微服务 面试题

微服务，又称微服务 架构，是一种架构风格，它将应用程序构建为以业务领域为模型的小型自治服务集合。

通俗地说，你必须看到蜜蜂如何通过对齐六角形蜡细胞来构建它们的蜂窝状物。他们最初从使用各种材料的小部分开始，并继续从中构建一个大型蜂箱。这些细胞形成图案，产生坚固的结构，将蜂窝的特定部分固定在一起。这里，每个细胞独立于另一个细胞，但它也与其他细胞相关。这意味着对一个细胞的损害不会损害其他细胞，因此，蜜蜂可以在不影响完整蜂箱的情况下重建这些细胞。

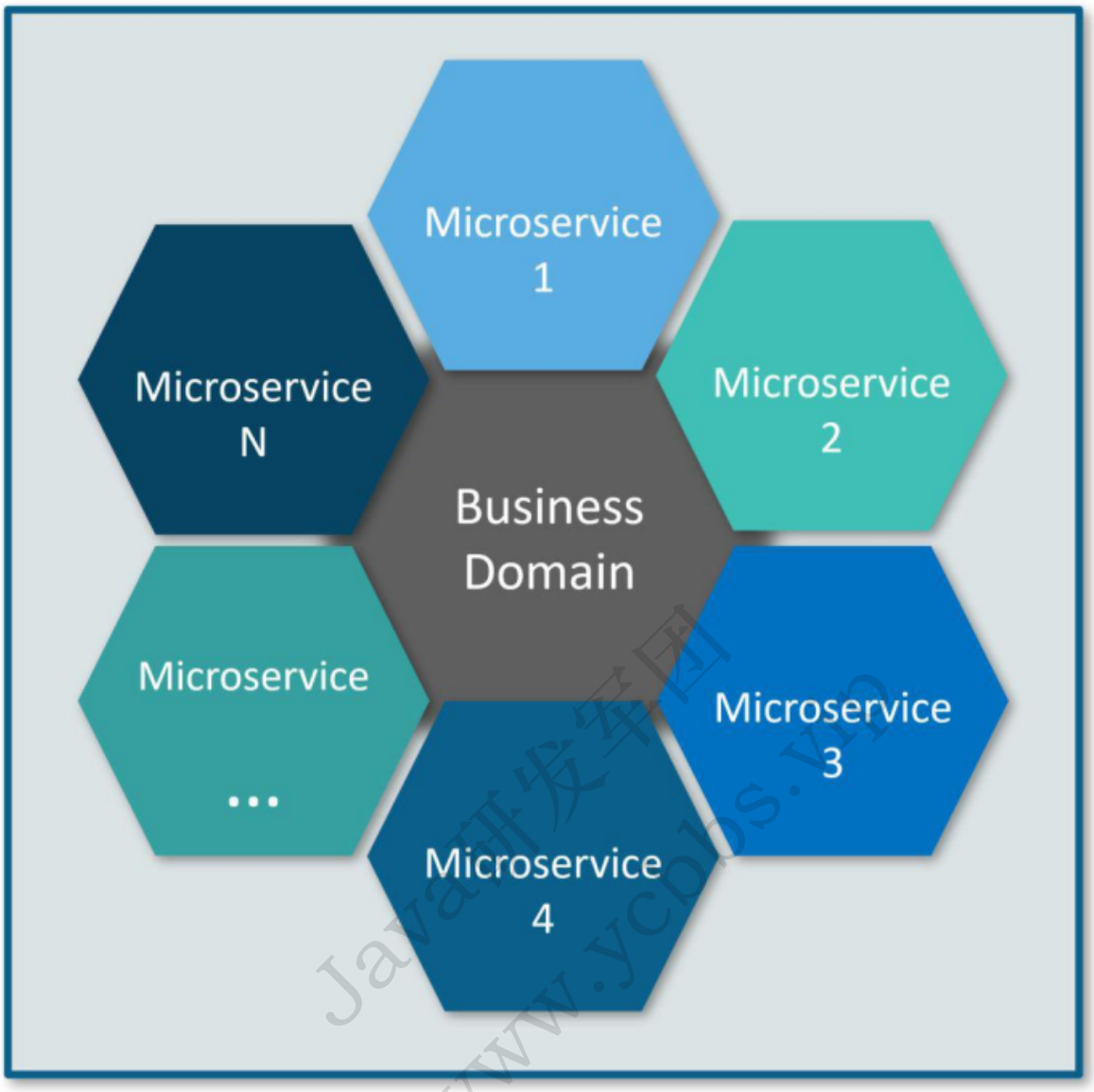

图 1：微服务的蜂窝表示 – 微服务访谈问题

请参考上图。这里，每个六边形形状代表单独的服务组件。与蜜蜂的工作类似，每个敏捷团队都使用可用的框架和所选的技术堆栈构建单独的服务组件。就像在蜂箱中一样，每个服务组件形成一个强大的微服务架构，以提供更好的可扩展性。此外，敏捷团队可以单独处理每个服务组件的问题，而对整个应用程序没有影响或影响最小。

2、微服务架构有哪些优势？

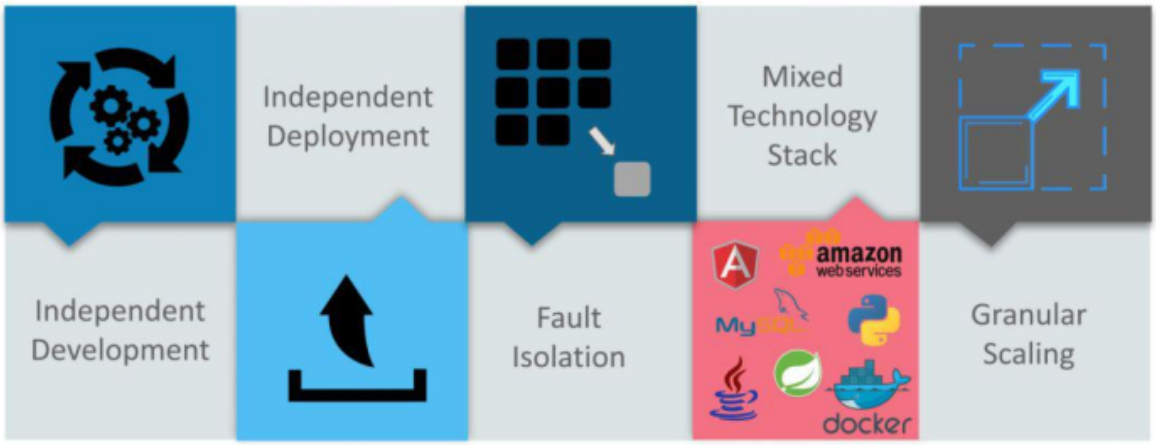

图 2：微服务的 优点 – 微服务访谈问题
 独立开发 – 所有微服务都可以根据各自的功能轻松开发
 独立部署 – 基于其服务，可以在任何应用程序中单独部署它们
 故障隔离 – 即使应用程序的一项服务不起作用，系统仍可继续运行
 混合技术堆栈 – 可以使用不同的语言和技术来构建同一应用程序的不同服务
 粒度缩放 – 单个组件可根据需要进行缩放，无需将所有组件缩放在一起

3、微服务有哪些特点？

图 3：微服务的 特点 – 微服务访谈问题
 解耦 – 系统内的服务很大程度上是分离的。因此，整个应用程序可以轻松构建，更改和扩展
 组件化 – 微服务被视为可以轻松更换和升级的独立组件
 业务能力 – 微服务非常简单，专注于单一功能
 自治 – 开发人员和团队可以彼此独立工作，从而提高速度
 持续交付 – 通过软件创建，测试和批准的系统自动化，允许频繁发布软件
 责任 – 微服务不关注应用程序作为项目。相反，他们将应用程序视为他们负责的产品
 分散治理 – 重点是使用正确的工具来做正确的工作。这意味着没有标准化模式或任何技术模式。开发人员可以自由选择最有用的工具来解决他们的问题
 敏捷 – 微服务支持敏捷开发。任何新功能都可以快速开发并再次丢弃

4、设计微服务的最佳实践是什么？

以下是设计微服务的最佳实践：

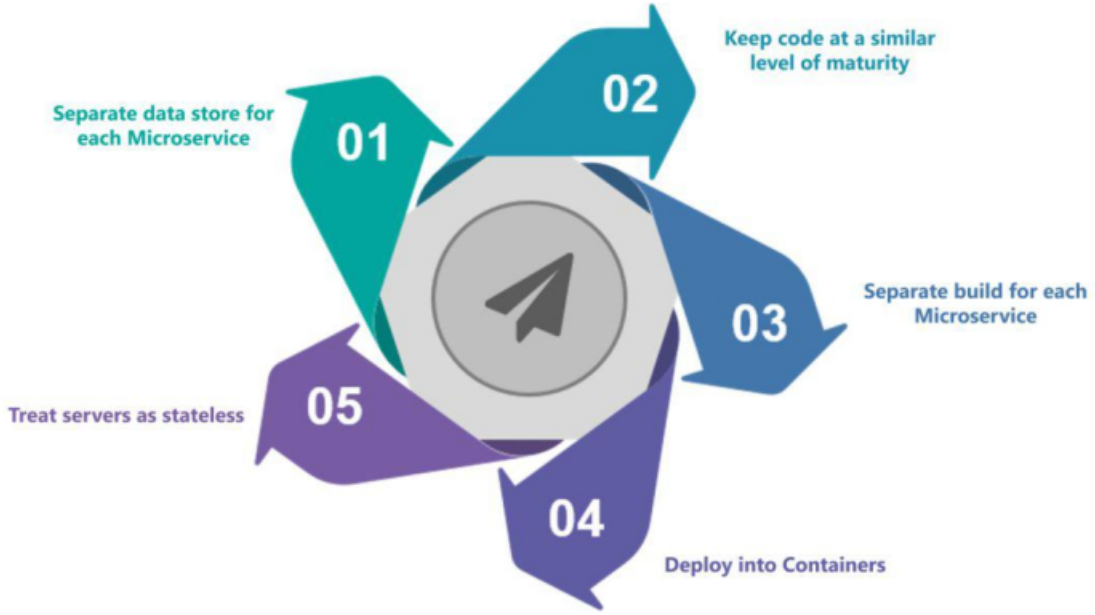

图 4：设计微服务的最佳实践 – 微服务访谈问题

5、微服务架构如何运作？

微服务架构具有以下组件：

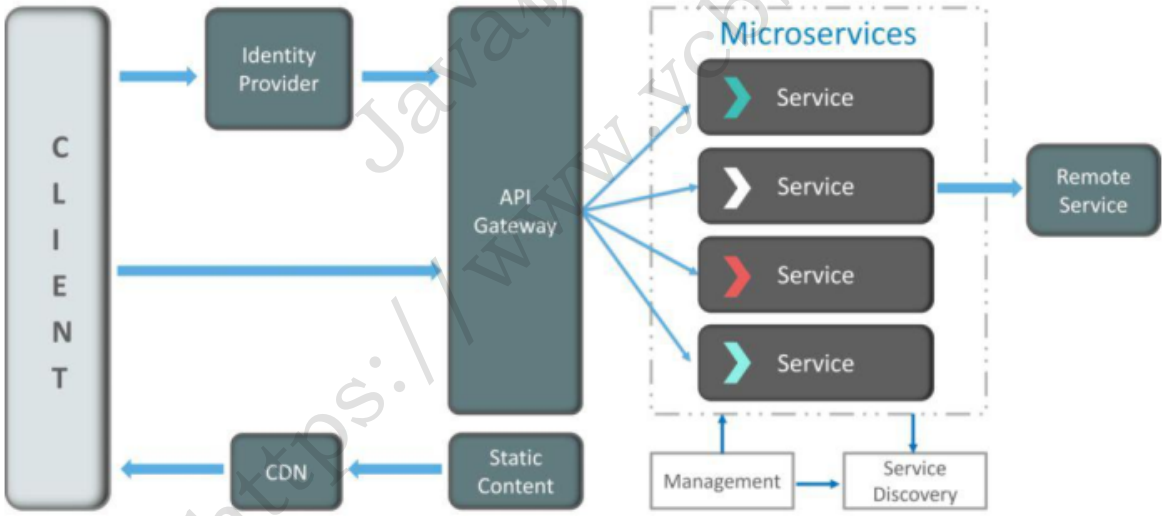

图 5：微服务 架构 – 微服务面试问题

- 客户端 – 来自不同设备的不同用户发送请求。
- 身份提供商 – 验证用户或客户身份并颁发安全令牌。
- API 网关 – 处理客户端请求。
- 静态内容 – 容纳系统的所有内容。
- 管理 – 在节点上平衡服务并识别故障。
- 服务发现 – 查找微服务之间通信路径的指南。
- 内容交付网络 – 代理服务器及其数据中心的分布式网络。
- 远程服务 – 启用驻留在 IT 设备网络上的远程访问信息。

6、微服务架构的优缺点是什么？

微服务架构的优点	微服务架构的缺点
自由使用不同的技术	增加故障排除挑战
每个微服务都侧重于单一功能	由于远程呼叫而增加延迟
支持单个可部署单元	增加了配置和其他操作的工作量
允许经常发布软件	难以保持交易安全
确保每项服务的安全性	艰难地跨越各种边界跟踪数据
多个服务是并行开发和部署的	难以在服务之间进行编码

7、单片，SOA 和微服务架构有什么区别？

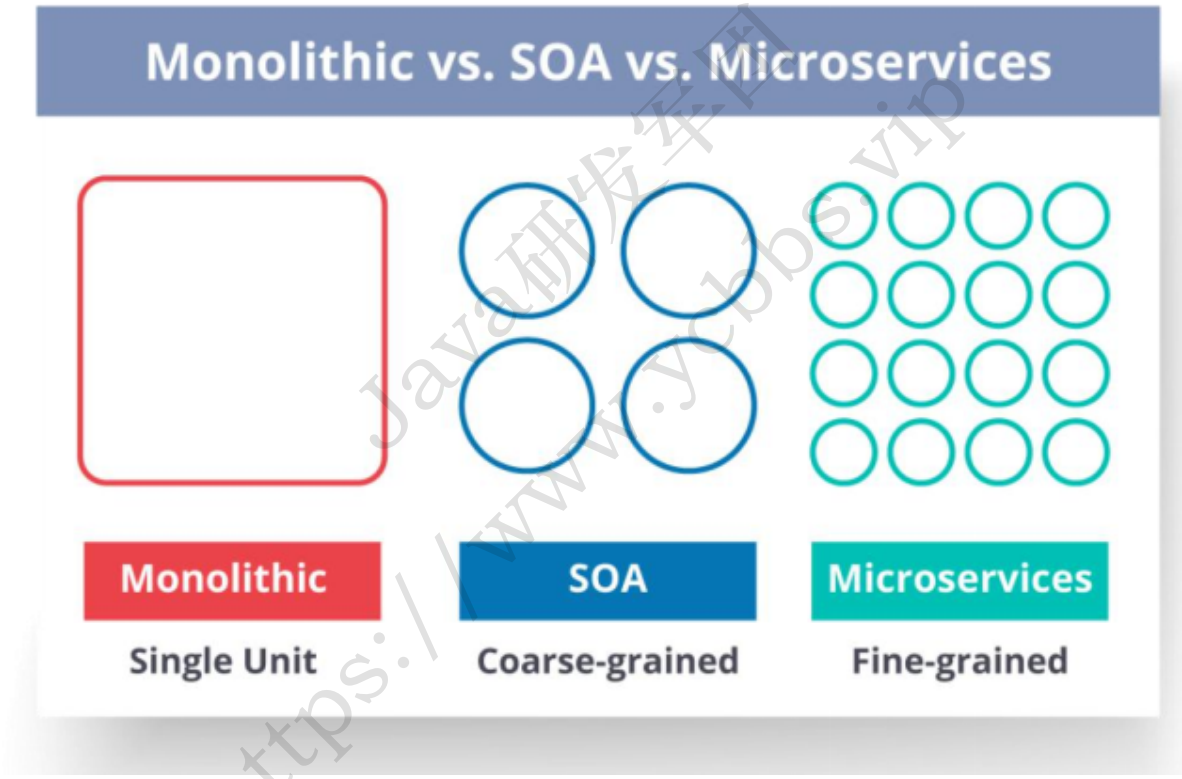

图 6： 单片 SOA 和微服务之间的比较 – 微服务访谈问题

□ 单片架构类似于大容器，其中应用程序的所有软件组件组装在一起并紧密封装。

□ 一个面向服务的架构是一种相互通信服务的集合。通信可以涉及简单的数据传递，也可以涉及两个或多个协调某些活动的服务。

□ 微服务架构是一种架构风格，它将应用程序构建为以业务域为模型的小型自治服务集合。

8、在使用微服务架构时，您面临哪些挑战？

开发一些较小的微服务听起来很容易，但开发它们时经常遇到的挑战如下。

□ 自动化组件：难以自动化，因为有许多较小的组件。因此，对于每个组件，我们必须遵循 Build，Deploy 和 Monitor 的各个阶段。

□ 易感性：将大量组件维护在一起变得难以部署，维护，监控和识别问题。它需要在所有组件周围具有很好的感知能力。

□ 配置管理：有时在各种环境中维护组件的配置变得困难。
□ 调试：很难找到错误的每一项服务。维护集中式日志记录和仪表板以调试问题至关重要。

9、SOA 和微服务架构之间的主要区别是什么？

SOA 和微服务之间的主要区别如下：

SOA	微服务
遵循" 尽可能多的共享 "架构方法	遵循" 尽可能少分享 "的架构方法
重要性在于 业务功能 重用	重要性在于" 有界背景 " 的概念
他们有 共同的 治理 和标准	他们专注于 人们的 合作 和其他选择的自由
使用 企业服务总线（ESB） 进行通信	简单的消息系统
它们支持 多种消息协议	他们使用 轻量级协议 ，如 HTTP / REST 等。
多线程， 有更多的开销来处理I / O.	单线程 通常使用Event Loop功能进行非锁定I / O 处理
最大化应用程序服务可重用性	专注于 解耦
传统的关系数据库 更常用	现代 关系数据库 更常用
系统的变化需要修改整体	系统的变化是创造一种新的服务
DevOps / Continuous Delivery正在变得流行，但还不是主流	专注于DevOps /持续交付

10、微服务有什么特点？

您可以列出微服务的特征，如下所示：

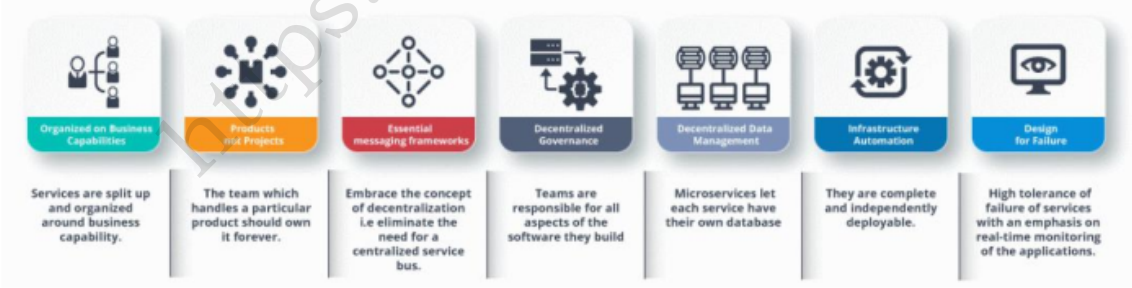

图 7：微服务的特征 – 微服务访谈问题

11、什么是领域驱动设计？

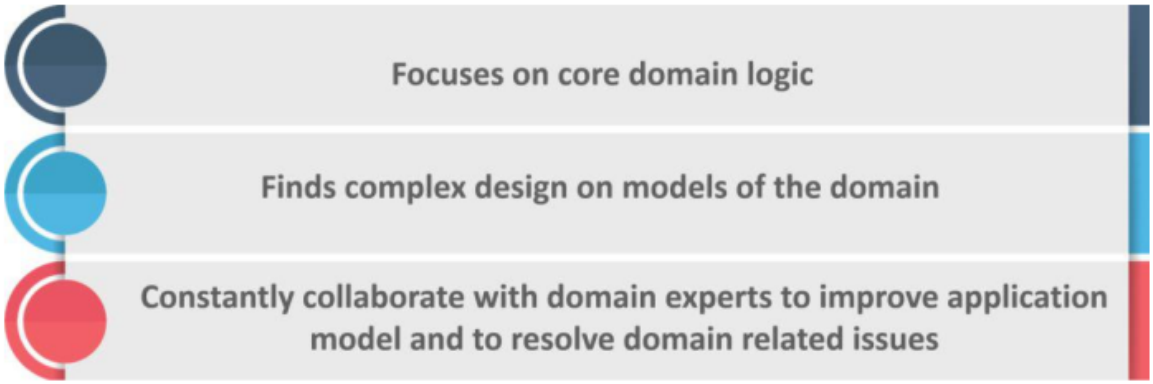

图 8： DDD 原理 – 微服务面试问题

12、为什么需要域驱动设计（DDD）？

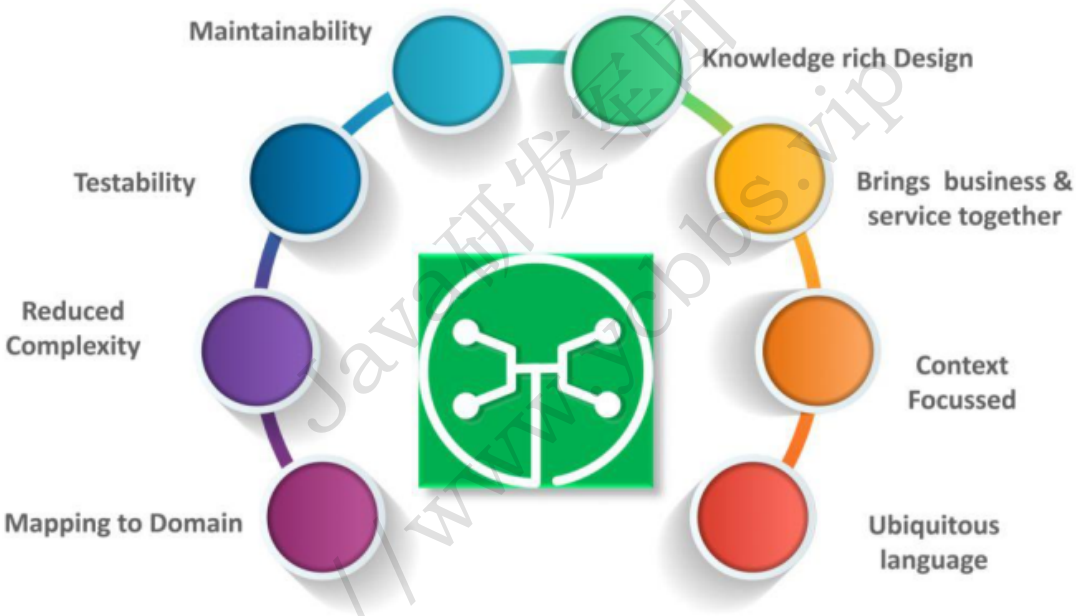

图 9：我们需要 DDD 的因素 – 微服务面试问题

13、什么是无所不在的语言？

如果您必须定义泛在语言（UL），那么它是特定域的开发人员和用户使用的通用语言，通过该语言可以轻松解释域。
无处不在的语言必须非常清晰，以便它将所有团队成员放在同一页面上，并以机器可以理解的方式进行翻译。

14、什么是凝聚力？

模块内部元素所属的程度被认为是凝聚力。

15、什么是耦合？

组件之间依赖关系强度的度量被认为是耦合。一个好的设计总是被认为具有高内聚力和低耦合性。

16、什么是 REST / RESTful 以及它的用途是什么？

Representational State Transfer（REST）/ RESTful Web 服务是一种帮助计算机系统通过 Internet 进行通信的架构风格。这使得微服务更容易理解和实现。微服务可以使用或不使用 RESTful API 实现，但使用 RESTful API 构建松散耦合的微服务总是更容易。

17、你对 Spring Boot 有什么了解？

事实上，随着新功能的增加，弹簧变得越来越复杂。如果必须启动新的 spring 项目，则必须添加构建路径或添加 maven 依赖项，配置应用程序服务器，添加 spring配置。所以一切都必须从头开始。Spring Boot 是解决这个问题的方法。使用 spring boot 可以避免所有样板代码和配置。因此，基本上认为自己就好像你正在烘烤蛋糕一样，春天就像制作蛋糕所需的成分一样，弹簧靴就是你手中的完整蛋糕。

图 10： Spring Boot 的因素 – 微服务面试问题

18、什么是 Spring 引导的执行器？

Spring Boot 执行程序提供了 restful Web 服务，以访问生产环境中运行应用程序的当前状态。在执行器的帮助下，您可以检查各种指标并监控您的应用程序。

19、什么是 Spring Cloud？

根据 Spring Cloud 的官方网站，Spring Cloud 为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，领导选举，分布式会话，集群状态）。

20、Spring Cloud 解决了哪些问题？

在使用 Spring Boot 开发分布式微服务时，我们面临的问题很少由 Spring Cloud解决。
 与分布式系统相关的复杂性 – 包括网络问题，延迟开销，带宽问题，安全问题。
 处理服务发现的能力 – 服务发现允许集群中的进程和服务找到彼此并进行通信。
 解决冗余问题 – 冗余问题经常发生在分布式系统中。
 负载平衡 – 改进跨多个计算资源（例如计算机集群，网络链接，中央处理单元）的工作负载分布。
 减少性能问题 – 减少因各种操作开销导致的性能问题。

21、在 Spring MVC 应用程序中使用 WebMvcTest 注释有什么用处？

在测试目标只关注 Spring MVC 组件的情况下，WebMvcTest 注释用于单元测试Spring MVC 应用程序。在上面显示的快照中，我们只想启动 ToTestController。执行此单元测试时，不会启动所有其他控制器和映射。

22、你能否给出关于休息和微服务的要点？

虽然您可以通过多种方式实现微服务，但 REST over HTTP 是实现微服务的一种方式。REST 还可用于其他应用程序，如 Web 应用程序，API 设计和 MVC 应用程序，以提供业务数据。
微服务是一种体系结构，其中系统的所有组件都被放入单独的组件中，这些组件可以单独构建，部署和扩展。微服务的某些原则和最佳实践有助于构建弹性应用程序。简而言之，您可以说 REST 是构建微服务的媒介。

23、什么是不同类型的微服务测试？

在使用微服务时，由于有多个微服务协同工作，测试变得非常复杂。因此，测试分为不同的级别。
 在底层，我们有面向技术的测试，如单元测试和性能测试。这些是完全自动化的。
 在中间层面，我们进行了诸如压力测试和可用性测试之类的探索性测试。
 在顶层， 我们的 验收测试数量很少。这些验收测试有助于利益相关者理解和验证软件功能。

24、您对 Distributed Transaction 有何了解？

分布式事务是指单个事件导致两个或多个不能以原子方式提交的单独数据源的突变的任何情况。在微服务的世界中，它变得更加复杂，因为每个服务都是一个工作单元，并且大多数时候多个服务必须协同工作才能使业务成功。

25、什么是 Idempotence 以及它在哪里使用？

幂等性是能够以这样的方式做两次事情的特性，即最终结果将保持不变，即好像它只做了一次。
用法：在远程服务或数据源中使用 Idempotence，这样当它多次接收指令时，它只处理指令一次。

26、什么是有界上下文？

有界上下文是域驱动设计的核心模式。DDD 战略设计部门的重点是处理大型模型和团队。DDD 通过将大型模型划分为不同的有界上下文并明确其相互关系来处理大型模型。

27、什么是双因素身份验证？

双因素身份验证为帐户登录过程启用第二级身份验证。

图 11： 双因素认证的表示 – 微服务访谈问题
因此，假设用户必须只输入用户名和密码，那么这被认为是单因素身份验证。

28、双因素身份验证的凭据类型有哪些？

这三种凭证是 ：

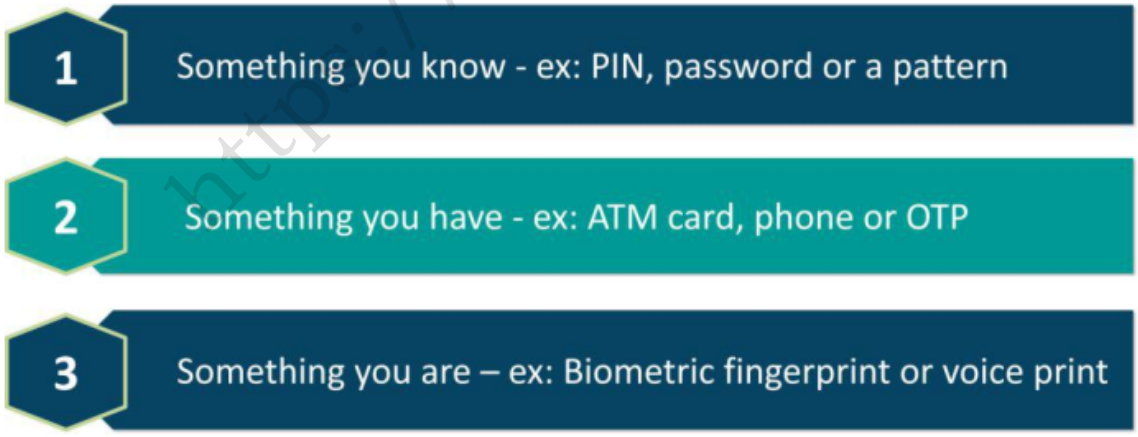

图 12： 双因素认证的证书类型 – 微服务面试问题

29、什么是客户证书？

客户端系统用于向远程服务器发出经过身份验证的请求的一种数字证书称为客户端证书。客户端证书在许多相互认证设计中起着非常重要的作用，为请求者的身份提供了强有力的保证。

30、PACT 在微服务架构中的用途是什么？

PACT 是一个开源工具，允许测试服务提供者和消费者之间的交互，与合同隔离，从而提高微服务集成的可靠性。

微服务中的用法

- 用于在微服务中实现消费者驱动的合同。
- 测试微服务的消费者和提供者之间的消费者驱动的合同。

查看即将到来的批次

31、什么是 OAuth？

OAuth 代表开放授权协议。这允许通过在 HTTP 服务上启用客户端应用程序（例如第三方提供商 Facebook，GitHub 等）来访问资源所有者的资源。因此，您可以在不使用其凭据的情况下与另一个站点共享存储在一个站点上的资源。

32、康威定律是什么？

“任何设计系统的组织（广泛定义）都将产生一种设计，其结构是组织通信结构的副本。” – Mel Conway

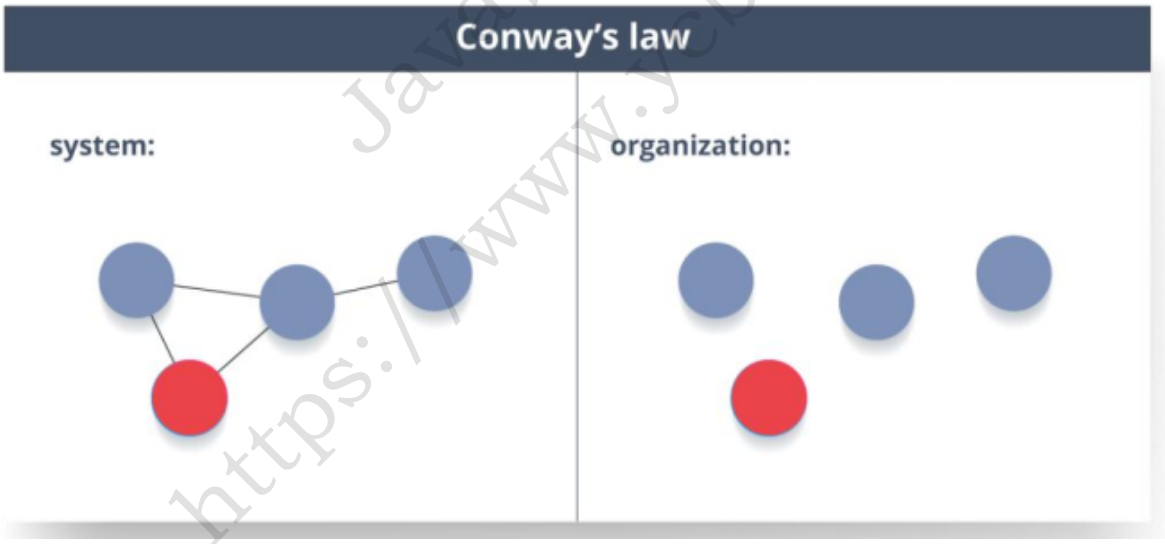

图 13： Conway 定律的表示 – 微服务访谈问题

该法律基本上试图传达这样一个事实：为了使软件模块起作用，整个团队应该进行良好的沟通。因此，系统的结构反映了产生它的组织的社会边界。

33、合同测试你懂什么？

根据 Martin Flower 的说法，合同测试是在外部服务边界进行的测试，用于验证其是否符合消费服务预期的合同。

此外，合同测试不会深入测试服务的行为。更确切地说，它测试该服务调用的输入＆输出包含所需的属性和所述响应延迟，吞吐量是允许的限度内

34、什么是端到端微服务测试？

端到端测试验证了工作流中的每个流程都正常运行。这可确保系统作为一个整体协同工作并满足所有要求。
通俗地说，你可以说端到端测试是一种测试，在特定时期后测试所有东西。

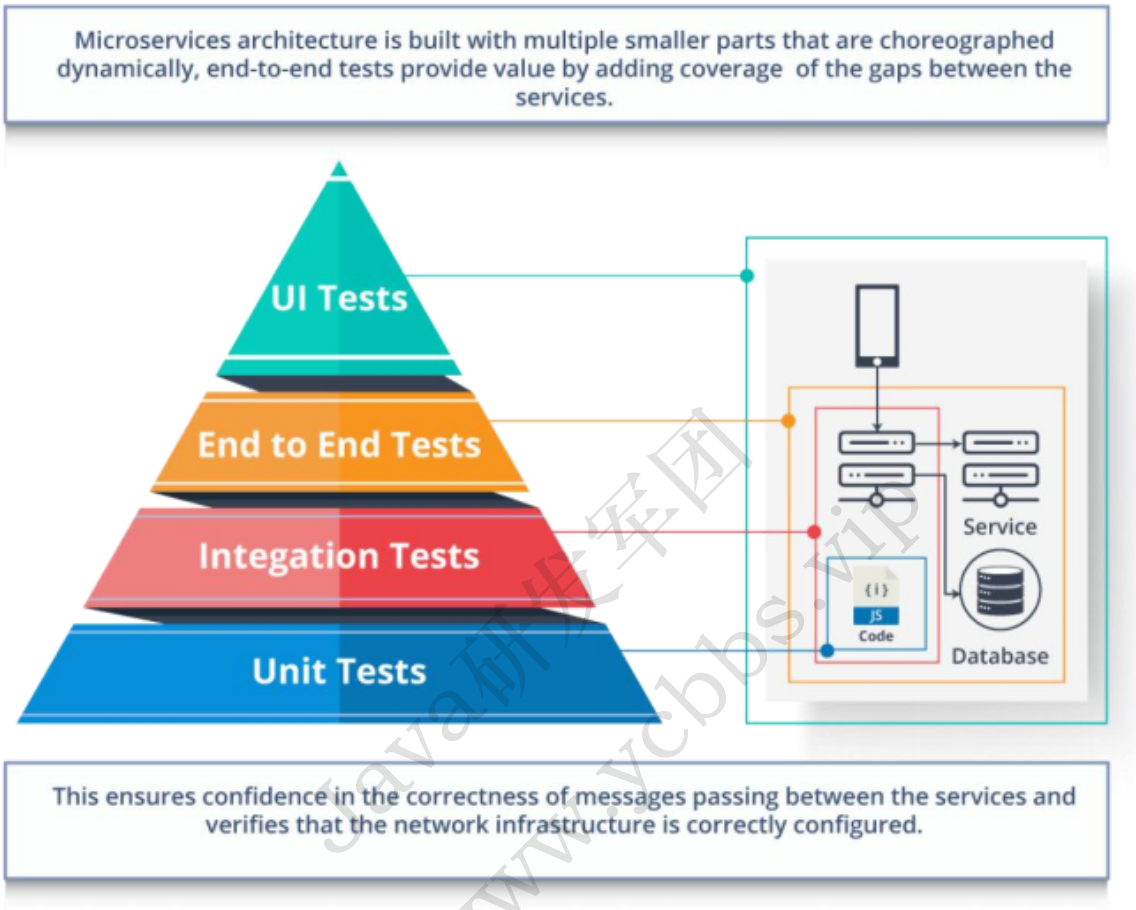

图 14：测试层次 – 微服务面试问题

35、Container 在微服务中的用途是什么？

容器是管理基于微服务的应用程序以便单独开发和部署它们的好方法。您可以将微服务封装在容器映像及其依赖项中，然后可以使用它来滚动按需实例的微服务，而无需任何额外的工作。

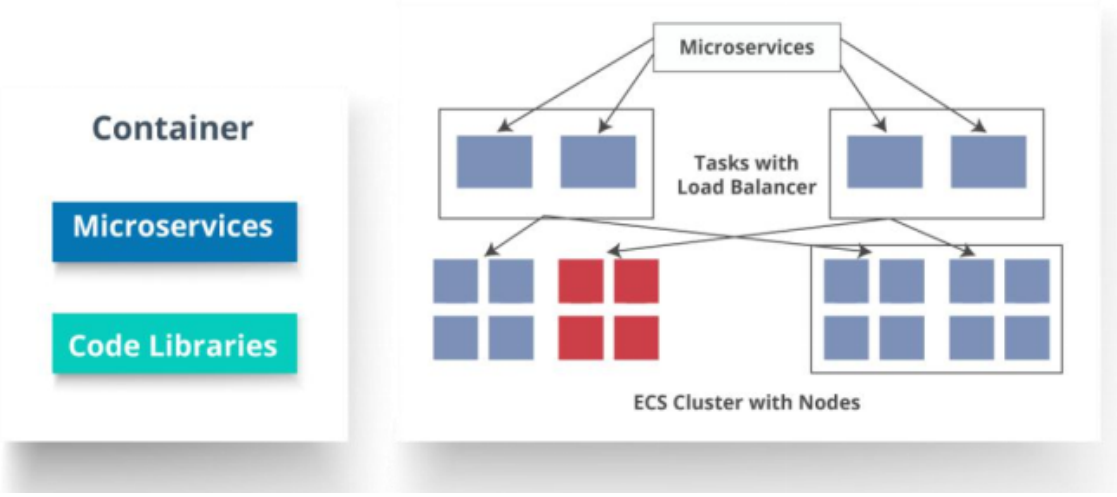

图 15： 容器的表示及其在微服务中的使用方式 – 微服务访谈问题

36、什么是微服务架构中的 DRY？

DRY 代表不要重复自己。它基本上促进了重用代码的概念。这导致开发和共享库，这反过来导致紧密耦合。

37、什么是消费者驱动的合同（CDC）？

这基本上是用于开发微服务的模式，以便它们可以被外部系统使用。当我们处理微服务时，有一个特定的提供者构建它，并且有一个或多个使用微服务的消费者。通常，提供程序在 XML 文档中指定接口。但在消费者驱动的合同中，每个服务消费者都传达了提供商期望的接口。

38、Web，RESTful API 在微服务中的作用是什么？

微服务架构基于一个概念，其中所有服务应该能够彼此交互以构建业务功能。因此，要实现这一点，每个微服务必须具有接口。这使得 Web API 成为微服务的一个非常重要的推动者。RESTful API 基于 Web 的开放网络原则，为构建微服务架构的各个组件之间的接口提供了最合理的模型。

39、您对微服务架构中的语义监控有何了解？

语义监控，也称为 综合监控， 将自动化测试与监控应用程序相结合，以检测业务失败因素。

40、我们如何进行跨功能测试？

跨功能测试是对非功能性需求的验证，即那些无法像普通功能那样实现的需求。

41、我们如何在测试中消除非决定论？

非确定性测试（NDT）基本上是不可靠的测试。所以，有时可能会发生它们通过，显然有时它们也可能会失败。当它们失败时，它们会重新运行通过。

从测试中删除非确定性的一些方法如下：

1、 隔离
2、 异步
3、 远程服务
4、 隔离
5、 时间
6、 资源泄漏

42、Mock 或 Stub 有什么区别？

存根

- 一个有助于运行测试的虚拟对象。
- 在某些可以硬编码的条件下提供固定行为。
- 永远不会测试存根的任何其他行为。

例如，对于空堆栈，您可以创建一个只为 empty（）方法返回 true 的存根。因此，这并不关心堆栈中是否存在元素。

嘲笑

- 一个虚拟对象，其中最初设置了某些属性。
- 此对象的行为取决于 set 属性。
- 也可以测试对象的行为。

例如，对于 Customer 对象，您可以通过设置名称和年龄来模拟它。您可以将 age设置为 12，然后测试 isAdult（）方法，该方法将在年龄大于 18 时返回 true。因此，您的 Mock Customer 对象适用于指定的条件。

43、您对 Mike Cohn 的测试金字塔了解多少？

Mike Cohn 提供了一个名为 Test Pyramid 的模型。这描述了软件开发所需的自动化测试类型。

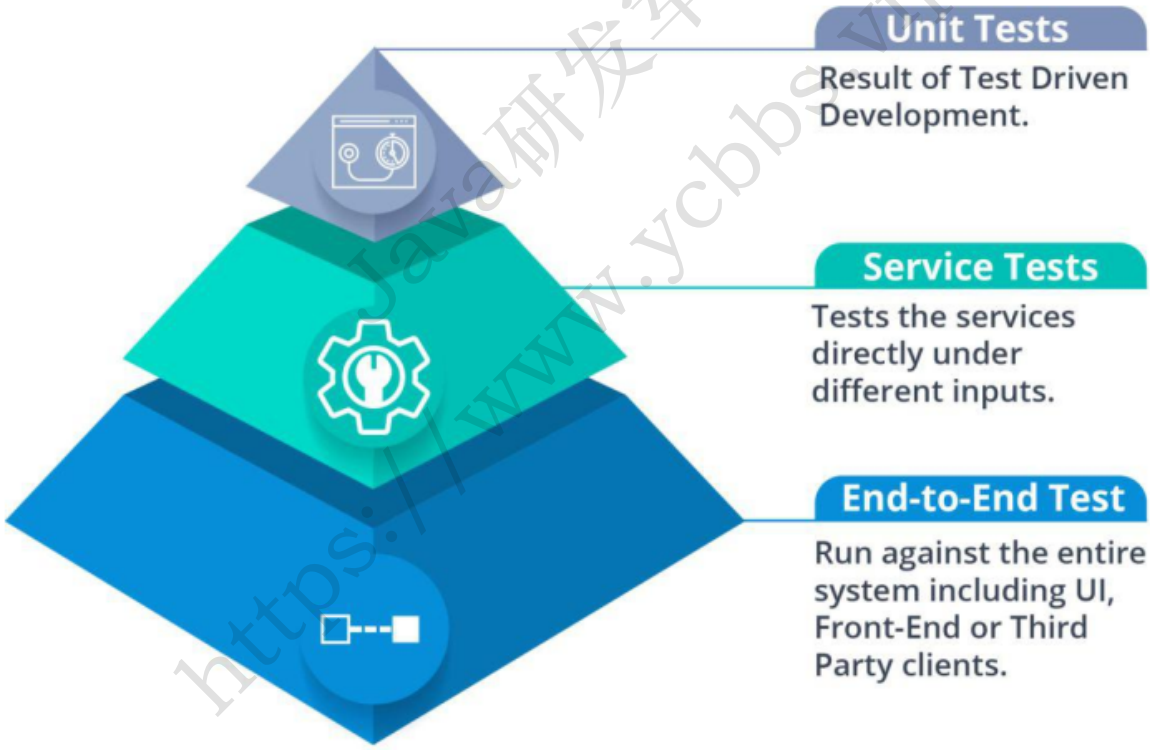

图 16： Mike Cohn 的测试金字塔 – 微服务面试问题

根据金字塔，第一层的测试数量应该最高。在服务层，测试次数应小于单元测试级别，但应大于端到端级别。

44、Docker 的目的是什么？

Docker 提供了一个可用于托管任何应用程序的容器环境。在此，软件应用程序和支持它的依赖项紧密打包在一起。因此，这个打包的产品被称为 Container，因为它是由 Docker 完成的，所以它被称为 Docker 容器！

45、什么是金丝雀释放？

Canary Releasing 是一种降低在生产中引入新软件版本的风险的技术。这是通过将变更缓慢地推广到一小部分用户，然后将其发布到整个基础架构，即将其提供给每个人来完成的。

46、什么是持续集成（CI）？

持续集成（CI）是每次团队成员提交版本控制更改时自动构建和测试代码的过程。这鼓励开发人员通过在每个小任务完成后将更改合并到共享版本控制存储库来共享代码和单元测试。

47、什么是持续监测？

持续监控深入监控覆盖范围，从浏览器内前端性能指标，到应用程序性能，再到主机虚拟化基础架构指标。

48、架构师在微服务架构中的角色是什么？

微服务架构中的架构师扮演以下角色：
□ 决定整个软件系统的布局。
□ 帮助确定组件的分区。因此，他们确保组件相互粘合，但不紧密耦合。
□ 与开发人员共同编写代码，了解日常生活中面临的挑战。
□ 为开发微服务的团队提供某些工具和技术的建议。
□ 提供技术治理，以便技术开发团队遵循微服务原则。

49、我们可以用微服务创建状态机吗？

我们知道拥有自己的数据库的每个微服务都是一个可独立部署的程序单元，这反过来又让我们可以创建一个状态机。因此，我们可以为特定的微服务指定不同的状态和事件。
例如，我们可以定义 Order 微服务。订单可以具有不同的状态。Order 状态的转换可以是 Order 微服务中的独立事件。

50、什么是微服务中的反应性扩展？

Reactive Extensions 也称为 Rx。这是一种设计方法，我们通过调用多个服务来收集结果，然后编译组合响应。这些调用可以是同步或异步，阻塞或非阻塞。Rx是分布式系统中非常流行的工具，与传统流程相反。希望这些微服务面试问题可以帮助您进行微服务架构师访谈。
翻译来源：
https://www.edureka.co/blog/interview-questions/microservices-interview-questions/