Ironic Documentation Analysis

Dave Welsch (@dwelsch-esi)

Introduction

This document analyzes the effectiveness of the Ironic open source software (OSS) project's documentation and website.

This analysis is the first step of a process that aims to enable contributors to improve the effectiveness of the Ironic documentation.

Effective documentation

We deliberately use the word "effectiveness" rather than "quality" in this work in an effort to use objective, goal-oriented criteria in evaluating and recommending changes to the documentation.

This analysis assumes that the purpose of technical documentation is to help the user of a product achieve particular objectives. The scope of these objectives ranges from individual tasks (use a command correctly, connect to a database) to large-scale goals (write a software product or module; configure a server farm). Objectives also vary by user role (new developer; software architect; IT administrator). Documentation effectiveness is therefore defined by how well it enables a user to succeed in achieving their objectives, *within every scope of a product's use and for all users*.

Most often*, the most effective form of user documentation is a repeatable process, laid out as a step-by-step procedure or task. For more complex goals, conceptual information is required for the user to build a mental model (a* schema*) of the system (software product or development environment, for example) in order to reason about the system and to derive novel methods and procedures. Finally, reference information is needed for the sake of completeness to fully describe the system.

With this in mind, our vision of effective documentation skews toward the instructional, with emphasis on procedures and tasks. Of course conceptual explanations and reference materials are needed as well. Instructions, however, form the backbone of good software documentation, the goal of which is to achieve real technical and business objectives.

*Per widely accepted technical documentation best practices

Purpose of this analysis

This document analyzes the current state of Ironic documentation. Its purpose is to provide project leaders with an informed understanding of potential problems in current project

documentation, especially with respect to the documentation's effectiveness as defined above.

This document:

- Analyzes the current Ironic technical documentation and website
- Discusses existing documentation in terms of a rubric based on best practices
- Recommends a program of key improvements with the largest return on investment

A companion document, ironic-implementation.md, follows up these recommendations with an actionable plan for improvement. A third document, ironic-issues.md, outlines a backlog of changes that can be made to improve the documentation. To the extent possible, these changes are independent of each other and require limited effort so that contributors can make progress by committing small blocks of time.

Scope of the analysis

The documentation discussed here includes the entire contents of the Ironic documentation website, as well as documentation for contributors and users on the Ironic OpenDev repository.

The Ironic documentation is written in ReStructured Text (RST) and is compiled using the Python-based Sphinx static site generator. The site's code is stored on the Ironic OpenDev repo. The documentation website is hosted by OpenDev on equipment donated by project contributors.

The material designated "Primary" in scope is the main concern of this analysis. The "Secondary" material is discussed if it is relevant, for example when linked from primary material or when it provides essential information not found in the primary material.

In scope:

Primary

- Ironic documentation: https://docs.openstack.org/ironic/latest/
- Ironic release notes: https://docs.openstack.org/releasenotes/ironic/
- Ironic API documentation: https://docs.openstack.org/api-ref/baremetal/

Secondary

- Ironic Bare Metal (a sort of marketing website): https://ironicbaremetal.org/
- Documentation repo: https://opendev.org/openstack/ironic
- Ironic software spec: https://opendev.org/openstack/ironic-specs
- Main project contributor guide: https://docs.openstack.org/contributors/
- Main project documentation contributor guide: https://docs.openstack.org/contributors/code-and-documentation/documentation.html

- Activation of Ironic top-level pages in the OpenStack documentation portal: https://opendev.org/openstack/openstack-manuals/src/branch/master/www/project-data/latest.yaml

Out of scope:

- Other Ironic repos: https://opendev.org/openstack/ironic*
- Other OpenStack repos and projects: https://opendev.org/openstack/
- Deprecated project documentation contributor guide: https://docs.openstack.org/doc-contrib-guide/writing-style/general-writing-guidelines.html
- The OpenStack documentation portal (except that the Ironic documentation must conform to portal's top-level organization scheme): https://opendev.org/openstack/openstack-manuals
- Other OpenStack documentation

How this analysis is organized

This document is divided into three sections that represent three major areas of concern:

- **1. Project documentation:** concerns documentation for users of the Ironic software, aimed at people who intend to use it
- **2. Contributor documentation:** concerns documentation for new and existing contributors to the Ironic OSS project
- **3. Website:** concerns the mechanics of publishing the documentation, and includes branding, website structure, and maintainability

Each section contains two parts:

- **Comments**: discusses the documentation in terms of a rubric with appropriate [criteria][cncf-doc-criteria] for the section and other observations about the existing documentation, with a focus on how it does or does not help Ironic users achieve their goals. The rubric criteria were developed by the Cloud Native Computing Foundation (CNCF) to assess the documentation of its many member projects. A few other sections have been added to discuss issues identified in preliminary interviews as important to Ironic that are not covered by the CNCF rubric.
- **Recommendations**: suggests changes that would improve the effectiveness of the documentation as defined in Effective Documentation. Each recommendation is rated on importance (low, medium, or high) and effort level (same).

An accompanying document, [ironic-implementation.md], breaks the recommendations down into concrete actions that can be implemented by project contributors. Its focus is on specific, achievable work that can be completed in constrained blocks of time. Ultimately, the implementation items are to be rendered as a set of issues or backlog items in Launchpad or a similar tracking system.

How to use these documents

Readers interested in the current state of the documentation and the reasoning behind the recommendations should read the section of this document pertaining to their area of concern:

1. Project documentation
2. Contributor documentation
3. Website and documentation infrastructure

Readers interested in recommended improvements but not the overall analysis should skip this document and read ironic-implementation.md.

Readers interested only in specific actionable improvements should skip to ironic-issues.md

Recommendations, requirements, and best practices

This analysis describes how documentation could be improved according to industry standards, best practices, and in some cases the analyst's experience. In most cases there is more than one way to do things. Few recommendations here are meant to be prescriptive. Rather, the recommended implementations represent the reviewers' experience with how to apply documentation best practices. In other words, borrowing terminology from the lexicon of [RFCs][rfc-keywords], the changes described here should be understood as "recommended" or "should" at the strongest, and "optional" or "may" in many cases. Any "must" or "required" actions are clearly denoted as such, and pertain to legal requirements such as copyright and licensing issues.

1. Project documentation

Comments

Style and conventions

> **Summary:** Page headings are often unhelpful and can cause confusion. Other style issues are more minor.

Many of the issues here relate to findability, including consistent use of navigational markers such as headings and product names to cue and orient the reader.

Naming and Labeling

Page headings often do not represent content as well as they should. In particular, a heading should clearly represent the purpose of the following content, whether that is a procedure, a reference, or a conceptual explanation. This is especially confusing in combination with the overly broad scope of the Search functionality.

There are inconsistencies in naming: of services, software programs, components. Changing the capitalization of a product name, for example, changes its meaning. Each OpenStack

product seems to have a name (Ironic, Swift, Neutron) and a functional name (Bare Metal service, Object Store service, Network service). These should both be capitalized consistently; “Ironic” is not the same as “ironic”. Each service also has a CLI command that must be typed literally, usually in all lower-case.

Typography

Typographic conventions: Again, consistency is key. Onscreen text such as CLI examples, file contents, file names, and commands should use a `monospace` font. Use italics, *not* capital letters, for emphasis, and do so sparingly.

Information architecture

> **Summary:** Documentation is mostly complete, but difficult to navigate. Often, instructional content (the information needed to solve a problem or complete a task) must be picked out from several different doc pages.
>
> Support channels are good, but are on services that may be unfamiliar to users of more mainstream open-source software resources, (I’m looking at you, GitHub).
>
> The documentation set lacks a clear Getting Started entry point and work path.
>
> Search capability encompasses all OpenStack documentation and cannot be limited to the Ironic doc set.
>
> OpenStack-wide infrastructure exists for internationalization, but no Ironic documentation seems to have been translated.
>
> There is guidance for contributing to OpenStack documentation and to the Ironic code, but not for contributing to Ironic documentation specifically.
>
> The Ironic documentation echoes the OpenStack documentation in containing much non-inclusive language.

Any Ironic documentation page indexed from the OpenStack documentation portal is constrained by OpenStack’s documentation conventions. See Appendix A: Conformance to OpenStack documentation.

Conceptual documentation

The Ironic documentation presents conceptual information throughout the documentation. Product overview information exists primarily at the beginning of the User Guide.

The conceptual overview intermingles these main issues:

- What is bare metal provisioning, and why it’s necessary
- How bare metal provisioning works on a physical server (PXE, IPMI, and similar concepts)
- How Ironic fits into and is invoked by the OpenStack architecture, including detailed sequence and state machine diagrams

Advanced Ironic users that I interviewed found the state diagram very useful.

Feature coverage

There seems to be fairly complete feature coverage in the Ironic documentation. Users I've interviewed have described being able to find information, but often having to track it down in the documentation.

There are many pages with multiple topics. Some contain multiple tasks. Some contain a mix of different types of topics, for example concepts and tasks. The TOC only shows two levels (and even then only when expanded by selecting the level 1 entry). As a result, many of these topics are unfindable except using search, and OpenStack's search functionality is broken. For example, how would one find any of the topics in Advanced features?

Ironic has several sub-projects split over multiple repositories. The documentation links to these projects, which is helpful.

Instructional documentation

Developers naturally tend to write documentation from a feature perspective, not a user perspective. This leads to the documentation that is *complete*, but in which instructional information (how to use the software) is incomplete, missing, or fragmented. This suboptimal instructional content is *inefficient* and limits the documentation's usefulness.

- *complete*: all features are meticulously documented somewhere
- *inefficient*: readers must make their own way through the docs to find what they need

Much of the Ironic instructional documentation is:

- Difficult to find
  - Organization is sub-optimal
  - Headings are ambiguous or do not identify the page as instructional
  - Tasks are intermixed with examples, conceptual information, and reference data on the same page
- Difficult to use
  - Many tasks assume extensive knowledge and omit prerequisites
  - Many tasks omit basic steps
  - Many tasks are not clearly written as instructions; for example, steps are not explicitly ordered or labeled as steps

The "happy paths" – the most common use cases – seem to be documented, but suffer from the following:

- Not organized by user role or use case, so are hard to find
- Split up among different sections of the documentation, organized by system component rather than procedurally

The Administrator's Guide is written to contain tasks, including admin tasks in the TOC such as:

- Hardware inspection
- Node deployment
- Node cleaning
- Node adoption (transfer from another bare-metal installation, e.g.)
- Node retirement
- Configuring RAID

Upgrading

There does not seem to be much in the way of tutorial documentation. This is of secondary concern, but might be helpful in illustrating some of the more elaborate happy path procedures.

Tasks are not clearly named according to user goals and are interspersed with, and often subordinate to, other information. Often, the page containing task content is titled for a feature that can be more or less related to the task. For example, instructions to enable a proxy for image download are on the "Ironic Python Agent" page.

The *instructional tone* of the Ironic documentation is sometimes suggestive rather than imperative. This means that, rather than telling the user what to do, the text provides suggestions that the user can or should use.

Support documentation

Ironic-specific support is linked from the Ironic Developer's Guide documentation pages. Listed community resources include the code repo; weekly meetings; a contributor whiteboard; and an IRC channel hosted on OFTC (Open and Free Technology Community), among others.

The main README page in the Ironic repo lists project resources, including a Launchpad page. Launchpad is also used for bug tracking.

Both of these locations are contributor resources. Support for other Ironic users is not found on the main documentation pages. Again, this is partly a function of the OpenStack documentation infrastructure, but it should be possible to list basic support resources on the Ironic doc landing page.

In general, the OpenStack community uses a different set of tools from most other open source projects. Getting accounts on the tools and getting acclimated to them requires some effort in addition to the ramp-up on the Ironic product itself. This could discourage new contributors and users who are not already familiar with the OpenStack project. This is not a value judgement about the merits of the OpenDev resource stack; rather, it's a recognition that GitHub has captured the lion's share of OSS development activity and that the resulting unfamiliarity with alternative platforms puts them at a disadvantage with new users.

Some of the more important Ironic and OpenStack resources and their more mainstream analogues are:

Resource Type	Ironic/OpenStack flavor	More familiar platform
code repository	OpenDev	GitHub
code contribution framework	Gerrit	GitHub pull requests
bug tracking	Launchpad	GitHub issues
chat	OFTC IRC	Slack
shared documents	Etherpad	Google Docs
video conferencing	OpenDev MeetPad	Zoom

There are Ironic-related channels on the OFTC (Open and Free Technology Community) IRC network, primarily **#openstack-ironic**. The OpenStack IRC channels are unlisted, meaning they cannot be discovered in a search. OFTC is bridged from matrix, which is more friendly. The Ironic IRC is publicly logged here:

https://meetings.opendev.org/irclogs/%23openstack-ironic/

Ironic meeting notes are at

https://meetings.opendev.org/meetings/ironic/.

Ironic Project Team Gathering (PTG) meeting notes and minutes are captured in EtherPad; for example

https://etherpad.opendev.org/p/ironic-ptg-april-2024.

The notes are released as a final document (from

https://opendev.org/openstack/ironic-specs).

New user content

> **Summary**: There is no clear "Getting Started" workflow.
>
> The installation instructions cover only one scenario; it seems likely this is not the only install case.

The new user path is a weakness of the Ironic documentation.

The variations on user roles for Ironic and the variety of deployment scenarios makes it challenging to direct readers to the correct onboarding path. However, this is all the more reason to make the attempt. The alternative is letting new users waste a colossal amount of time trying to sort out what they need to do and where they need to go.

There is no explicitly labeled "getting started" guide. There is no clear new user roadmap. At the end of the Installation Guide is a section called Next Steps that doesn't link to anything.

Installation is documented (but see caveats about the project's instructional documentation). The Installation Guide documents one specific scenario (prefaced by a user story featuring personas), an operator building a small cloud with a user provisioning small numbers of instances through the Compute API. There are several installation scenarios elsewhere in the documentation; these are not introduced in a manner that helps choose between them (though in some cases the scenario will be obvious to the user; for example integrating with OpenStack compute). Step-by-step instructions are not explicit in some scenarios.

Sample code is abundant in task sections and configuration references, though some of it is in the form of examples in which the user must replace placeholder variables.

Different operating systems (OSes) play a role in Ironic, but these are largely out of scope of this analysis:

- Installed instances can have different OSes, but Ironic is unconcerned with the OS at this point (an image is an image).
- There are a number of OpenStack and Ironic deployment schemes for various OSes, but they are out of the scope of this analysis.
- A number of OSes are supported by Ironic Python Agent Builder, also out of scope of this analysis.

The Standalone configuration mode page gives configuration settings for /etc/ironic/ironic.conf, but doesn't tell you to copy a sample config file here. Is there one you're supposed to use? There's one in configuration/, but no indication that's where you go for it.

Content maintainability & site mechanics

> **Summary**: Inconsistent page naming coupled with the lack of an Ironic-specific search tool makes finding information difficult.
>
> No Ironic documentation has been translated to other languages, though the OpenStack infrastructure exists to do so.

Search

The Search control in the page header menu searches multiple releases of the entire OpenStack documentation set. Ironic documentation is not searchable separately from the entire OpenStack documentation set.

If there is a way to finesse the search to include only Ironic documentation, I'd include that instruction prominently somewhere on the landing page.

The titles displayed in the search results are often misleading. Since the search is OpenStack-wide, a title such as "User Guide" or "How to Contribute" could be for any OpenStack project.

Internationalization

OpenStack has a documentation translation project, with various pieces of documentation completed to varying percentages. Translations to 52 languages are represented, but it does not look like any Ironic documentation is included.

Translation for all OpenStack projects seems spotty, probably a result of not enough volunteer translators. The translation project seems to be the approved internationalization path for OpenStack and is out of scope of this analysis.

Versioning

The documentation seems to be up to date. Releases are versioned (in OpenDev branches) and release notes are available for OpenStack named releases. It's not clear how non-synced releases (minor releases that are not on the OpenStack release cadence) are documented.

API

The TOC explicitly exposes an API reference. The API reference is generated from code; the source is in a separate directory (api-ref) in the Ironic doc repo.

Content creation processes

> **Summary**: The Ironic Developer's Guide does not include information for documentation contributors.

There doesn't seem to be any process documented for contributing to documentation for Ironic. There is an OpenStack Documentation Contributor Guide, which (I'm told) has been deprecated.

The code release process (as documented in the [Developer's Guide][https://docs.openstack.org/ironic/latest/contributor/index.html] and So You Want to Contribute … pages) includes references to updating the documentation, but is vague on how to do this.

There is a Quickstart for OpenStack documentation contributors that seems to be current. This guide describes how to commit documentation changes.

It's not clear who is responsible for documentation updates, or who the website owner is.

Inclusive language

> **Summary**: Could be better.

The Ironic documentation contains several words on the Inclusive Naming Initiative's "Replace Immediately" and "Strongly Consider Replacing" word lists, including "master", "abort", and "sanity check". Much of this language is in the contributor documentation.

The documentation also includes many instances of potentially ableist language, using terms like "simply" and "easy".

Recommendations

Overall recommendations are to: - Reorganize page information to: - Separate conceptual, instructional, and reference information - Organize instructional information by user role, by revising the Installation, Administrator, User, and Developer guides around instructional information. - Rewrite instructions with definite, numbered steps. - Rename headings to better reflect the content of pages and aid in using the Search tool. - Write a Getting Started guide, with an introduction at the top of the main Ironic landing page. Link to Getting Started from each of the OpenStack-listed guides (Installation, Admin, User, etc.). - Revise the Installation guide to outline all major install scenarios. - Generate APIs from annotated code. - Eliminate the worst non-inclusive language.

Information architecture

Importance	Effort Level
High	High

Separate conceptual, task, and reference information throughout the documentation. - Use gerunds ("-ing" verbs) for task headings ("Rescuing a Node", "Deploying a Bare Metal Server"). - Label references as such: "API Reference" and so forth. - Use a description for concept headings ("Architectural Overview", "Bare Metal Deployment").

Conceptual documentation

It might not be practical to change this, but it was suggested more than once during interviews that some of the states in the state diagram have names that can be confused with actions, including "Delete".

Instructional documentation

Rewrite instructions for tasks and procedures to be step-by-step instructions with an imperative purpose. Here are some examples:

Page	**Suggestive style**	**Replace with**
/admin/upgrade-guide.html	This document outlines various steps and notes for operators to consider when upgrading …	This document provides instructions for upgrading …
/install/standalone.html	This guide explains how to configure and use the Bare Metal service standalone, i.e. without other OpenStack services.	None; this is good.

Page	Suggestive style	Replace with
/configuration/index.html	… The following pages describe configuration options that can be used to adjust the service to your particular situation.	The following pages describe the available configuration options.
/admin/index.html	If you are a system administrator running Ironic, this section contains information that may help you understand how to operate and upgrade the services.	This section provides instructions for system administrators needing to operate and upgrade the services.

Make instructions step-by-step procedures. If there are technical or knowledge prerequisites, state them at the beginning of the procedure. Remove extraneous information; if reference material is needed (for example, if there are multiple configuration options to choose among), provide a link to the reference. In cases where the instructions might diverge (for example, when Ironic behaves differently depending on the hardware in question), enumerate the outcomes and give choices or point to sub-tasks as a solution. It's better to build a decision tree than to provide vague instructions.

Put instructional topics in the Guide for its most likely user role: Administrator, User, or Developer (Contributor).

Take non-installation instructional material (roughly, Enrollment and everything after it) and put it in the Administrator's guide. Alternatively, explain in the Getting Started guide that Installation includes enrolling a single node to activate and test the scheduler.

Support documentation

Normally a project has support resources listed in a persistent location on the website, such as in the dropdown menu, a persistent header, or the footer. If this is not possible, create an Ironic support resources page and link to it prominently on the Ironic documentation landing page (and elsewhere as appropriate).

For the benefit of those new to the OpenDev/OpenStack community, provide a brief explanation of the community's platforms, listing their analogues in the GitHub world. Provide links to help and tutorial resources.

Resource Type	Ironic/OpenStack flavor	More familiar platform
code repository	OpenDev	GitHub

Resource Type	Ironic/OpenStack flavor	More familiar platform
code contribution framework	Gerrit	GitHub pull requests
bug tracking	Launchpad	GitHub issues
chat	OFTC IRC	Slack
shared documents	Etherpad	Google Docs
video conferencing	OpenDev MeetPad	Zoom

New user content

Importance	Effort Level
High	Medium

Put Getting Started information on the landing page. Link from there to whatever tasks the user needs to go to.

Put a single link to the Getting Started page from the top of each of the other guides (Installation, User, Admin, …) since a certain percentage of users will jump to each of those sections directly from the OpenStack doc landing page.

Write a Getting Started page that defines Ironic's main use cases. If the use cases have different setup, installation, or configuration requirements, direct users to the correct workflow for their situation.

At the end of the Getting Started guide(s), write a Next Steps section that lays out what the user can do from there and clearly links to instructional information for each option.

Revise the Installation guide to give instructions for every important installation scenario. The most expedient way to do that might be to expand [Reference Deploy Architectures][] to include other important scenarios (currently only "Small cloud with trusted tenants" is listed).

When documenting configuration settings in instructions, give specific examples. If the user genuinely has more than one option, list and describe the options and/or link to a reference. Likewise the configuration file itself: Provide it in the procedure, or link to it.

Content maintainability & site mechanics

Importance	Effort Level
Medium	Medium

Search

Since search is OpenStack-wide and cannot be adjusted within Ironic, make sure page headings are descriptive and reflect the content and purpose of the page.

If no other remedy is available, consider including “Ironic” or “Bare Metal” in top-level page headings. Even though it’s redundant and funny-looking in the Ironic documentation, it will make search results much easier to interpret.

API

Generate API documentation from code rather than maintaining the documentation separately.

Content creation processes

Importance	Effort Level
Medium	Low

Include a page with documentation-specific contributor information in the Ironic [Developer’s Guide][https://docs.openstack.org/ironic/latest/contributor/index.html], So You Want to Contribute … pages) and OpenStack documentation Quickstart.

Create an OWNERS or MAINTAINERS file in the docs directory, or specifically add documentation owners to the Ironic maintainers list.

Inclusive language

Importance	Effort Level
Medium	Medium

Replace the most offensive use of terms in the Inclusive Naming Initiative’s word lists.

2. Contributor documentation

Comments

In many ways, the contributor documentation is better than the user documentation.

Ironic contributor documentation includes a getting-started workflow, links to the Ironic community channels, and instructions on using contributor tools.

There are two names for contributors in the OpenStack documentation. They are called “contributors”, as is conventional in open source software; however, the documentation for contributors is called the “Developer’s Guide”. Probably nothing can be done about this, especially at the individual project level.

Communication methods documented

> **Summary:** . - Ironic (and other OpenStack projects) have active discussions ongoing in OFTC IRC channels. There are links to Ironic’s community forums and lists. Meetings are documented.

Ironic has links to its OpenDev repository, bug tracker, and contributor tools.

Beginner friendly issue backlog

Issues are tracked and triaged. Documentation issues are tagged. Issues are described and well labeled.

There is no “good first issue” label, but there is a “low-hanging-fruit” issue.

There seem to be stale issues in the bug system dating back to 2014.

New contributor getting started content

Procedures, tools, and support for contributing to the project are documented.

The community is accessible from the documentation website, though not through page menus or headings.

There is a contributor Getting Started document.

Project governance documentation

Information about project governance is not immediately apparent on the documentation website or in the repository.

Recommendations

Beginner friendly issue backlog

Importance	Effort Level
Low	Low

Add a “good first issue” label or its equivalent.

Project governance documentation

Importance	Effort Level
Medium	Low

Document the project governance rules and procedures in the repository. If they are the same as OpenStack’s, provide a link.

3. Website

Comments

The website analysis is abbreviated with respect to the CNCF rubric. Most aspects of the website implementation and hosting are prescribed by OpenStack. Issues with web infrastructure such as search result quality and indexing of subdocuments is outside the scope of this analysis, namely, the quality of the Ironic documentation. Consequently, comments are limited to casual observation only.

Single-source requirement

Ironic documentation page source is contained in a doc directory in the main Ironic OpenDev repo. This seems to be customary throughout the OpenStack projects. Likewise, the REST API documentation has its own directory in the same repo. While not in a separate website repo, this is adequate for maintaining the docs and meets the spirit CNCF single-source requirement.

Usability, accessibility and devices

Most static page generation tools deliver documentation in a framework that adapts to the viewing device. The following criteria are delivered by OpenStack's use of the Sphynx framework.

The website is usable from a mobile device. Doc pages are readable; layout is reasonable on small devices. Tables of contents and search are accessible on mobile devices (though search is indiscriminate – see Content Maintainability). A mobile-first design does not make sense for Ironic or for OpenStack in general.

Accessibility features seem basic and unimproved: - Color contrasts are adequate (using OpenStack's framework and theme) - Website features can be accessed with keyboard only (though this is cumbersome) - Text-to-speech experience is untested.

Branding

Branding is completely consistent with OpenStack's branding, since Ironic conforms to OpenStack's framework.

Case studies/social proof

The Ironic documentation follows OpenStack's documentation framework and offers no case study or social proof elements. This content is available elsewhere, for example through the OpenInfra Project and OpenStack.org.

Maintenance planning

Website maintenance is the same as for all other OpenStack documentation.

- The website tooling, Sphinx, is supported by the OpenStack community.
- Website maintainers do not seem abundant within the community.
- The site builds in a reasonable time on a local machine.
- Site maintainers presumably have adequate permissions. If not, the project's maintainers are responsive to requests for access.

Other

The website is accessible via HTTPS.

HTTP requests redirect automatically to HTTPS.

The PDF version of the documentation is a nice-to-have (generated essentially for free by the Python Sphinx setup, I'm guessing). The PDF version is not branded; in fact, it seems completely generic. I'm guessing it uses default formatting. This seems to be the case for all OpenStack documentation.

Recommendations

No recommendations.

Appendices

Appendix A: Conformance to OpenStack documentation

This section is a summary of the structure prescribed by OpenStack for integrating product documentation with its documentation set.

Ironic is constrained to follow the OpenStack documentation requirements. This means that any Ironic doc pages linked from the OpenStack documentation portal must be one of the approved OpenStack top-level portal pages.

The OpenStack portal pages are briefly described below. Except for the Contributor Guide, all the listed pages can be included on the OpenStack main documentation portal by setting a flag in a master doc configuration file. OpenStack projects are not required to include these pages, but they're the only ones allowed on the OpenStack doc portal.

- **Installation Guide**: Everything to do with installing from packages, including dependencies, database requirements, and configuration. Production installation is not documented here. *??Where is it documented – Admin Guide? ??*
- **Configuration reference**: Automatically (generated by oslo.config) and manually written reference information.
- **CLI reference**: Automatically generated (using cliff Sphinx integration) or manually written CLI command line tool reference documentation.
- **Administrator Guide**: Information about how to configure and operate the software, including the project's administrator guide.
- **User Guide**: Information for end users, including (but not limited to) instructional material like procedures, tasks, run books, and tutorials; conceptual information like technical overviews; API references.
- **Reference Guide**: Reference information not included elsewhere.
- **API reference**: The REST API reference, if one exists for the product.

The OpenStack documentation guidelines also recommend creating a Contributor Guide that is specific to the project:

- **Contributor Guide**: Information for project contributors. This guide is never linked from the OpenStack documentation portal.

The complete OpenStack doc requirements are here.

The OpenStack documentation requirements do not specify the content of project documentation aside from the guidelines above and the names of the top-level doc page "hooks". The documentation can therefore be arranged however makes the most sense for the project behind the prescribed top-level landing pages. If necessary, other landing pages can be written, they just won't be included in the OpenStack doc portal.

Appendix B: Glossary

Bare Metal

1. Describes a physical server running application software with no intervening virtualization layer.

2. In the OpenStack community, sometimes used as a synonym for Ironic (1). Usually phrased "Bare Metal service".

Bare Metal Service

Same as Bare Metal (2).

Conceptual documentation

An explanatory documentation topic. Examples include technical overviews, state diagrams, and background information.

Documentation topic

A unit of documentation covering a single information topic. Examples include a single task; an API reference, or in some cases a single API; and an explanation of a single concept. A topic can be a single web page, though in practice a page often contains multiple topics.

Image

A disk image. The image is copied to the server and becomes the volume used by the new bare metal instance.

Instructional documentation

Documentation topics that provide "how-to" information, including procedures, tasks, and tutorials.

Imperative

Presenting a direct instruction. The following are examples of *imperative* sentences:

> Create an Ironic database that is accessible by the user.
>
> Update the node's `driver_info` field.

IPMI

Intelligent Platform Management Interface. A specification for a subsystem that autonomously manages and monitors a computer independently of the computer's CPU.

Ironic

1. OpenStack's bare-metal server provisioning service.
2. Poignantly contrary to what was expected or intended.
3. Self-consciously or in parody, usually as a nihilistic form of social commentary.

OpenStack

A blanket project for a collection of open source cloud infrastructure component software. Ironic is an OpenStack component.

OSS

Open-source software.

PXE

Preboot Execution Environment. A specification that enables a computer's BIOS and NIC to boot a computer from the network rather than a disk.

Reference documentation

A topic that lists, defines, and explains options, parameters, functions, and so on of a system such as an API or CLI. A reference is exhaustive, as distinct from an example.

TOC

Table of contents

Topic

A documentation topic.