软件部署专题笔记报告

201900302030 邵嘉明

软件部署是整个软件工程的生命周期中必不可少的一环，在现有常见软件工程架构下，C/S模型下无论是客户端还是服务端的应用code都需要进行从开发环境到生产环境的迁移工作，而这一工作就是我们一般说的“部署”。

部署是一个从编码开始到真实用户在生产环境中与系统交互结束的过程。 如果这个过程是完全自动化的——也就是说，如果没有人为干预——那么它被称为持续部署。 如果该过程在将系统（部分）投入生产之前是自动化的，并且此最后步骤需要人工干预（可能是由于法规或政策），则该过程称为持续交付。

为了加快发布速度，我们需要引入部署管道的概念：工具和活动的序列，从您将代码签入版本控制系统开始，到您的应用程序部署完成以供用户发送请求时结束。在这些点之间，一系列工具集成并自动测试新提交的代码，测试集成代码的功能，并测试应用程序的负载性能、安全性和许可证合规性等问题。

部署管道中的每个阶段都发生在为支持阶段隔离和执行适合该阶段的操作而建立的环境中。主要环境如下：

代码是在单个模块的开发环境中开发的，并在其中进行独立的单元测试。 一旦通过测试并经过适当的审查，代码将提交给版本控制系统，该系统会触发集成环境中的构建活动。

集成环境构建用户服务的可执行版本。 持续集成服务器编译您的新代码或更改的代码，以及用户服务其他部分的最新兼容代码版本，并为用户服务构建一个可执行映像。 集成环境中的测试包括来自各种模块的单元测试（现在针对构建的系统运行）以及专门为整个系统设计的集成测试。 当各种测试通过后，构建的服务将被提升到暂存环境。

暂存环境测试整个系统的各种质量。这些包括性能测试、安全测试、许可证一致性检查，以及可能的用户测试。 对于嵌入式系统，这是物理环境模拟器（向系统提供合成输入）的地方。 通过所有暂存环境测试（可能包括现场测试）的应用程序被部署到生产环境，使用蓝/绿模型或滚动升级。 在某些情况下，部分部署用于质量控制或测试市场对提议的更改或提供的反应。

一旦进入生产环境，服务就会受到密切监控，直到各方对其质量有一定程度的信心。 在这一点上，它被认为是系统的正常部分，并与系统的其他部分一样受到关注。

在每个环境中执行一组不同的测试，将测试范围从开发环境中单个模块的单元测试扩展到集成环境中构成您的服务的所有组件的功能测试，并以广泛的质量测试结束在暂存环境和生产环境中的使用监控。

但并非一切都按计划进行。 如果开发者在软件进入其生产环境后发现问题，通常需要在解决缺陷时回滚到以前的版本。

架构选择会影响可部署性。 例如，通过采用微服务架构模式（参见第 5.6 节），负责微服务的每个团队都可以做出自己的技术选择； 这消除了以前在集成时发现的不兼容问题（例如，选择使用哪个版本的库不兼容）。 由于微服务是独立的服务，这样的选择不会造成问题。

同样，持续部署的心态迫使开发者在开发过程的早期考虑测试基础设施。 这是必要的，因为为持续部署而设计需要持续的自动化测试。 此外，能够回滚或禁用功能的需求导致了有关机制的架构决策，例如功能切换和接口的向后兼容性。 这些决定最好尽早做出。

有关于持续部署和自动化测试，这点我略有体会，此前我在工程实习时亦或是在独自完成项目时，就有遇到类似的场景，需要我去编写jest单元测试进行自动化提测，持续部署需要我预定义shell命令去实现，在本次我们组的软件工程大实验项目中，我也将会将我之前学到的自动化测试和持续部署方面的知识和用法用于我们的小组作业，打造更为自动化、智能化的软件工程开发体验。