1、开源开发人员角色定位和分工；

模块设计师、需求分析师、数据库设计师、架构师、系统分析师、项目组长、项目经理、软件开发工程师、美工、需求调研员、系统分析员

2、创建和开展开源项目需要注意的问题有哪些；

为何追踪并审查代码

简单来说，如果您的公司没有追踪开发人员如何使用开源代码的方式和地点，那么您将面临不遵守适用开源许可证的风险——无论是在法律费用和修正错误所花费的工程时间两个方面都是一种昂贵的途径。忽视开源法律义务也会影响贵公司在开源社区的声誉。

开源项目办公室帮助集中制定围绕开源消费、分发和发布来集中制定方针和政策，追踪代码来源及其使用情况，并确保组织不违反其合规义务。

理想情况下，开源项目包含一个在法律顾问的帮助下开发的完整的合规项目。在本指南中，我们将介绍合规项目的一个重要方面：您关于使用、发布和分发开源代码的方针与流程。

“一个精心设计的开源合规流程应同时确保遵守开源许可证条款，并帮助企业保护自己的知识产权，以及第三方供应商免受意外披露和/或其他后果的影响。”

企业可以通过维护开源合规项目获得几点好处：

获得技术优势——因为兼容的软件组合更易于服务、测试、升级和维护。

识别开源代码部分——将发现在多个产品和组织的某些部分使用了哪些代码，且/或对开源战略是具有高度战略价值和益处。

说明与使用开源组件相关的成本与风险——当代码经过多轮审查时，这将是显而易见的。

建立社区信任——如果发生合规性挑战，这样的项目可以展示一个正在进行的善意行为模式。

为适合收购、销售或者新产品或服务的发布做好准备——这是一种罕见的好处，但在完成此类交易之前，合规性保证是强制性的。

建立供应链可信度——可以提高整个软件供应链的合规性，包括提升原始设备制造商（OEM）和下游供应商的合规性。

合规角色与责任

在开源计划中，需要创建一个特定的开源合规团队，该团队的任务是确保开源的合规性。

这个核心团队，通常被称为 审计团队或开源审查委员会(Open Source Review Board)（OSRB），由来自工程团队和产品团队的代表，一名或多名法律顾问以及 合规人员(Compliance Officer)组成（合规人员通常是开源项目经理）。

各个部门中的其他人员也为开源合规工作提供源源不断的基础：文件、供应链、企业开发、IT、本地化和监督整个开源战略的 开源执行委员会(Open Source Executive Committee)（OSEC）。但与核心团队不同，这些扩展团队的成员只是根据从开源审查委员会（OSRB）收到的任务，在兼职的基础上开展工作。

开源审查委员会（OSRB）负责创建开源合规战略和一套决定企业如何在日常基础上实施这些规则的流程。该战略确立了必须采取的措施来保证合规性，并为员工如何与开源软件进行互动提供了一套主要原则。它包括了开源的审批、获取与使用的正式流程，以及发布开源软件或经开源许可证授权的软件。

使用开源代码的一个简单方针

使用办法是所有合规项目的重要组成部分。这套规则包含在您的开源战略文件中（您有一份开源战略文件，对吗？），并提供给所有人以便参考。

使用办法确保任何成为产品基础的（专有的、第三方的或开源的）软件都已经过审核、审查与审批。该办法还确保在产品送达客户之前，贵公司已经制定了一个计划来履行因使用各种软件组件所产生的许可证义务。

无需制作一份冗长或复杂的文件。一个优秀的开源使用办法包括六个简单规则：

在将开源代码整合到产品中之前，工程师必须获得开源审查委员会（OSRB）的许可。

从第三方接收的软件必须经过审核，以识别其包含的所有开源代码，这样可以确保在产品发货之前许可证的义务得以履行。

所有软件都必须经过审核和审查，包括所有的专有软件组件。

产品必须在客户收货前履行开源许可证义务。

即使开源组件是一样的，对于在一个产品中使用给定的开源组件的许可也不等于其他部署许可。

任何变更的组件都必须经过审批流程。

代码审查过程的五个阶段

一旦制定办法，就必须计划并创建一个更易于应用办法中规定的流程。您的工作是帮助开发人员顺利地进行开源应用并为开源项目做贡献。

“如果您的代码审查过程过于繁琐，您将会放慢创新的进程，或是为开发人员完全规避流程提供好借口。“

3、开源开发为什么要进行版本控制，如何进行版本控制；

如果说什么是软件开发项目一定要使用的基础工具，那么版本控制系统应该算最重要的部分。不管是个人开发或是团队协作开发，都可以通过版本控制系统获得巨大的好处。

　　没有版本控制系统的话，代码可能被别人或自己不小心覆盖或遗失、也不知道是谁因为什么原因改了这段代码、也没办法可以复原回前几天的修改。有了版本控制系统，开发人员只要将每次程式码的变更都纪录（Commit）起来，并且透过版本控制系统中进行更新。

　　有了版本控制系统，我们可以浏览所有开发的历史纪录，掌握团队的开发进度，而且作任何修改都不再害怕，因为你可以轻易的复原回之前正常的版本。我们也可以透过分支和标签的功能来进行软件发行的不同版本，例如稳定版本、维护版本和开发中版本。

　　很多项目需求方还没有明白开发的定义，这里必须要跟大家说一点老生常谈的段子：“开发永远是个过程，而不是结果。”所以开发者一定要使用版本控制系统，Git或Mercurial是免费开源的版本系统系统、随处可用的网络、便宜的云端服务器，甚至有现成的第三方服务Github。

　　如果你还没有使用的话，建议马上为你的软件开发项目建立版本控制。接下来是几点使用版本控制系统的建议：

　　1.将所有东西都放进版本控制系统

　　是的，所有项目开发过程中的产出物都放到版本控制系统之中，这包括了程序源代码、测试程序、文件、设定档、各种自动化脚本等等。除了新成员可以很容易拉出最新的版本马上开始工作之外，我们也希望在测试环境、正式环境中，也可以随时更新到我们所指定的版本，因此将所有变更的纪录保存起来是非常重要的。

　　例如，数据库的变更也必须纳入版本控制。首先，在数据库中纪录它目前的版本编号。接着我们每次的修改都透过一个自动化脚本来执行，并将这个脚本放入版本控制之中，而不是手动用指令去修改纲要。这样的好处是团队中每个人都可以透过版本控制系统看到这个变更，并且升级他的数据库。测试和正式的服务器环境，也会透过这个脚本来自动进行升级。笔者熟悉的Ruby on Rails中就有内置这样的Migration机制，其他程序语言也有类似的数据库工具。

　　另外，以服务器应用来说，光是有源代码还是无法100%让软件工作起来，我们还需要知道服务器的配置设定，包括基础网络设定、操作系统设定、依赖的套件版本等等。因此最好这些配置设定也可以纳入版本管理系统之中。近年来云端技术的进步，已经可以将这些基础设施设定当作程序，无缝地让每个成员和所有环境都使用完全相同的设定，减少出错的可能性。

　　2.频繁且适当大小的递交

　　如果久久才递交一次修改到版本控制系统，那么你只是把版本控制系统当作一种备份工具而已，而没有享受到它真正的好处。频繁的递交可以帮助团队开发进度的透明化，减少多人开发时的代码冲突。当多人同时修改同一块代码时，解决代码冲突是很麻烦的事情。还有，我们也希望每一次的递交有适当的粒度大小，也就是每个提交的内容应该有高度相关性和独立性。例如是一个小功能或是一个小改进。如果你同时在做新功能A和修旧Bug，那么就应该分开两次递交。语法错误无法建构的程序也不应该提交从而造成团队困扰。

　　有良好大小的代码提交习惯的好处除了版本的历史纪录更加清楚之外，我们可以非常轻易的做代码的复原或移植，假设上述的新功能A有问题，我们可以只复原A而不影响修好的Bug，或是只挑选修Bug的移植到不同分支去。

　　3.良好的递交信息

　　每一次的递交程序员都必须附上一段解释信息，说明修改的内容和原因。这除了可以帮助团队合作之外，更重要的是让软件有更好的维护性，以便将来备查，以下的场景相信开发者都不陌生：

　　软件发现一个Bug，然后指派给你修复。

　　你追代码追到一段看不懂的程序，也没有任何注释。

　　透过版本控制系统，你找到了同事在一年前加了这行，递交的信息是BUG或简单的错误提示。

　　同事还在公司，可是他也不记得当初是哪一个BUG了。或是他已经下班或离职了，反正找不到。

　　你强行改了这行代码，发布出去。

　　这个功能是修好了，但是发现另一个功能又出现问题。

　　继续加班到凌晨，悲催ing....

一个好的递交信息应该包括一行摘要信息，描述你为什么做这段变更，可能是新增、移除、修正某个功能，而不是描述新增或修改哪些档案，重点应放在备注为什么修改而不是这段是bug这么简单。因为修改了哪些档案和行数我们看版本差异就知道了，无须重复描述。如果你发现很难摘要，那可能表示你包含太多变更在同一次递交了，请试着拆开。

4、https://github.com/A1ansmorn