同类型的自动化测试用例进行分层分级管理。分层是指按测试目标分层，如单元测试、集成测试和端到端的测试。分级是指按反馈速度和反馈质量进行分级，在资源不充裕的情况下，将各种测试用例放入不同的级别中（如提交构建任务和次级构建任务），并按线性方式顺序执行。

（4）测试验证环境的准备。有时测试环境的准备时间较长，耽误持续集成的时间，我们就需要将测试环境的准备工作进行梳理，减少人工参与，保证测试环境的准备时间。如果测试用例较多，我们可能需要同时准备多套测试环境，此时测试环境的自动化准备工作更是非常重要。

（5）优化编码规范扫描。之前我们可以使用了最少量的编译规范。随着实践的深入，团队可以逐步增加、修改和调整编码规范，使其符合团队代码质量的要求。

（6）生成数据报告。让每个人随时都能够方便地掌握当前的代码质量状态。

6．工程师改变习惯，并提升技能

持续集成是一个积极的团队协作实践，要求工程师主动提前集成，而不是推迟集成。这与人们“推迟风险”的心理相违背，需要工程师改变工作习惯，将大块任务尽可能拆分成细粒度的工作。同时，频繁运行的自动化测试套件依赖于自动化测试用例的稳定性与可靠性。因此，需要工程师投入一些时间来学习相关的方法与技巧。关于自动化测试的实施方式与编写要求参见第10章。

9.4.2　分支策略与部署流水线

细心的读者已经发现，持续集成实践中的提交构建和次级构建已经组成了一个简单的部署流水线。正如第1章所述，最早提出部署流水线的概念时，也是基于持续集成实践。但是，如果需要多个团队协作共同开发大型复杂软件（例如手机操作系统）时，就需要仔细设计团队之间的持续集成方式。

一般来说，团队间的持续集成方式与团队所采纳的代码分支策略密切有关。可以确定的是，每创建一条分支，都应该立即创建与其对应的部署流水线，直至该分支的生命周期结束（被弃用或者合并回去）。

1．“主干开发，主干发布”策略

如果软件产品只有一个代码仓库，并且团队采用“主干开发，主干发布”的方式，那么，团队只需要对该代码主干做持续集成。也就是说，开发该产品的团队只需要架设一个持续集成服务，关注代码主干的代码变更即可。GoCD团队在产品未正式发布以前，就是采用这种方式做持续集成，如图9-10所示。

2．“主干开发，分支发布”策略

如果软件产品只有一个代码仓库，并且团队采用“主干开发，分支发布”的方式，那么，团队每当准备发布时应该创建新的发布分支，并为这个发布分支创建对应的部署流水