欢迎贡献！

我们欢迎以各种形式来为 Hyperledger 做贡献，这里有很多事可以做！

参与之前，请先回顾一下 行为准则 。保证文明合法是很重要的。

注解

如果你想贡献此文档，请查看 贡献者文档格式 。

贡献的方法

不管作为用户还是开发者，这里都有很多为 Hyperledger Fabric 做贡献的方法。

作为一个用户：

- 提出功能或改进的建议
- 报告错误
- 帮助测试在 发布路线 上即将发布的史诗（Epic）。通过 Jira 或者 RocketChat 联系分配到史诗的人。

作为一个作者或者文档编写者：

- 运用你在Fabric的经验更新文档，改进已存在的话题，创建新的话题。修改文档是开始成为贡献者的捷径， 让其它用户更好的理解Fabric，和增加您开源贡献历史的好方法。
- 参与一个您选择语言的翻译以保持Fabric文档的更新。Fabric文档支持许多种语言，英语、中文、 马拉雅拉姆语和巴西葡萄牙语–为什么不加入一个团队让你喜欢的文档保持最新的版本呢？你能 发现一个用户，作者和开发者协同合作的友好社区。
- 开始翻译Fabric文档还未支持的一门语言。中国团队、马拉雅姆团队、使用葡萄牙语的巴西团 队都是这样开始的，你也可以！ 这需要更多工作，你需要建立一个作者社区，组织投稿；但是 真的很高兴见到Fabric文档支持您的语言。

跳转至`贡献文档`_开始你的旅途。

作为一个开发者：

- 如果你的时间不多，可以考虑选择一些 “help-wanted” （需要帮助的）任务，参考 修复问题和认领正在进行的故事 （Story）。
- 如果你是全职开发，可以提出一个新的特性（参考 提出功能或改进的建议 ）并带领一个团队来实现它，或者加入已存在的史诗的团队。如果你在 发布路线 发现了一个你感兴趣的史诗，请及时通过 Jira 或者 RocketChat 联系分配到任务的人，和他们一起完成这个史诗。

获取 Linux Foundation 账号

为了参与到 Hyperledger Fabric 项目开发中来，你首先需要一个 Linux 基金会账号。然后你要使用你的 LF ID 来访问所有 Hyperledger 社区的工具，包括 Jira issue 管理， RocketChat, 和 Wiki （仅用于编辑）。

按照如下步骤创建 Linux 基金会账号，如果你没有的话。

1. 前往 Linux Foundation ID 网站 。
2. 在出现的表单中选择 `I need to create a Linux Foundation ID` 。
3. 等几分钟，然后查看带有如下主题的邮件：”Validate your Linux Foundation ID email”。
4. 打开接收到的 URL 来验证你的邮箱地址。
5. 确定你的浏览器显示了如下信息 `You have successfully validated your e-mail address` 。
6. 访问 Jira issue 管理，或 RocketChat。

贡献文档

将文档编写作为您的第一次变更是一个好主意。它很快也很容易，能帮助您检查机器是否配置正确（包括需预装的软件）， 也能让您熟悉贡献的流程。下述主题能帮助您开始：

- 修改文档
- 国际语言
- 贡献者文档格式

项目治理

正如我们的 章程 中所说，Hyperledger Fabric 是在一个开放治理的模型下管理的。项目和子项目由维护者主导。新的子项目可以指定一组初始的维护人员，这些维护人员将在项目首次批准时由顶级项目的现有维护人员批准。

维护者

Fabric 由项目的顶级 维护者 领导。维护者负责评审和合并提交评审的所有补丁，并在超级账本技术委员会的方针下指导项目的技术发展路线。

成为一名维护者

项目的维护者会时不时地根据以下标准来考虑添加或者删除维护者：

- 提供（足质足量的）PR记录
- 在项目中表现出足够的领导能力
- 展现出引导项目工作和贡献者的能力

现有的维护者可以提交变更到 维护者 文件中。一个提名的维护者可以由大多数现有的维护者批准通过成为正式的维护者。一旦批准通过，变更就会被合并同时该维护者也会被添加到维护者组中。

维护者可能会因为明确的辞职、长时间的不活动（超过3个月或者更长的时间），或者因为违反相关的行为准则 或持续表现出糟糕的判断而被移出维护者的队列。移除维护者的提案也需要大多数人同意。在

恢复贡献和评审（一个月或更长时间）之后，应恢复因不活动而被移除的维护人员。

发布节奏

Fabric 的维护者已经确定了每个季度大致的发布节奏（请参考 发布说明）。我们会不定时创建稳定的 LTS（long term support，长期支持版本）release 分支，同时 master 分支会持续增加新的特性。我们会在 RocketChat 的 #fabric-release 频道讨论这些内容。

提出功能或改进的建议

首先，请查看 JIRA 确保之前没有开启或者关闭的相同功能的提案。如果没有，为了开启一个提案，我们建议创建一个 Jira 的史诗（Epic）或者故事（Story），选择一个最合适的环境，并附上一个链接或者内嵌一个提案的页面，说明这个特性是做什么的，如果可能的话，描述一下它应该如何实现。这有助于说明为什么应该添加这个特性，例如确定需要该特性的特定用例，以及如果实现该特性的好处。一旦 Jira 的 issue 被创建了，并且描述中添加了附加的或者内嵌的页面或者一个公开的可访问的文档链接，就可以向 fabric@lists.hyperledger.org 邮件列表发送介绍性的电子邮件，邮件中附上 Jira issue 的链接，并等待反馈。

对建议性的特性的讨论应该在 JIRA issue 本身中进行，这样我们就可以在社区中有一个统一的方式来找到这个设计的讨论。

获得3个或者更多的维护者对新特性的支持将会大大提高该特性相关的变更申请被合并到下一次发布的可能性。

维护者会议

维护者会定期举行维护者会议。会议的目的是计划和审核发布进程，以及讨论项目和子项目的技术和维护方向。

维护者会议详情请查看 wiki 。

如上所述新特性或增强的建议应该在维护者的会议上进行探讨、反馈和接受。

发布路线

Fabric 相关的发布路线的史诗维护在 JIRA 上。

交流

我们使用 RocketChat 来进行交流或者使用 Google Hangouts™ 进行屏幕分享。我们的开发计划和优先级在 JIRA 上进行发布，同时我们也花大量的时间在 mailing list 上进行讨论才做决定。

贡献指南

安装准备

在我们开始之前，如果你还没有这样做那你可能需要检查一下您是否已经在将要开发区块链应用或者运行 Hyperledger Fabric 的平台上安装了运行所需的环境。

获得帮助

如果你试图寻找一种途径来寻找专家援助或者解决一些问题，我们的 社区 会为您提供帮助。我们在 Chat ，IRC（#hyperledger on freenode.net）以及 mailing lists 中都可以找到。我们大多数人都很乐意提供帮助。唯一愚蠢的是你不去问。问题实际上是帮助改进项目很好的方法，因为它们使我们的文档更加清晰。

报告错误

如果你是一个用户，并且发现了错误，请使用 JIRA 来提交问题。在您创建新的 JIRA 问题之前，请尝试搜索是否有人已经提过类似的问题，确保之前没有人报告过。如果之前有人报告过，那么你可以添加评论表明你也期望这个问题被修复。

注解

如果缺陷与安全相关，请遵循 Hyperledger 安全问题处理流程 。

如果以前没有报告过，你可以提交一个 PR 并在提交信息里描述问题和修复措施，或者你也可以创建一个新的 JIRA。请尝试为其他人提供足够多的信息以重现该问题。该项目的维护人员会在24小时之内回复您的问题。如果没有，请通过评论提出问题，并要求对其进行评审。您还可以在 Hyperledger Chat 中将问题发布到相关的 Hyperledger Fabric 频道中。比如，可以将文档问题在 `#fabric-documentation` 中进行广播，数据存储问题可以在 `#fabric-ledger` 中广播，以此类推。

提交你的修复

如果你在 JIRA 上提交了你发现的问题，并希望修复它，我们很乐意并且非常欢迎。请将 JIRA 问题分配给自己，然后您可以提交 PR。详细流程请参考 GitHub Contributions。

修复问题和认领正在进行的故事

查看 问题列表 找到你感兴趣的内容。您也可以从 “求助” 列表中寻找。明智的做法是从相对直接和可实现的任务开始，并且这个任务是未被分配的。如果没有分配给别人，请将问题分配给自己。如果你无法在合理的时间内完成，请加以考虑并且取消认领，如果你需要更多的时间，请添加评论加以说明，你正在积极处理问题。

审核提交的 PR

另一种贡献和了解 Hyperledger Fabric 的方法是帮助维护人员审查开放的 PR。实际上维护者的工作很辛苦，他们需要审查所有正在提交的 PR 并且评估他们是否应该被合并。您可以查看代码或者修改文档，测试更改的内容，并告知提交者和维护者您的想法。完成审核或测试后，只需要添加评论和投票，即可完成回复 PR。评论“我在系统 X 上尝试过这个 PR，是正确的”或者“我在系统 X 上运行这个 PR 发现了一些错误”将帮助维护者进行评估。因此，维护人员也能够更快地处理 PR，并且每个人都能从中获益。

从浏览 GitHub 上开放的 PR 开始你的贡献。

PR 过期

随着 Fabric 项目的壮大，开放的 PR 积压也开始增多。几乎所有项目都面临的一个问题是如何高效的管理积压，Fabric 也不例外。为了便于 Fabric 相关项目积压的 PR 的管理，我们将介绍由机器人（bots）程序强制执行的过期策略。这和其他大型项目管理他们的 PR 积压是一样的。

PR 过期策略

Fabric 项目管理员将自动监控所有拖延着的活动的 PR。如果一个 PR 两周没有更新，就会为该 PR 增加一条表示提醒的评论，要求更新 PR 以完成任何未完成的评论，或者在放弃撤销 PR。如果拖延的 PR 继续两周没有更新，它就会被自动取消。如果一个 PR 从最初提交起已经超过两个月了，尽管还是活跃状态，它也将被标记为供维护者审核。

如果一个提交的 PR 通过了所有的检查但是超过 72 小时（3天）还没有被审核，它将会每天被推送到 #fabric-pr-review 频道，直到收到一个审核评论。

该策略适用于所有 Fabric 官方项目（fabric、fabric-ca、fabric-samples、fabric-test、fabric-sdk-node、fabric-sdk-java、fabric-gateway-java、fabric-chaincode-node、fabric-chaincode-java、fabric-chaincode-evm、fabric-baseimage 和 fabric-amcl）。

设置开发环境

接下来，在本地开发环境中 构建项目 ，以确保所有配置都是正确的。

什么是好的 PR?

- 一次只包含一个变更。不是5个，3个，或者10个。仅仅一个变更。为什么呢？因为它限定了问题的范围。如果我们有一个回归，相对影响了很多代码的复杂更改，我们更容易识别错误的提交。
- 在 JIRA 的故事中包含一个链接。为什么？因为 a)我们希望追踪你的进度以便更好地判断我们可以传递什么信息。 b) 因为我们可以证明这次变更是有效的。在很多情况下，会有很多围绕提交变更的讨论，我们希望将它链接到它的本身。
- 每次变更都包含单元或者集成测试（或者对已有测试的修改）。不仅仅是正确的测试。同样也要包括一些异常测试来捕获错误。在你写代码的时候，你有责任去测试它并且证明你的变更是正确的。为什么呢？因为没有这些，我们无法知道你的代码是否真的正确地工作。
- 单元测试不要有额外的依赖。你应该使用 `go test` 或者等价的语言测试方式来运行单元测试。任何需要额外依赖的测试（例如需要用脚本来运行另一个组件）需要适当的 mocking。任何除了单元测试以外的测试根据定义都是集成测试。为什么？因为很多开源软件开发者都实用测试驱动的开发方式。他们关注一个目录下的测试用例，一旦代码变更了他们采用测试去判断他们的代码是否正确。相比当代码变更后运行整个项目来说，这是非常高效的。请参考 单元测试的定义 在脑海中建立单元测试的标准，以此来写出高效的单元测试。
- 最小化每个 PR 的代码行数。为什么？因为维护者每天同样也有工作。如果你发送1000或者2000行代码，你认为维护者需要多久才能审查完你的代码？尽可能地保证你的变更在200-300行左右。如果你有一个比较大的变更，可以将它分解为比较小的几个独立的变更。如果要添加一组新功能来满足一个需求，请在测试中分别添加它们，然后编写满足需求的代码。当然，总会有意外。如果你增加一些

小变动然后添加了300行测试，你将会被宽恕;-)如果你需要做一个变更，而且影响比较广或者生成了很多代码（protobufs 等）。同样，也是个例外。

注解

大的 PR，例如那些大于300行的 CR 将有可能不会通过，并且你可能被要求重构以符合本指南。

- 写一个有意义的提交信息。包括55个或者更少字符的标题，后面跟一行空行，然后跟上更全面的关于变更的描述。每个变更必须包括对应的变更的 JIRA 编号（例如[FAB-1234]）。这个可以在标题中，但是同样需要包括在消息正文中。

注解

示例提交信息:

```
[FAB-1234] fix foobar() panic

Fix [FAB-1234] added a check to ensure that when foobar(foo string)
is called, that there is a non-empty string argument.
```

最后，要有回应。不要让一个 PR 因为评审意见而不了了之，这样会导致它需要进行 rebase。这只会进一步延迟合并，给你带来更多的工作，比如修复合并冲突。

法律材料

Note: 每一个源文件必须包括 Apache Software License 2.0。可以参考 license header.

我们尽可能努力让贡献变得简单。这个协议为我们提供了贡献相关的法律相关的知识。我们使用和 Linux® Kernel 社区 一样的管理贡献的方法 Developer's Certificate of Origin 1.1 (DCO) 来管理 Hyperledger Fabric。

我们只要求在提交要审查的补丁时，开发者在 commit 消息中带上他们的离线签名即可。

这里是一个离线签名的签名例子，表示提交者接受 DCO 约定：

```
Signed-off-by: John Doe <john.doe@example.com>
```

你可以使用 `git commit -s` 在提交的时候自动带上你的签名。

相关主题

- Using Jira to understand current work items
- Setting up the development environment
- Building Hyperledger Fabric

- Configuration
- Configuration
- Coding guidelines
- Adding or updating Go packages
- Generating gRPC code
- Adding or updating Go packages