅是做到这一点，你的底层中间件研发，和分布式系统研发的经验和能力就很强了！

这儿给大家稍微说点别的，有很多人说，老师，我学了 Netty，我学了并发编程，但是感觉平时不怎么用啊！难道就是面试的时候用用？当然不是了！Netty、并发这些东西，其实都是在研发底层的中间件和分布式系统的时候才用的!所以我们正好在这里完美的实践所有的技术！

这个阶段结束之后，我们的课程整体代码量起码达到 50 万行+

11、项目阶段九：中间件以及分布式系统的源码剖析（180 小时）

这个阶段，我们就是要能对常见的分布式 RPC 框架、分布式协调中间件、分布式缓存、分布式消息队列、分布式搜索的底层原理都非常深刻的理解！积累扎实的技术功底！

在整个亿级流量高并发系统架构，我们都开发好，以及部署上线之后，大家应该对各种底层的中间件技术，都有了非常深刻的理解了！接着我们就会在实战之后，来兼顾深度，我们会深入剖析各种中间件系统的源码！

（1）Dubbo 源码剖析
（2）ZooKeeper 源码剖析
（3）Redis 内核剖析
（4）RocketMQ 源码剖析：虽然我们课程里用了 RabbitMQ，但是 RabbitMQ 是 erlang 写的，不适合我们阅读源码，所以我们会对业内非常优秀的 RabbitMQ 深入剖析源码
（5）Sharding-JDBC 源码剖析
（6）MyCat 源码剖析：虽然我们课程不用 mycat，但是作为目前非常火的，我们会深入剖析其源码
（7）Tomcat 源码剖析
（8）Elasgticsearch 源码剖析
（9）Kafka 源码剖析

通过这个阶段剖析大量的开源中间件以及分布式系统的源码剖析，我们会积累扎实的内功，实际上，上面这些源码都深度剖析以及理解过后，我们就可以深入理解分布式系统的 RPC 框架、分布式协调中间件、分布式缓存、分布式消息队列、分布式数据库中间件、Web 服务器、分布式搜索等常用核心技术的底层原理，积累扎实的内功修为。

12、项目阶段十：JDK+JVM+基础组件源码剖析（120 小时）

这个阶段，我们要能够对 JDK、JVM 以及基础组件的底层源码和原理有深入的理解！

上个阶段，在做性能优化的时候，实际上我们大量的跟一些基础组件打了交道，比如 JDK 中的并发、集合、IO 相关类，还有比如 JVM 这块，连接池这块，那么在这个阶段，我们在大量的进行各种性能优化手段的实践之后，同时对那些背后的技术和原理都进行了深入的学习之后，在这里就可以对常见的基础类库的源码进行剖析。

（1）JDK 中的集合、并发、IO 相关的源码剖析
（2）常用基础组件的剖析：数据库连接池、日志组件，等等
（3）对 JVM 我们会更加深入底层的学习内核原理，甚至尝试自己动手写一个简单的 JVM 出来，深入底层，积累扎实的功底

13、项目阶段十一：SaaS 云平台架构（120 小时）

这个阶段，我们要能够掌握如何研发一套按需租用的 SaaS 云平台架构！

最后这一块，是将我们之前做好的一整套系统，比如高并发、高可用、高性能的电商系统，以及各种公司框架，包括中间件系统，全部以 SaaS 云平台的模式开放出去，将我们的技术储备暴露出去，让别的公司按需使用，按需租用！这块我们将会讲解一整套的 SaaS 多租户云平台架构！

14、常见问题 Q&A

1、这里面这么大的课程量，老师能录制的完吗？

放心，我们项目阶段一，240 小时，大部分都是 3 月份一个月讲完的，而且 3 月份一个月，我一边讲课，一边写代码，手写 10 万行代码！所以不用担心我讲不完，只要你能确保自己每周 20~30 小时的学习时间，紧紧跟上我就行！

2、这个课程量也太大，时间周期太长了吧！这么多课程哪里学的完？

不好意思，自己仔细看下我的课程设计，我认为包含了一个架构师需要掌握的所有东西，很多细项都没有给出来。所以我只是按照我认为最好的课程标准去设计课程，去讲课程！而且

根据目前第一批学员的学习反馈来看，只要你够努力，每周用 20~30 小时学习，就能跟上进度，不掉队，1 年多学完绝对没问题！

如果你想每周就学习几个小时，然后轻轻松松，几个月后成为技术大牛，可能存在于武侠小说里吧！我不认可这种好逸恶劳的态度！记住一句话：吃得苦中苦方为人上人！要在技术上有大的突破，必须拿出一股拼劲来，硬拼 1 年多，最后实现技术上的脱胎换骨！

3、这么多课程 1 年多能讲完吗？

能，因为有些阶段我是并行同时讲的，比如项目阶段三和项目阶段八，就是并行的，所以根据之前的录制经验以及时间预估，差不多就是 1 年多可以讲完！

4、那我是不是要学到 1 年多以后才能出去跳槽啊？我等不及了啊！

谁说的一定要 1 年多以后才能跳槽？我这里每个阶段的课程量都非常大！我的定位就是，每个阶段讲完，你就是这个领域的技术专家！比如微服务这块，高并发这块，你学完几个阶段，而且是跟着我们走完整大型真实的项目，那么你在这个领域的积累已经在行业里远超别人！你可能学到一半儿，就可以出去跳槽，你的技术实力有很扎实的功底，绝对没问题！薪资涨一大截是肯定的！

5、如果还有疑问，可以找我的助理，小棠（QQ：2564484767）；或者是架构班一期的学员，Jeverson（QQ：770396102）

15、关于答疑、助教、职业发展规划和跳槽指导

目前石杉码农学院，已经完全正规化运作，后续会持续招收学员，所以是不可能我一个人来讲课和给所有人答疑的！我们接下来会启用助教模式，从第一批学员中抽取学的最好、理解最深、经验最丰富的学员出来作为我的助教！

针对日常学习的普通技术类问题：有助教学长会给大家进行答疑，助教同学一般每天都会定时看下相关的问题，然后回复大家

针对自己公司里的技术问题：这个没有任何一个培训课程是可以保证给学员解答自己公司里遇到的问题的！一般来说建议大家有这类工作里的问题，抛对应的学员群里，跟大家交流！如果还是有疑问，可以找我，我尽量给大家看看！但是因为是公司里的问题，我本人也不在现场，所以是不能保证给解决的！

针对职业发展规划和跳槽指导：我会对每个学员 1 对 1 的指导如何定制修改简历、如何跳槽、如何面试、如何谈 offer、如何规划自己的职业生涯，这块严格依赖于我本人的行业经

验，所以只能我来办！目前来看我的指导效果非常好，比如有的同学本来自己去找可能就找20k 薪资的，有我全程指导，可以拿到 25k，甚至 28k。

顺带提一句，每个同学都有机会可以成为助教，只要在自己的班级里学的最好，最快，那么就有机会脱颖而出，成为助教！如果成为助教，有很多的福利，比如每个月我都是支付助教工资的，可能不到一年，自己付这点儿学费就回来了！

16、关于课程永久性免费更新的事情

但凡熟悉我的人都知道，我跟之前两个网站合作的时候，都是在学员已经付费的情况下，坚持后面又用大量的时间，免费更新课程的！这个事情熟悉我的人有目共睹！所以我们的架构师课程也是一样！

之前提过了，从 2018 年 3 月开始全力录制，差不多会到 2019 年 9 月，上面所有东西都结束，那么在这之后呢？很简单，我会每年都持续更新最新的技术到这个课程里去，比如某个技术出了最新的版本？业内有最新的流行技术出现了？都会更新到课程里去!那么对于老学员来说，就是永久性的持续性免费学习后面的更新的课程！绝对不用二次付费！

17、关于课程学完之后的薪资问题

我也懒得说一些虚头巴脑的数字，给大家抛几个真实的数字，截止目前，我的第一批学员刚学了开头部分课程，但是有我的跳槽指导下，有人从 10k 涨到 20k，有人从 18k 到 26k，有人从 20k 到 28k，有人从 22k 到 30k，而且都是去还算不错的互联网公司。所以好好学，薪资都不是问题，出去可以斩获一堆 offer。

如果真的把这个课程全部学完了，或者学完了 70%~80%的大部分课程，那么出去 35k，甚至 40k，我相信都有可能！前提是，你自己必须很努力很努力的，每周付出 30+小时的时间，努力跟着我学习！

18、关于课程的费用、分期付款、学员筛选相关的问题

重点，课程费用目前是 2 万，大家自己看下这个课程量，项目，以及真实生产环境的演示，以及我承诺的永久免费更新，就知道是否值得了，不想多言！

其次，很多同学说一下子拿出 2 万太吃力了，理解！所以我们是支持支付宝付款的，支付宝上可以用自己绑定的信用卡付款，然后**自己对信用卡账单分期还款就好了，如果分 12 期，一般信用卡利息很低，也就年化 7%吧，每个月也就还 1800 左右，1 年还清了**。所以相信如果真的想学的同学，走分期方式，学费压力应该不大吧。况且我都说了，自己学到一半了，想跳槽，我指导，你的工资涨幅绝对比你自己出去找工作要高很多，这就赚回来了吧！

接着，说下报名的问题，**我们是走双向选择的，也就是学员选择我们，我们也要选择学员！**所以不是说，愿意交钱，就可以报名参加我们的学习，预约报名之后，每一期开班之前，我们会走较为严格的多轮筛选，我们会考核学员对我的了解程度、认可程度、信任程度、过往工作经验、技术能力、平时的学习时间、对待学习的态度、以及是否有努力的学习劲头，等等吧！我们也会筛选那批最有潜力，最好的学员，进来学习！否则大家换位思考一下，如果某个同学有很多地方跟我们不 match，结果交了钱进来没学好，怎么办呢？我们要对所有人负责，我们要确保每个人都能学的很好，所以只挑选最合适的学员！

19、关于重金投入防盗版的声明

我们的课程很庞大，耗费大量的精力去录制，这次我们自己做自己的品牌，对盗版的态度是：0 容忍！

1、我们所有的课程，全部是申请国家**合法的版权**的！如果有盗版现象，是受国家法律保护的，根据国家法律，盗版行为，轻则罚款，重则判刑！

2、所有学员要参加进来，必须**全部实名制**！每个人需要发给我们身份证照片，同时我们会对每个学员视频，现场验证真人跟身份证是否相符，而且还会将身份证现场联网验证，确保用的都是真实的身份证！

3、我们目前已经投入了很大一笔资金在加密上，我们是走了多重技术加密的，安全度远远高于市面上普通的加密系统！

4、我们已经业内最知名的维权骑士达成合作关系：只要定位到盗版人，维权骑士会代理我们全权起诉，直到盗版分子被法院判刑以及缴纳罚款！各位记住，做人要有最基本的道德，行的正坐得直，勿以恶小而为之，不要干偷鸡摸狗的盗版事情！我们付出大量的心血和努力在做课程，如果一个人整天想着来盗版，那么我们必定要让他受到法律的制裁，轻则数万、数十万、上百万的罚款！重则判刑坐牢！

20、盗版悬赏令！盗版悬赏令！盗版悬赏令！

目前我们针对所有人，发出盗版悬赏令，如果有人发现外面有人在合谋要多人

盗版合买我们的课程，或者搞我们的课程的盗版众筹！只要发现来告诉我们，同时协助我们获取足够的证据，一举抓获带头的盗版分子，协助我们提供证据让盗版分子被判刑或者罚款，我们直接给一个免费学习这个架构师课程的名额，同时根据举报人的具体贡献给出几千元到几万元不等的现金奖励！如果有发现有可疑分子，可以直接联系我的助理，小棠（QQ：2564484767）！

21、我们的官方网站和官方微信公众号

石杉码农学院的官方网站：www.zhonghuashishan.com

石杉码农学院的官方微信公众号：石杉码农

22、重要资金转账安全提醒

所有人请注意，为了避免大家遇到不法分子的诈骗陷阱，特在这里提醒，任何石杉码农学院的学费相关事宜，无论是支付宝还是银行转账，请认准只能是向对公账户（北京石杉网络科技有限公司）进行转账，请大家注意不要向其他任何个人账户或者其他公司账户进行转账！