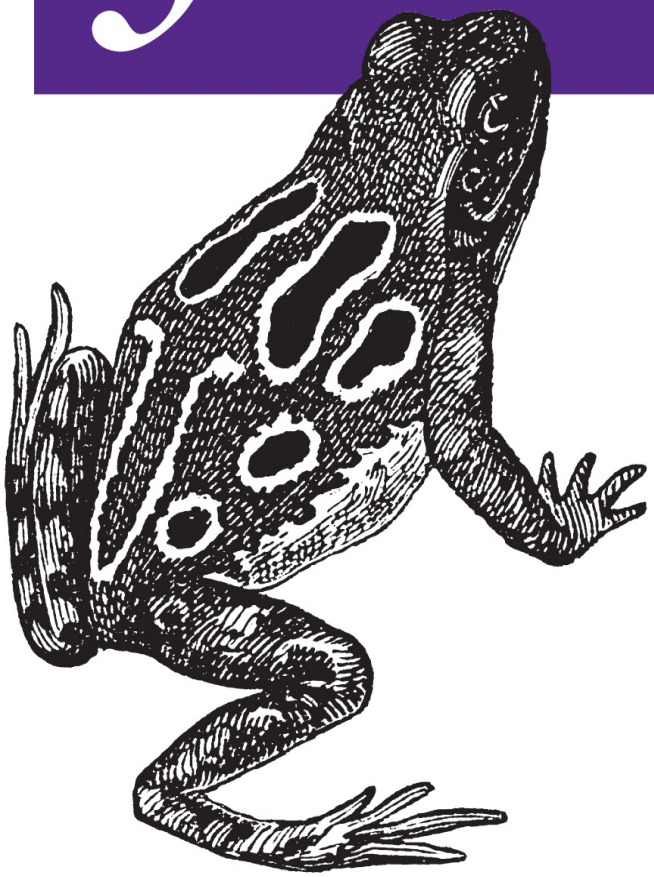
Continuous Integration for the Masses

Also Covers Hudson

Jenkins

The Definitive Guide



John Ferguson Smart

Foreword by Kohsuke Kawaguchi, creator of Jenkins/Hudson

O'REILLY®

Table of Contents

Jenkins权威指南

通过Jenkins可以实现流水线式的软件开发，Jenkins是一个开源的软件，他改变了团队对于Continuous Integration(CI)的认识。这本书将会教你如何使用Jenkins来使你的构建，集成，发布，开发流程自动化，证明了CI可以节省你的时间，金钱和解决你许多头疼的问题。

通过本书你可以：

- 学习如何安装，配置和使你的Jenkins服务器更加安全；
- 组织和监视general-purpose构建作业；
- 集成自动化测试来验证构建是否正确，构建代码质量报告；
- 建立有效的团队通知策略和其中所用到的技术；
- 配置构建流水线，参数化作业，矩阵式build和其他高级作业；
- 管理Jenkins服务器集群来运行分布式构建；
- 实现自动化部署和持续交付；

Jenkins介绍

简介

Continuous Integration(CI)是现代软件开发领域的基石，它改变了团队对于整个开发过程的理解。一个好的CI架构能够使得从开发到部署顺序进行，更快地发现和修复bug,最终给客户带来更多的价值。每个专业的开发团队，无论打还是小都应该采用CI。

Jenkins介绍

Jenkins来源于Hudson。在2009年，Oracle收购了Sun并继承了Hudson的基础代码。在2011年，Oracle和开源社区关系紧张，hudson分成了两个项目：

- Jenkins:被大多数的Hudson开发者所运营
- Hudson:被Oracle所控制

简介

Continuous Integration(CI)是现代软件开发领域的基石，它改变了团队对于整个开发过程的理解。一个好的CI架构能够使得从开发到部署顺序进行，更快地发现和修复bug,最终给客户带来更多的价值。每个专业的开发团队，无论打还是小都应该采用CI。

持续集成(CI)基础

在引入CI之前，开发团队的很大一部分精力都被用在集成阶段。在这个阶段中，每个独立的开发者或者每个团队将他们开发的各个模块集成为一个可以工作的产品。在这个过程中可能会出现几个月的冲突解决。并且各种问题在这个过程中都会出现而且非常难以解决。最终可能导致发布延迟和其它无法预计的损失。持续集成的出现解决了这些问题。

持续集成是一个监视你版本控制系统改变的软件。每当代码改变的时候，这个工具自动的编译和测试你的应用。如果出现了错误，这个工具立即通知开发者，因此开发者能够立即修复问题。而且它还能自动地监测代码质量和测试覆盖率。可视化的代码质量度量方案能够鼓励开发者不断地改进他们的代码。

持续部署（Continuous Deployment）是自动地讲每次成功的构建直接部署到生产环境中。

持续发布（Continuous Delivery）与持续部署（Continuous Deployment）有稍微的不同。持续发直接布的版本通过了所有的自动化测试和其他的质量检测手段，可以通过点击的方式完全自动化的部署到生产环境中，并且用户就可以直接使用了。然而，这个过程不是自动的，它是由业务决定最好的发布时间而不是由IT直接发布的。

持续集成需要整个团队的维护。例如：

- 自动化的不需要认为参与的构建过程；
- 出现失败的构建需要立即修复；
- 部署的过程需要自动化不需要人为参与；
- 代码需要由足够的测试来保证代码的质量；

Jenkins介绍

Jenkins起源于Hudson,占据了很大的市场份额，可被各种大小的团队和用不同语言(.NET,Ruby,Groovy,Grails,PHP等)开发的项目使用。优点如下：

- 易用性；
- 可扩展。插件覆盖了版本控制系统，构建工具，代码质量度量工具，通知工具，和其它外部系统进行集成，UI自定义等；
- 活跃的社区；
- 稳定版本的支持。Long-term Support(LTS)发布。

在团队中引入CI

阶段1-没有构建服务器

软件在开发者的机器上通过Ant或其它脚本手动构建，代码保存在中央源码仓库中，但是开发者不是经常提交本地的修改。每次需要发布的时候，开发者手动合并修改，这个过程是相当痛苦的。

阶段2-晚上进行构建

在这个阶段，团队有构建服务器，自动化的构建在晚上进行。构建过程只是简单的编译代码，没有可靠的和可重复的单元测试。然而，开发人员每天提交代码。如果某个开发人员提的代码和其他人的代码冲突的话，构建服务器会在第二天通过邮件通知团队。所以又一段时间构建是处于失败状态的。

阶段3-晚上进行构建并进行自动化测试

团队对CI和自动化测试越来越重视。无论什么时候版本管理系统中的代码改变了都会触发编译构建过程，团队成员可以看到是代码中的什么改变触发了这个构建。并且，构建脚本会编译应用并且会执行一系列的单元测试或集成测试。除了邮件，构建服务器还可以通过其他方式通知团队成员，如：IM。失败的构建被快速的修复。

阶段4-代码质量度量

自动化的代码质量和测试覆盖率的度量手段有助于评价代码的质量和测试的有效性。代码质量的构建会产生API文档。

阶段5-更加认真地对待测试

CI和测试紧密相关。如今，像测试驱动开发被广泛地使用，使得对自动化的构建更加有信心。应用不仅仅是简单地编译和测试，而是如果测试成功会被自动的部署到一个应用服务器上来进行更多的综合的end-to-end测试和性能测试。

阶段6-验收测试和更加自动化的部署

验收测试驱动的开发被使用，使得我们能够了解项目的状态。这些自动化的测试使用行为驱动的开发和测试驱动的开发工具来作为交流和文档工具，发布非开发人员也能读懂的测试结果报告。由于测试在开发的早起就已经被自动化的执行了，所以我们能更加清楚地了解到什么已经做了，什么还没有做。每当代码改变或晚上，应用被自动化地部署到测试环境中，以供QA团队测试。当测试通过之后，软件的一个版本将被手工部署到生产环境中，团队也可以在出现问题的时候回滚之前的发布。

阶段7-持续部署

对自动化的单元测试，集成测试和验收测试的信心使得我们可以使用自动化的部署技术将软件直接部署到生产环境中。但是测试还是有可能不能真正的反映现实的环境。

使用

开始使用Jenkins

环境准备

CI工具最基本的功能是监视版本控制系统中代码，并且当代码改变的时候拉去和构建最新的版本。

安装Java

安装完成之后，通过java -version检测是否安装正确。

安装Git

- Ubuntu或其它Debian系统：`sudo apt-get install git-core`
- Fedora或其它基于RPM的系统：`sudo yum install git-core`

创建GitHub账号

1. 配置SSH keys

```
ssh-keygen -t rsa -b 4096 -C "gwpost@qq.com"
sudo apt-get install xclip
xclip -sel clip < ~/.ssh/id_rsa.pub
```

2. fork示例代码仓库

`git clone git@github.com:gwpost/game-of-life.git`

自动化测试

如果你没有在CI中使用自动化测试，那么你将错过许多项目中重要的东西。没有自动化测试的CI只是用来进行自动化构建的一个工具。CI的一个重要的原则就是每一次buid都是应该是可信的，你必须客观地决定是否某个build能够继续之后的build过程。如果没有自动化测试，你必须进行手工测试，这在CI中是非常不可取的。

在你的应用中集成自动化测试可以使用以下两种方法：

- 测试驱动开发（TDD）和行为驱动开发（BDD）
- 单元测试

> Jenkins还可以做其它类型的自动化测试，如：integration tesing,web testing,functional testing,performance testing,load testing。

Jenkins也可以和其它技术结合，如：行为驱动开发，验收测试驱动开发。它们的集合可以作为项目开发者和其它利益相关者的交流工具。BDD的框架像easyb,fitnesse,jbehave,rspec,Cucumber,能够形成测试人员/用户等可以看得懂的验收测试报告。使用这些工具，Jenkins可以报告项目的进度和加快开发人员和非开发人员之间的交流。

对于现在已经存在的项目，采用重写单元测试的方法不太实际。可以采用回归测试。例如可以采用像Selenium或WebDriver之类的测试工具来进行web应用的测试

自动化单元和集成测试

Jenkins可以很好的展示你的测试结果，然而，你必须写出合适的单元测试和配置相应的脚本来使自动化能够正确运行。在Jenkins中集成单元测试工具使非常容易的，如今有很多的单元测试工具可以使用，其中xUnit家族是使用的比较多的。在JAVA领域，Junit是使用的比较多的，尽管TestNG是一个有很多的新特性的单元测试框架。

这些工具也可以在集成测试/功能测试/web测试中使用。许多web测试工具（Selenium，WebDriver，Watir）可以生成xUnit兼容的报告。

为了能够尽可能快的得到反馈，你的测试必须划分为几类，从运行最快的单元测试，之后运行集成测试，最后运行比较慢的功能或web测试。

配置测试报告

一旦构建过程产生了测试结果，你需要配置Jenkins来展示它们。

对于Maven项目，你需要做下面两步：

- 在构建中加入 `mvn test` ；
- 在确保执行了测试之后，需要告诉Jenkins测试报告的路径；

> 提示：对于非JAVA项目，需要安装 `xUnit插件` 。安装完成插件之后需要配置测试报告的路径。因为Jenkins需要将这些报告转化为Junit报告以便他们能在Jenkins中展示。

展示测试结果

Jenkins区分failed构建和unstable构建的区别：

- failed构建：表示测试失败或构建任务失败（比如：编译错误）
- unstable构建：是代码不满足质量要求（比如：代码覆盖率）。

运行时间过长的测试可能表示有性能问题。

代码覆盖率

代码覆盖率表示在测试中有多少代码被执行了，但是并不能表现代码的质量。

代码覆盖率分析需要很多CPU和内存资源会减慢构建作业，因为这个原因，我们需要在另外一个作业中执行代码覆盖率分析，并且在单元测试和集成测试执行成功之后再运行。

代码覆盖率插件：Cobertura and Emma or Clover

代码覆盖率分析需要三步：

1. 装饰代码来让代码可以通过计数器来记录每行代码的执行次数，将所有的数据存储在特定的文件里（如：cobertura.ser）；
2. 执行测试，产生一个数据文件其中包含每行的代码执行的次数；
3. 讲数据文件转化为更加稳定的格式，如：XML或HTML;