Best Practices Badges / Templates

Templates for Incubation Status

https://bestpractices.coreinfrastructure.org/en

Project Checklist

- [ ] Wiki space Maintenance
- [ ] Github or Gerrit repositories
- [ ] Graphic Set
- [ ] Whitepaper
- [ ] Training Documentation Packet
- [ ] Diverse group of Maintainers

Templates Resource Checklist:

Whitepaper Template	Use Case	Graphics	RFP	Presentations	EdX Courses	Webinar Basics	Github Basics	Blog Basics

Format for Standard

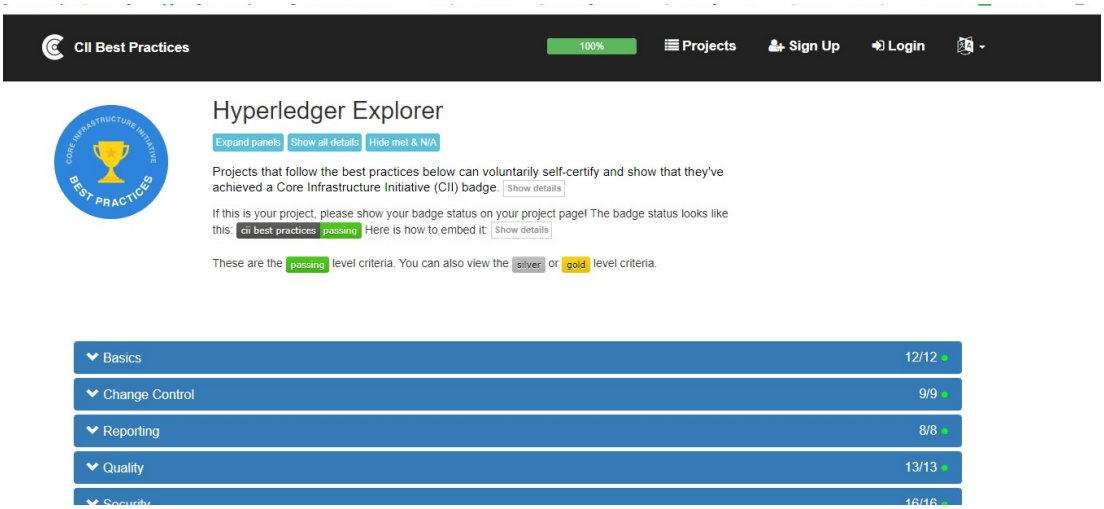
To exit incubation, as project must have Sufficient user documentation
The project must including enough documentation for anyone to test or deploy any of the modules.



Requirements for BEST PRACTICE BADGE :

Self Certify for CII Badge:

Core Infrastructure Initiative



***UPDATE CII TO INCLUDE COMMUNITY READINESS AND DOCUMENTATION.

Met

Basics:		
• **Identification**	• What is the human-readable name of the project? What is a brief description of the project? • What is the URL for the project (as a whole)? • What is the URL for the version control repository (it may be the same as the project URL)? • **What programming language(s) are used to implement the project?** • What is the Common Platform Enumeration (CPE) name for the project (if it has one)?	
• **Basic project website content**	• The project website MUST succinctly describe what the software does (what problem does it solve?). [description_good] The project website MUST provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software. [interact] how to CONTRIBUTE. The information on how to contribute MUST explain the contribution process (e.g., are pull requests used?) (URL required) [contribution] Non-trivial contribution file in repository. Projects on GitHub by default use issues and pull requests, as encouraged in documentation. The information on how to contribute SHOULD include the requirements for acceptable contributions (e.g., a reference to any required coding standard). (URL required) [contribution_requirements]	
• **FLOSS license** **What license(s) is the project released under?**	• License The software produced by the project MUST be released as FLOSS. [floss_license] It is SUGGESTED that any required license(s) for the software produced by the project be approved by the Open Source Initiative (OSI). [floss_license_osi] The Apache-2.0 license is approved by the Open Source Initiative (OSI). The project MUST post the license(s) of its results in a standard location in their source repository. (URL required) [license_location] Non-trivial license.	

• **Documentation**	• The project MUST provide basic documentation for the software produced by the project. [documentation_basics] The project MUST provide reference documentation that describes the external interface (both input and output) of the software produced by the project. [documentation_interface] .	
• **Other**	• The project sites (website, repository, and download URLs) MUST support HTTPS using TLS. [sites_https] The project MUST have one or more mechanisms for discussion (including proposed changes and issues) that are searchable, allow messages and topics to be addressed by URL, enable new people to participate in some of the discussions, and do not require client-side installation of proprietary software. [discussion] The project SHOULD provide documentation in English and be able to accept bug reports and comments about code in English. [english] Other general comments about the project:	
Change Control		
Public version-controlled source repository	The project MUST have a version-controlled source repository that is publicly readable and has a URL. [repo_public] Show details Mirror Repository on GitHub, which provides public git repositories with URLs. Source control via GitHub The project's source repository MUST track what changes were made, who made the changes, and when the changes were made. [repo_track] Repository on GitHub, which uses git. git can track the changes, who made them, and when they were made. Source control via GitHub Repository on GitHub, which uses git. git can track the changes, who made them, and when they were made. Repository on GitHub, which uses git. git can track the changes, who made them, and when they were made. To enable collaborative review, the project's source repository MUST include interim versions for review between releases; it MUST NOT include only final releases. [repo_interim] Show details Source control via GitHub It is SUGGESTED that common distributed version control software be used (e.g., git) for the project's source repository. [repo_distributed] Show details Mirror repository on GitHub, which uses git. git is distributed. Repository on GitHub, which uses git. git is distributed.	
Unique version numbering	The project results MUST have a unique version identifier for each release intended to be used by users. [version_unique] The project uses Git tags, see change log It is SUGGESTED that the Semantic Versioning (SemVer) format be used for releases. [version_semver] It is SUGGESTED that projects identify each release within their version control system. For example, it is SUGGESTED that those using git identify each release using git tags. [version_tags] Hyperledger Explorer version control, and tagging	
Release notes	The project MUST provide, in each release, release notes that are a human-readable summary of major changes in that release to help users determine if they should upgrade and what the upgrade impact will be. The release notes MUST NOT be the raw output of a version control log (e.g., the "git log" command results are not release notes). Projects whose results are not intended for reuse in multiple locations (such as the software for a single website or service) AND employ continuous delivery MAY select "N/A". (URL required) [release_notes] https://github.com/hyperledger/blockchain-explorer/tree/master/release_notes, and the CHANGELOG found at https://github.com/hyperledger/blockchain-explorer/blob/master/CHANGELOG.md The release notes MUST identify every publicly known vulnerability with a CVE assignment or similar that is fixed in each new release, unless users typically cannot practically update the software themselves. If there are no release notes or there have been no publicly known vulnerabilities, choose "not applicable" (N/A). [release_notes_vulns] If any known vulnerability was fixed, it is described in for example, https://github.com/hyperledger/blockchain-explorer/blob/master/release_notes/v0.3.8.md#known-vulnerabilities	
REPORTING		
Bug-reporting process		
Vulnerability report process		
Quality		
Working build system		
Automated test suite		

New functionality testing		
Warning flags		
Security		
Secure development knowledge		
Use basic good cryptographic practices		
Secured delivery against man-in-the-middle (MITM) attacks		
Other security issues		
Static code analysis		
Dynamic code analysis		

Example : Explorer

Best practices

- All project names must be prefaced with “Hyperledger.”
- Ensure the name is uniquely identifiable. Anticipate and remove potential confusion.
    - Are there closely named brands in the tech industry or elsewhere?
    - Is it similar to popular existing open source projects?
    - Is your project also a common word? If it’s too generic, searching for it and enforcing trademark compliance may be difficult.
- The name should give people some understanding of what the technology does and/or how people can use it.
- Who does your project most appeal to? Understand your target user and what kind of names or brands they respond to.
- Think carefully about what the name evokes. It sets the tone and intent, and will ideally inspire action.
- Consider incorporating a thematic mascot (e.g., Ursa’s bear, Aries’ ram, Burrow’s marmot).
- Aim for no more than eight characters so it will be quick to type and the logo won’t take up too much space.
- Think inclusively.
    - Avoid complex acronyms and unnecessarily technical terminology/jargon. How will the name will translate to other cultures or people that may join the project in the future?
    - If you select a name related to an inside joke, make sure the origin story is one you’re willing to share with the masses as part of the brand narrative.
    - Make sure the name is easy to remember and easy to pronounce. Consider potential mispronunciations.

Project Checklist

- [ ] Wiki space Maintenance
- [ ] Github or Gerrit repositories
- [ ] Graphic Set
- [ ] Whitepaper
- [ ] Training Documentation Packet
- [ ] Diverse group of Maintainers

Templates Resource Checklist:

On-the-fly Style Guide for Hyperledger Publications	Hyperledger Style Guide for Publications • *Last updated September 23, 2019* • Compiled by Gordon Graham (gordon@thatwhitepaperguy.com)and Bobbi Muscara *"I put together this style guide on-the-fly while working on publications for Hyperledger. When I encounter an inconsistency or a questionable term, I decide which way seems to make the most sense. I personally dislike the Chicago Manual of Style because it so often says 'either approach is acceptable.' I'm not saying these decisions are perfect and must stand forever, but I have followed these guidelines in the first four or five white papers I've revised and in several case studies I've written over the past year."—Gordon*
White Paper Standards	• Compiled by Gordon Graham (gordon@thatwhitepaperguy.com) and Bobbi Muscara

To exit incubation, as project must have Sufficient user documentation
The project must including enough documentation for anyone to test or deploy any of the modules.



Requirements for BEST PRACTICE BADGE :

Best practices

- All project names must be prefaced with "Hyperledger."
- Ensure the name is uniquely identifiable. Anticipate and remove potential confusion.
  - Are there closely named brands in the tech industry or elsewhere?
  - Is it similar to popular existing open source projects?
  - Is your project also a common word? If it's too generic, searching for it and enforcing trademark compliance may be difficult.
- The name should give people some understanding of what the technology does and/or how people can use it.
- Who does your project most appeal to? Understand your target user and what kind of names or brands they respond to.
- Think carefully about what the name evokes. It sets the tone and intent, and will ideally inspire action.
- Consider incorporating a thematic mascot (e.g., Ursa's bear, Aries' ram, Burrow's marmot).
- Aim for no more than eight characters so it will be quick to type and the logo won't take up too much space.
- Think inclusively.
  - Avoid complex acronyms and unnecessarily technical terminology/jargon. How will the name will translate to other cultures or people that may join the project in the future?
  - If you select a name related to an inside joke, make sure the origin story is one you're willing to share with the masses as part of the brand narrative.
  - Make sure the name is easy to remember and easy to pronounce. Consider potential mispronunciations.

Project Checklist

- [ ] Wiki space Maintenance
- [ ] Repositories
- [ ] Github or Gerrit repositories
- [ ] Graphic Set
- [ ] Whitepaper
- [ ] Training Documentation Packet

- [ ] Mooc
- [ ] Webinars

blocked URL