

Rubin Observatory

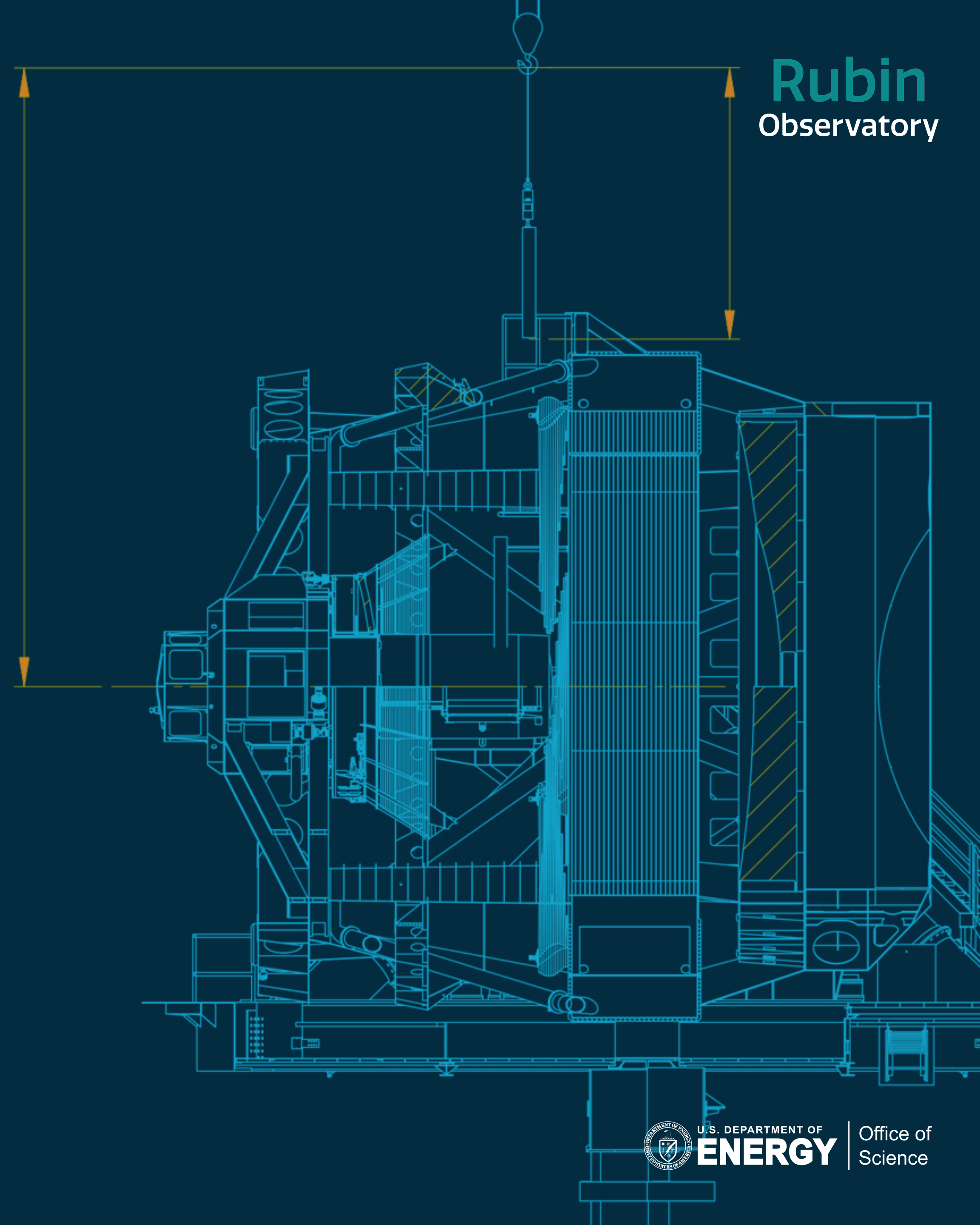
Documentation Contribution Primer

Jonathan Sick
Slack/GitHub: @jonathansick

Rubin Observatory Operations Boot Camp | October 14, 2020



Docs *or it didn't happen*



Rubin Observatory Operations Boot Camp | October 14, 2020



U.S. DEPARTMENT OF
ENERGY

Office of
Science

What you will learn

- Gain a broad understanding of our documentation system and patterns.
- Learn how to write technical notes.
- Learn how to contribute to user guides and software documentation.

Outline

1. Documentation system
2. Creating documents and technical notes
3. ReStructuredText primer
4. Writing user guides and Python software documentation
5. Python API documentation (Numpydoc docstrings)
6. Content style guide

Documentation system concepts: Docs-like-code

- User guides for a software system are embedded in the software's GitHub repository (or in their own GitHub repository if the documentation covers multiple software repositories).
 - *Documentation is naturally versioned with the underlying software.*
 - *Documentation contribution review using Pull Requests.*
 - *Git and GitHub skills are necessary for technical writing.*
- Sphinx ([sphinx-doc.org](https://www.sphinx-doc.org)) is our main tool for building documentation sites.
 - *ReStructuredText is the primary format for writing content, along with Jupyter Notebooks for tutorials.*
 - *SQuaRE builds custom Sphinx extensions to improve doc production; inquire at #dm-docs on Slack.*
- Continuous integration (CI) and deployment (CD)
 - *Documentation sites are tested, built, and deployed to the web from the CI/CD server configured for a project (e.g. Jenkins or GitHub Actions).*
- Documentation is hosted on lsst.io, a platform powered by *LSST the Docs*.
 - *Documentation is publicly available on the web.*

Documentation system concepts: lsst.io

- Documentation portal for search and browsing.
 - www.lsst.io
- Each documentation project is a subdomain.
 - E.g. pipelines.lsst.io, nb.lsst.io
- The "main" version of any documentation project is served from the root path.
 - E.g. pipelines.lsst.io/
 - The meaning of "main" is configurable, per-project
- Alternative versions are served with `/v/{version}/` path prefixes.
 - E.g. pipelines.lsst.io/v/daily/, pipelines.lsst.io/v/v20_0_0/
- Dashboard for versions.
 - E.g. pipelines.lsst.io/v/

www.lsst.io

Documentation portal

The screenshot shows the Rubin Observatory website. At the top, there's a navigation bar with the "Rubin Observatory" logo, links for "Advanced search" and "About", and a "Dark mode" toggle. Below the navigation bar is a large hero image of the observatory at night with the text "Find Rubin Observatory technical docs and software." and a search bar. The "Featured guides" section includes two cards: "LSST Science Pipelines" and "Rubin Science Platform Notebook Aspect User Guide". The "Documents" section provides a search bar and a grid of document categories.

Rubin Observatory [Advanced search](#) [About](#) ☐ Dark mode

Find Rubin Observatory technical docs and software.

Featured guides

LSST Science Pipelines

The LSST Science Pipelines are designed to enable optical and near-infrared astronomy in the "big data" era. While they are being developed to process the data for the Rubin Observatory Legacy Survey of Space and Time (Rubin's LSST), our command line and programming interfaces can be extended to address any optical or near-infrared dataset.

[lsst/pipelines_lsst_io](#)

Rubin Science Platform Notebook Aspect User Guide

The Notebook Aspect enables you to do your science at the LSST Data Facility, with the LSST Science Pipelines, a full suite of development tools, and your own Python code. The Notebook Aspect is powered by the JupyterLab project.

[lsst-dm/nb_lsst_io](#)

Documents

[Search in Rubin Observatory technical documents](#), or browse by series:

DMTN Data Management Technotes	DMTR Data Management Test Reports	ITTN IT Technotes	LDM LSST Data Management
LPM	LSF	PSTN	RTN

Key types of documentation

Controlled documents

<https://developer.lsst.io/project-docs/change-controlled-docs.html>

- Requirements, specifications, design, test reports, and policies.
- Are approved by a control board.
- Officially published in DocuShare (docushare.lsstcorp.org)
- Many (but not all!) such documents are drafted on GitHub/lsst.io using LaTeX.

Series: LPM, LSE, LDM, etc.

Technical notes

<https://developer.lsst.io/project-docs/technotes.html>

- A medium for ground-up technical communication to your team, other teams, and beyond: Ideas, designs, proposals, experiments, literature reviews, and more. Even paper preprints.
- Combines the organization of a document archive with the immediacy and low overhead of a wiki.
- Migrate to guides (→) to document living systems and products.
- RST/Sphinx or LaTeX format

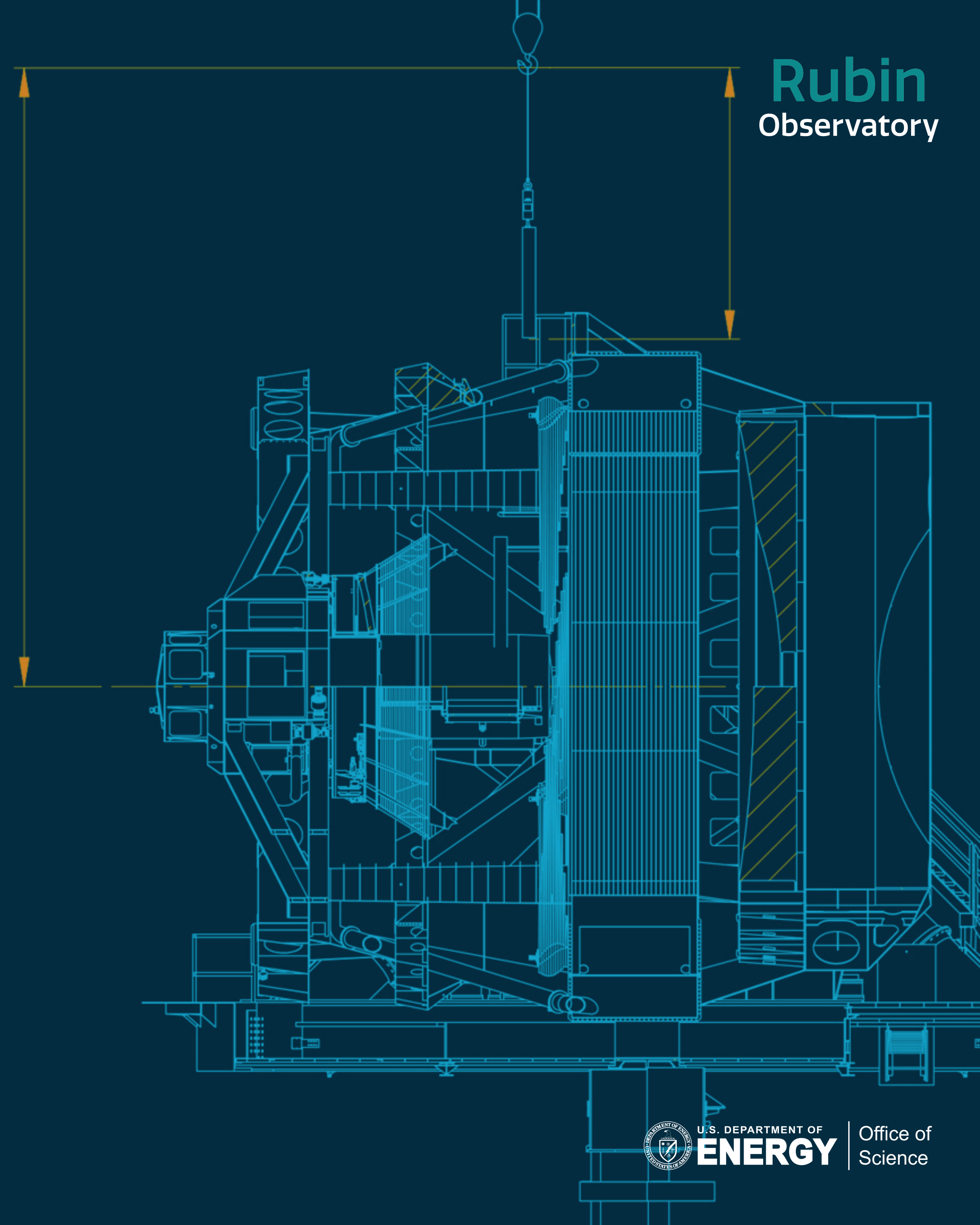
Technote series: DMTN, SQR, TSTN, SMTN, PSTN, RTN, etc.

Guides

- Multi-page websites that document a product, system, or service.
- Usually built with Sphinx (sphinx-doc.org) and Documenteer (documenteer.lsst.io).
- User-facing or developer/team-facing (or both!).
- Content types:
 - Conceptual guides
 - How-tos
 - Tutorials
 - References

Technical notes

Rubin
Observatory



Rubin Observatory Operations Boot Camp | October 14, 2020

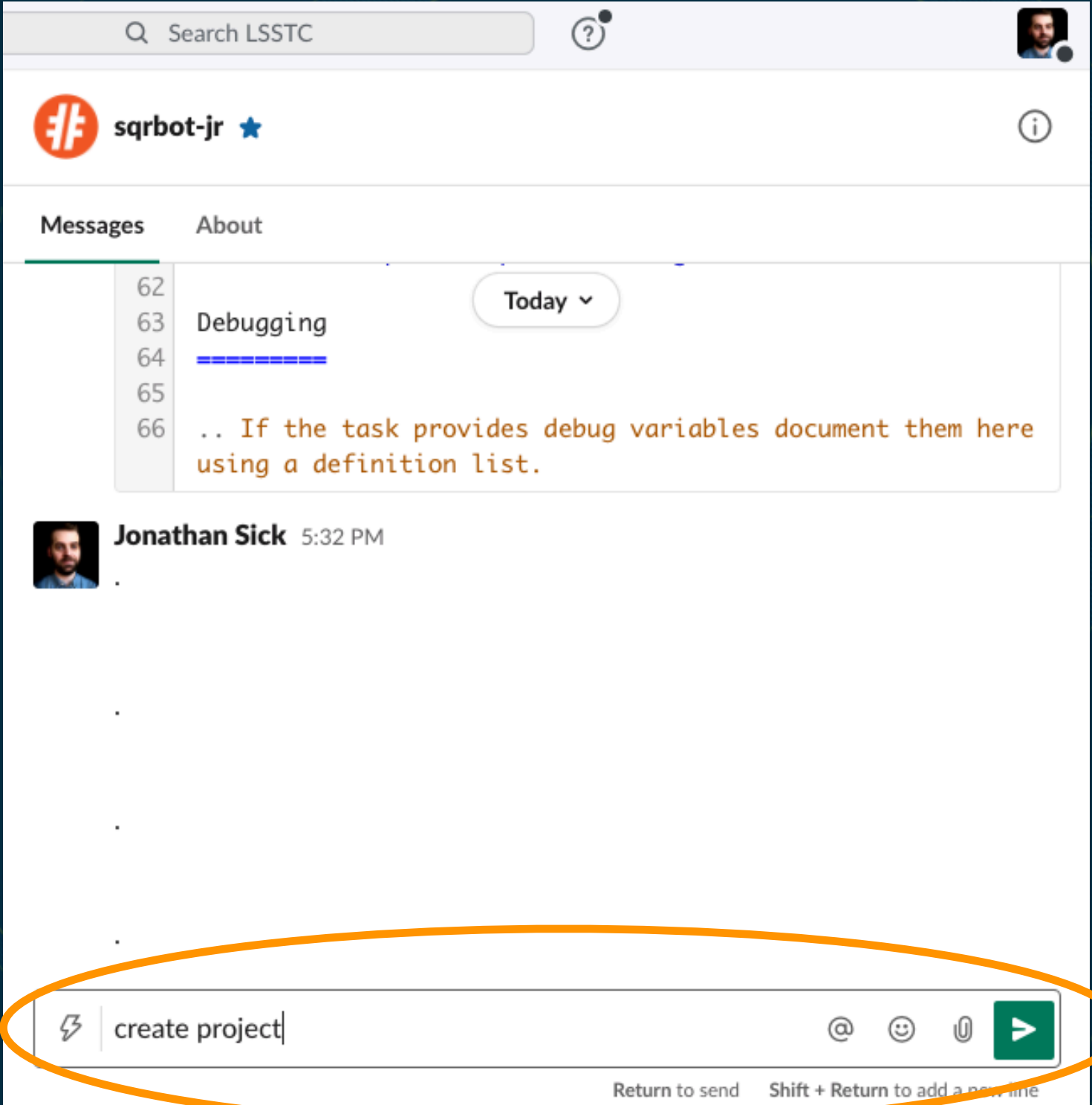


U.S. DEPARTMENT OF
ENERGY

Office of
Science

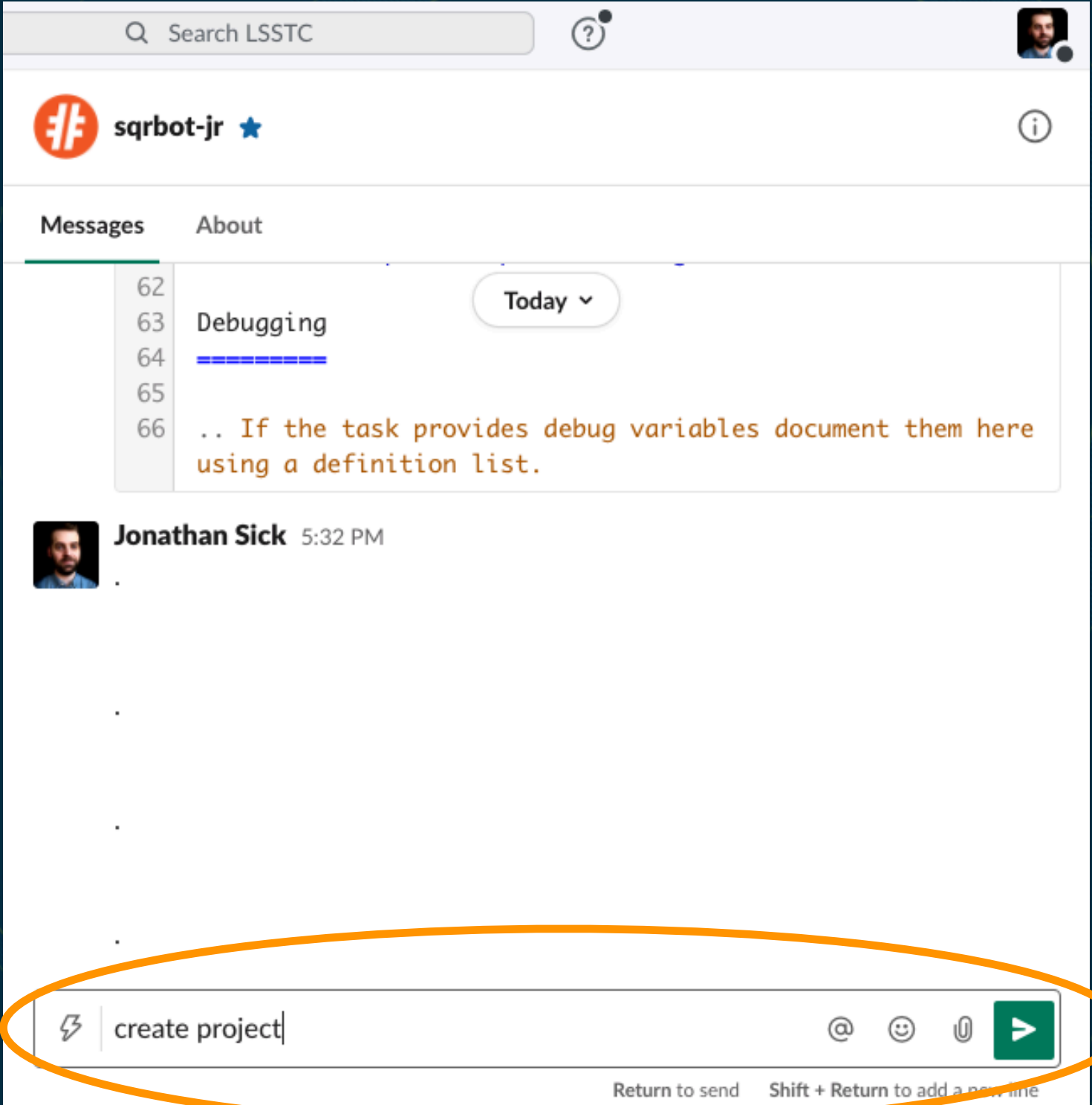
Let's make a technical note

Step 1

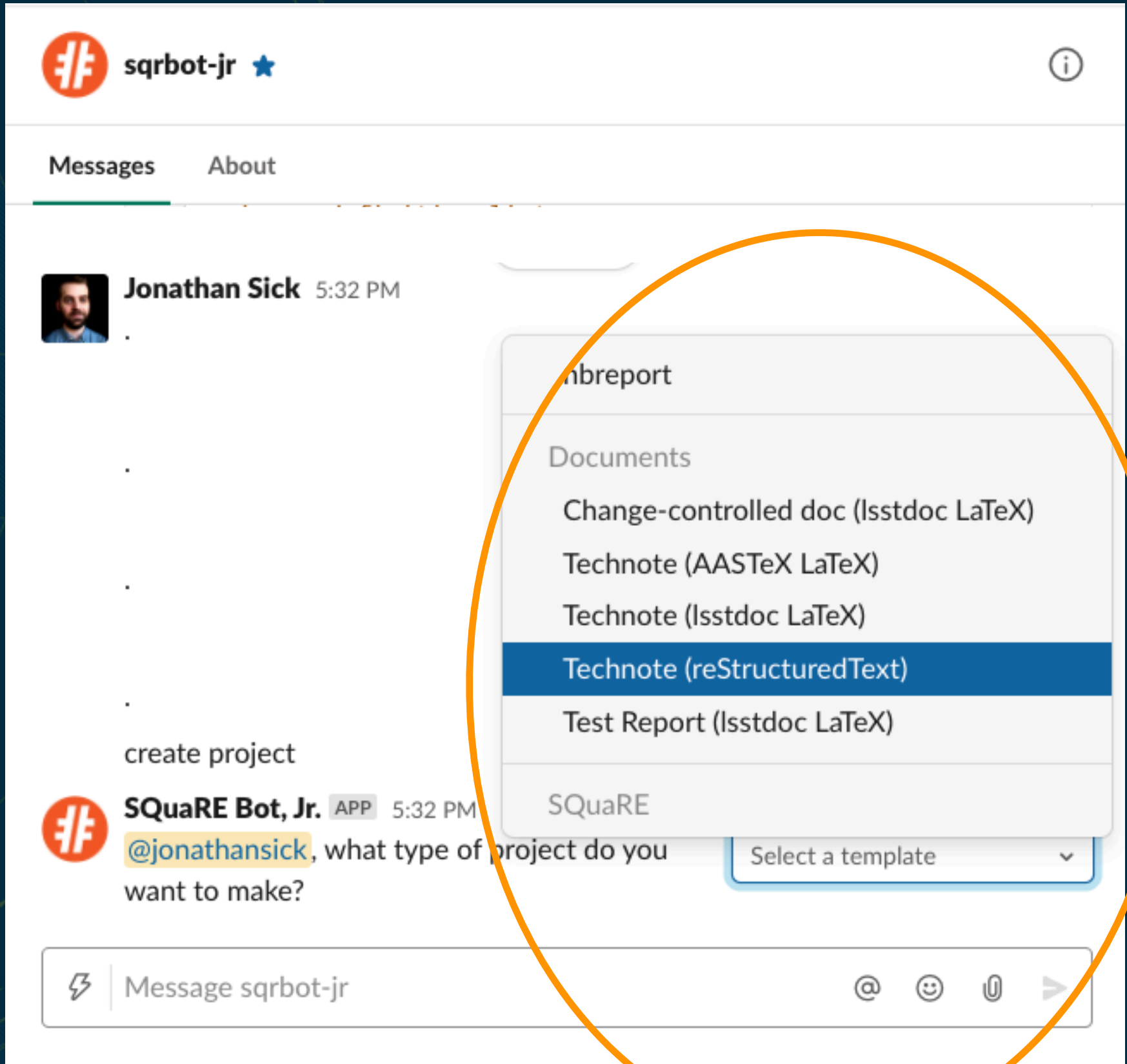


Let's make a technical note

Step 1

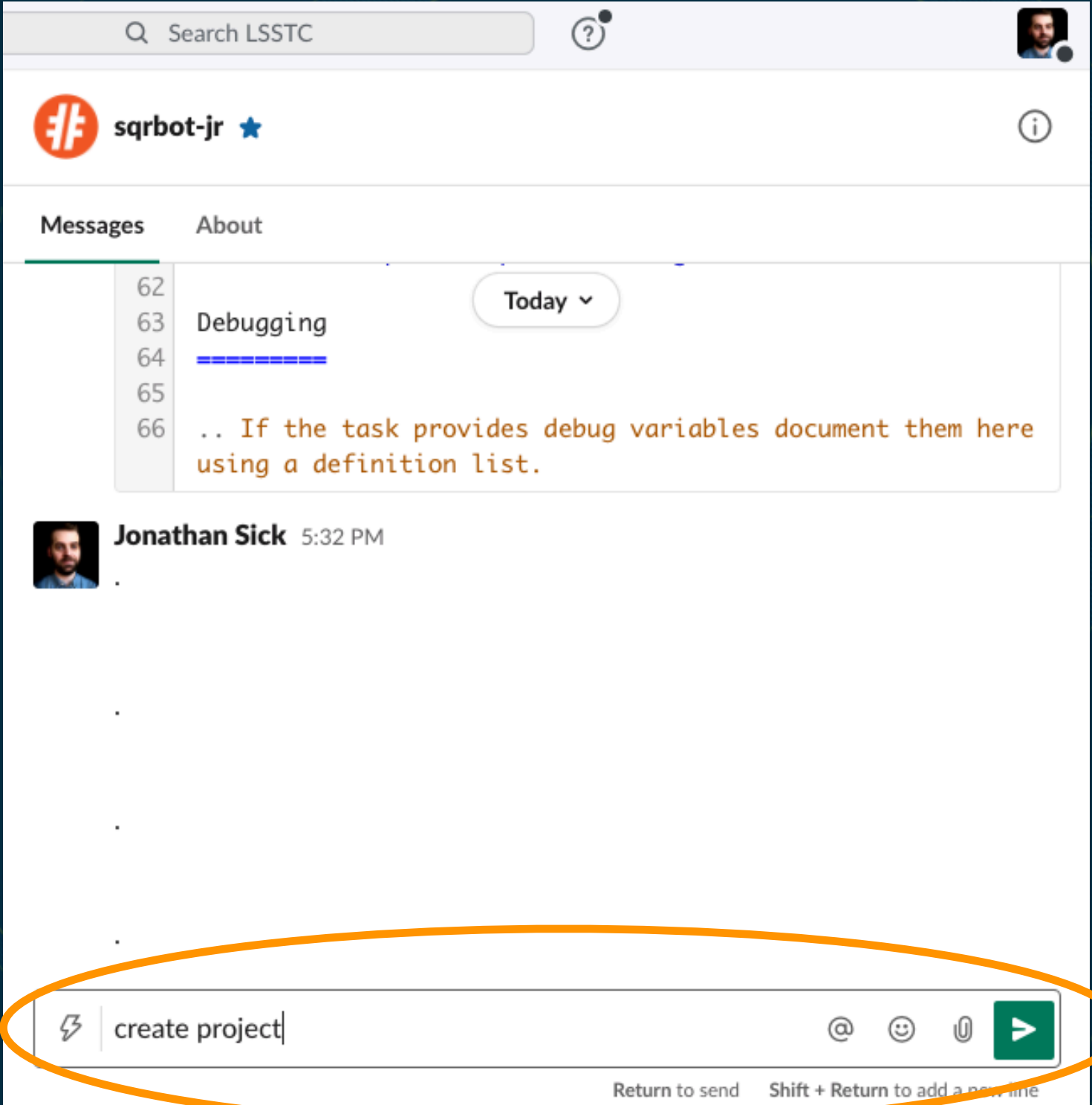


Step 2

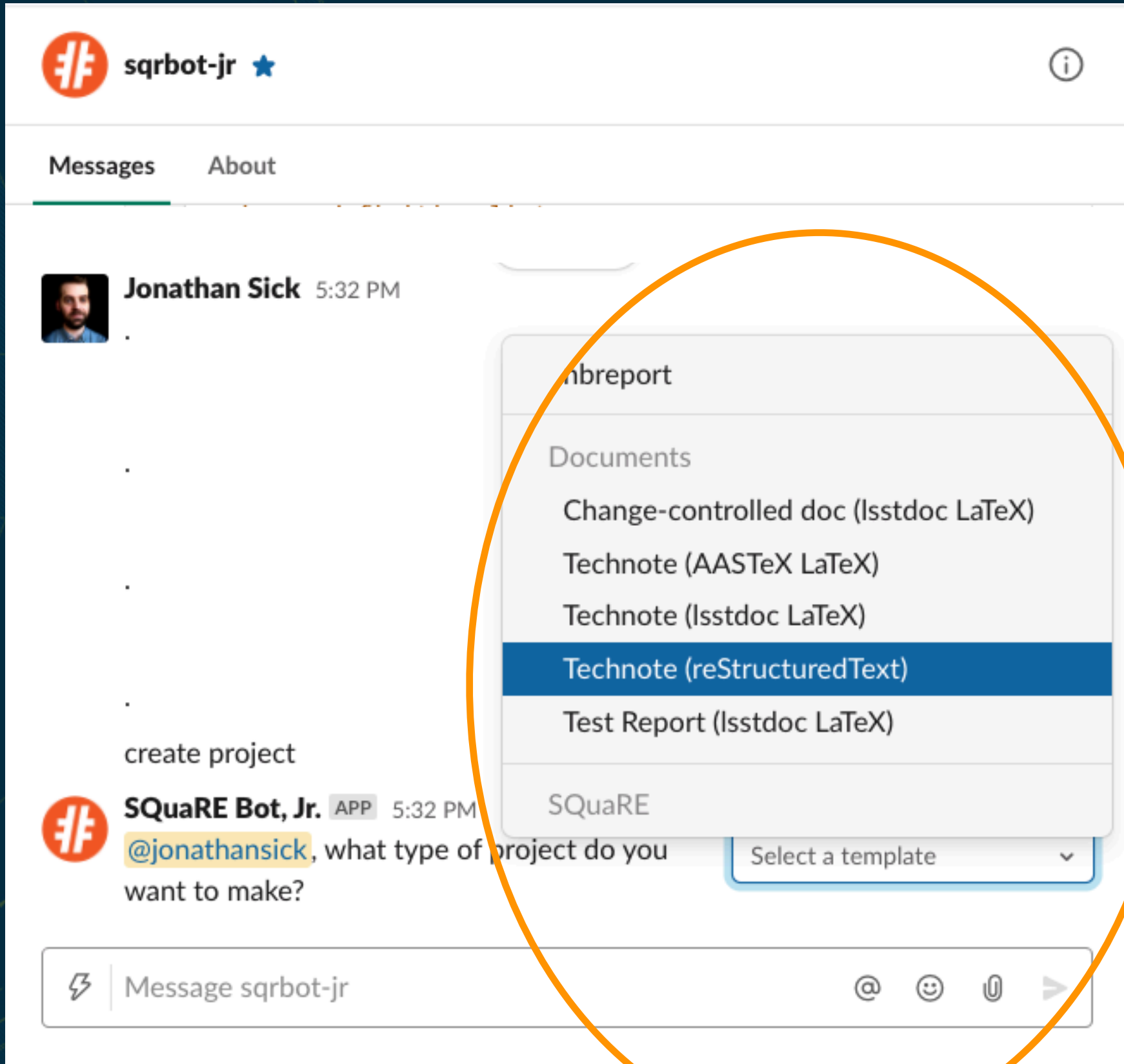


Let's make a technical note

Step 1





Step 2



Step 3

The screenshot shows the "Create an rst technote" form. The form has a title field with the text "Technical note demonstration". Below the title field, there's a note: "Don't include the document's handle and avoid rst markup." The "Abstract or description" field contains the text: "Some of the possible applications for technotes are: report the results of a project, such as a data processing or software development experiment. Announce a new technology, serving as a high-level overview complementing". Below the abstract field, there's a note: "You can use reStructuredText here." The "Series" field has a dropdown menu with the text "Test". The "Initial copyright holder" field has a dropdown menu with the text "Association of Universities for Research in Astronomy, Inc. (AU...". At the bottom, there's a link: "Learn more about SQuaRE Bot, Jr." and two buttons: "Cancel" and "Submit".


Let's make a technical note


**sqrbot-jr** 


Messages





About

create project

**SQuaRE Bot, Jr.** APP 5:32 PM
[@jonathansick](#), got it! I'll start working on your Technote (reStructuredText) repository right away. I'll keep you updated!


 **3 replies** Last reply today at 5:36 PM

 Message sqrbot-jr





Thread

sqrbot-jr

**SQuaRE Bot, Jr.** APP Today at 5:32 PM
[@jonathansick](#), got it! I'll start working on your Technote (reStructuredText) repository right away. I'll keep you updated!


3 replies

**SQuaRE Bot, Jr.** APP < 1 minute ago
[@jonathansick](#), the documentation URL will be:
<https://testn-100.lsst.io>.
That page will give a 404 error until the first build completes. Hold tight!

**SQuaRE Bot, Jr.** APP < 1 minute ago
[@jonathansick](#), the new repository is:
<https://github.com/lsst-sqre-testing/testn-100>
You can start working on it!
If I have any extra work to do, I'll send a PR and let you know in this thread. (edited)

lsst-sqre-testing/testn-100
Technical note demonstration

Website	Last updated
https://testn-100.lsst.io	a few seconds ago

 lsst-sqre-testing/testn-100 | Today at 5:36 PM | Added by GitHub

github.com

Search or jump to...

Pull requests

Issues

Marketplace

Explore

+

+

+

lsst-sqre-testing / testn-100

Watch

0

Star

0

Fork

0

<> Code

Issues

Pull requests

Actions

Security

Insights

Settings

master

1 branch

0 tags

Go to file

Add file

Code

jonathansick and SQuaRE Bot Initial commit

024969d 2 minutes ago

1 commits

.github/workflows	Initial commit	2 minutes ago
_static	Initial commit	2 minutes ago
lsstbib	Initial commit	2 minutes ago
.gitignore	Initial commit	2 minutes ago
COPYRIGHT	Initial commit	2 minutes ago
LICENSE	Initial commit	2 minutes ago
Makefile	Initial commit	2 minutes ago
README.rst	Initial commit	2 minutes ago
conf.py	Initial commit	2 minutes ago
index.rst	Initial commit	2 minutes ago
local.bib	Initial commit	2 minutes ago
metadata.yaml	Initial commit	2 minutes ago
requirements.txt	Initial commit	2 minutes ago

README.rst

testn-100 lsst.io CI passing

Technical note demonstration

TESTN-100

Some of the possible applications for technotes are: report the results of a project, such as a data processing or software development experiment. Announce a new technology, serving as a high-level overview complementing user documentation. Propose an architecture, possibly becoming the subject of a request for comment (RFC).

Links:

- Publication URL: <https://testn-100.lsst.io>
- Alternative editions: <https://testn-100.lsst.io/v>
- GitHub repository: <https://github.com/lsst-sqre-testing/testn-100>
- Build system: <https://github.com/lsst-sqre-testing/testn-100/actions/>

About

Technical note demonstration

[testn-100.lsst.io](#)

Readme

CC-BY-4.0 License

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)


Languages

TeX 99.3%

Other 0.7%

Rubin
Observatory

11

U.S. DEPARTMENT OF
ENERGY

Office of
Science

github.com

Search or jump to...

Pull requestsIssuesMarketplaceExplore

🔔+👤

lsst-sqre-testing / testn-100

Watch0Star0Fork0

<> CodeIssuesPull requestsActionsSecurityInsightsSettings

master1 branch0

jonathansick and SQuaRE Bo

.github/workflows

_static

lsstbib

.gitignore

COPYRIGHT

LICENSE

Makefile

README.rst

conf.py

index.rst

local.bib

metadata.yaml

requirements.txt

README.rst

testn-100 lsst.io CI passing

Technical note demo

TESTN-100

Some of the possible applicatio
data processing or software de
a high-level overview complem
becoming the subject of a requ

Links:

• Publication URL: <https://te>

• Alternative editions: <https;>

• GitHub repository: <https://>

• Build system: <https://github>

lsst-sqre-testing / testn-100

Watch0Star0Fork0

<> CodeIssuesPull requestsActionsSecurityInsightsSettings

mastertestn-100 / index.rstGo to file...

jonathansick Initial commit ✓Latest commit 024969d 3 minutes agoHistory

1 contributor

62 lines (38 sloc)1.7 KBRawBlame📄✎🗑

tocdepth:1

Note

This technote is not yet published.

Some of the possible applications for technotes are: report the results of a project, such as a data processing or software development experiment. Announce a new technology, serving as a high-level overview complementing user documentation.

Propose an architecture, possibly becoming the subject of a request for comment (RFC).

© 2020 GitHub, Inc.

TermsPrivacySecurityStatusHelpContact GitHubPricingAPITrainingBlogAbout

lsst-sqre-testing / testn-100

CodeIssuesPull requests

master1 branch0 forks

jonathansick and SQuaRE Bot

.github/workflows

_static

lsstbib

.gitignore

COPYRIGHT

LICENSE

Makefile

README.rst

conf.py

index.rst

local.bib

metadata.yaml

requirements.txt

README.rst

lsst-sqre-testing / testn-100

CodeIssuesPull requests

master1 contributor

jonathansick Initial commit

62 lines (38 sloc)1.7 KB

tocdepth: 1

Note

This technote is not yet published.

Some of the possible applications for development experiment. Announce a

Propose an architecture, possibly bec

lsst-sqre-testing / testn-100

CodeIssuesPull requestsActionsSecurityInsightsSettings

testn-100 / index.rst

Edit filePreview changes

1..

2Technote content.

3

4See <https://developer.lsst.io/restructuredtext/style.html>

5for a guide to reStructuredText writing.

6

7Do not put the title, authors or other metadata in this document;

8those are automatically added.

9

10Use the following syntax for sections:

11

12Sections

13=====

14

15and

16

17Subsections

18-----

19

20and

21

22Subsubsections

23^^^^^^^^

24

25To add images, add the image file (png, svg or jpeg preferred) to the

26_static/ directory. The reST syntax for adding the image is

27

28.. figure:: _static/filename.ext

29:name: fig-label

30

31Caption text.

32

33Run: ``make html`` and ``open _build/html/index.html`` to preview your work.

34See the README at <https://github.com/lsst-sqre/lsst-technote-bootstrap> or

35this repo's README for more info.

36

Commit changes

Update index.rst

Add an optional extended description...

jsick@lsst.org

Choose which email address to associate with this commit

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests](#)

lsst-sqre-testing / testn-100

CodeIssuesPull requests

master1 branch0

jonathansick and SQuaRE Bo

.github/workflows

_static

lsstbib

.gitignore

COPYRIGHT

LICENSE

Makefile

README.rst

conf.py

index.rst

local.bib

metadata.yaml

requirements.txt

README.rst

testn-100 lsst.io CI passing

Technical note demo

TESTN-100

Some of the possible applicatio
data processing or software de
a high-level overview complem
becoming the subject of a requ

Links:

Publication URL:

GitHub repository:

lsst-sqre-testing / testn-100

CodeIssuesPull request

mastertestn-100 / index.rst

jonathansick Initial commit

1 contributor

62 lines (38 sloc)1.7 KB

tocdepth:1

Note

This technote is not yet published.

Some of the possible applications for
development experiment. Announce a

Propose an architecture, possibly bec

© 2020 GitHub, Inc. TermsPrivacy

lsst-sqre-testing / testn-100

CodeIssuesPull requests

mastertestn-100 / index.rst

jonathansick Initial commit

1 contributor

62 lines (38 sloc)1.7 KB

Edit filePreview changes

1..

2Technote content.

3

4See <https://developer.lsst.io/restr>

5for a guide to reStructuredText writ

6

7Do not put the title, authors or ot

8those are automatically added.

9

10Use the following syntax for section

11

12Sections

13=====

14

15and

16

17Subsections

18-----

19

20and

21

22Subsubsections

23^^^^^^^^^^^^

24

25To add images, add the image file (

26_static/ directory. The reST syntax

27

28.. figure:: /_static/filename.ext

29:name: fig-label

30

31Caption text.

32

33Run: ``make html`` and ``open _bui

34See the README at <https://github.c>

35this repo's README for more info.

36

Commit changes

Update index.rst

Add an optional extended descript

jsick@lsst.org

Choose which email address to associate

Commit directly to the master

Create a new branch for this

testn-100.lsst.io

TESTN-100: Technical note demonstration

Jonathan Sick

Latest Revision: 2020-10-13

Note

This technote is not yet published.

Some of the possible applications for technotes are: report the results of a project, such as a
data processing or software development experiment. Announce a new technology, serving as
a high-level overview complementing user documentation.

Propose an architecture, possibly becoming the subject of a request for comment (RFC).

© Copyright 2020, Association of Universities for Research in Astronomy, Inc. (AURA). Last
updated on Oct 13, 2020.

Built with Sphinx using a theme provided by Read the Docs.

Search or jump to...

lsst-sqre-testing / testn-100

CodeIssuesPull requests

master1 branch0

jonathansick and SQuaRE Bot

.github/workflows

_static

lsstbib

.gitignore

COPYRIGHT

LICENSE

Makefile

README.rst

conf.py

index.rst

local.bib

metadata.yaml

requirements.txt

README.rst

testn-100 lsst.io CI passing

Technical note demonstration

TESTN-100

Some of the possible applications for data processing or software development, a high-level overview complete with becoming the subject of a request.

Links:

Publication URL: <https://testn-100.lsst.io>

Alternative editions: <https://testn-100.lsst.io>

GitHub repository: <https://github.com/lsst-sqre/testn-100>

Build system: <https://github.com/lsst-sqre/testn-100>

Search or jump to...

lsst-sqre-testing / testn-100

CodeIssuesPull requests

mastertestn-100 / index.rst

jonathansick Initial commit

1 contributor

62 lines (38 sloc)1.7 KB

tocdepth:1

Note

This technote is not yet published.

Some of the possible applications for development experiment. Announce a becoming the subject of a request.

Propose an architecture, possibly becoming the subject of a request.

© 2020 GitHub, Inc. Terms Privacy

Search or jump to...

lsst-sqre-testing / testn-100

CodeIssuesPull requests

mastertestn-100 / index.rst

jonathansick Initial commit

1 contributor

62 lines (38 sloc)1.7 KB

tocdepth:1

Note

This technote is not yet published.

Some of the possible applications for development experiment. Announce a becoming the subject of a request.

Propose an architecture, possibly becoming the subject of a request.

© 2020 GitHub, Inc. Terms Privacy

LSST

Large Synoptic Survey Telescope

TESTN-100: Technical note demonstration

Edition: master

Switch editions

Edit on GitHub

1..

2Technote content.

3

4See <https://developer.lsst.io/restructuredtext> for a guide to reStructuredText writing.

5

6

7Do not put the title, authors or other metadata in this file; those are automatically added.

8

9

10Use the following syntax for section titles:

11

12Sections

13=====

14

15and

16

17Subsections

18-----

19

20and

21

22Subsubsections

23^^^^^^^^^^^^

24

25To add images, add the image file (e.g., `fig.png`) to the `_static/` directory. The reST syntax is:

26

27

28.. figure:: `/_static/`filename.ext

29:alt: fig-label

30

31Caption text.

32

33Run: `make html` and `open _build/html/index.html` in a web browser.

34See the README at <https://github.com/lsst-sqre/testn-100> for more info.

35

36

jsick@lsst.org

Choose which email address to associate with this commit

Commit directly to the master branch

Create a new branch for this commit

TESTN-100: Technical note demonstration

Jonathan Sick

Latest Revision

Note

This technote is not yet published.

Some of the possible applications for development experiment. Announce a becoming the subject of a request.

Propose an architecture, possibly becoming the subject of a request.

© Copyright 2020 LSST, Inc. All rights reserved. updated on October 1, 2020

Built with Sphinx

LSST

Large Synoptic Survey Telescope

SQR-049: Science Platform token management design

Edition: master

Switch editions

Edit on GitHub

1Abstract

2Scope

3Overview

4Storage

5API

6Web UI

7auth_request API

8References

SQR-049: Science Platform token management design

Russ Allbery

Latest Revision: 2020-09-28

1 Abstract

Authentication tokens will be used by the science platform as web authentication credentials, for API and service calls from outside the Science Platform, and for internal service-to-service and notebook-to-service calls. This document lays out the technical design of the token management component, satisfying the requirements given in SQR-044.

2 Scope

This design covers the token component in isolation. The user management, group management, and quota components will be designed separately. That may result in some changes to this design as the rest of the system is built. If so, this document will be updated accordingly.

In addition to the requirements in SQR-044, see SQR-039 for a discussion of authentication and authorization for the Science Platform.

This specification will be implemented in Gafaelfawr, the authentication and authorization service for the Rubin Science Platform.

2.1 User metadata

Eventually, metadata about the user as opposed to their session or authentication token (full name, group memberships, UID, etc.) will be stored in a separate user management system that will provide an API to retrieve that information given an authentication token. However, that component of the overall identity management system has not yet been built. Instead, authorization is currently reliant on user metadata communicated via OAuth 2.0 or OpenID Connect and encoded in the resulting identity token.

Therefore, as an interim measure, user metadata is associated with each authentication token and stored with it. The token service provides an API to retrieve that metadata. The storage elements and APIs to support this are flagged below and should be considered temporary.

User-created tokens will, temporarily, inherit the user metadata (including group membership) from the session token used to create that user token. This means group membership will be encoded in the session data for that token and the token will have to be reissued to change that information, contrary to the design in SQR-044 and SQR-039. This will be fixed once the user metadata component is available as a separate service.

3 Overview

Kubernetes

Science Platform

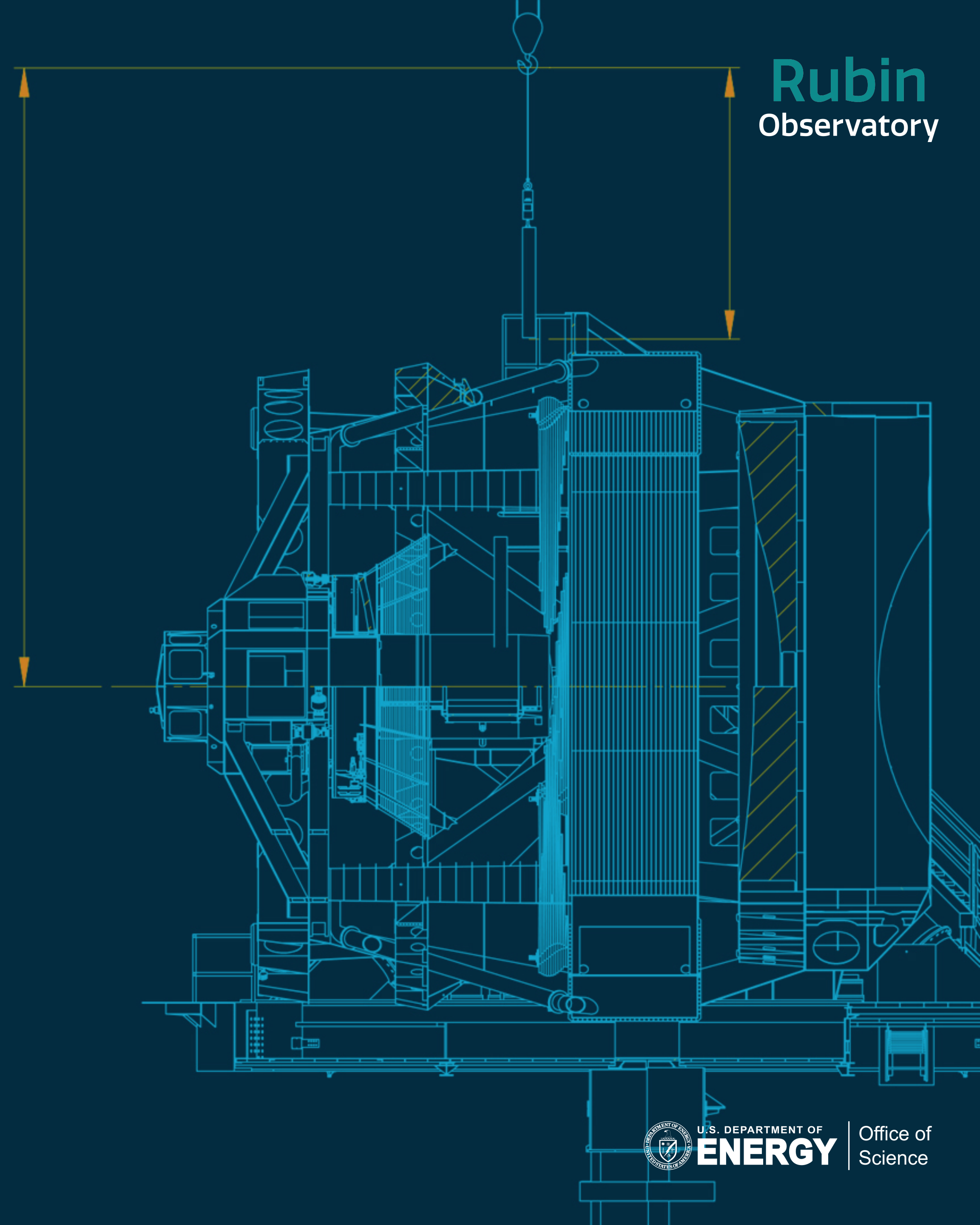
Science Platform

<https://developer.lsst.io/restructuredtext/style.html>

ReStructuredText

ReStructuredText is the format we use for writing most types of documentation:

- Python docstrings
- User guides (pipelines.lsst.io, developer.lsst.io, nb.lsst.io)
- Technical notes



ReStructuredText

Inline syntax

Bold

- ▶ ****bold****

Italic

- ▶ **italic**

Monospace (inline code)

- ▶ ```monospace```



ReStructuredText

Formatting paragraphs

Put a single blank line between paragraphs.

```

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Proin facilisis pharetra neque, at semper nulla mattis auctor.
Proin semper mollis enim eget interdum.

```

```

Mauris eleifend eget diam vitae bibendum.
Praesent ut aliquet odio, sodales imperdiet nisi.
Nam interdum imperdiet tortor sed fringilla.
Maecenas efficitur mi sodales nulla commodo rutrum.
Ut ornare diam quam, sed commodo turpis aliquam et.

```



**One-sentence per line
formatting is strongly
recommended for better
Git diffs.**

ReStructuredText

Sectioning

These under/overline styles define each level of section hierarchy:

```
#####  
Page title  
#####  
  
...  
  
Section heading  
=====  
  
...  
  
Subsection heading  
-----  
  
...  
  
Subsubsection heading  
^^^^^^
```



Page title
...
Section heading
...
Subsection heading
...
Subsubsection heading

ReStructuredText

Linking to sections using the "ref" role

```
.. _sec-a:
```

Section A

=====

See `:ref:`sec-b``.

```
.. _sec-b:
```

Section B

=====

See the `:ref:`previous
section <sec-a>``.

Section A

See Section B.

Section B

See the previous section.

ReStructuredText

Linking to pages using the "doc" role

File tree

```

.
├── page-a.rst
├── page-b.rst
├── tasks
│   ├── task1.rst
│   └── task2.rst
└── tutorials
    └── tutorial-a.rst
  
```

Content in tutorial-a.rst

```

:doc: `../page-a`
:doc: `/page-a`
:doc: `this task <../tasks/task1>`
  
```

Rendered output of tutorial-a.html

[Page A](#)
 → [Page A](#)
[this task](#)

ReStructuredText Hyperlinks

``GitHub <https://github.com>`_.`

You can find the code on GitHub_.



GitHub.

You can find the code on GitHub.

ReStructuredText

Externally-defined hyperlinks

The ``lsst-dm organization`_` on GitHub_.

```
.. _lsst-dm organization: https://github.com/lsst-dm
.. _GitHub: https://github.com
```



The [lsst-dm organization](https://github.com/lsst-dm) on [GitHub](https://github.com).

ReStructuredText

Multiple labels for externally-defined hyperlinks

The ``lsst-dm organization`_` on GitHub_. ``lsst-dm`_`

```
.. _lsst-dm:
.. _lsst-dm organization: https://github.com/lsst-dm
.. _GitHub: https://github.com
```



The lsst-dm organization on GitHub. lsst-dm

ReStructuredText

Link to Python APIs

Link to a function: ``numpy.sin``.
 Link to class: ``astropy.table.Table``.
 Link to a method: ``astropy.table.Table.write``.
 Link to a builtin: ``list``.



Link to a function: [numpy.sin](#).
 Link to a class: [astropy.table.Table](#).
 Link to a method: [astropy.table.Table.write](#).
 Link to a builtin: [list](#).



Because of "Intersphinx" you can link to not only LSST's APIs, but also the Python Standard library and most third-party packages.

ReStructuredText

Link to Python APIs (short forms)

Link to a function: ``~numpy.sin``.

Link to class: ``~astropy.table.Table``.

Link to a method: ``~astropy.table.Table.write``.



Link to a function: [sin](#).

Link to a class: [Table](#).

Link to a method: [write](#).

ReStructuredText

Link to Science Pipelines tasks & configs

```
:lsst-task: `~lsst.pipe.tasks.calibrate.CalibrateTask`  
:lsst-config-field: `~lsst.pipe.tasks.calibrate.CalibrateTask.photoCal`
```



[CalibrateTask](#)
[photoCal](#)

ReStructuredText

Unordered lists

- First item.
- Second item.
- Third item.

Another paragraph for third item.

- Fourth item.

ReStructuredText

Ordered lists

#. First item.

#. Second item.

#. Third item.

Another paragraph for
third item.

#. Fourth item.

1. First item.
2. Second item.
3. Third item ➤

➔ Another paragraph for
third item.

4. Fourth item.

ReStructuredText

Definition lists

Term A
Description of A.

Term B
Description of B:

- nested
- list

Term C
Description of C.

Term A
Description of A.

Term B
Description of B:

- nested
- list

Term C
Description of C.

ReStructuredText

Showing code

```
.. code-block:: python
   :emphasize-lines: 2
```

```
def hello():
    print('Hello world')
```

— or —

```
.. literalinclude:: example.py
   :language: python
   :emphasize-lines: 2
```


ReStructuredText

Tables

```
.. list-table:: Slack channels
   :header-rows: 1

   * - Channel
     - Purpose
   * - ``#dm``
     - General Data Management discussion.
   * - ``#dm-docs``
     - Documentation engineering and writing.
```

 See also "csv-table"

Fig. 1 Slack channels

Channel	Purpose
#dm	General Data Management discusssion.
#dm-docs	Documentation engineering and writing.

ReStructuredText

Math

Write inline expressions: `:math:`\sigma_{\mathrm{mean}} = \sigma / \sqrt{N}``.
Or write block expressions:

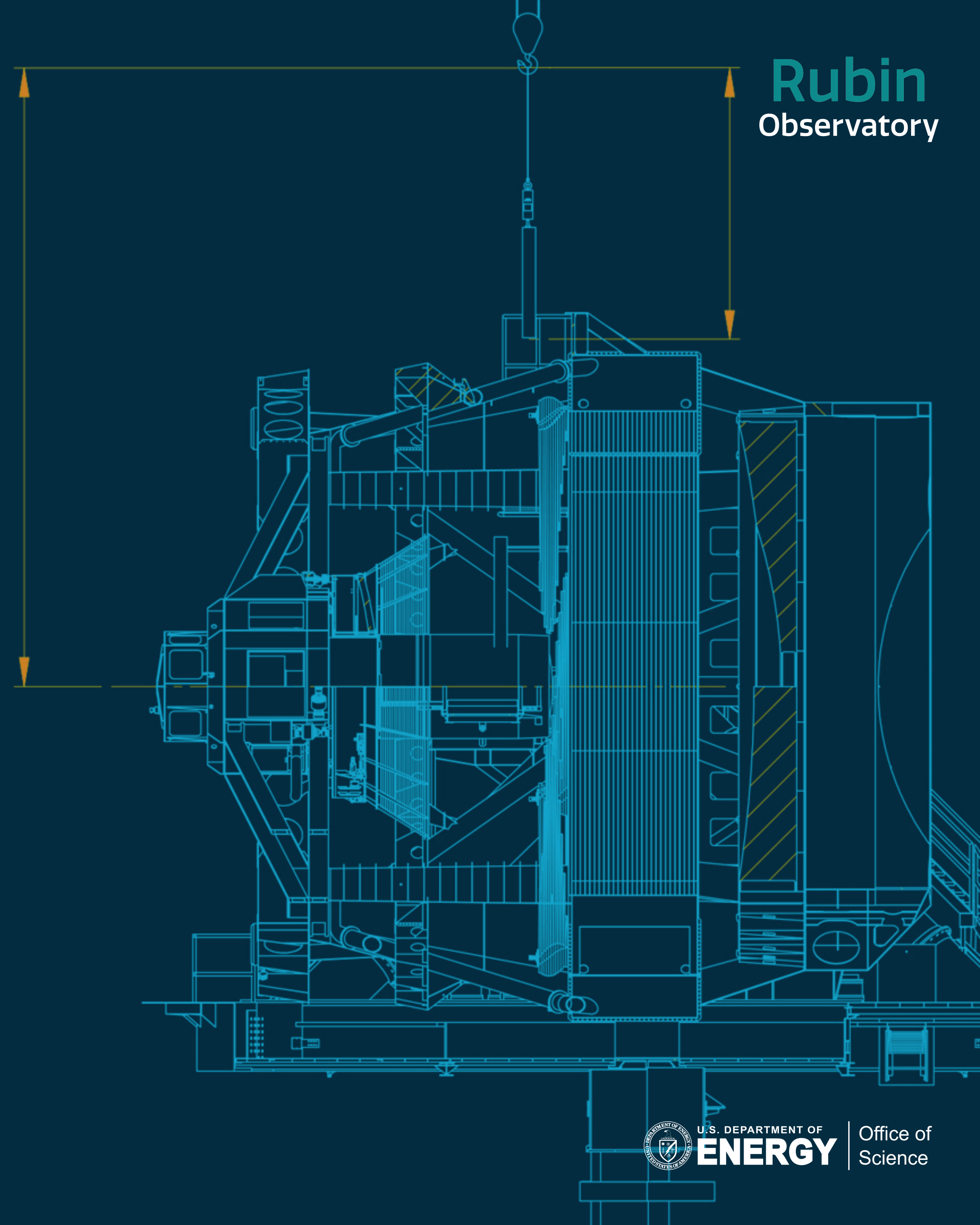
```
.. math:: \sigma_{\mathrm{mean}} = \frac{\sigma}{\sqrt{N}}
```

:label: sigma

```
:eq: `Link to equation <sigma>`.
```


<https://developer.lsst.io/index.html#part-dm-stack>

Writing docs for Science Pipelines



Rubin Observatory Operations Boot Camp | October 14, 2020



U.S. DEPARTMENT OF
ENERGY

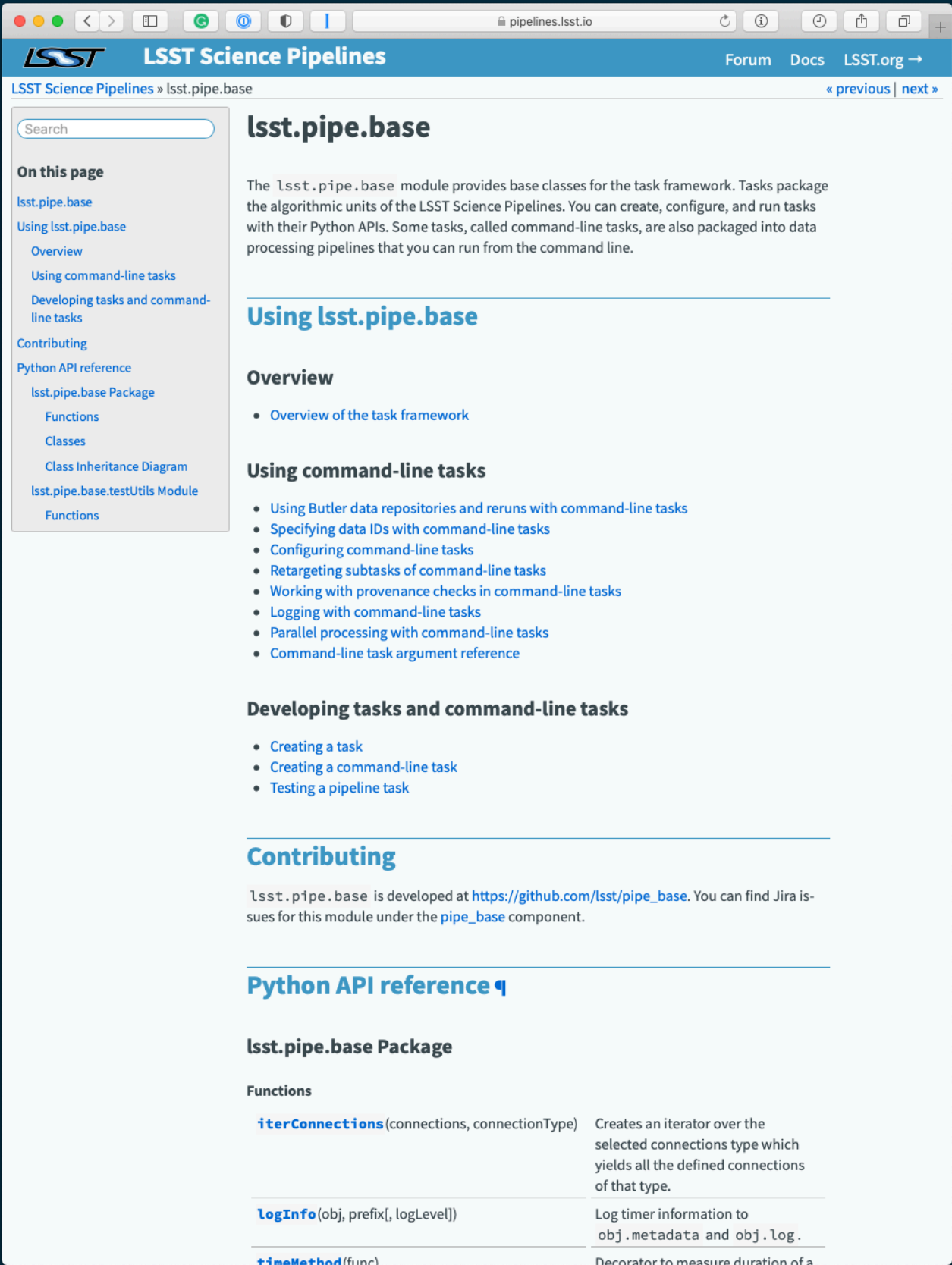
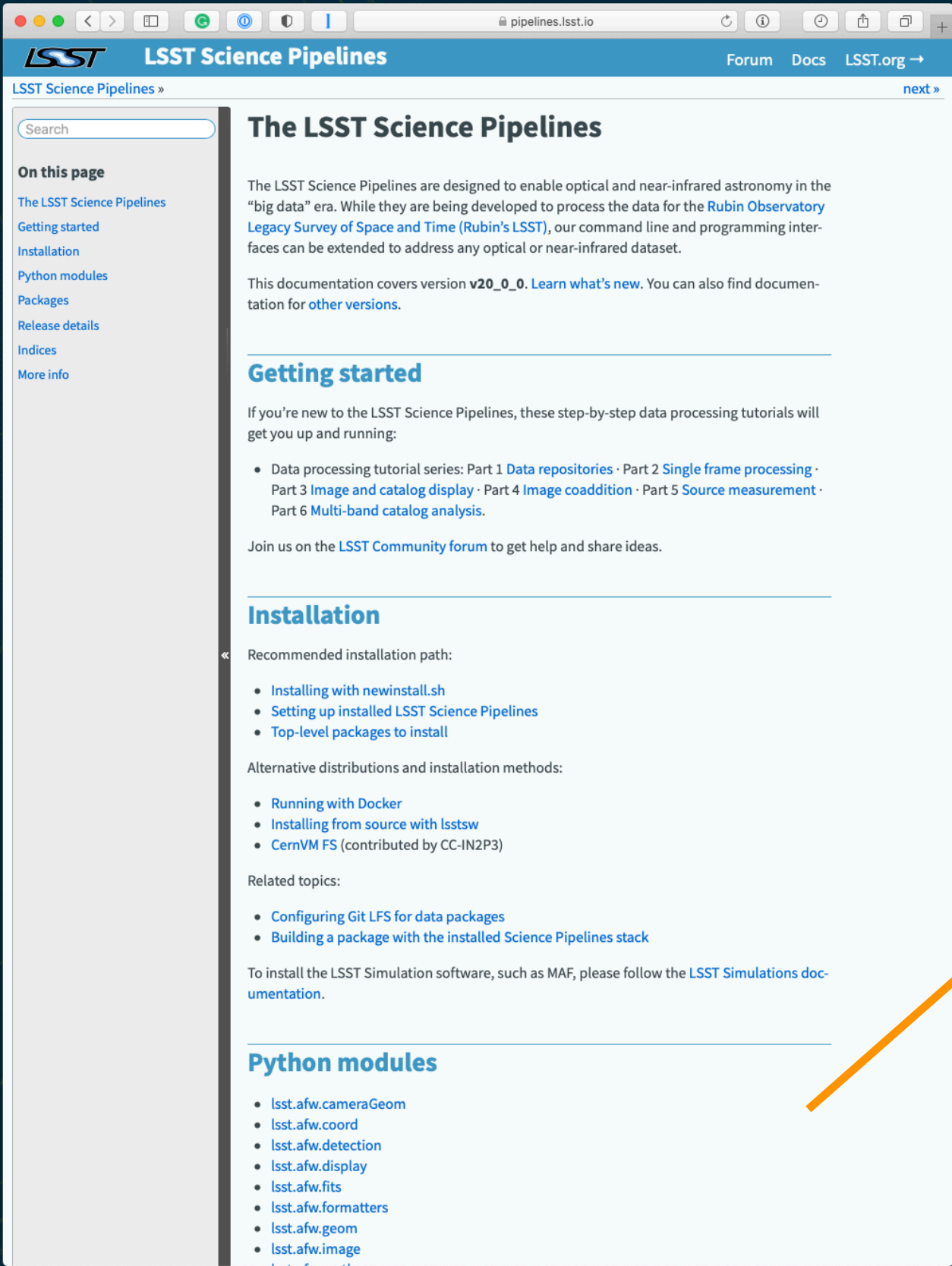
Office of
Science

Science Pipelines documentation

The Science Pipelines documentation project (pipelines.lsst.io) is exceptional because it is built from multiple source repositories:

github.com/lsst/pipelines_lsst_io

[doc/](#) directories of individual packages

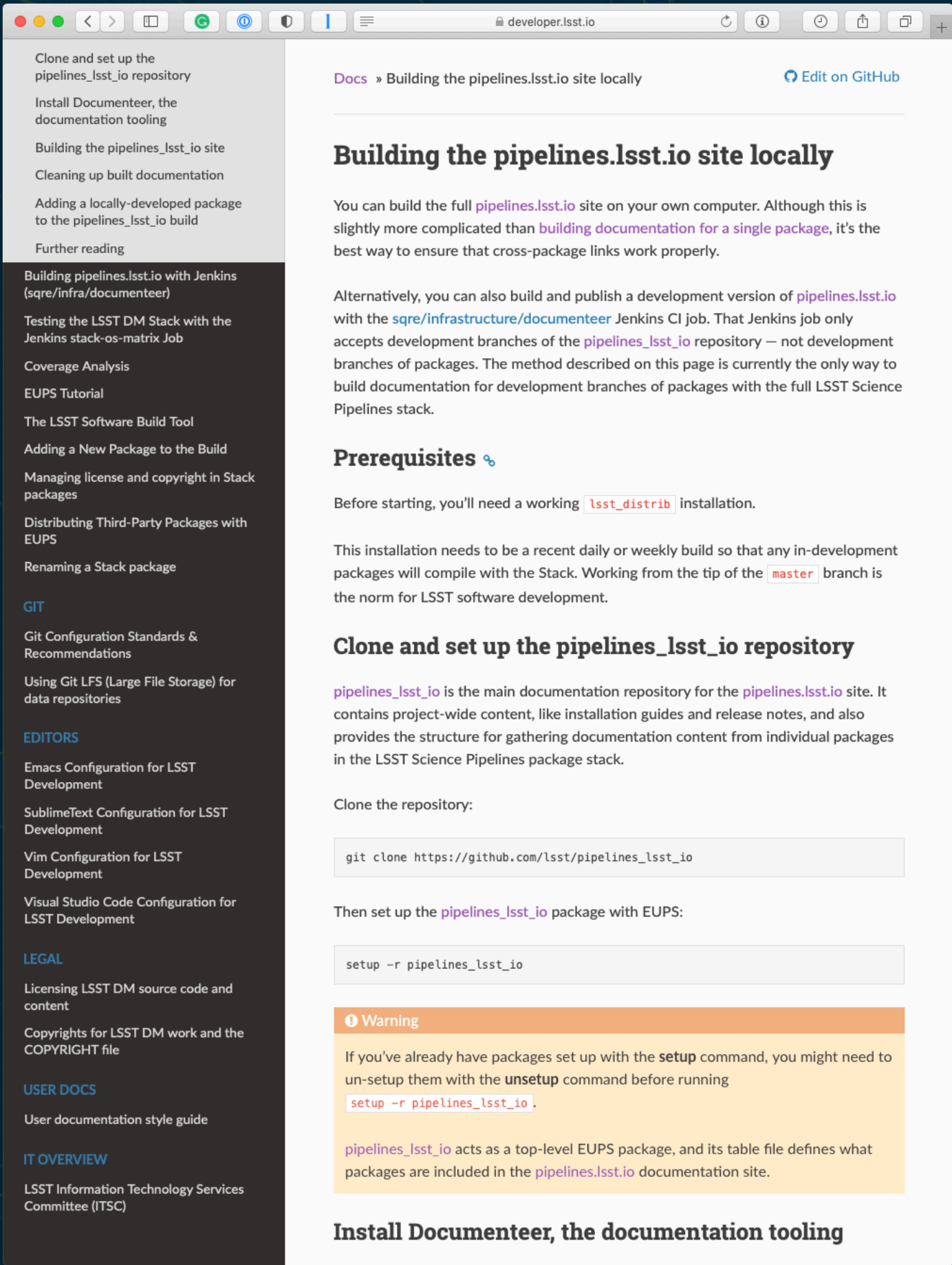


Further reading: <https://developer.lsst.io/stack/documentation-system-overview.html>

Building Science Pipelines documentation

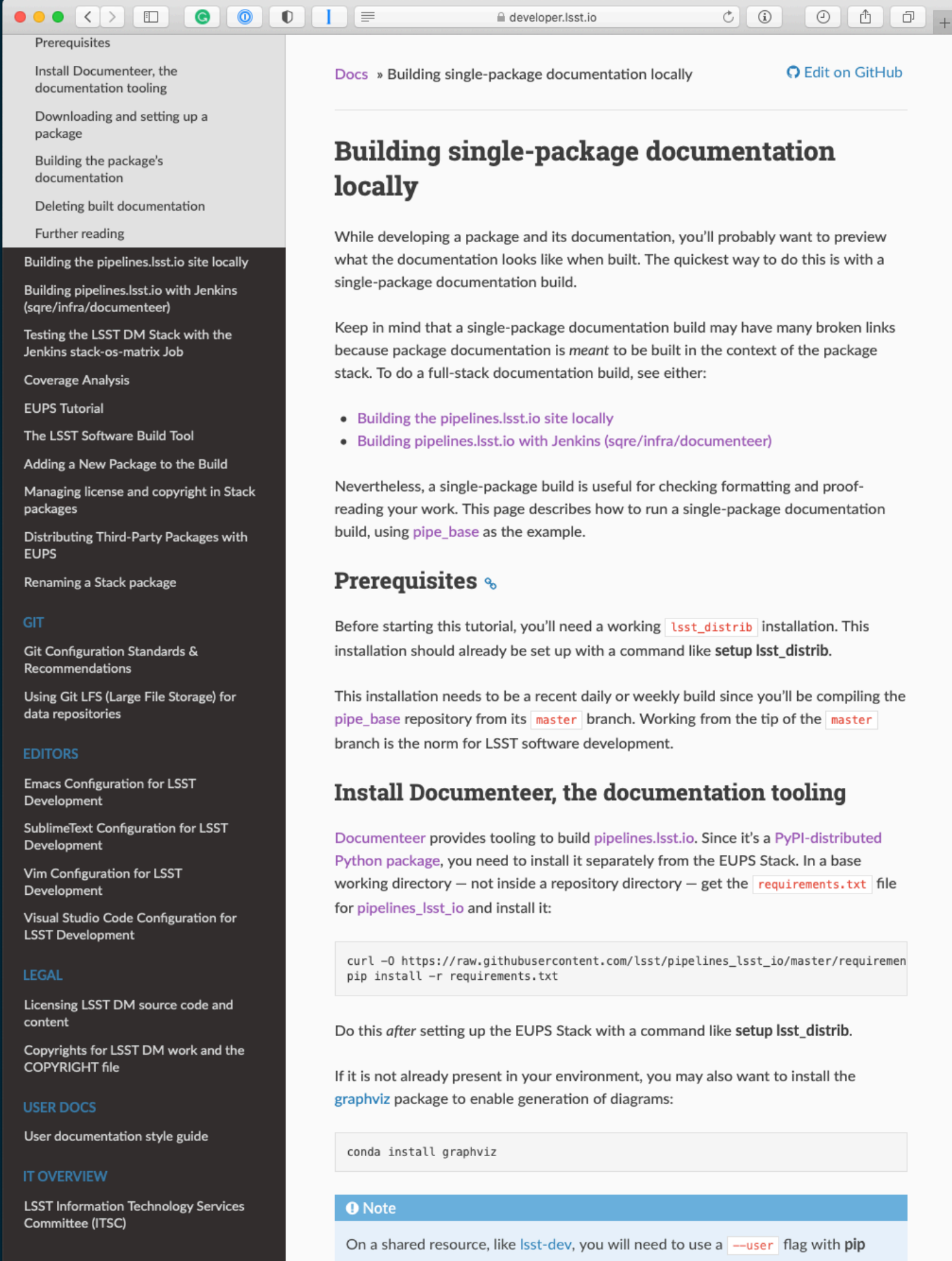
<https://developer.lsst.io/stack/building-pipelines-lsst-io-locally.html>

Mode 1: Building the whole documentation project.



<https://developer.lsst.io/stack/building-single-package-docs.html>

Mode 2: Building single package documentation for tight-loop development.



Science Pipelines documentation

The doc/ directory of a package

```
doc/
├── SConscript
├── conf.py
├── doxygen.conf.in
├── index.rst
├── lsst.example/
│   ├── index.rst
│   ├── scripts
│   └── tasks
└── manifest.yaml
```

<https://developer.lsst.io/stack/module-homepage-topic-type.html>

<https://developer.lsst.io/stack/argparse-script-topic-type.html>

<https://developer.lsst.io/stack/task-topic-type.html>

Adding a documentation page

What type of documentation is it?

Tutorials

- Learning-oriented
- For “newcomers” to get started
- A self-contained lesson that the reader performs
- Provides an example

Concept guides

- Understanding-oriented
- Explains
- Provides background & context

How-to guides

- Goal-oriented
- Shows how to solve a specific problem
- Series of steps

Reference guides

- Information-oriented
- Accurate and complete
- Examples:
 - Numpydoc, Doxygen “docstrings”
 - Task reference pages

These types can be fractal (an example in a API reference).
Often just link from one page to another (we’re on the web).

Credit: <https://www.divio.com/en/blog/documentation/>³⁴

Adding a documentation page

Elements of a good documentation page

- **Self-contained**

No previous page, no next page. But lives within the site's web.

- **Specific and limited purpose**

A topic (i.e., a page) should have a specific, well-defined purpose.

- **Conform to type**

A type is like a template.

- **Link richly**

Let the reader forage for information. Think of Wikipedia.

- **Establish context**

Use the first paragraph to establish the topic's purpose and its relationships to other topics.

- **Assume the reader is qualified**

Link to introductory topics, don't repeat introductory concepts.

- **Stay on one level**

Don't evolve level of detail like a narrative.

Based on "Every Page is Page One" by Mark Baker

Adding a documentation page

1. Create a page (rst file) in the module documentation directory or a subdirectory.
2. Add that page to the "toctree" in the parent index.rst file.

```
doc
├── SConscript
├── conf.py
├── doxygen.conf.in
├── index.rst
├── lsst.example
│   ├── index.rst
│   ├── howto-get-butler-dataset.rst
│   ├── scripts
│   └── tasks
└── manifest.yaml
```

doc/lsst.example/index.rst

```
.. py:currentmodule:: lsst.example

.. _lsst.example:

#####
lsst.example
#####

.. _lsst.example-using:

Using lsst.example
=====

.. toctree::
   :maxdepth: 1

   howto-get-butler-dataset
```


Adding a documentation page

In any documentation page you write, the first paragraph has two roles:

1. State what the reader will learn from the page.
2. Establish the context of this doc, usually by linking to overviews and related docs.

```
.. _howto-get-butler-dataset:
```

```
#####  
How to get a dataset with the Butler  
#####
```

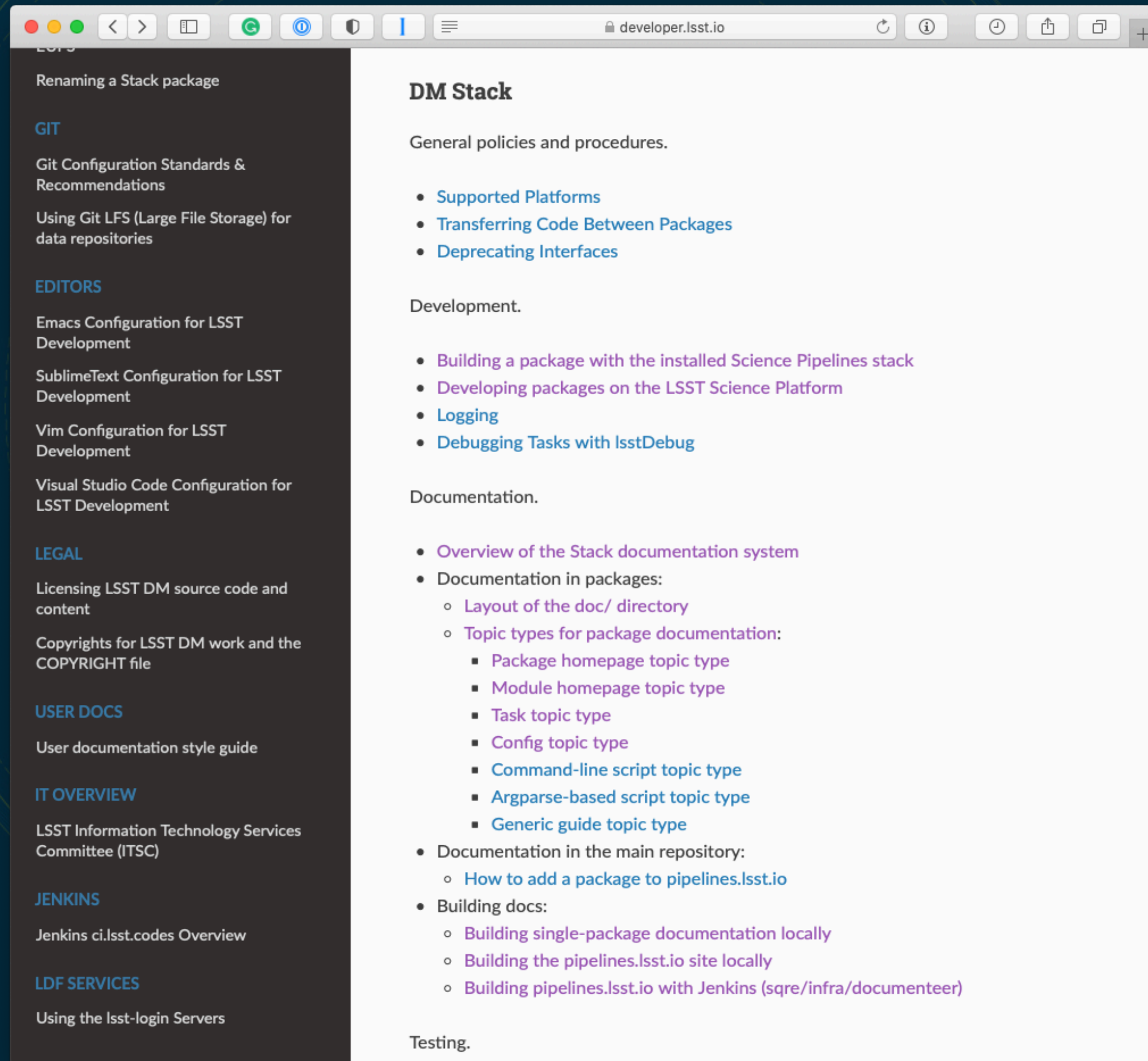
With the LSST Science Pipelines, you don't directly interact with data files, such as FITS. Instead, datasets are managed by the Butler in a repository. This page describes how to get a dataset using the Butler Python class.

```
<the actual how-to goes here>
```

link to a conceptual
overview page

link to an API reference

Science Pipelines documentation templates

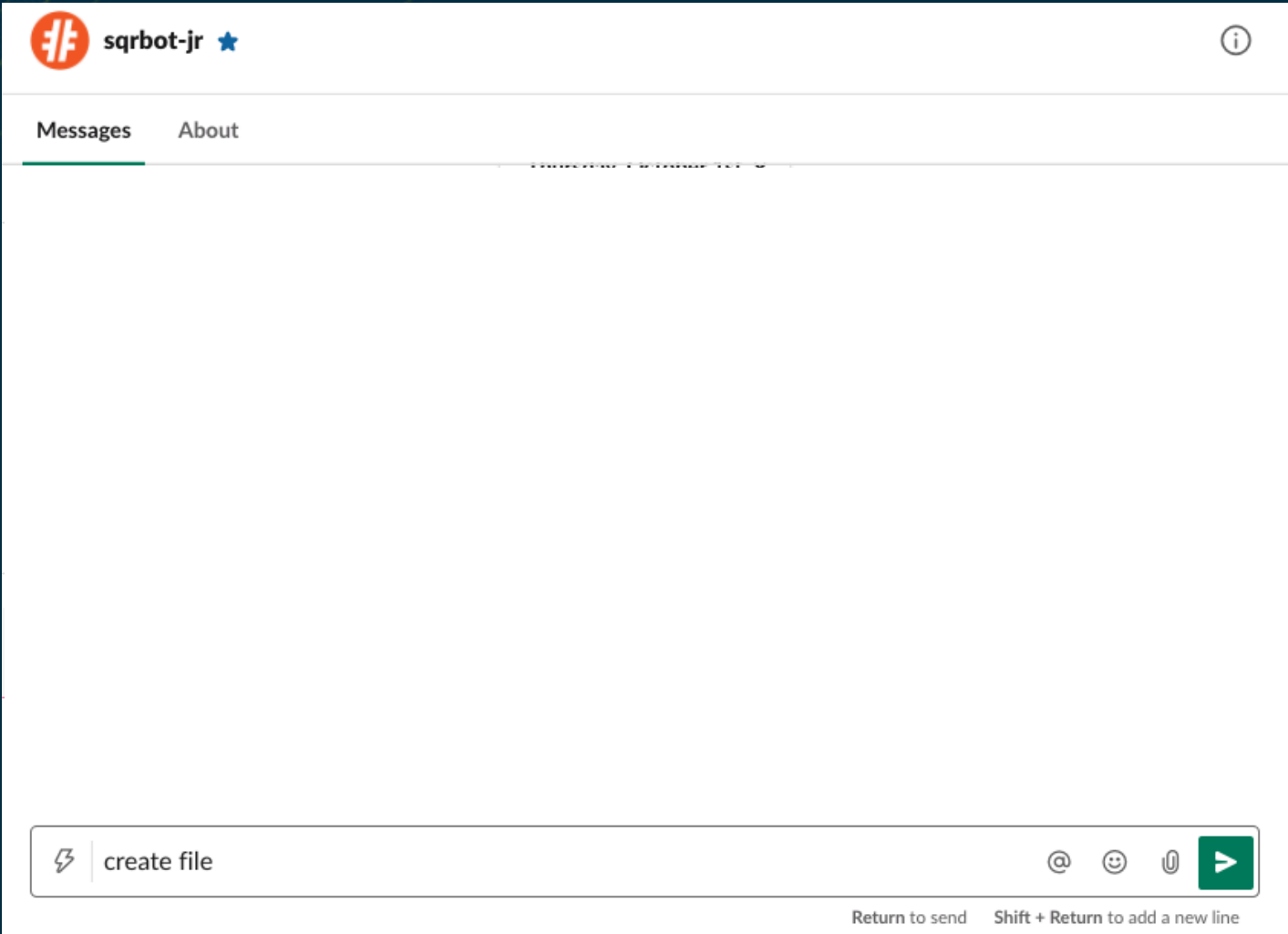


- Templates give the documentation consistency, which is important to readers' success.
- We have a number of templates already, more will be added as we design the types of documentation content we need.



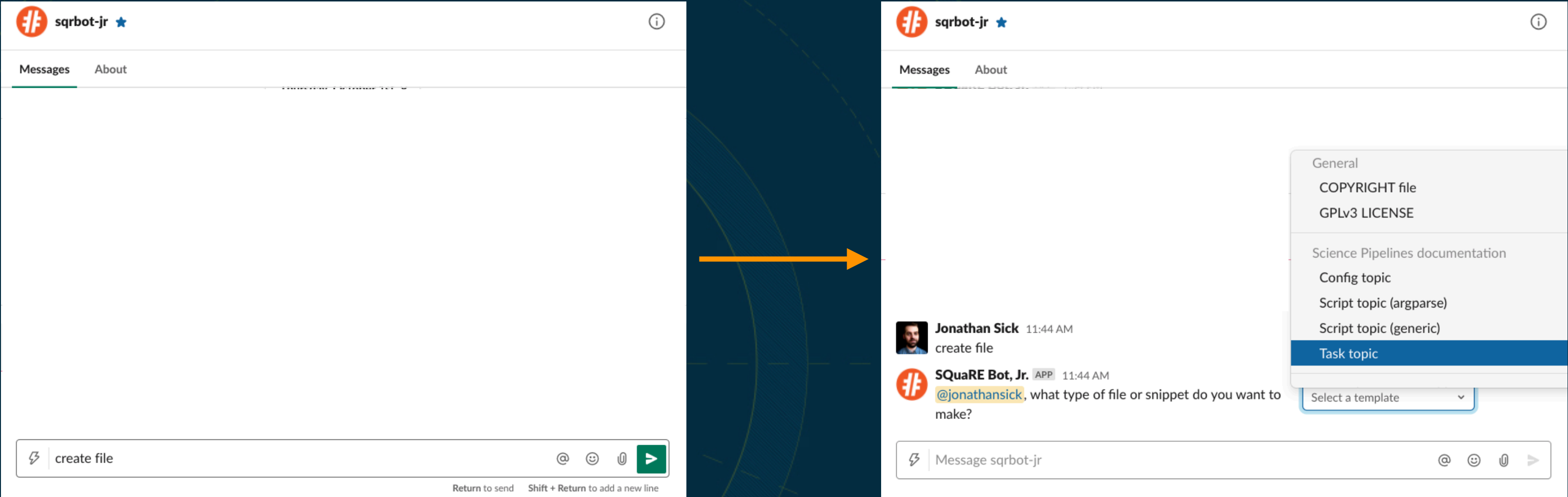
Page templates (topic types in technical writing jargon) are listed in the Developer Guide.

Science Pipelines documentation templates



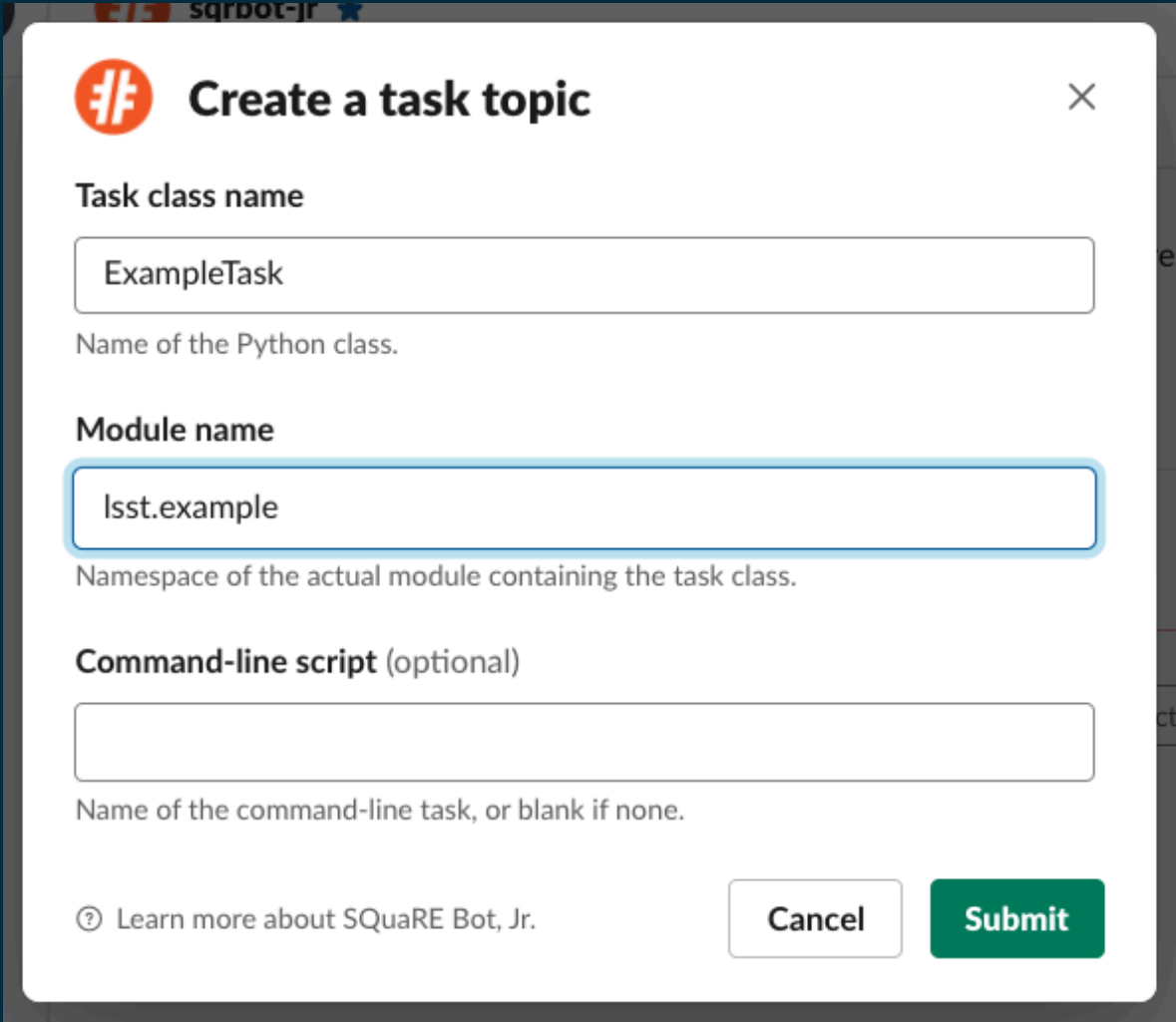
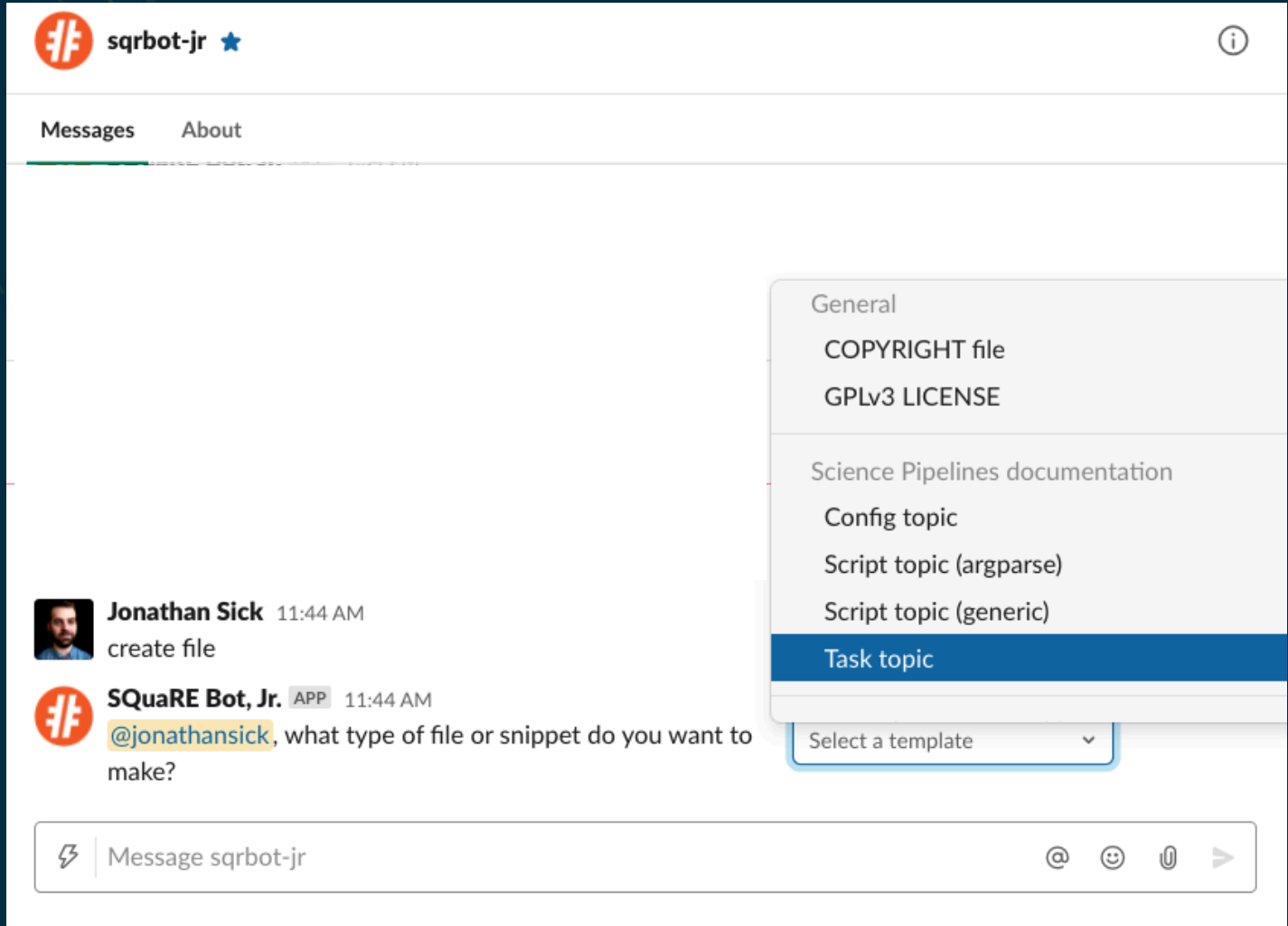
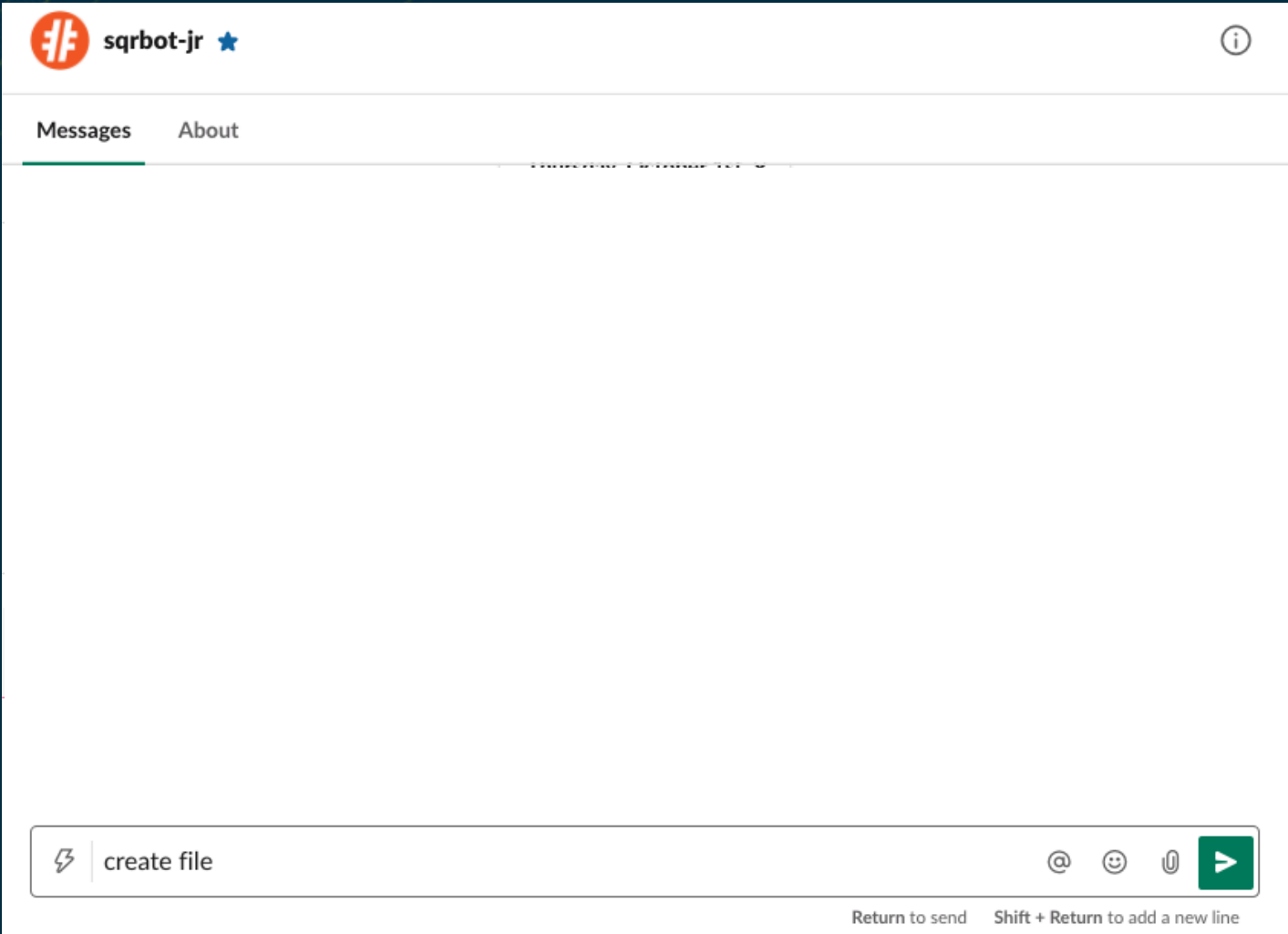
Direct message @sqrbot-jr in Slack to create a new file from a template.

Science Pipelines documentation templates



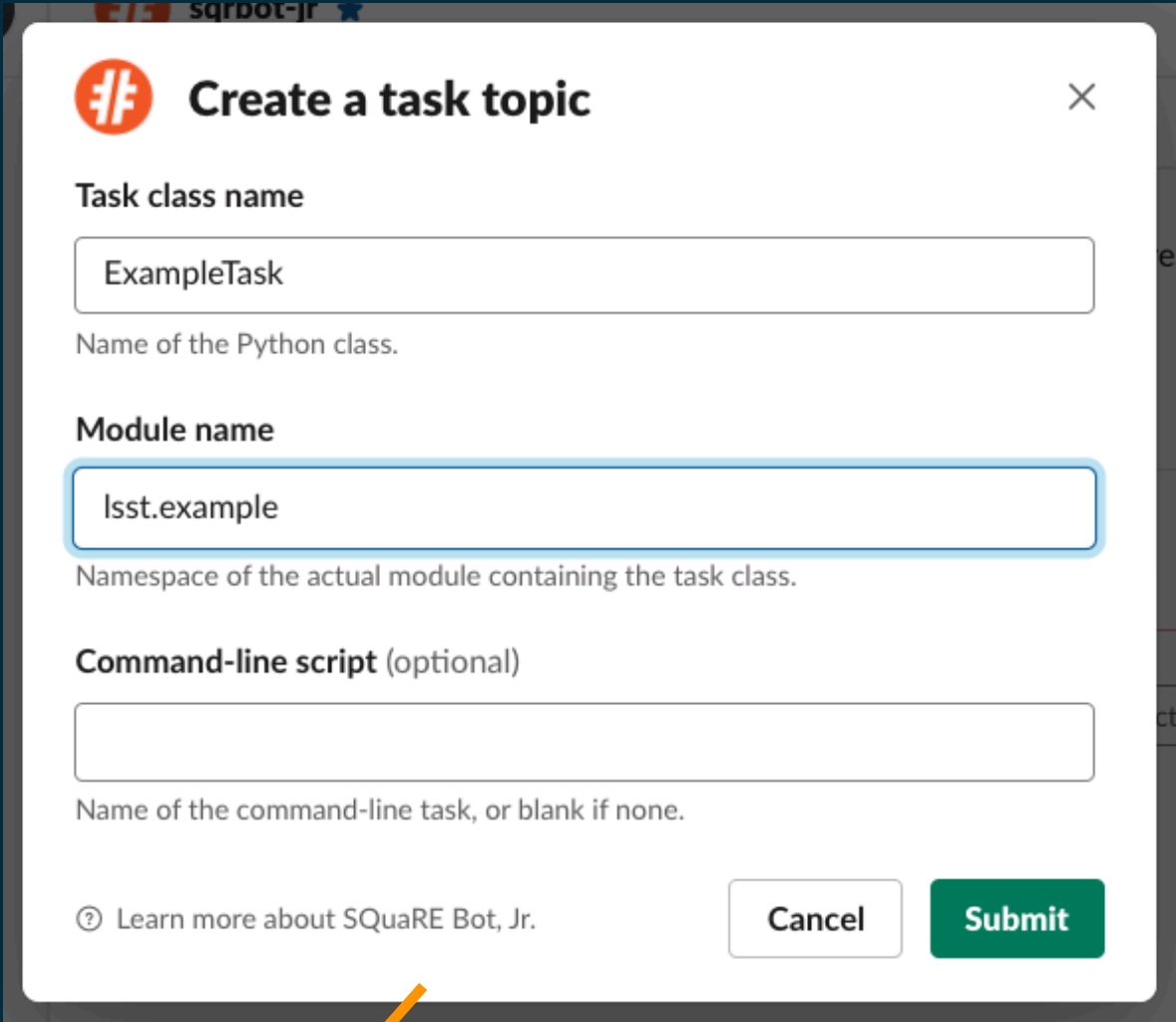
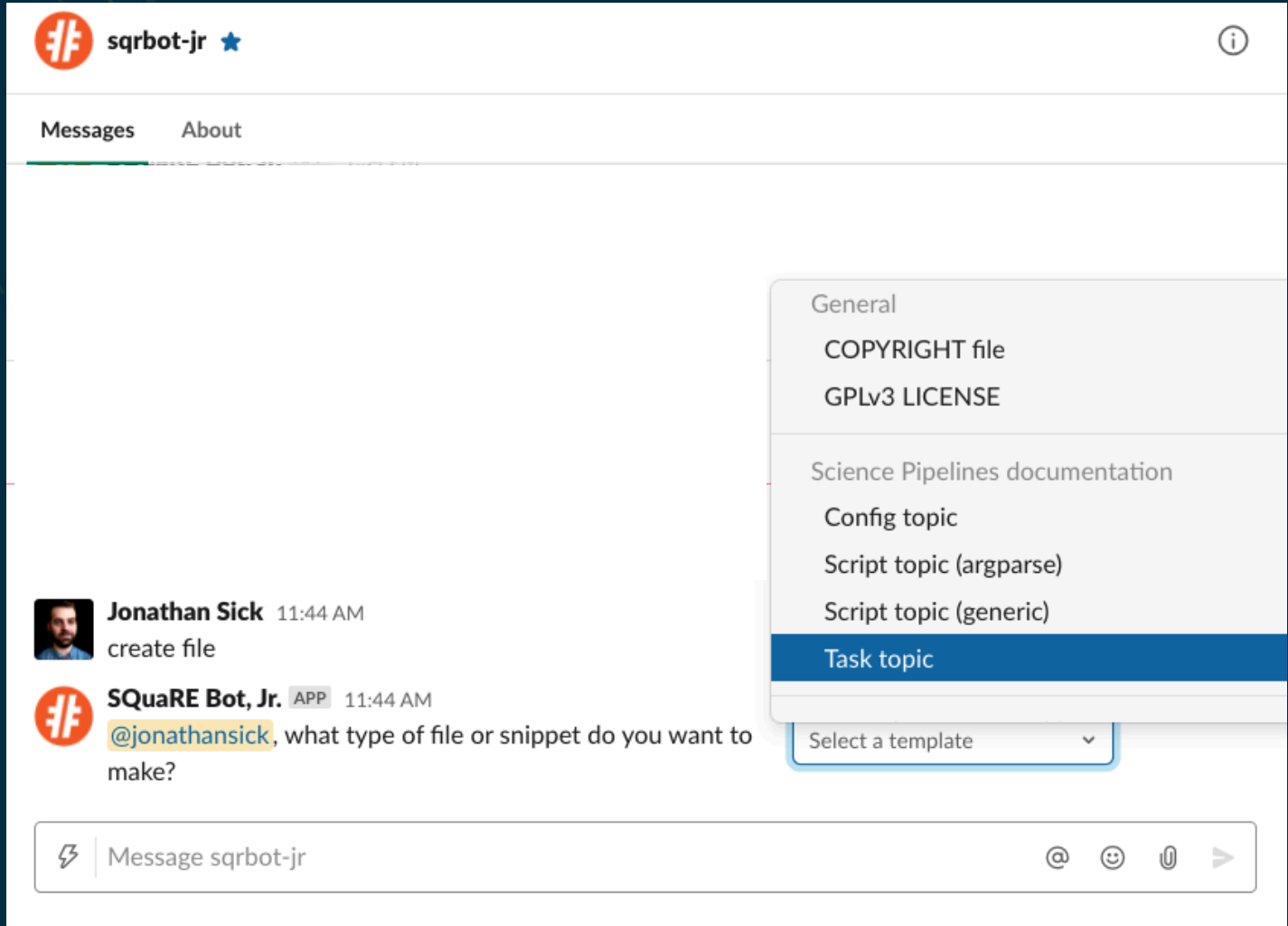
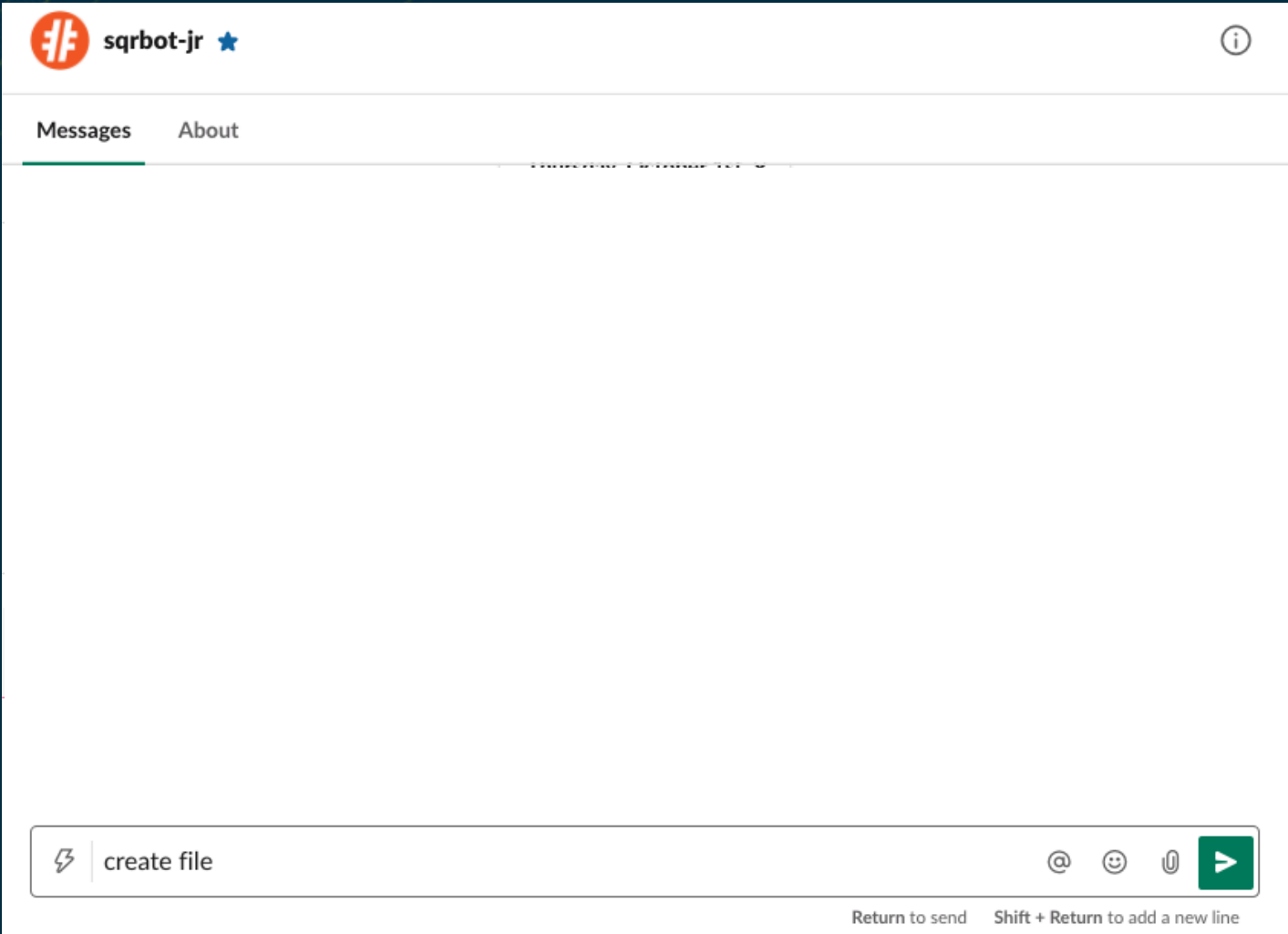
Direct message @sqrbot-jr in Slack to create a new file from a template.

Science Pipelines documentation templates

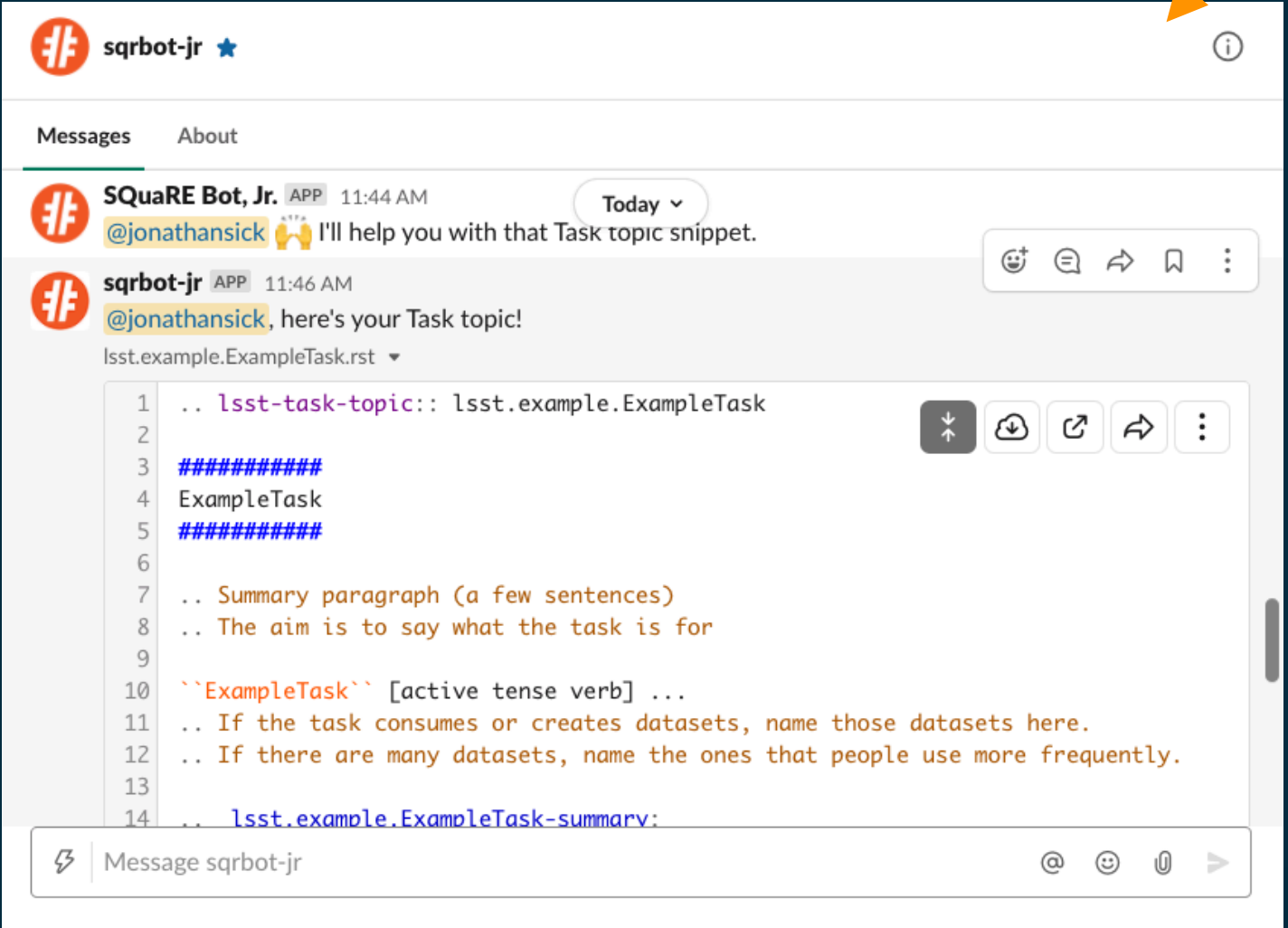


Direct message @sqrbot-jr in Slack
to create a new file from a
template.

Science Pipelines documentation templates



Direct message @sqrbot-jr in Slack to create a new file from a template.



Module homepage topic type

LSST

LSST Science Pipelines

Forum

Docs

LSST.org →

LSST Science Pipelines » lsst.pipe.base

« previous | next »

Search

On this page

lsst.pipe.base

Using lsst.pipe.base

Overview

Using command-line tasks

Developing tasks and command-line tasks

Contributing

Python API reference

lsst.pipe.base Package

Functions

Classes

Class Inheritance Diagram

lsst.pipe.base.testUtils Module

Functions

lsst.pipe.base

The `lsst.pipe.base` module provides base classes for the task framework. Tasks package the algorithmic units of the LSST Science Pipelines. You can create, configure, and run tasks with their Python APIs. Some tasks, called command-line tasks, are also packaged into data processing pipelines that you can run from the command line.

Using lsst.pipe.base

Overview

- Overview of the task framework

Using command-line tasks

- Using Butler data repositories and reruns with command-line tasks
- Specifying data IDs with command-line tasks
- Configuring command-line tasks
- Retargeting subtasks of command-line tasks
- Working with provenance checks in command-line tasks
- Logging with command-line tasks
- Parallel processing with command-line tasks
- Command-line task argument reference

Developing tasks and command-line tasks

- Creating a task
- Creating a command-line task
- Testing a pipeline task

Contributing

lsst.pipe.base is developed at https://github.com/lsst/pipe_base. You can find Jira issues for this module under the [pipe_base](#) component.

Python API reference

lsst.pipe.base Package

Functions

`iterConnections`(connections, connectionType)

Creates an iterator over the selected connections type which yields all the defined connections of that type.

`logInfo`(obj, prefix[, logLevel])

Log timer information to `obj.metadata` and `obj.log`.

`timeMethod`(func)

Decorator to measure duration of a

Rubin Observatory Operations Boot Camp | October 14, 2020

U.S. DEPARTMENT OF
ENERGY | Office of
Science

Task topic type

LSST

LSST Science Pipelines

Forum Docs LSST.org →

LSST Science Pipelines » [lsst.ip.diffim](#) » NumberSciSourcesMetricTask [« previous](#) [next »](#)

Search

On this page

[NumberSciSourcesMetricTask](#)

[Processing summary](#)

[Python API summary](#)

[Butler datasets](#)

[Input datasets](#)

[Retargetable subtasks](#)

[Configuration fields](#)

[connections](#)

[saveMetadata](#)

NumberSciSourcesMetricTask

NumberSciSourcesMetricTask computes the number of science sources created when processing data through image differencing (as the `ip_diffim.numSciSources` metric). It requires source catalogs (a `src` dataset) as input, and can operate at image-level or coarser granularity.

Processing summary

FractionDiaSourcesToSciSourcesMetricTask reads source catalogs (`src` datasets) and adds up the number of sources in those catalogs.

Python API summary

```
from lsst.ip.diffim.metrics import NumberSciSourcesMetricTask
```

`class` **NumberSciSourcesMetricTask** (***kwargs*)
Task that computes the number of cataloged science sources...

attribute **config**
Access configuration fields and retargetable subtasks.

method **run** (*sources*)
Count the number of science sources...

See also

See the **NumberSciSourcesMetricTask** API reference for complete details.

Butler datasets

Input datasets

`sources`
The catalog type for science sources (default: `src`).

Retargetable subtasks

No subtasks.

Configuration fields

connections

Data type
`lsst.pipe.base.config.Connections`

Script topic type

LSST

LSST Science Pipelines

Forum

Docs

LSST.org →

LSST Science Pipelines » [lsst.verify](#) » [dispatch_verify.py](#)

[« previous](#) | [next »](#)

Search

On this page

[dispatch_verify.py](#)

[positional arguments](#)

[optional arguments](#)

[Environment arguments](#)

[SQUASH API arguments](#)

dispatch_verify.py

Upload LSST Science Pipelines Verification **Job** datasets to the SQUASH dashboard.

Job JSON files can be created by **`lsst.verify.Job.write`** or **`lsst.verify.output_quantities`**. A **Job** dataset consists of metric measurements, associated blobs, and pipeline execution metadata. Individual LSST Science Pipelines tasks typically write separate JSON datasets. This command can collect and combine multiple Job JSON datasets into a single Job upload.

Configuration

`dispatch_verify.py` is configurable from both the command line and environment variables. See the argument documentation for environment variable equivalents. Command line settings override environment variable configuration.

Metadata and environment

`dispatch_verify.py` can enrich Verification Job metadata with information from the environment. Currently `dispatch_verify.py` supports the Jenkins CI and the LSST Data Facility (LDF) execution environments.

In the Jenkins CI execution environment (`--env=jenkins`) the following environment variables are consumed:

- `BUILD_ID` : ID in the CI system
- `BUILD_URL` : CI page with information about the build
- `PRODUCT` : the name of the product built, e.g. 'validate_drp'
- `dataset` : the name of the dataset processed, e.g. 'validation_data_cfht'
- `label` : the name of the platform where it runs

If `--lsstsw` is used, additional Git branch information is included with Science Pipelines package metadata.

In the LSST Data Facility execution environment (`--env=ldf`) the following environment variables are consumed:

- `DATASET` : the name of the dataset processed, e.g 'HSC RC2'
- `DATASET_REPO_URL` : a reference URL with information about the dataset
- `RUN_ID` : ID of the run in the LDF environment
- `RUN_ID_URL` : a reference URL with information about the run
- `VERSION_TAG` : the version tag of the LSST software used, e.g. 'w_2018_18'

Note: currently it is not possible to gather Science Pipelines package metadata in the LDF environment, thus if `--env=ldf` is used `--ignore-lsstsw` is also used by default in this environment.

```
usage: dispatch_verify.py [-h] [--test] [--write PATH] [--show]
                        [--ignore-blobs] [--env {jenkins,ldf}]
                        [--lsstsw PATH] [--package-repos [PATH [PATH ...]]]
                        [--ignore-lsstsw] [--url URL] [--user USER]
                        [--password PASSWORD]
                        json [json ...]
```

More information is available at <https://pipelines.lsst.io>.

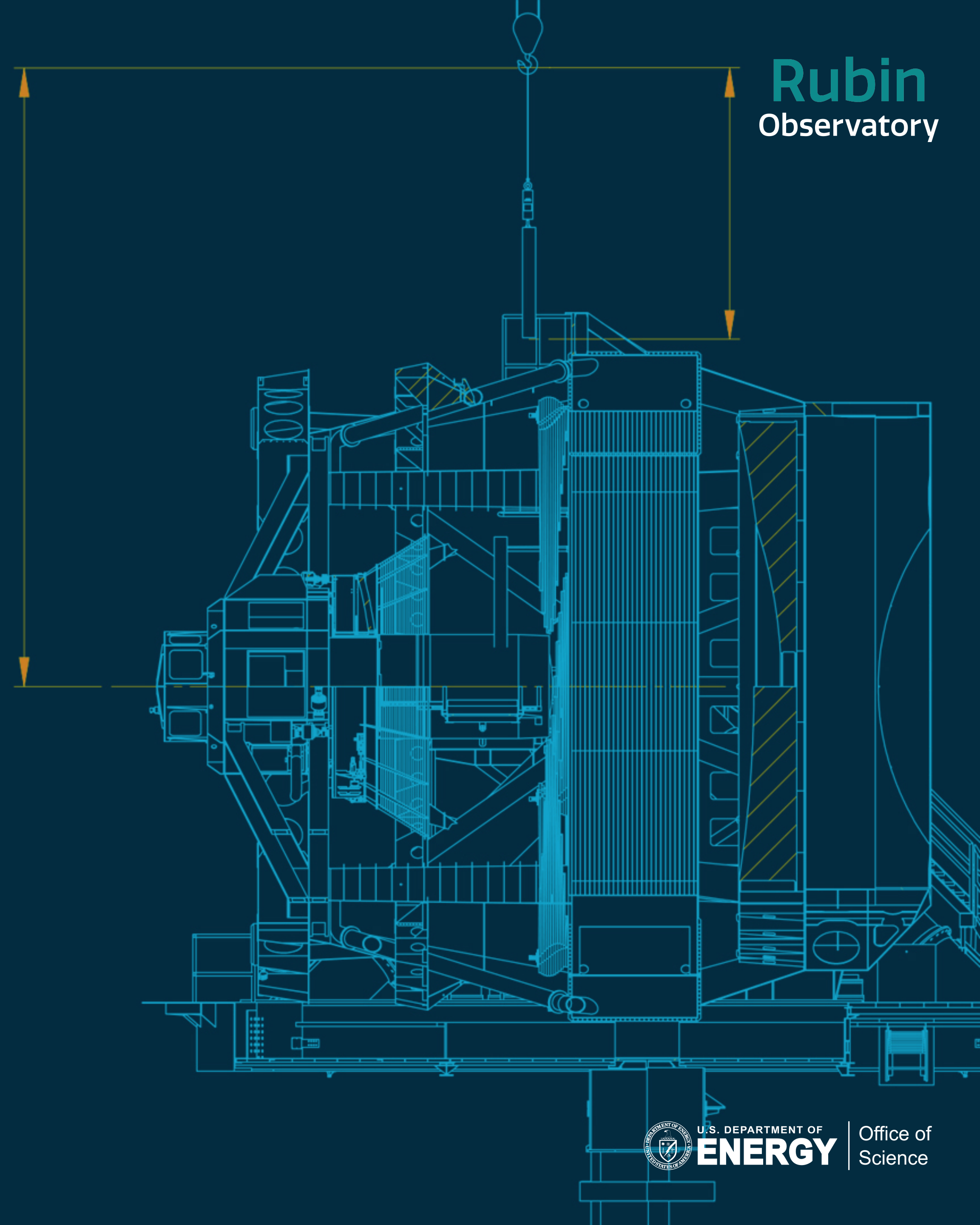
positional arguments

<https://developer.lsst.io/python/numpydoc.html>

Python docstrings

The docstrings in Python code are compiled into the API reference sections of software documentation guides, such as [pipelines.lsst.io](https://developer.lsst.io).

Following the docstring format is important for documentation builds.



Python docstrings

The basics

```
def upload():  
    pass
```


Python docstrings

The basics

```
def upload():
    """Upload content to the server."""
    pass
```

Start and end the docstring with three double quotes. If docstring spans multiple lines put closing quotes on their own line.

Use imperative mood for summary sentence.
~~Uploads content to the server.~~

Wrap after the 79th character column.

Python docstrings

Function/method: Parameters

```
def upload(body):
    """Upload content to the server.
```

Parameters

```
body : `bytes`
```

The content to upload, already encoded as bytes.

```
"""
```

```
pass
```

② Type description. Use single-backticks to link type names.

① Variable name

③ Indent description by four spaces.

For details: <https://developer.lsst.io/python/numpydoc.html#parameters>

Python docstrings

Function/method: Parameters

```
def upload(body, contentType='application/octet-stream'):
    """Upload content to the server.

    Parameters
    -----
    body : `bytes`
        The content to upload, already encoded as bytes.
    contentType : `str`, optional
        The media type of the ``body``. Common types:

        ``application/octet-stream``
            A binary file (default).
        ``application/text``
            Text content.
        ``application/json``
            A JSON-formatted document.
    """
    pass
```

For details: <https://developer.lsst.io/python/numpydoc.html#parameters>

Python docstrings

Function/method: Returns

```
def upload(body, contentType='application/octet-stream'):
    """Upload content to the server.

    Parameters
    -----
    [...]

    Returns
    -----
    url : `str`
        The canonical URL where the content can be accessed.
    """
    pass
```


Python docstrings

Function/method: Raises (exceptions)

```
def upload(body, contentType='application/octet-stream'):
    """Upload content to the server.

    Parameters
    -----
    [...]

    Returns
    -----
    [...]

    Raises
    -----
    lsst.example.UploadError
        Raised if a connection cannot be made with the server, or if the
        server returns a 500 error.
    lsst.example.AuthError
        Raised if the client cannot be authenticated or is not
        authorized.
    """
    pass
```


Python docstrings

Class attribute docstrings

```
class Point:
    """A cartesian coordinate.

    Parameters
    -----
    x : `float`
        The ``x``-axis coordinate.
    y : `float`
        The ``y``-axis coordinate.
    """

    x = None
    """The ``x``-axis coordinate (`float`)."""

    y = None
    """The ``y``-axis coordinate (`float`)."""

    def __init__(self, x, y):
        self.x = x
        self.y = y
```


Python docstrings

Class properties

```
class Point:
    """A cartesian coordinate.

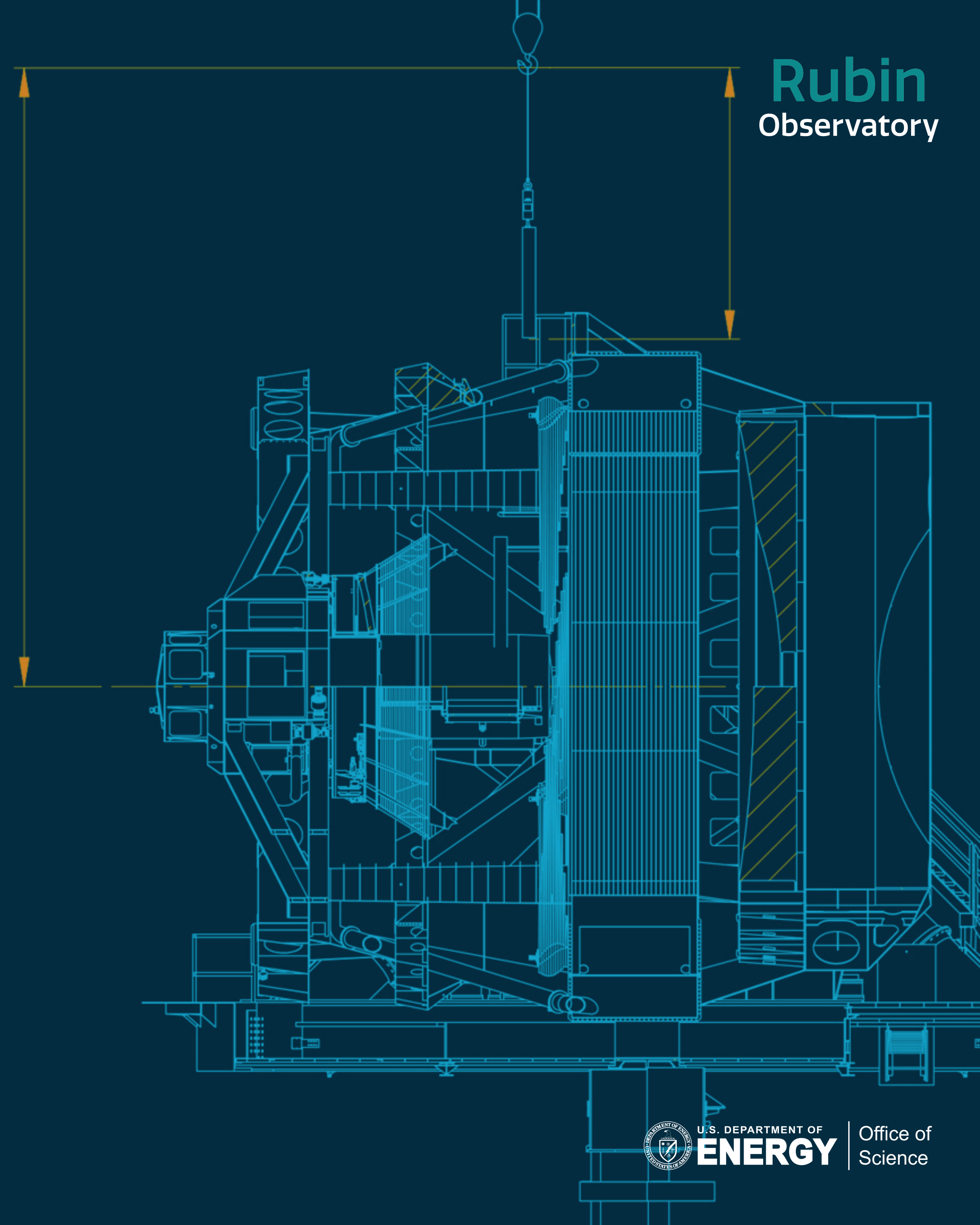
    Parameters
    -----
    x : `float`
        The ``x``-axis coordinate.
    y : `float`
        The ``y``-axis coordinate.
    """

    @property
    def x(self):
        """The ``x``-axis coordinate (`float`)."""
        return self._x

    def __init__(self, x, y):
        self._x = x
        self._y = y
```


<https://developer.lsst.io/user-docs/index.html>

Content style guide



Rubin
Observatory



Rubin Observatory Operations Boot Camp | October 14, 2020



U.S. DEPARTMENT OF
ENERGY

Office of
Science

The style guide is here to help you

In DM, we have primarily adopted the Google Content Style guide:
developers.google.com/style/

The screenshot shows the Google developer documentation style guide website. The browser address bar displays `developers.google.com`. The page title is "Google developer documentation style guide". The left sidebar contains a navigation menu with sections: Introduction, Style guide (highlighted), Other editorial resources, What's new, Key resources, Word list, Product names, Text-formatting summary, General principles, Style and tone, Documenting future features, Writing accessible documentation, Writing for a global audience, Writing inclusive documentation, Avoiding excessive claims, Using other sources, Language and grammar, Abbreviations, Active voice, Anthropomorphism, Articles (a, an, the), Capitalization, Clause order, Contractions, Plurals in parentheses, Possessives, Prepositions, Present tense, Pronouns, Second person, Spelling, Verb forms in reference documentation, Punctuation, Colons, Commas, Dashes, Ellipses, Exclamation points, Hyphens, Parentheses, Periods, Pluralizing a single letter, Quotation marks, Semicolons, and Slashes. The main content area has a breadcrumb trail: Home > Products > Style. It includes a "Rate and review" button and a "Highlights" section with a table of contents: Table of contents, Introduction, Tone and content, Language and grammar, Formatting, punctuation, and organization, and Images. A "Key Point" section states: "An overview of some of the highlights of the style guide." The "Introduction" section explains that the style guide covers a lot of material and provides an overview of its most important points. The "Tone and content" section lists five bullet points: "Be conversational and friendly without being frivolous.", "Don't pre-announce anything in documentation.", "Use descriptive link text.", "Write accessibly.", and "Write for a global audience." The "Language and grammar" section lists five bullet points: "Use second person: 'you' rather than 'we.'", "Use active voice: make clear who's performing the action.", "Use standard American spelling and punctuation.", "Put conditional clauses before instructions, not after.", and "For usage and spelling of specific words, see the word list." The "Formatting, punctuation, and organization" section lists one bullet point: "Use sentence case for document titles and section headings."

Writing for the web

- “ Users won't read your text thoroughly in a word-by-word manner.
- “ The first two paragraphs must state the most important information.
- “ Start subheads, paragraphs, and bullet points with information-carrying words.



From: F-Shaped Pattern For Reading Web Content by Jakob Nielsen, <http://ls.st/fzp>.

Writing for the web

- Most people don't read the web sentence-by-sentence.
- Use lots of meaningful headers.
- 2–3 sentence paragraphs.
- First one or two words of a paragraph should convey meaning.
- Use lists and typographic elements for structure.

Tone: be conversational without being frivolous

- ✗ Please note that completion of the task requires the following prerequisite: executing an automated memory management function.
- ✗ Then—BOOM—just garbage-collect (or *collecter des garbâge*, as they say in French), and you're golden.
- ✓ To clean up, call the `collectGarbage()` method.



The most straightforward sentence is often the best sentence.



Avoid sounding "clever."



Avoid jokes unless you know what you're doing.



Contractions are fine.

From <https://developers.google.com/style/tone>

Use second person



In the previous tutorial in the series we used `processCcd.py` to calibrate a set of raw Hyper Suprime-Cam images. Now we'll learn how to use the LSST Science Pipelines to inspect `processCcd.py`'s outputs by displaying images and source catalogs in the DS9 image viewer. In doing so, we'll be introduced to some of the LSST Science Pipelines' Python APIs.



In the previous tutorial in the series you used `processCcd.py` to calibrate a set of raw Hyper Suprime-Cam images. Now you'll learn how to use the LSST Science Pipelines to inspect `processCcd.py`'s outputs by displaying images and source catalogs in the DS9 image viewer. In doing so, you'll be introduced to some of the LSST Science Pipelines' Python APIs.

From <https://developers.google.com/style/person>

Use second person: speak directly to the user

- ✗ When setting up a product, a user may specify a version, or accept the current default.
- ✓ When setting up a product, you may specify a version, or accept the current default.



User documentation must be actionable for the user.

From <https://developers.google.com/style/person>

Use the active voice

- ✗ The service is queried, and an acknowledgment is sent.
- ✓ Send a query to the service. The server sends an acknowledgment.



Passive? If you can add "by monkeys" to a sentence, it's passive.



Active voice makes it clear who performs the action, which makes for more understandable documentation.

From <https://developers.google.com/style/voice>

Use the passive voice when the actor is unimportant to the user

- ✓ | The database was purged in January.
- ✓ | Over 50 conflicts were found in the file.

From <https://developers.google.com/style/voice>

Write in the present tense

- ✗ Send a query to the service. The server will send an acknowledgment.
- ✓ Send a query to the service. The server sends an acknowledgment.
- ✗ You can send an unsubscribe message. The server would then remove you from the mailing list.
- ✓ If you send an unsubscribe message, the server removes you from the mailing list.

<https://developers.google.com/style/tense>

Don't anthropomorphize software and hardware systems

✗ The `doApCorr` configuration field tells the `ForcedPhotCcdTask` whether to apply aperture corrections.

✓ Set the `doApCorr` configuration field to `True` to enable aperture corrections.



Software doesn't:

- tell
- know
- want
- ...

See <https://developers.google.com/style/anthropomorphism>

Avoid ableist language

- ✗ Replace the dummy variable in this example with the appropriate variable.
- ✓ Replace the placeholder variable in this example with the appropriate variable.

<https://developers.google.com/style/inclusive-documentation>

Use sentence case for section headlines

✗ | LSST-Related Notebook Repositories

✓ | LSST-related notebook repositories

✗ | Getting Notebooks from GitHub

✓ | Getting notebooks from GitHub

<https://developers.google.com/style/headings>

Introducing a list



Use the LSST Science Pipelines to:

- do forced photometry
- measure galaxy shapes
- apply photometric and astrometric calibrations



You can use use the LSST Science Pipelines any of the following purposes:

- To perform forced photometry of a set of images.
- To measure the shapes of galaxies.
- To apply photometric and astrometric calibrations.



The introductory sentence must be complete. Don't complete it with list items.

See <https://developers.google.com/style/lists>

List tips

- ✓ Use parallel structure.
- ✓ Add a period if the item is a complete sentence.
- ✓ Don't add add punctuation if the item is a single word.
- ✓ Use the appropriate type of list:
 - Bullet lists for unordered items
 - Numbered lists for procedures.
 - Definition lists to pair terms with definitions or explanations.

See <https://developers.google.com/style/lists>



Introducing a code sample



Run:

`setup lsst_distrib`
to set up the LSST Science Pipelines.

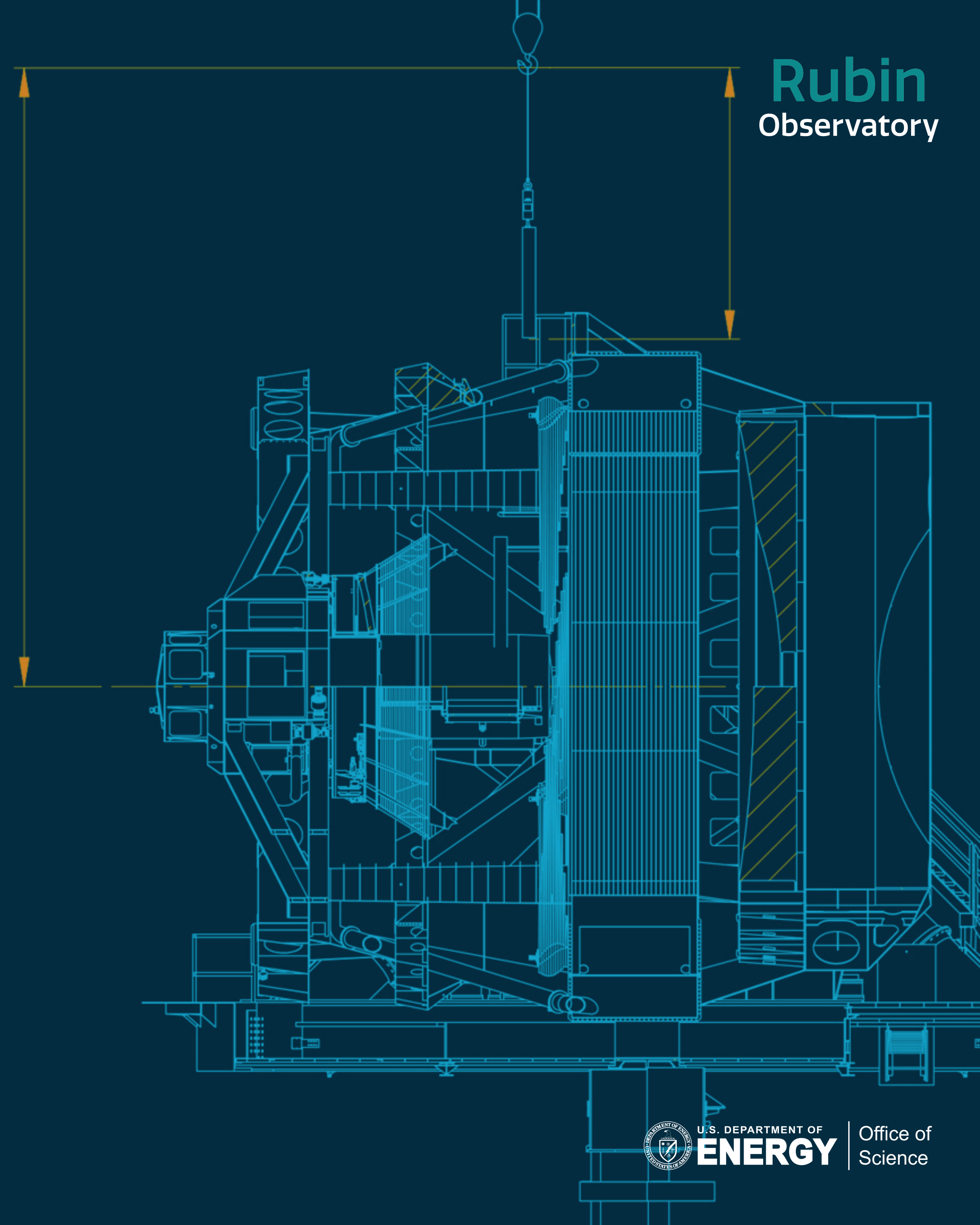


Run this command to set up the LSST Science Pipelines:

`setup lsst_distrib`

Whew, that was a lot to learn.

Don't worry, everything is documented and you can ask for help on Slack.



Resources for learning more

Slack

#dm-docs channel for help writing docs

User documentation style guide

<https://developer.lsst.io/user-docs/index.html>

ReStructuredText style guide

<https://developer.lsst.io/restructuredtext/style.html>

Docstrings

<https://developer.lsst.io/python/numpydoc.html>

Stack developer guide (including writing docs for pipelines.lsst.io)

<https://developer.lsst.io/index.html#dm-stack>

Writing technotes

<https://developer.lsst.io/project-docs/technotes.html>

Documentation portal

<https://www.lsst.io>

