Hyperledger Fabric

Contributions Welcome!

欢迎贡献！

We welcome contributions to Hyperledger in many forms, and there’s always plenty to do!

我们欢迎以多种形式向Hyperledger贡献，而且总是有很多事情要做！

First things first, please review the Hyperledger Code of Conduct before participating. It is important that we keep things civil.

首先，请在参与前查看“超级账本行为准则”。重要的是我们要保持文明。

一、Project Governance

一、项目治理

Hyperledger Fabric is managed under an open governance model as described in our charter. Projects and sub-projects are lead by a set of maintainers. New sub-projects can designate an initial set of maintainers that will be approved by the top-level project’s existing maintainers when the project is first approved.

如我们的章程所述，Hyperledger结构是在开放式管理模式下管理的。项目和子项目由一组维护人员领导。新的子项目可以指定一组初始的维护人员，当项目第一次获得批准时，这些维护人员将由顶级项目的现有维护人员批准。

1、Maintainers

1、维护者

The Fabric project is lead by the project’s top level maintainers. The maintainers are responsible for reviewing and merging all patches submitted for review, and they guide the overall technical direction of the project within the guidelines established by the Hyperledger Technical Steering Committee (TSC).

织物项目由项目的顶级维护人员领导。维护人员负责审查和合并提交审查的所有补丁程序，并在Hyperledger技术指导委员会（TSC）制定的指导方针范围内指导项目的总体技术方向。

2、Becoming a maintainer

2、成为维护者

The project’s maintainers will, from time-to-time, consider adding or removing a maintainer. An existing maintainer can submit a change set to the MAINTAINERS.rst file. A nominated Contributor may become a Maintainer by a majority approval of the proposal by the existing Maintainers. Once approved, the change set is then merged and the individual is added to (or alternatively, removed from) the maintainers group. Maintainers may be removed by explicit resignation, for prolonged inactivity (3 or more months), or for some infraction of the code of conduct or by consistently demonstrating poor judgement. A maintainer removed for inactivity should be restored following a sustained resumption of contributions and reviews (a month or more) demonstrating a renewed commitment to the project.

项目的维护人员将不时考虑添加或删除维护人员。现有的 Maintainer 可以将更改集提交给 Maintainers.rst 文件。指定的供款人可以通过现有维护人对提案的多数批准而成为维护人。一旦获得批准，变更集就被合并，个人被添加到维护人员组（或者从维护人员组中删除）。维护人员可通过明确辞职、长期不活动（3 个月或更长时间）、违反行为准则或持续表现出较差的判断力而被解雇。在持续恢复捐款和审查（一个月或一个月以上）后，应恢复因不活动而被移除的维护人员，以表明对项目的新承诺。

3、Release cadence

3、释放节奏

The Fabric maintainers have settled on a quarterly (approximately) release cadence (see releases). We are also actively considering adopting an LTS (long term support) release process, though the details of this are still being worked out by the maintainers. Follow the discussion on the #fabric-maintainers channel in Chat.

Fabric 维护人员已确定每季度（大约）发布一次（参见发布）。我们也在积极考虑采用 LTS（长期支持）发布过程，尽管维护人员仍在研究这方面的细节。按照“聊天”中有关“面料维护者”频道的讨论进行操作。

4、Making Feature/Enhancement Proposals

4、提出功能/增强建议

First, take time to review JIRA to be sure that there isn’t already an open (or recently closed) proposal for the same function. If there isn’t, to make a proposal we recommend that you open a JIRA Epic or Story, whichever seems to best fit the circumstance and link or inline a “one pager” of the proposal that states what the feature would do and, if possible, how it might be implemented. It would help also to make a case for why the feature should be added, such as identifying specific use case(s) for which the feature is needed and a case for what the benefit would be should the feature be implemented. Once the JIRA issue is created, and the “one pager” either attached, inlined in the description field, or a link to a publicly accessible document is added to the description, send an introductory email to the fabric@lists.hyperledger.org mailing list linking the JIRA issue, and soliciting feedback.

首先，花时间审查 JIRA，以确保同一职能部门还没有开放（或最近关闭）的提案。如果没有，为了提出建议，我们建议您打开一个圣战史诗或故事，无论哪一个似乎最适合的情况和链接或内联的“一个寻呼机”的建议，说明该功能将做什么，如果可能的话，它可能如何

实现。它还将有助于说明应添加功能的原因，例如确定需要该功能的特定用例以及实现该功能将带来哪些好处的用例。一旦创建了 JIRA 问题，并且在描述字段中添加了“一个寻呼机”，或者在描述中添加了指向公共可访问文档的链接，请向 fabric@lists.hyperledger.org 邮件列表发送介绍性电子邮件，链接 JIRA 问题，并征求反馈。

Discussion of the proposed feature should be conducted in the JIRA issue itself, so that we have a consistent pattern within our community as to where to find design discussion.

对建议的特性的讨论应该在 JIRA 问题本身中进行，这样我们就可以在社区中找到一致的模式来找到设计讨论的位置。

Getting the support of three or more of the Hyperledger Fabric maintainers for the new feature will greatly enhance the probability that the feature’s related CRs will be included in a subsequent release.

获得三个或更多的 HyperledgeFabric 维护人员对新功能的支持将大大提高该功能的相关 CRS 将包含在后续版本中的可能性。

5、Maintainers meeting

5、维护人员会议

The maintainers hold a bi-weekly meeting every other Wednesday at 9 am ET on Zoom. Please see the community calendar for details.

维护人员每隔一个星期三上午 9 点举行一次双周例会。详情请参阅社区日历。

The purpose of the maintainers meeting is to plan for and review the progress of releases, and to discuss the technical and operational direction of the project and sub-projects.

维护人员会议的目的是规划和审查发布的进度，并讨论项目和子项目的技术和操作方向。

New feature/enhancement proposals as described above should be presented to a maintainers meeting for consideration, feedback and acceptance.

如上所述的新功能/增强建议应提交给维护人员会议以供考虑、反馈和接受。

6、Release roadmap

6、发布路线图

The Fabric release roadmap of epics is maintained in JIRA.

Epics 的 Fabric 发布路线图保存在 Jira 中。

7、Communications

7、通信

We use RocketChat for communication and Google Hangouts™ for screen sharing between developers. Our development planning and prioritization is done in JIRA, and we take longer running discussions/decisions to the mailing list.

我们使用 rocketchat 进行通信，使用 google hangouts™在开发人员之间共享屏幕。我们的开发计划和优先顺序是在 JIRA 中完成的，我们需要更长的时间来对邮件列表进行讨论/决策。

二、Contribution guide

二、贡献指南

1、Install prerequisites

1、安装必备组件

Before we begin, if you haven’t already done so, you may wish to check that you have all the prerequisites installed on the platform(s) on which you’ll be developing blockchain applications and/or operating Hyperledger Fabric.

在我们开始之前，如果您还没有这样做，您可能希望检查您是否在开发区块链应用程序和/或操作超级账本结构的平台上安装了所有先决条件。

2、Getting a Linux Foundation account

2、获取 Linux 基金会帐户

In order to participate in the development of the Hyperledger Fabric project, you will need a Linux Foundation account. You will need to use your LF ID to access to all the Hyperledger community development tools, including Gerrit, Jira and the Wiki (for editing, only).

为了参与 HyreDeGeFrand 项目的开发，您需要一个 Linux 基础帐户。您需要使用您的 LF ID 访问所有的超级账本社区开发工具，包括 gerrit、jira 和 wiki（仅用于编辑）。

3、Getting help

3、得到帮助

If you are looking for something to work on, or need some expert assistance in debugging a problem or working out a fix to an issue, our community is always eager to help. We hang out on Chat, IRC (#hyperledger on freenode.net) and the mailing lists. Most of us don’t bite :grin: and will be glad to help. The only silly question is the one you don’t ask. Questions are in fact a great way to help improve the project as they highlight where our documentation could be clearer.

如果您正在寻找可以解决的问题，或者在调试问题或解决问题时需要一些专家帮助，我们的社区总是乐于提供帮助。我们在聊天室、IRC（freenode.net 上的 hyperledger）和邮件列表上闲逛。我们大多数人不咬人：咧嘴一笑：很乐意帮忙。唯一愚蠢的问题就是你不问的问题。事实上，问题是帮助改进项目的一个很好的方法，因为它们强调了我们的文档可能更清晰的地方。

4、Reporting bugs

4、报告错误

If you are a user and you have found a bug, please submit an issue using

JIRA. Before you create a new JIRA issue, please try to search the existing items to be sure no one else has previously reported it. If it has been previously reported, then you might add a comment that you also are interested in seeing the defect fixed.

如果您是用户，并且发现了错误，请使用 JIRA 提交问题。在创建新的 JIRA 问题之前，请尝试搜索现有项目，以确保以前没有其他人报告过该问题。如果之前已经报告过，那么您可以添加一条注释，说明您也有兴趣看到修复的缺陷。

Note

注释

If the defect is security-related, please follow the Hyperledger security bug reporting process.

如果缺陷与安全相关，请遵循 Hyperledger 安全错误报告流程。

If it has not been previously reported, create a new JIRA. Please try to provide sufficient information for someone else to reproduce the issue. One of the project’s maintainers should respond to your issue within 24 hours. If not, please bump the issue with a comment and request that it be reviewed. You can also post to the relevant Hyperledger Fabric channel in Hyperledger Chat. For example, a doc bug should be broadcast to #fabric-documentation, a database bug to #fabric-ledger, and so on…

如果之前没有报告，创建一个新的 JIRA。请尽量为其他人提供足够的信息来重现问题。项目的一个维护人员应该在 24 小时内响应您的问题。如果没有，请发表评论，并要求对问题进行审查。您还可以在“超级账本聊天”中发布到相关的“超级账本结构”频道。例如，一个文档错误应该广播到结构文档，一个数据库错误应该广播到结构分类帐，等等…

5、Submitting your fix

5、提交修复程序

If you just submitted a JIRA for a bug you’ve discovered, and would like to provide a fix, we would welcome that gladly! Please assign the JIRA issue to yourself, then you can submit a change request (CR).

如果你刚刚提交了一个你发现的 bug 的 jira，并想提供一个修复，我们会很高兴地欢迎！请将 JIRA 问题分配给您自己，然后您可以提交变更请求（CR）。

Note

注释

If you need help with submitting your first CR, we have created a brief tutorial for you.

如果您在提交第一个 CR 时需要帮助，我们为您创建了一个简短的教程。

6、Fixing issues and working stories

6、解决问题和工作故事

Review the issues list and find something that interests you. You could also check the “help-wanted” list. It is wise to start with something relatively straight forward and achievable, and that no one is already assigned. If no one is assigned, then assign the issue to yourself. Please be considerate and rescind the assignment if you cannot finish in a reasonable time, or add a comment saying that you are still actively working the issue if you need a little more time.

查看问题列表并找到您感兴趣的内容。您还可以查看“需要帮助”列表。从相对直接和可实现的事情开始是明智的，而且还没有人被指派。如果没有分配任何人，则将问题分配给自己。如果您不能在合理的时间内完成任务，请考虑并取消任务；如果您需要更多的时间，请添加一条评论，说明您仍在积极处理该问题。

7、Reviewing submitted Change Requests (CRs)

7、审查提交的变更请求（CRS）

Another way to contribute and learn about Hyperledger Fabric is to help the maintainers with the review of the CRs that are open. Indeed maintainers have the difficult role of having to review all the CRs that are being submitted and evaluate whether they should be merged or not. You can review the code and/or documentation changes, test the changes, and tell the submitters and maintainers what you think. Once your review and/or test is complete just reply to the CR with your findings, by adding comments and/or voting. A comment saying something like “I tried it on system X and it works” or possibly “I got an error on system X: xxx ” will help the maintainers in their evaluation. As a result, maintainers will be able to process CRs faster and everybody will gain from it.

另一种贡献和学习 Hyperledger 结构的方法是帮助维护人员审查打开的 CRS。实际上，维护人员的任务很困难，必须审查所有提交的 CRS，并评估它们是否应该合并。您可以查看代码和/或文档更改，测试更改，并告诉提交者和维护者您的想法。一旦您的审查和/或测试完成，只需通过添加评论和/或投票回复 CR 和您的发现。一条评论说“我在系统 x 上尝试过并且它工作了”，或者“我在系统 x:x x x 上出错了”，将帮助维护人员进行评估。因此，维护人员能够更快地处理 CRS，每个人都能从中获益。

Just browse through the open CRs on Gerrit to get started.

只需浏览 gerrit 上打开的 CRS 就可以开始了。

8、Setting up development environment

8、建立发展环境

Next, try building the project in your local development environment to ensure that everything is set up correctly.

接下来，尝试在本地开发环境中构建项目，以确保正确设置所有内容。

What makes a good change request?

什么是一个好的变更请求？

One change at a time. Not five, not three, not ten. One and only one. Why? Because it limits the blast area of the change. If we have a regression, it is much easier to identify the culprit commit than if we have some composite change that impacts more of the code.

一次换一次。不是五，不是三，不是十。只有一个。为什么？因为它限制了爆炸面积的变化。如果我们有一个回归，那么识别罪犯所犯的错误要比我们有一些影响更多代码的复合

变更容易得多。

Include a link to the JIRA story for the change. Why? Because a) we want to track our velocity to better judge what we think we can deliver and when and b) because we can justify the change more effectively. In many cases, there should be some discussion around a proposed change and we want to link back to that from the change itself.

包括一个链接到圣战故事的变化。为什么？因为 a）我们希望跟踪我们的速度，以便更好地判断我们认为我们能够交付什么以及何时交付；b）因为我们可以更有效地证明变更的合理性。在许多情况下，应该围绕一个提议的变更进行一些讨论，我们希望从变更本身返回到这个讨论。

Include unit and integration tests (or changes to existing tests) with every change. This does not mean just happy path testing, either. It also means negative testing of any defensive code that it correctly catches input errors. When you write code, you are responsible to test it and provide the tests that demonstrate that your change does what it claims. Why? Because without this we have no clue whether our current code base actually works.

在每次更改中包括单元和集成测试（或对现有测试的更改）。这也不意味着只是愉快的路径测试。它还意味着对任何能够正确捕获输入错误的防御代码进行负面测试。当您编写代码时，您有责任测试它，并提供测试来证明您的更改执行了它所要求的操作。为什么？因为如果没有这个，我们就不知道当前的代码库是否真的有效。

Unit tests should have NO external dependencies. You should be able to run unit tests in place with go test or equivalent for the language. Any test that requires some external dependency (e.g. needs to be scripted to run another component) needs appropriate mocking. Anything else is not unit testing, it is integration testing by definition. Why? Because many open source developers do Test Driven Development. They place a watch on the directory that invokes the tests automagically as the code is changed. This is far more efficient than having to run a whole build between code changes. See this definition of unit testing for a good set of criteria to keep in mind for writing effective unit tests.

单元测试应该没有外部依赖性。您应该能够使用 Go 测试或该语言的等效测试来运行单元测试。任何需要一些外部依赖性的测试（例如，需要编写脚本来运行另一个组件）都需要适当的模拟。任何其他东西都不是单元测试，而是定义为集成测试。为什么？因为许多开源开发人员进行测试驱动的开发。他们在目录上放置一个监视，当代码更改时自动调用测试。这比在代码更改之间运行整个构建要高效得多。请参阅单元测试的定义，了解编写有效单元测试时要记住的一组标准。

Minimize the lines of code per CR. Why? Maintainers have day jobs, too. If you send a 1,000 or 2,000 LOC change, how long do you think it takes to review all of that code? Keep your changes to < 200-300 LOC, if possible. If you have a larger change, decompose it into multiple independent changes. If you are adding a bunch of new functions to fulfill the requirements of a new capability, add them separately with their tests, and then write the code that uses them to deliver the capability. Of course, there are always exceptions. If you add a small change and then add 300 LOC of tests, you will be forgiven;-) If you need to make a change that has broad impact or a bunch of generated code (protobufs, etc.). Again, there can be exceptions.

最小化每个 CR 的代码行。为什么？维护人员也有日常工作。如果发送 1000 或 2000 个 loc 更改，您认为需要多长时间来检查所有代码？如果可能，将您的更改保持在<200-300 个位置。如果您有一个更大的变更，请将其分解为多个独立的变更。如果要添加一组新功能来满足新功能的需求，请在测试中分别添加它们，然后编写使用它们来交付功能的代码。当然，总会有例外。如果您添加一个小的更改，然后添加 300 个 loc 的测试，那么如果您需要进行具有广泛影响的更改或一组生成的代码（protobufs 等），您将被原谅；-）。同样，也有例外。

Note

注意

Large change requests, e.g. those with more than 300 LOC are more likely than not going to receive a -2, and you' ll be asked to refactor the change to conform with this guidance.

较大的变更请求（例如，具有 300 个以上 loc 的变更请求）更有可能收到-2，并且您将被要求重构变更以符合本指南。

Do not stack change requests (e.g. submit a CR from the same local branch as your previous CR) unless they are related. This will minimize merge conflicts and allow changes to be merged more quickly. If you stack requests your subsequent requests may be held up because of review comments in the preceding requests.

不要堆叠变更请求（例如，提交来自与您以前的变更请求相同的本地分支机构的变更请求），除非它们是相关的。这将最小化合并冲突，并允许更快速地合并更改。如果您将请求堆叠起来，那么随后的请求可能会因为前面请求中的审阅注释而被搁置。

Write a meaningful commit message. Include a meaningful 55 (or less) character title, followed by a blank line, followed by a more comprehensive description of the change. Each change MUST include the JIRA identifier corresponding to the change (e.g. [FAB-1234]). This can be in the title but should also be in the body of the commit message. See the complete requirements for an acceptable change request.

编写一个有意义的提交消息。包括一个有意义的 55 个字符（或更少）的标题，后面是一个空行，后面是对更改的更全面的描述。每个更改必须包括与更改相对应的 JIRA 标识符（例如[FAB-1234]）。这可以在标题中，但也应该在提交消息的正文中。见可接受变更请求的完整要求。

Note

注意

That Gerrit will automatically create a hyperlink to the JIRA item. e.g.

该 gerrit 将自动创建到 jira 项目的超链接。例如

```
[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.
```

Finally, be responsive. Don' t let a change request fester with review comments such that it gets to a point that it requires a rebase. It only further delays getting it merged and adds more work for you - to remediate the merge conflicts.

最后,要有回应。不要让一个变更请求因评审意见而恶化,这样它就需要一个重新平衡。它只会进一步延迟合并，并为您添加更多工作-以修复合并冲突。

三、Legal stuff

三、法律材料

Note: Each source file must include a license header for the Apache Software License 2.0. See the template of the license header.
注意：每个源文件必须包含 Apache 软件许可证 2.0 的许可证头文件。请参见许可证头的模板。

We have tried to make it as easy as possible to make contributions. This applies to how we handle the legal aspects of contribution. We use the same approach—the Developer’s Certificate of Origin 1.1 (DCO)—that the Linux® Kernel community uses to manage code contributions.

我们努力使尽可能容易地作出贡献。这适用于我们如何处理捐款的法律方面。我们使用与开发人员的源代码证书 1.1（DCO）相同的方法——Linux®内核社区用于管理代码贡献。

We simply ask that when submitting a patch for review, the developer must include a sign-off statement in the commit message.

我们只要求在提交一个补丁进行审查时,开发人员必须在提交消息中包含一个签准语句。

Here is an example Signed-off-by line, which indicates that the submitter accepts the DCO:

下面是一个由第行签名的示例，表明提交者接受 DCO：

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

签字人：john doe<john.doe@example.com>

You can include this automatically when you commit a change to your local git repository using git commit -s.

当您使用 git commit-s 提交对本地 git 存储库的更改时，可以自动包含此内容。

四、Related Topics

四、相关主题

Maintainers
维护者
Using Jira to understand current work items
使用 JIRA 了解当前工作项
Setting up the development environment
建立发展环境
Building Hyperledger Fabric
构建 Hyperledger 结构
Building outside of Vagrant
流浪者外的建筑
Configuration
配置
Requesting a Linux Foundation Account

请求 Linux 基金会帐户
Working with Gerrit
与 Gerrit 合作
Reviewing Using Gerrit
使用 Gerrit 查看
Viewing Pending Changes
查看挂起的更改
Submitting a Change to Gerrit
向 Gerrit 提交变更
Reviewing a Change
审阅更改
Gerrit Recommended Practices
Gerrit 推荐做法
Coding guidelines
编码指南
Generating gRPC code
生成 GRPC 代码
Adding or updating Go packages
添加或更新 Go 包