<https://blog.csdn.net/u013890660/article/details/21375345>

一个好的办法是每日构建（daily builds）。 每日构建意味着自动地，每天，完整地构建整个代码树、（译者按：“代码树”，原文为source tree，意思是将整个项目源代码的目录，子目录，文件的位置尽可能事先固定下来，这样在开发过程中各个模块间，各个文件间的相对位置都不会混乱。源代码树指的就是一个项目所有的已经组织好的代码文件。通常代码树应该用版本控制软件管理起来。虽然这个概念很基本，但是据我的观察，国内还是有软件公司在这方面做的不够好的，所以有必要解释一下。）

http://column.ibeifeng.com/wp-includes/js/tinymce/plugins/wordpress/img/trans.gif

自动地 － 因为你设定代码每天在固定的时间构建。在Unix环境下使用cron，在windows下使用“任务计划”。

每天 － 或者更频繁. 当然每天构建的次数越多越好啦。但是有时候构建次数还是有上限的，原因和版本控制有关系，等会儿我会谈到的。

完整地 －很可能你的代码有多个版本。多语言版本，多操作系统版本，或者高端低端版本。每日构建（daily build）需要构建所有这些版本。并且每个文件都需要从头编译，而不是使用编译器的不完美的增量编译功能。

以下是每日构建（daily build）能带来的好处：

1. 当一个bug被修正了，测试者可以很快得到最新的修正后的版本开始重新测试，以验证bug是否真正地被修复了。
2. 开发人员可以更加确定他们对代码做的修改不会破坏1024个操作系统上的任何一个版本。验证这一点不需要在他们的机器上安装OS/2（IBM公司生产的PC机操作系统）。
3. 那些每天将修改过的代码导入（check in）版本控制服务器的开发人员知道，他们对一个模块导入的修改不会拖别的开发人员的后腿。拖后腿的意思是，那些开发别的模块的程序员使用这个修改过的模块，出了问题，于是他们自己的模块也没有办法开发下去了。每日构建则不会有人拖后腿。如果把一个开发团队比作一台PC机，那么团队中的一个程序员对某个模块的修改导致其他人无法开发别的模块，相当于PC机发生了蓝屏。当一个程序员忘记把他（她）新建立的文件导入到repository（指版本控制服务器上的代码树）时，这种开发过程中的“蓝屏”会经常发生。因为在这个程序员自己的计算机上有这个文件，所以他（她）构建 这个程序没有问题。但是其他程序员是从版本控制服务器上导出（check out）代码的，由于服务器上没有这个文件，他们遇到了链接错误（link error），无法继续工作了。
4. 外部团队（例如市场销售部门，进行beta测试的一些客户）可以获得一个比较稳定的版本，这样对他们开展自己的工作比较有利。
5. 假如你将每日构建出的二进制文件（例如一个可执行程序，一个dll等等）存档管理，那么当你发现一个非常奇怪的，无法解决的bug时，你可以通过对这些文件进行二进制搜索（binary search）来确定什么时候这个bug第一次出现。如果有对代码进行了完善的版本控制，你也可以找出是谁在何时对代码进行的导入（check in）导致了这个bug。
6. 当开发者修正了测试者报告的一个错误时，如果测试者同时报告了发现错误时的构建的版本，开发人员可以直接在那个版本中测试是否bug真正被修复了。（译者按：测试者报告出现了一个bug，只是报告了一个错误症状，而错误的原因可能有多个，每个原因可能在不同的模块中。前文中的方法是为了避免只修正了一个模块中一个原因，别的模块由于在变动，于是掩盖了而不是修复了bug）

以下是如何进行每日构建（daily build）的具体步骤。你需要用最快的电脑建立一个每日构建服务器。写一个脚本，可以自动从版本控制服务器中导出（check out）完整的代码，（你当然使用版本控制，不是吗？），然后对代码从头开始进行构建（build），要构建所有的版本。如果你有一个安装打包程序，也要在脚本中自动运行。所有会卖给最终用户的东西都要包括在构建过程中。把构建出来的版本放在各自的的目录里，不同时间构建的相同版本也应该按日期整理好，不要相互覆盖。每天固定的时间运行这样的脚本。

1. 最关键的是所有这些步骤都应该由脚本自动化完成，从导出（check out）代码到将最终产品放在网上供用户下载（当然在开发阶段，产品是放在一台测试服务器上的）。要保证开发过程中的任何东西的任何记录是由文档记录的而不是由某个人的脑子来记录的，这是唯一的办法。否则你会碰到这样的情况，产品需要发布了，可是只有Shaniqua知道如何将产品打包的，可是他刚刚被巴士撞了。在Juno公司（本文作者工作过的公司之一），要进行每日构建，你唯一需要学会的东西就是双击每日构建服务器桌面上的一个快捷方式。
2. 如果你在发行程序的当天发现了一个小bug，没有问题。修正它，然后重新运行每日构建脚本，现在你可以安安心心的发行程序了。当然，黄金定律是每日构建脚本应该是把所有的事情从头做一遍，遵循这个定律就不会有什么问题。
3. 将你的编译器的警告参数设到最大（在微软的VC中是-W4 ; 在GCC中是-Wall），当遇到任何一个最微小的警告时就应该停下来。
4. 如果每日构建失败了，可能整个开发团队的工作会因此进行不下去。当务之急是立刻找出原因，使得每日构建能成功进行下去。某天也许你会一天运行好几次每日构建脚本的。
5. 每日构建一旦失败，应该自动地将失败用email通知整个团队。提取错误日志中的恰当部分并包括在email正文中也不是很难。每日构建脚本也可以将当前的状态报告整理成一个html文件，自动发布到一个所有人都可以访问的web服务器上，这样测试者可以很快知道那个版本的构建是成功的。
6. 当我在微软的excel团队中工作时，我们的一个有效办法是，谁导致每日构建（daily build）失败，他（她）就得负责照看当日的每日构建（译者按：微软通常每日构建都在半夜），直到有另一个人导致构建失败而接替他（她）。这样做当然可以使每个开发者都小心不要因为自己代码的错误破坏了构建，更重要的是团队中的每个人都可以学会每日构建（daily build）的原理。
7. 如果你的团队在同一个时区工作，在午饭时间进行每日构建（daily build）是个不错的主意。午饭前每个程序员导入（check in）代码，这样当程序员在吃饭时，构建系统在工作。当程序员吃完饭回来时，如果每日构建失败了，所有的人也都在，可以尽快找出失败的原因。当构建系统运作起来时，没有人再会担心别人导入（check in）代码会妨碍自己的工作了。.
8. 如果你的团队同时在两个时区工作，计划好每日构建（daily build）的时间使得一个时区的工作不会影响另一个时区的工作。在Juno公司，纽约程序员在晚上7：00导入（check in）到版本控制服务器。如果他们的导入导致构建失败，印度Hyderabad市（译者按：印度科技重镇）的程序员在纽约时间晚上8：00以后的工作几乎无法进行下去。我们每天进行两次每日构建（daily build），每次构建的时间都在两地的程序员回家之前，这下就没有问题了。