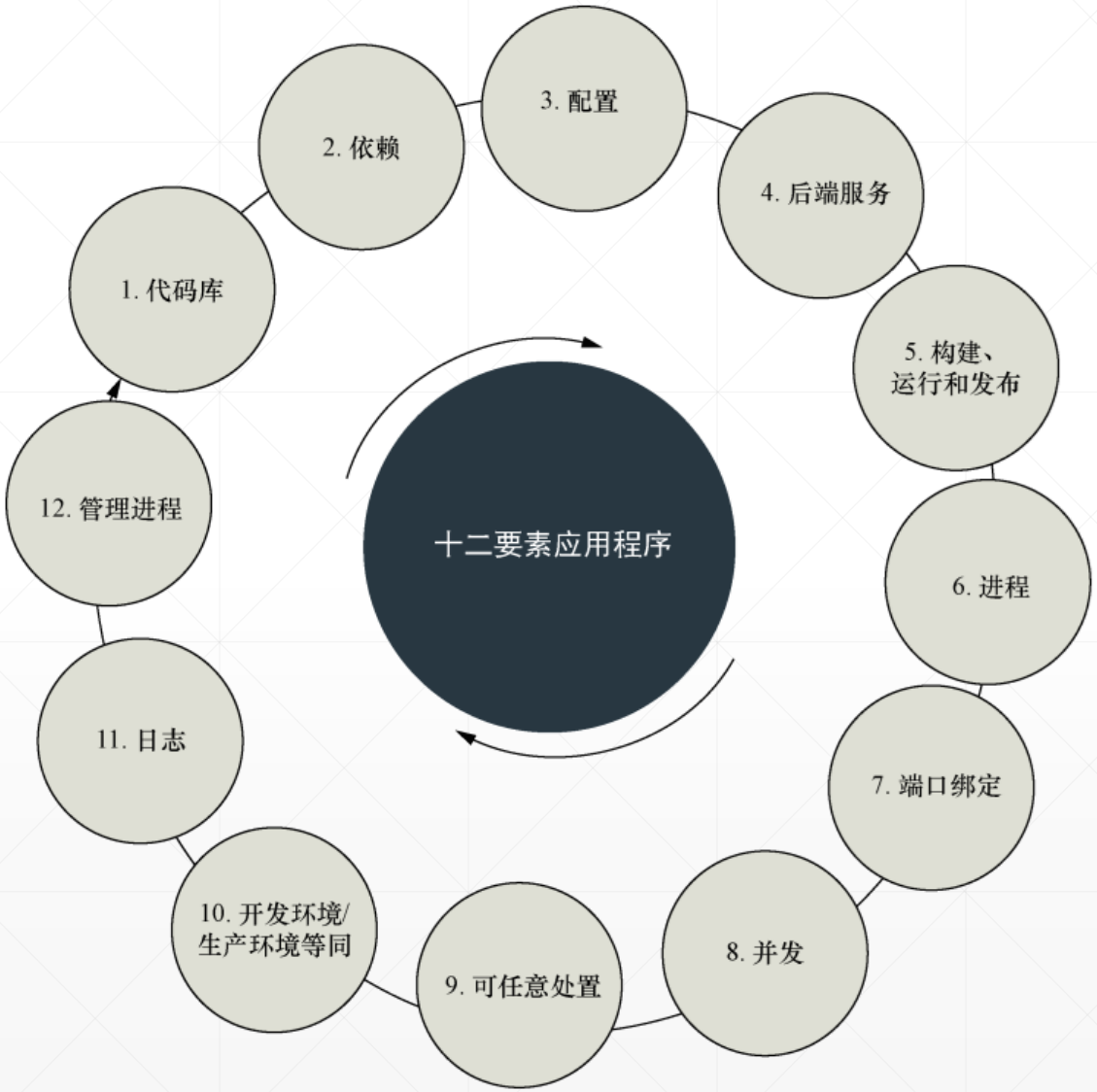


微服务构建十二要素

北京理工大学计算机学院
金旭亮

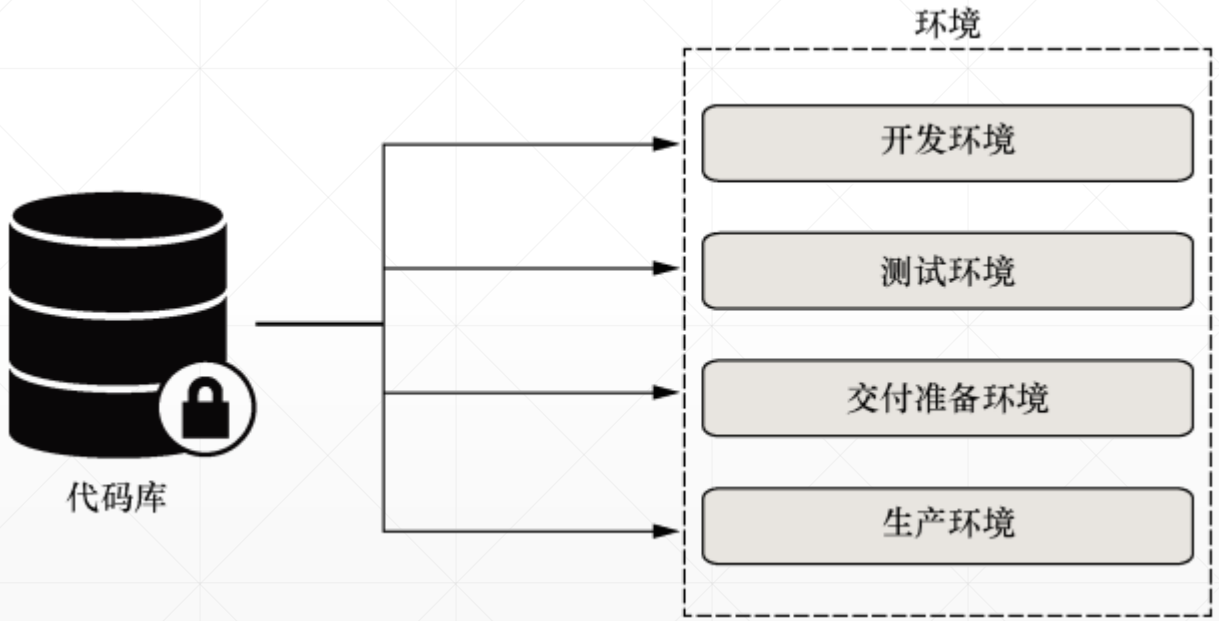
微服务构建十二要素

由Heroku公司的开发人员在2002年总结出来。



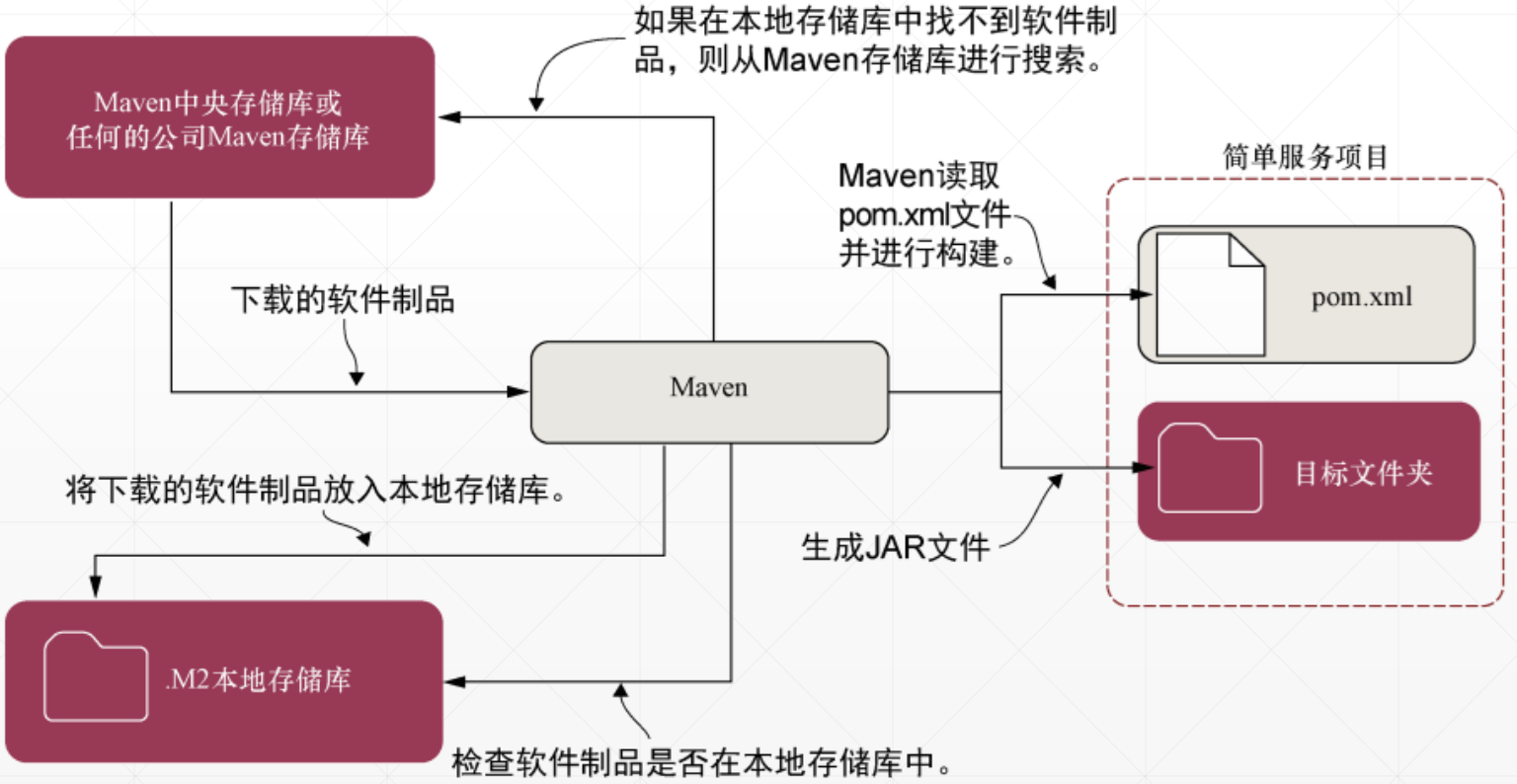
要素一：代码库

每个微服务都应该有一个单独的、在源代码版本管理系统管理下的代码库。



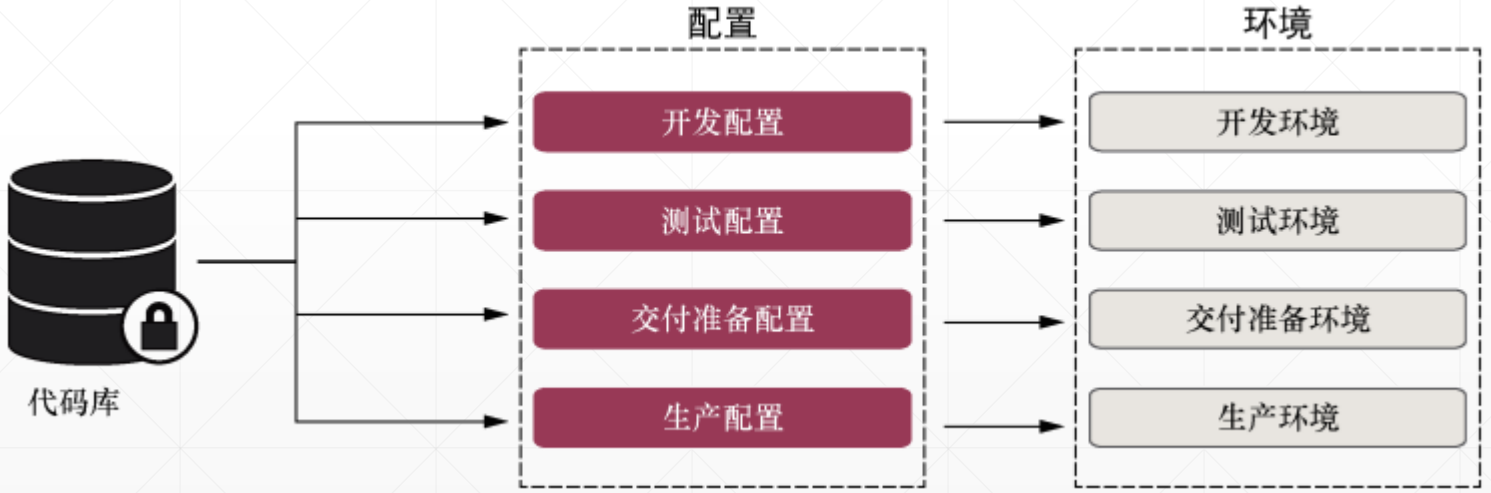
要素二：依赖

应该通过自动化的构建工具（如Maven或Gradle）来管理项目依赖，而不要手工管理，并且应该尽可能地统一各个微服务所使用的Java版本。



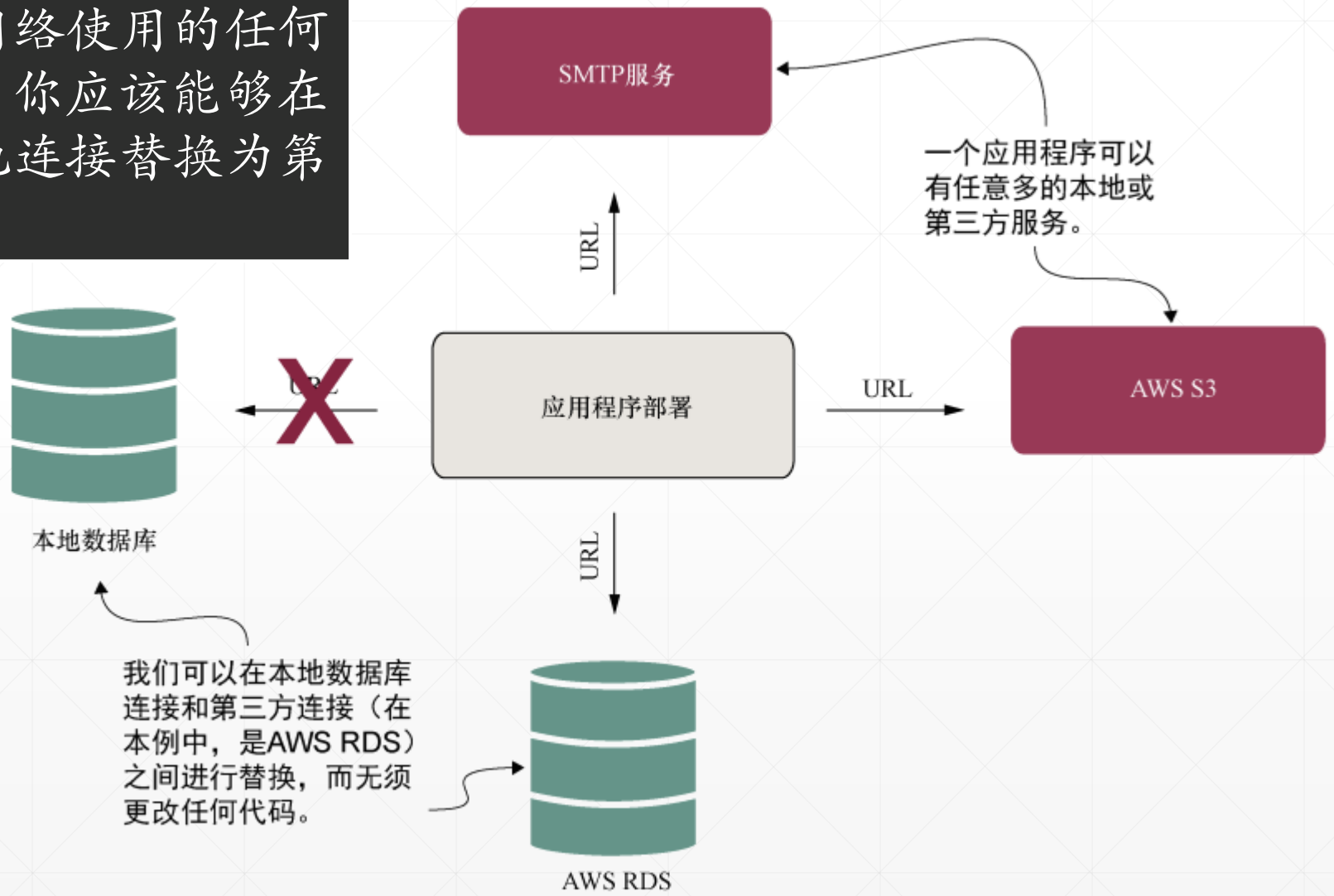
要素三：配置

应该将配置独立出来，与可部署的微服务本身完全分离。这样做的好处之一，是可以很方便地对配置进行同步和更新，并且可以使用专用的微服务集中管理整个系统的配置信息。



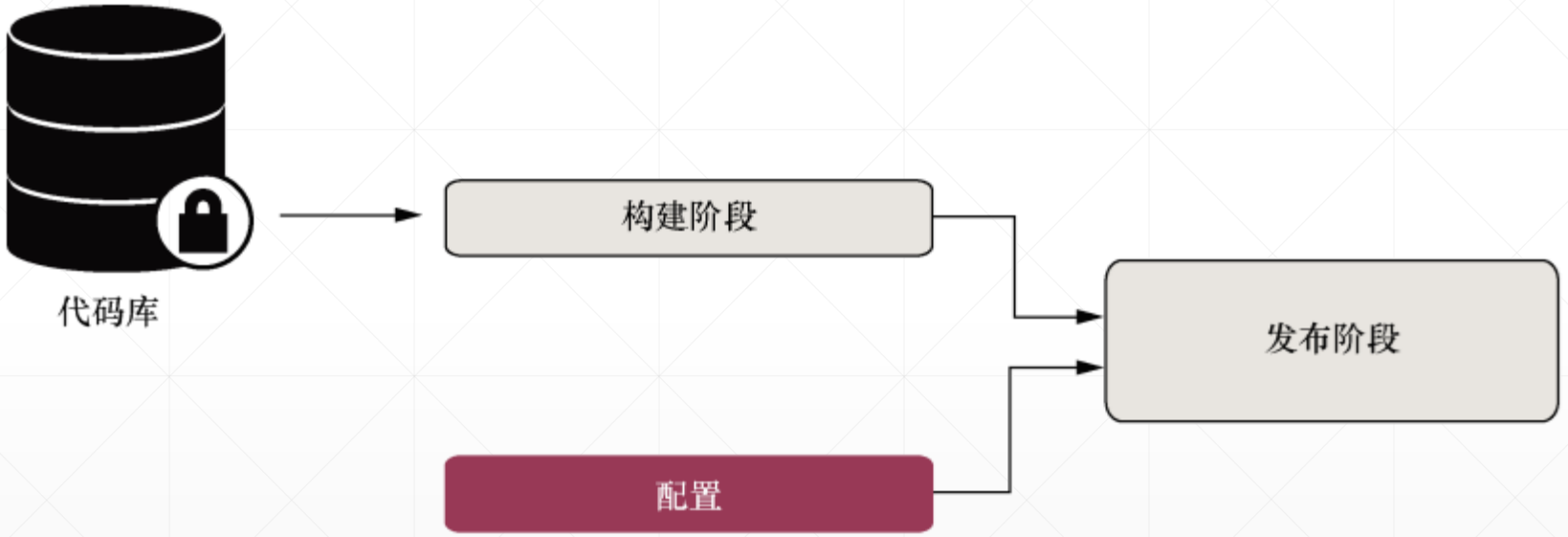
要素四：后端服务

后端服务是应用程序通过网络使用的任何服务。在部署应用程序时，你应该能够在不更改代码的情况下将本地连接替换为第三方连接。



要素五：构建、发布和运行

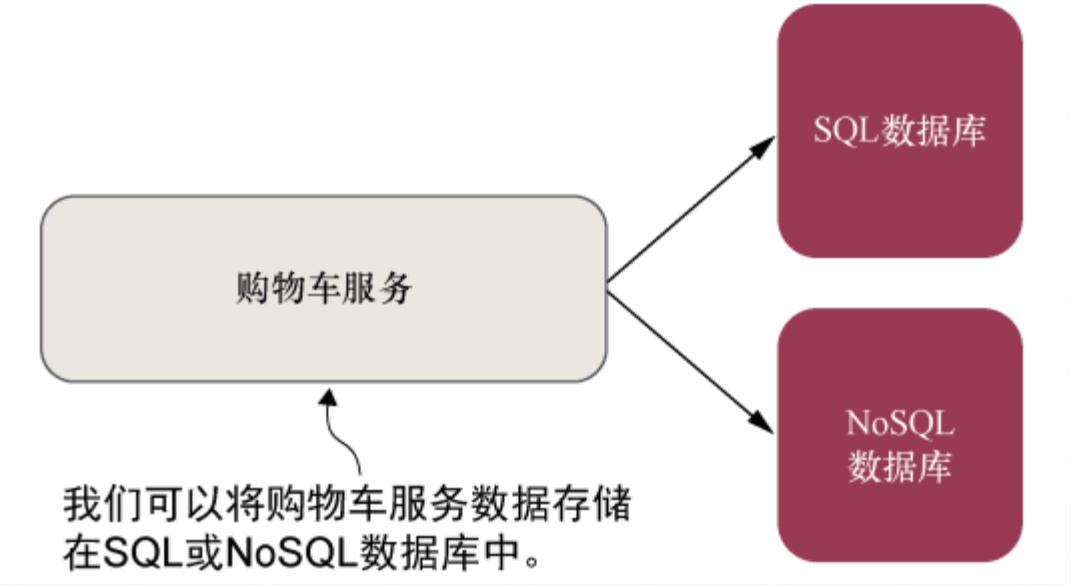
严格区分微服务的构建、发布和运行阶段。



微服务本身应该独立于外部运行环境，仅需调整配置参数，就能在不同的环境中运行。已构建服务是不可变的，不能更改。如果非要修改，需要重新构建并重新部署。

要素六：进程

微服务应该始终是无状态的，并且应该只包含执行请求的事务所必需的信息。微服务可以随时终止和替换，而不必担心服务实例的丢失会导致数据丢失。如果有存储状态的特定要求，则必须通过内存缓存（如 Redis）或后端数据库来完成。



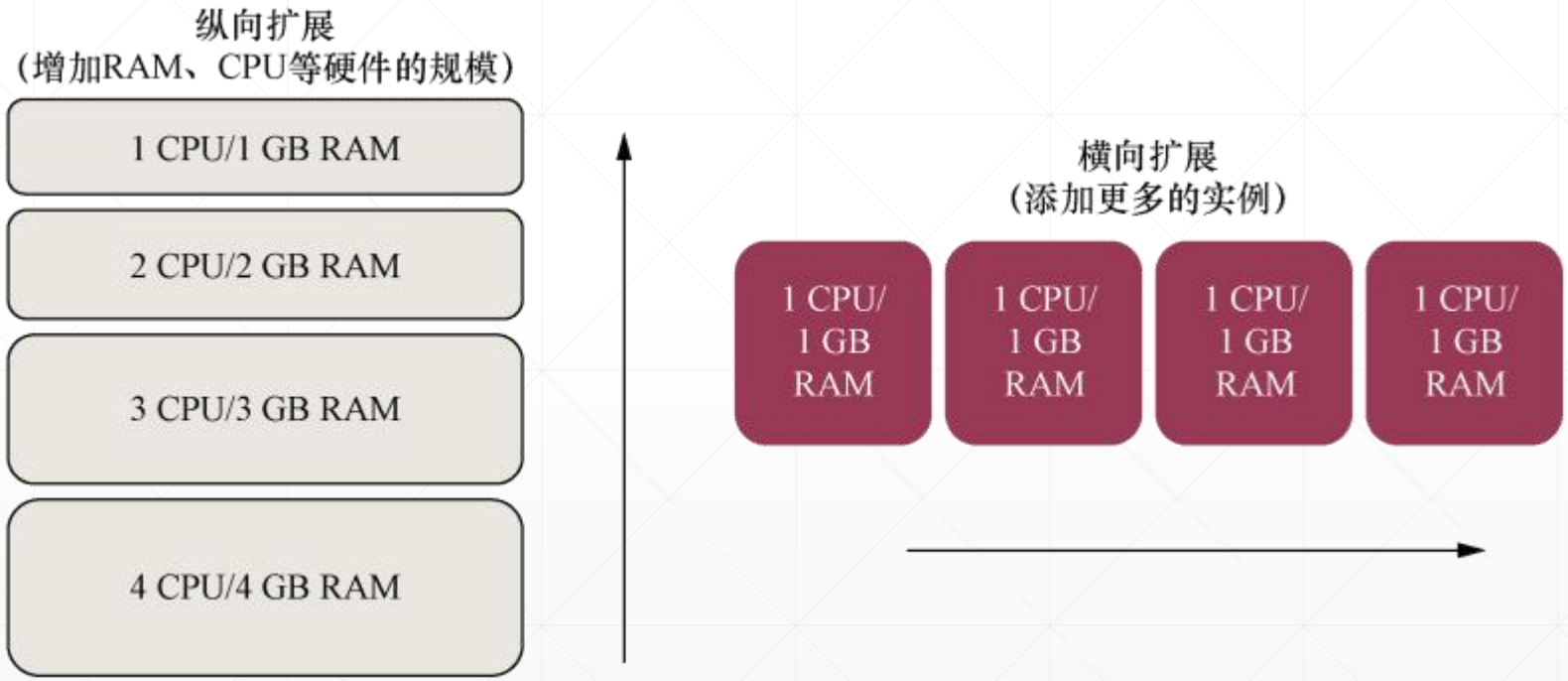
要素七：端口绑定

- 在微服务架构中，微服务在打包的时候应该是完全独立的，可运行的微服务中要包含一个运行时引擎。运行服务时不需要依赖于外部的Web或应用程序服务器。

- 服务应该在命令行上自行启动，并且可通过公开的HTTP端口立即访问。

要素八：并发

当系统负载大时，特定的微服务，应该可以通过运行更多的实例实现横向扩展。



为实现此目的，现在多使用容器（Docker）部署和运行微服务。

要素九：可任意处置

- 微服务可以根据需要随时启动和停止，当功能调整或配置更改时，能快速部署上线。
- 如果某个微服务的实例由于底层硬件或其他原因而出现故障，我们可以关闭它而不影响其他微服务，并且可以重新启动它的另一个实例。

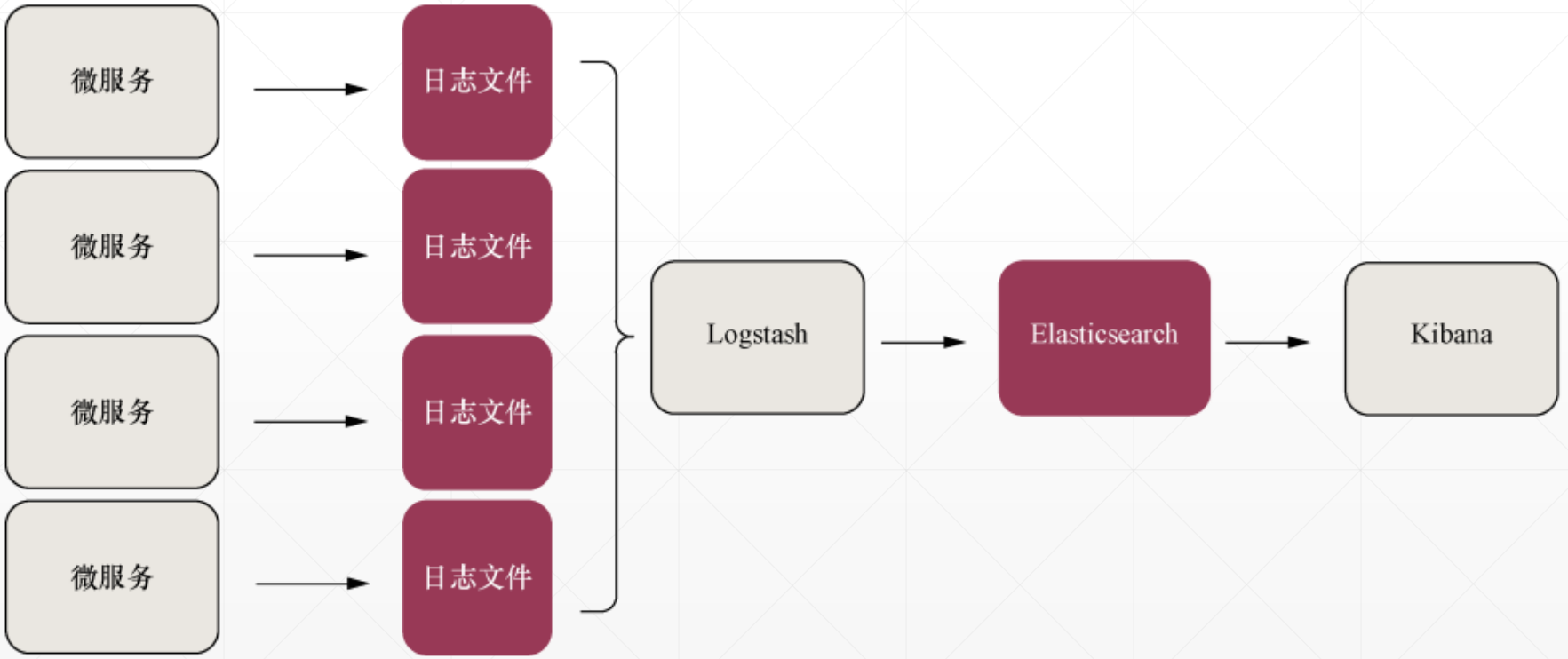
要素十：开发环境/生产环境等同

- 尽可能让不同的环境（如开发环境、交付准备环境、生产环境）保持相似。

- 代码一提交就应该被测试，然后尽快从开发环境推进到生产环境，这就要求开发团队应该配置一条“持续集成”和“自动部署”的软件“流水线”。

要素十一：日志

日志对于微服务管理至关重要。在写入日志时，应该通过专用工具来管理日志，这些工具收集日志并将它们写到一个集中位置。微服务永远不应该关心这是如何发生的，它只需要专注于将日志条目写入标准输出（stdout）。



要素十二：管理进程

- 微服务的管理，应该通过特定的脚本来完成，这些脚本保存于源代码仓库中。
- 这些脚本应该是可重复的，并且在每个运行的环境中都是不可变的（脚本代码不需要针对每个环境进行修改）。

小结



- 这十二个要素，只是一种建议和指南，不需要当成必须遵循的“法律”。
- 我们需要了解这十二个要素应用的场景，以及它们被提出的理由和原因，然后，再依据自己要面对的具体应用场景，分析与设计出自己的微服务技术解决方案。
- 注意不要犯“教条主义”错误，规则是死的，人是活的，一切都应该从实际出现，因地制宜，实事求是。