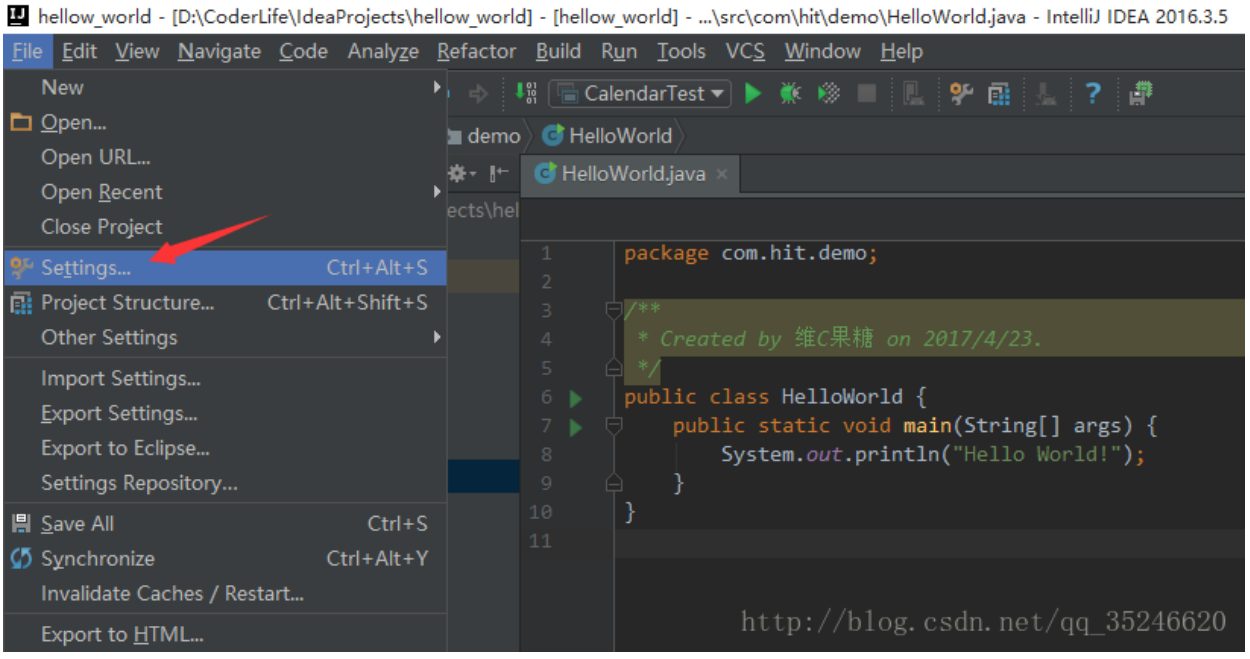
我们已经了解了很多关于 IntelliJ IDEA 的使用技巧，但是一个人进行项目开发更趋向于理想化，更多的则是团队协同开发，这时就需要了解一个非常重要的概念，那就是“**版本控制**”。

起初，并没有关于版本控制的概念，在协同开发的时候，大家都是自己保持项目代码，或者互相拷贝代码，这样在合并代码的过程中就难免遇到很多不兼容的问题；这就促使“集中式版本控制系统（CVCS）”的出现，例如 SVN、CVS 等，但这仍然有一个风险，那就是如果源码库出现问题，导致项目代码丢失，那么大家手里的都是部分代码，就算勉强合并到一起，也不能保证项目源码的准确性；因此，这又促使“分布式版本控制系统（DVCS）”的出现，例如 Git，它的好处显而易见，每个人从源码库中检出的代码，都是作为一份独立的、完整的拷贝代码存在，这时就算源码库出现问题，甚至源码丢失，那么任何一个人的代码都可以作为源码进行共享，从而大大提高了协同开发的抗风险能力。

因此，本文更倾向于推荐大家使用分布式版本控制系统。不过在一般情况下，仅需要下载一个版本控制系统的客户端即可，在这里，根据操作系统分别推荐一个非常好用的版本控制系统客户端：

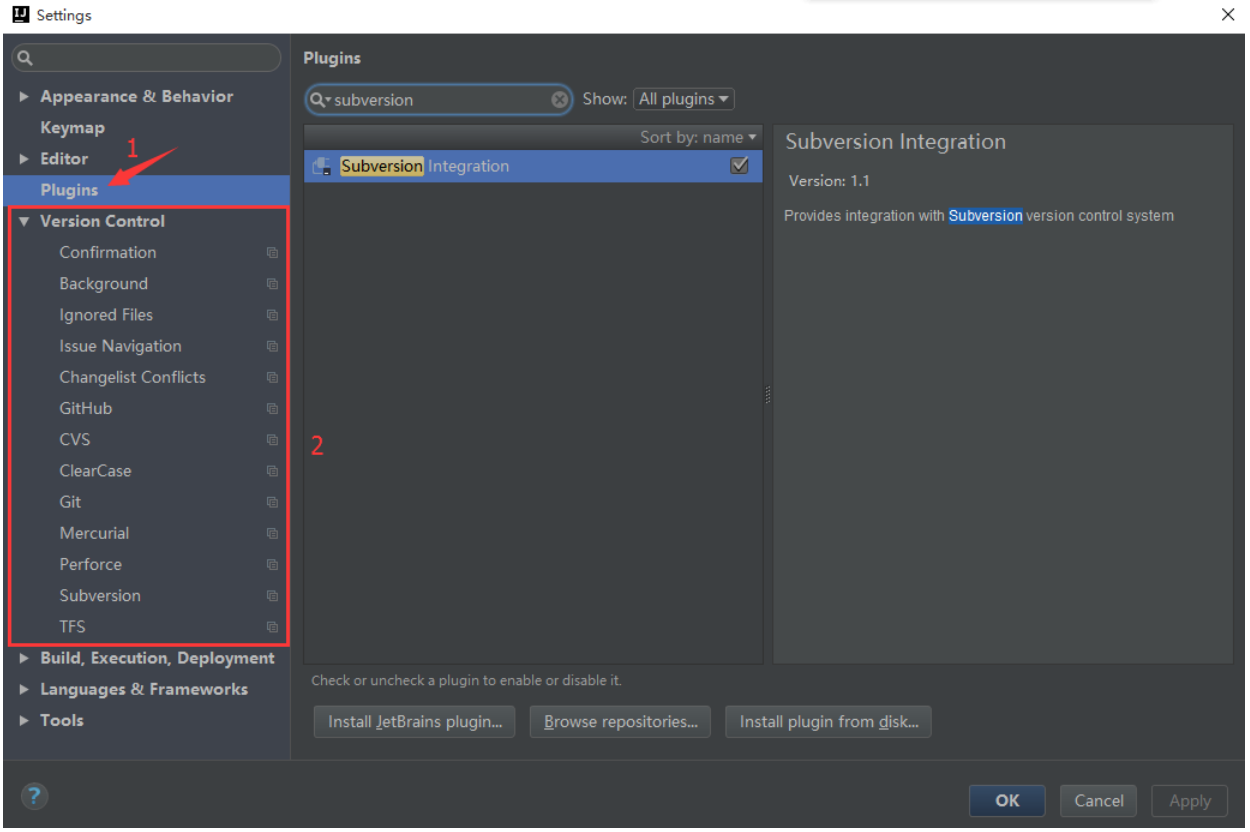
- Windows 版本控制系统客户端：**TortoiseSVN**；
- Mac 版本控制系统客户端：**CornerStone**.

接下来，咱们就进入主题，正式开始介绍 IntelliJ IDEA 中的版本控制机制：



1

如上图所示，点击Settings，进行如下界面：




2

- 标注1：Plugins，插件；
- 标注2：Version Control，版本控制。

如上图所示，标记出了“插件”和“版本控制”两个选项。有些人可能会认为 IntelliJ IDEA 自带了 SVN 或者 Git 等版本控制系统，因此只要安装了 IntelliJ IDEA 就可以使用版本控制系统的所有功能啦，这显然是一个错误的想法。IntelliJ IDEA 只是自带了对这些版本控制系统的支持插件，但是咱们想使用什么版本控制系统仍然得安装什么版本控制系统的客户端，否则照样用不了。

如上图中 **标注1** 所示，IntelliJ IDEA 对版本控制的支持都是以插件的方式来实现的。旗舰版默认支持目前主流的版本控制软件包括：GitHub、CVS、ClearCase、Git、、Mercurial、Perforce、Subversion（SVN） 和 TFS 等。