XFEL ReadTheDocs Documentation

Release 1.0

S. Hauf

Jun 16, 2021

Contents

Contents:

CHAPTER 1

Introduction

Documentation for software-related projects should be written such that it can be hosted on this XFEL internal Read The Docs server. Specifically, this implies that:

- you version your documentation on GitLab and add a projec to this server. See *Adding a New Project from GitLab*
- you write your documentation in RST. See See *How To Document*
- you follow a few simple policies.

CHAPTER 2

Karabo-Related Documentation

2.1 Migrating Devices

For legacy devices and projects linked to Karabo, documentation migration needs to be performed. If you already have run *sphinx-quickstart* to start your device documentation, you do not have to do this again, but can instead adapt your *conf.py* to reflect the settings given for quickstart below.

If you first run ‘sphinx-quickstart, please make choices according to the following suggestions

Run *sphinx-quickstart* from within the *./doc* folder:

```
Welcome to the Sphinx 1.2.3 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:
```

You should choose . as the root path, i.e. things will be mostly treated relative to your *doc* folder:

```
You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]: y
```

Please answer yes here. This way you can later add the build directory to a *.gitignore* file:

```
Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:
```

Please leave _ here to have things consistent:

```
The project name will occur in several places in the built documentation.
> Project name: XFEL ReadTheDocs
> Author name(s): S. Hauf
```

You should enter a human-readable and well suited project name here, frequently the device name, and the primary authors(s):

```
Sphinx has the notion of a "version" and a "release" for the
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version: 1.0
> Project release [1.0]:
```

Give a meaningful project version and release number here.

> The file name suffix for source files. Commonly, this is either ".txt" or ".rst". Only files with this suffix are considered documents. > Source file suffix [.rst]:

Do not change this away from *.rst*:

```
One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:
```

Your master document *always* should be called *index*:

```
Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/n) [n]: n
```

We do not create epub documentation, in fact it will fail on read the docs!

> Please indicate if you want to use one of the following Sphinx extensions: > autodoc: automatically insert docstrings from modules (y/n) [n]: y

For anything containing code this is strongly recommended to be set to *y*:

```
> doctest: automatically test code snippets in doctest blocks (y/n) [n]: n
```

This is optional, if you know how to use the doctest feature:

```
> intersphinx: link between Sphinx documentation of different projects
(y/n) [n]: n
```

The global configuration (see later) will take care of this:

```
> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]: y
```

Allowing the .. *todo::* directive is usually a good option:

```
> coverage: checks for documentation coverage (y/n) [n]:
```

Optionally, you can enable this feature:

```
> pngmath: include math, rendered as PNG images (y/n) [n]: n
```

Set this to *n* as we will be using *mathjax*.

> mathjax: include math, rendered in the browser by MathJax (y/n) [n]: y

Set this to *y* to enable math rendering:

```
> ifconfig: conditional inclusion of content based on config values (y/n) [n]: y
```

Set this to *yes* to enable conditional rendering:

```
> viewcode: include links to the source code of documented Python objects (y/n) [n]: y
```

You can optionally set this to *y*:

```
A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. `make html' instead of invoking sphinx-build
directly.
> Create Makefile? (y/n) [y]: n
> Create Windows command file? (y/n) [y]: n
```

We do not need make files as we will host on RTD, but if you want to run local builds of your documentation, e.g. for testing, set "Create Makefile" to *y*:

```
Creating file ./source/conf.py.
Creating file ./source/index.rst.
Creating file ./Makefile.

Finished: An initial directory structure has been created.

You should now populate your master file ./source/index.rst and create other␣
↪documentation
source files. Use the Makefile to build the docs, like so:
make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.
```

After you have created or edited your *conf.py* to reflect this settings, add the following import statement at the very beginning:

```
from rtd_conf import global_conf
```

and comment the lines:

```
extensions = [
'sphinx.ext.autodoc',
'sphinx.ext.intersphinx',
'sphinx.ext.todo',
'sphinx.ext.mathjax',
'sphinx.ext.ifconfig',
'sphinx.ext.viewcode',
'sphinx.ext.graphviz',
]
```

if you do not need aditional extension (they are already globally configured), or change to:

```
extensions += ['other_extension']
```

if you need specific extensions.

Further comment:

```
intersphinx_mapping = {'http://docs.python.org/': None}
```

Finally, make sure to point your project configuration on RTD to the *requirements.txt* in your doc folder.

CHAPTER 3

Non-Karabo-Related Documentation

For non-Karabo projects you need to create this yourself through a relatively simple procedure. You can use the *sphinx-quickstart* utility to generate an initial *conf.py* (the file that configures documentation processing), folder structure and *index.rst* (your top-level index).

3.1 Starting A Sphinx Project

From within the directory you would like to create documentation in run *./sphinx-quickstart*. You should then see the following sequence of queries, where suggested answers have already been chosen.

```
Welcome to the Sphinx 1.2.3 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Enter the root path for documentation.
> Root path for the documentation [.]:

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]: y

Inside the root directory, two more directories will be created; "_templates"
for custom HTML templates and "_static" for custom stylesheets and other static
files. You can enter another prefix (such as ".") to replace the underscore.
> Name prefix for templates and static dir [_]:

The project name will occur in several places in the built documentation.
> Project name: XFEL ReadTheDocs
> Author name(s): S. Hauf

Sphinx has the notion of a "version" and a "release" for the
```

(continues on next page)

(continued from previous page)

```
software. Each version can have multiple releases. For example, for
Python the version is something like 2.5 or 3.0, while the release is
something like 2.5.1 or 3.0a1.  If you don't need this dual structure,
just set both to the same value.
> Project version: 1.0
> Project release [1.0]:

The file name suffix for source files. Commonly, this is either ".txt"
or ".rst".  Only files with this suffix are considered documents.
> Source file suffix [.rst]:

One document is special in that it is considered the top node of the
"contents tree", that is, it is the root of the hierarchical structure
of the documents. Normally, this is "index", but if your "index"
document is a custom template, you can also set this to another filename.
> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:
> Do you want to use the epub builder (y/n) [n]: n

Please indicate if you want to use one of the following Sphinx extensions:
> autodoc: automatically insert docstrings from modules (y/n) [n]: y
> doctest: automatically test code snippets in doctest blocks (y/n) [n]: n
> intersphinx: link between Sphinx documentation of different projects (y/n) [n]: y
> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]: y
> coverage: checks for documentation coverage (y/n) [n]:
> pngmath: include math, rendered as PNG images (y/n) [n]: y
> mathjax: include math, rendered in the browser by MathJax (y/n) [n]: y
Note: pngmath and mathjax cannot be enabled at the same time.
pngmath has been deselected.
> ifconfig: conditional inclusion of content based on config values (y/n) [n]: y
> viewcode: include links to the source code of documented Python objects (y/n) [n]: y

A Makefile and a Windows command file can be generated for you so that you
only have to run e.g. `make html' instead of invoking sphinx-build
directly.
> Create Makefile? (y/n) [y]: y
> Create Windows command file? (y/n) [y]: n

Creating file ./source/conf.py.
Creating file ./source/index.rst.
Creating file ./Makefile.

Finished: An initial directory structure has been created.

You should now populate your master file ./source/index.rst and create other␣
↪documentation
source files. Use the Makefile to build the docs, like so:
make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.
```

3.2 Including InterSphinx Information for all Local Projects

In the next step you should adapt your *conf.py* which you can find in the directory you started *sphinx-quickstart* from. Specifically, you should include the following code snippet at the end:

```
#we add a custum function to work with intersphinx
import slumber
import json

# the RTDHOST should be set to localhost if you only compile on RTD
# otherwise it needs to be set to the server hosting the internal RTD
RTDHOST = 'https://in.xfel.eu/readthedocs
api = slumber.API(base_url='{}/api/v1/'.format(RTDHOST))
projects = api.project.get()['objects']
isphinx = {'python': ('http://python.readthedocs.io/en/latest/', None),
           'numpy': ('http://numpy.readthedocs.io/en/latest/', None),
           'scipy': ('http://scipy.readthedocs.io/en/latest/', None)}

for proj in projects:
    isphinx[proj['slug'].replace('-', '')] = \
     ('{}/docs/{}/en/latest'.format(RTDHOST, proj['slug']), None)


intersphinx_mapping = isphinx
```

Check *Referencing across Projects* on how to link across projects hosted on RTD.

3.3 Adding GraphViz Extension for Inline Graphs

To enable graphs as shown in *Graphs* make sure you have the corresponding extension enabled in your *conf.py*:

```
# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = [
    'sphinx.ext.autodoc',
    'sphinx.ext.intersphinx',
    'sphinx.ext.todo',
    'sphinx.ext.mathjax',
    'sphinx.ext.ifconfig',
    'sphinx.ext.viewcode',
    'sphinx.ext.graphviz',
]
```

3.4 Enabling Conditional Rendering of Developer Info

To enable conditional rendering of developer information, add the following line to your *conf.py* and make sure the *sphinx.ext.ifconfig* extension is in your extension list.

```
def setup(app):
    app.add_config_value('includeDevInfo', 'false', 'env')
```

Warning: Do not enable usage of custom themes in your *conf.py*. The theme files are not available on the RTD server and thus building you project may either fail or in the very least lead to aweful rendering in the browser.

CHAPTER 4

Projects with Special Dependencies

In case your project has special (external) dependencies it is likely that the build will fail, as the RTD server does not know of this dependencies. You can however add a *requirements.txt* file to make RTD aware of these dependencies and run the documentation build in a virtual environment with these dependencies. In your documentation root (where the *conf.py* file is located) place a *readthedocs.requirements.txt* file. It should have the same format as any *pip* requirements file:

```
slumber>=1.0
matplotlib
```

This would for instance assure that *slumber* with a version higher 1.0 and a version of *matplotlib* is installed.

CHAPTER 5

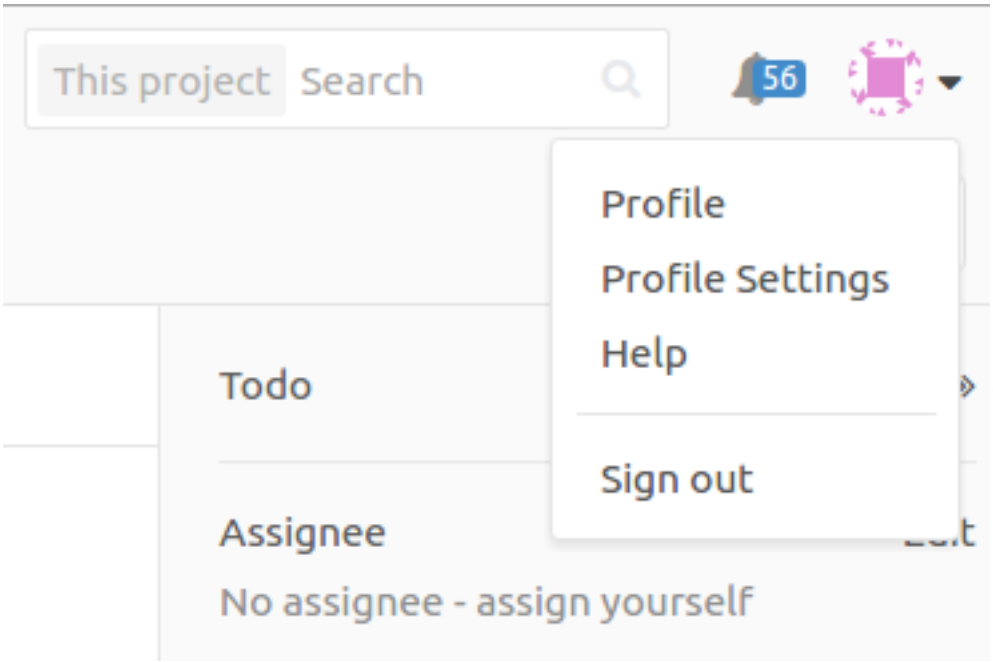
Adding a New Project from GitLab

Adding a new project from GitLab is simple. Before you start, you need to assure that a few prerequisites are met:

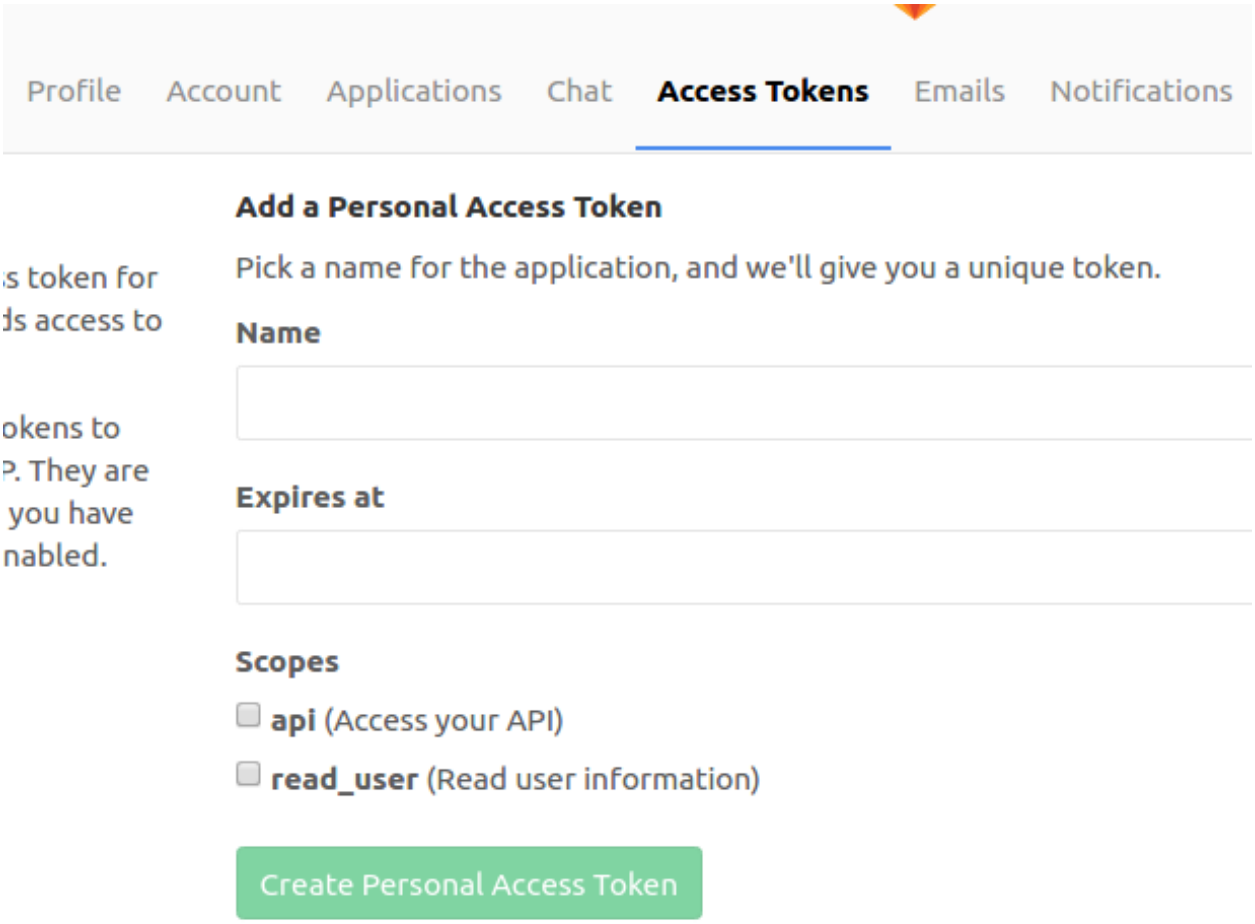
- your project has a sphinx compatible *conf.py* somewhere in its directory structure and a sphinx build with this files succeeds. See *Non-Karabo-Related Documentation* for information on how to create such a file.
- you do not have any special theme or style set in your *conf.py*. If you do please remove them.
- your project is hosted on our GitLab server. If it is publically available you can skip on to *Adding Your Project to RTD*, if not read on at *Adding an Access Token*.

5.1 Adding an Access Token

For private GitLab projects RTD needs some way to access these. As we cannot provide ssh keys for every project, and project's may have different members the acccess token offers a solution to this. To generate a personal access token navigate to your profile settings on GitLab and then on to "Access Tokens".



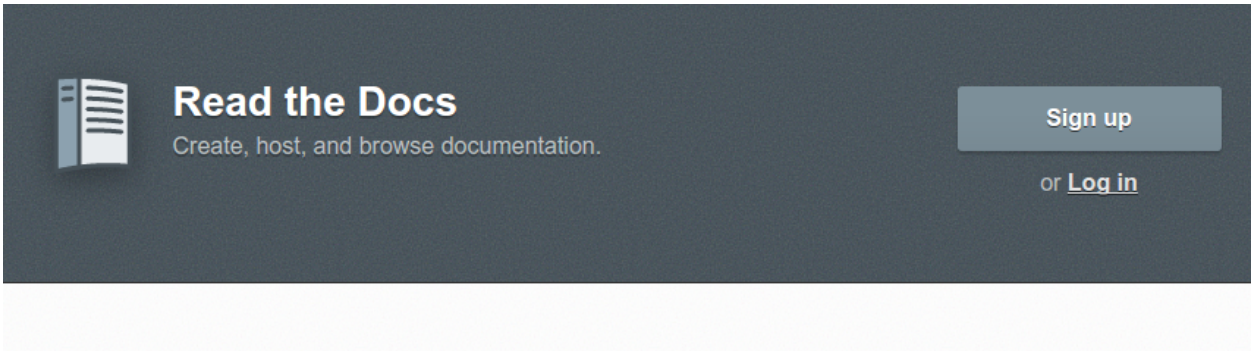
In the access token dialog give it a meaninful name, e.g. readthedocs, and select api as "scope". Clicking on "Create Personal Access Tokens" will create a token - be sure to save this somewhere, as it will not be retrievable after you leave the page.



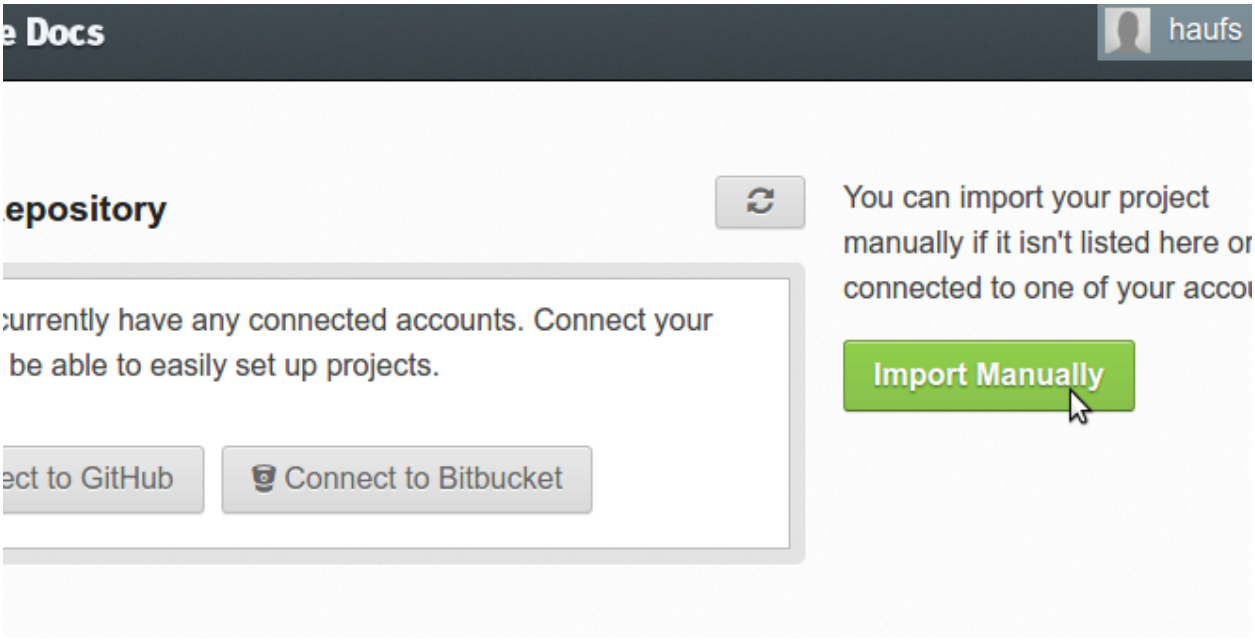
Copy this token for later usage.

5.2 Adding Your Project to RTD

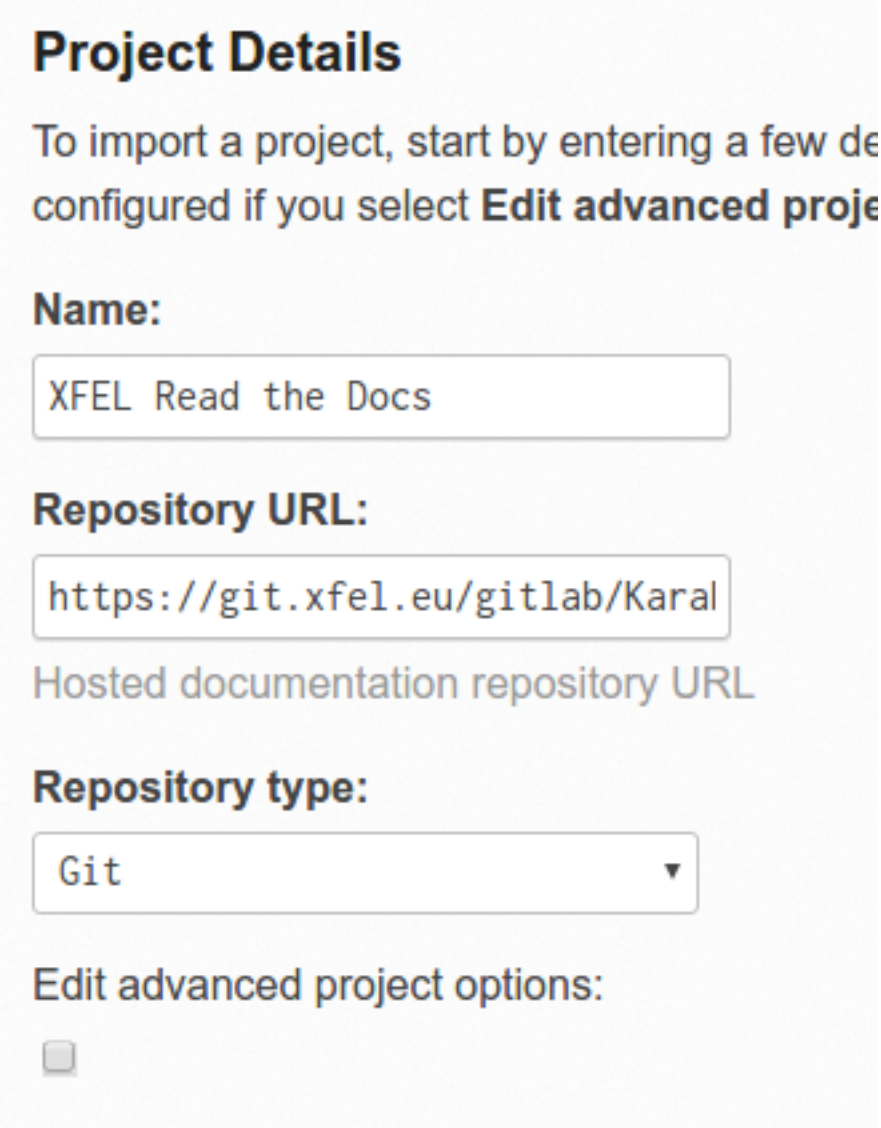
To add your project to RTD navigate to the RTD website and make sure you are logged in. You can log-in with your LDAP credentials. Given you are a member of the right group you will have the rights to create projects.



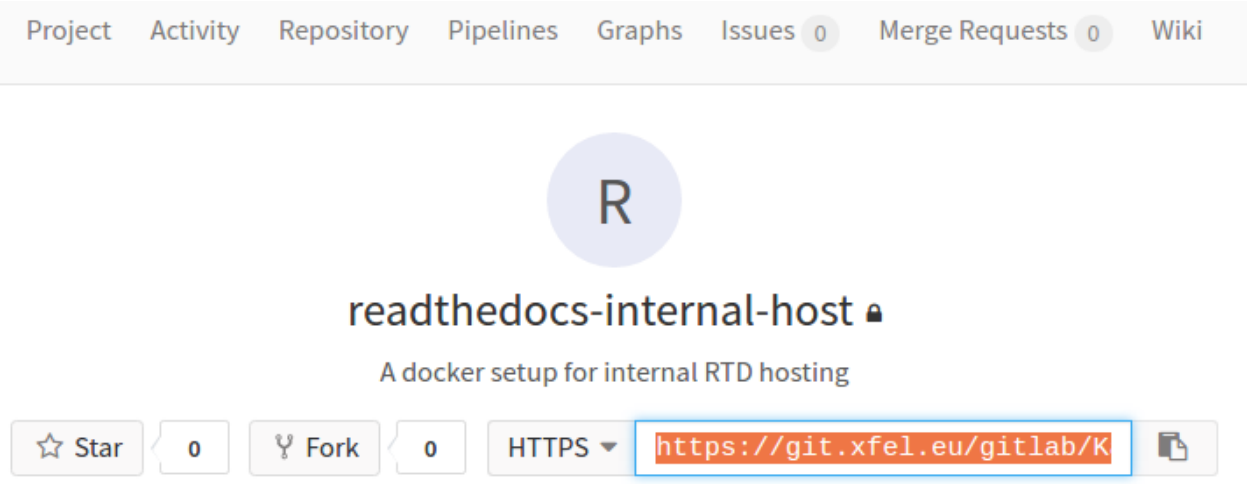
On the next page you can add a new project. For use with XFEL's internal GitLab server select *Import Manually*.



You will then be asked to give a project name (please choose something appropriate for your project), as well as the Gitlab location.

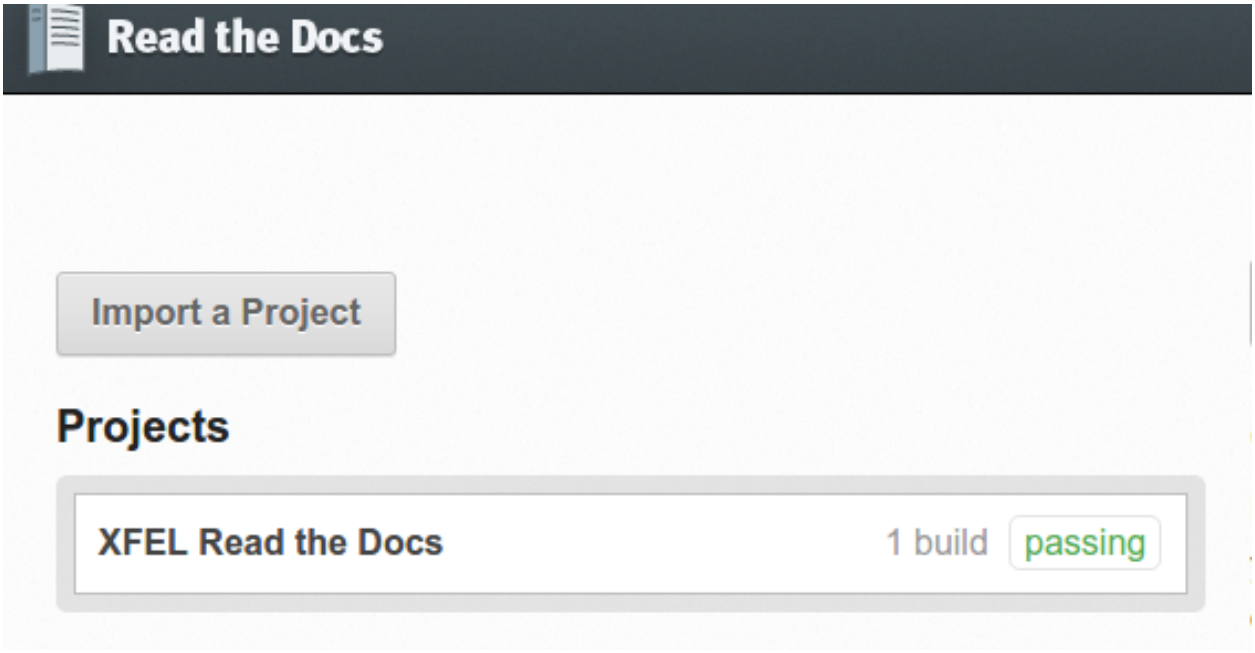


The location can be copied from GitLab on the project main page. Make sure you select the *https* address, not *ssh*:

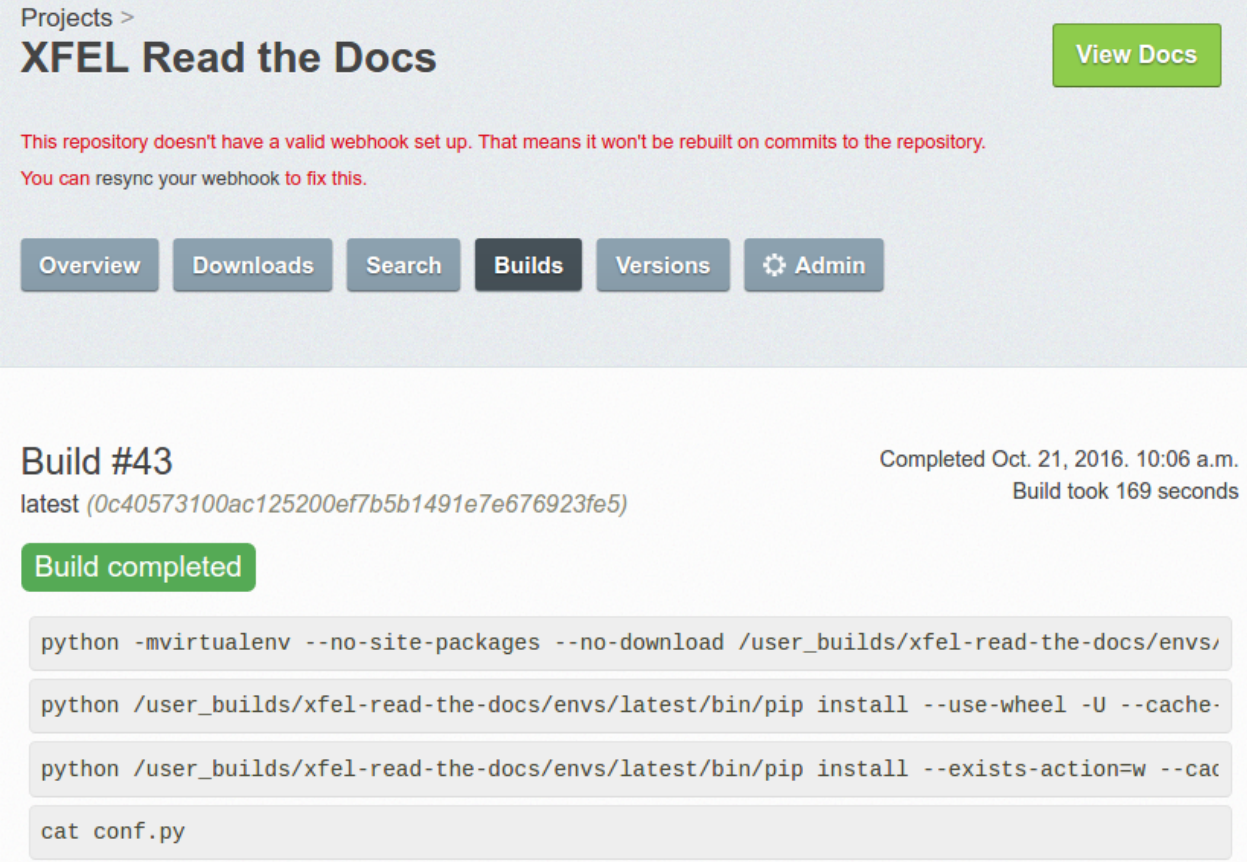


In case your repository is private you will now need to insert the access token into this address. It should have the form: *https://YOUR_LOGIN:YOUR_TOKEN@git.xfel.eu/...*

Where "YOUR_LOGIN" is your gitlab user name. Upon confirmation this page may not directly redirect. This depends on how much there is to clone. You can however go back to the main page and then click on your projects:

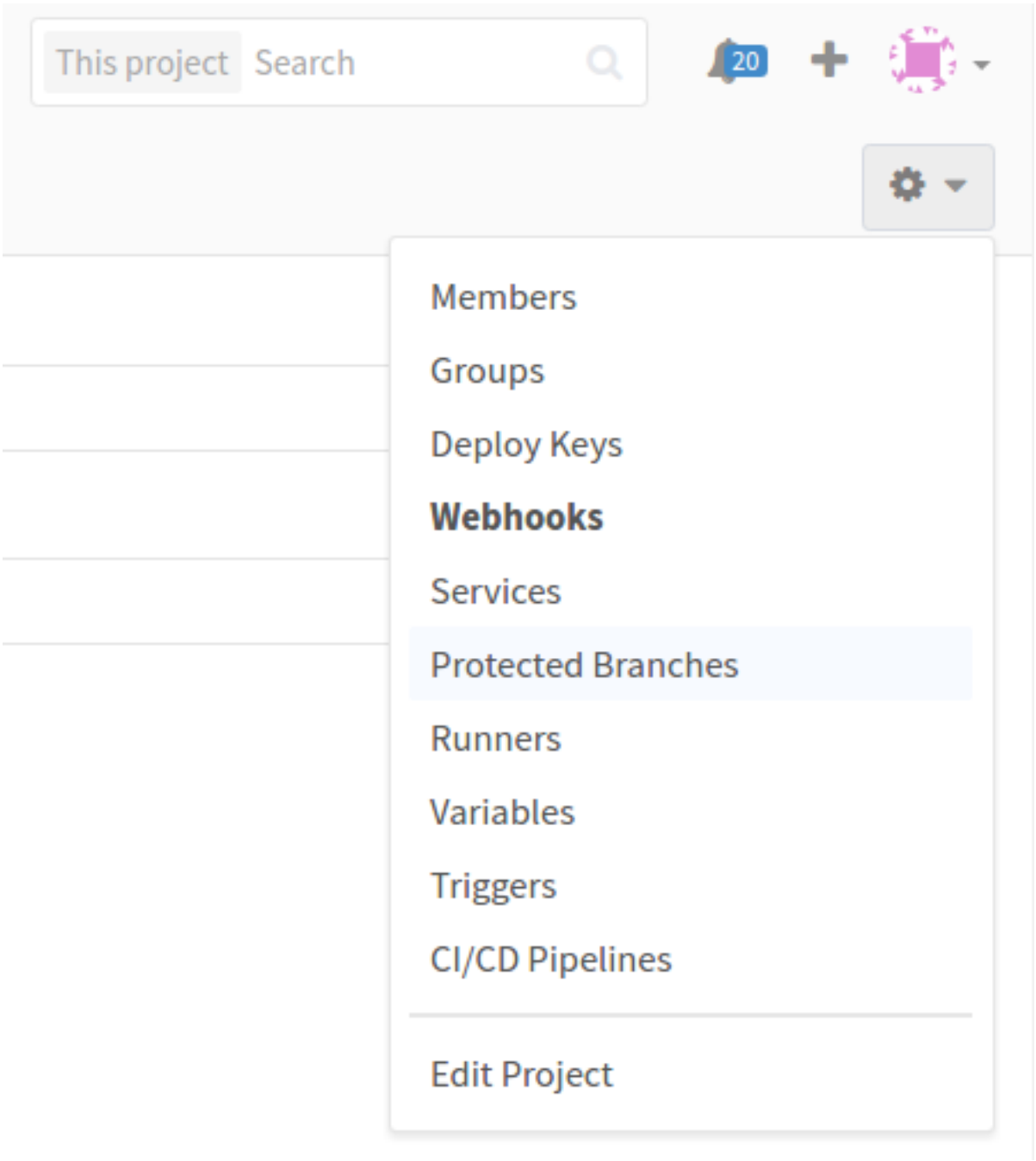


In the *Builds* menu you can check the status of the documentation build for your project.

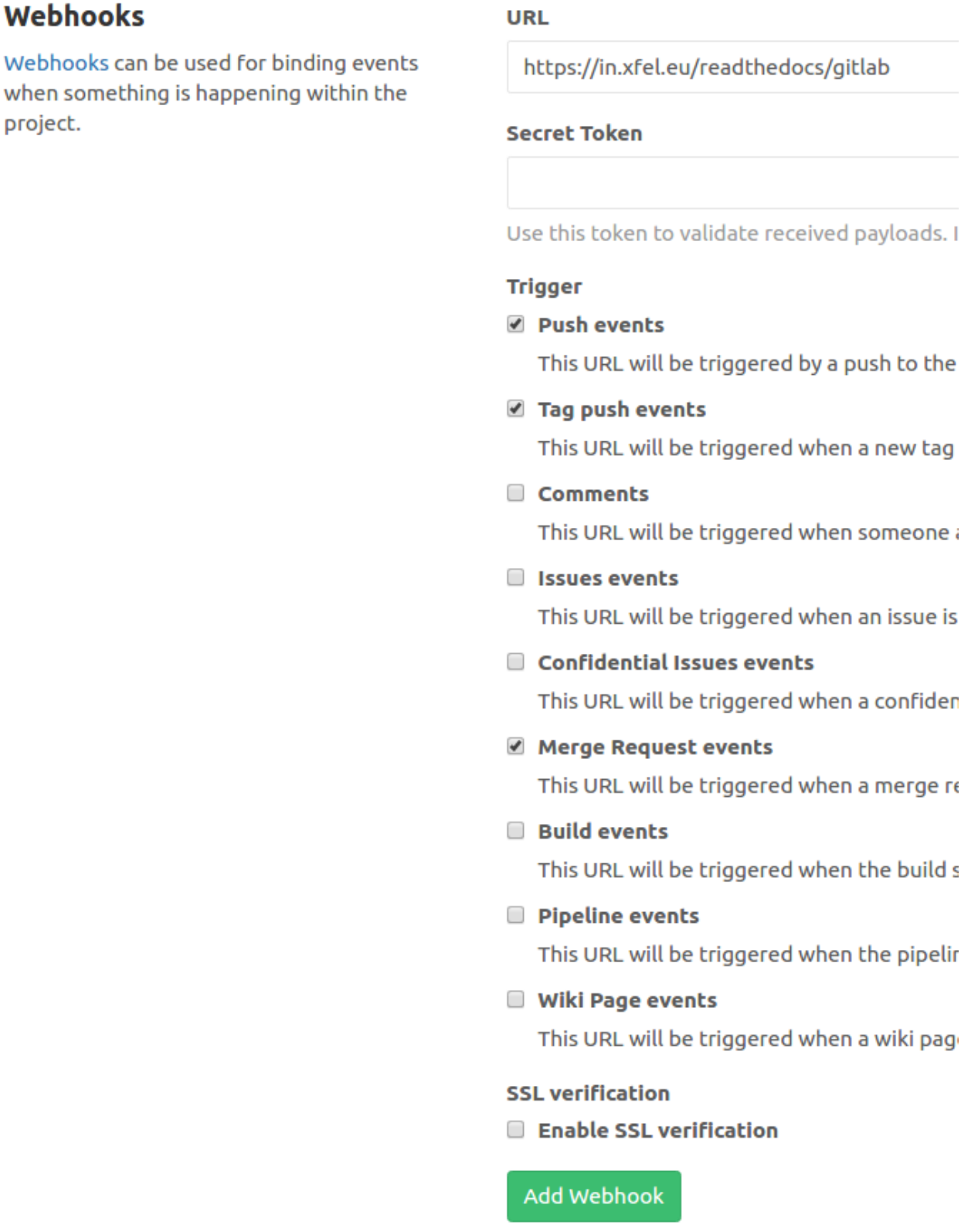


5.3 Enabling GitLab Webhooks

Note that RTD complains that you do not yet have a webhook set-up for this project. This means that new documentation builds will not be trigger upon commits to your project's git. We can remedy this in GitLab. Click on your project menu and select the *Webhooks* entry:



In the following dialog add the URL to XFEL's RTD server in the appropriate field: *https://in.xfel.eu/readthedocs/gitlab*. Select for which events on GitLab a new documentation build should be triggered. For Projects it is recommended to select *Push events*, for devices it is recommended to select *Tag events*.



After clicking *Add Webhook* the new webhook will appear in the bottom list. You can test it from there (will trigger a

documentation build of your project).

CHAPTER 6

Referencing across Projects

Assuming you have a *conf.py* configured to download the intersphinx bindings from XFEL's read-the-docs server (*Non-Karabo-Related Documentation*) use the following syntax to link across hosted projects: *:ref:'Karabo<karabo:introduction>'*. This will create a link to the Karabo project's introduction, referenced by its label *introduction*: Karabo. The project identifier, *karabo* in this case, corresponds to the URI-slug of the respective projects, with all dashes - replaced. You can deduce it from the url of the project. For

> Karabo this is *exflkarabo.desy.de/karabo/en/latest/*, the slug is the part right after the server address, hence *karabo*.

CHAPTER 7

Including in-code docs

Rendition of Karabo framework doc strings already works, this section describes how this can be achieved for doc strings in Karabo devices.

7.1 Steps required

An example Karabo devices including doc strings rendition is karaboDevices/puffin, viewing puffin files and performing the steps outlined below should get doc strings visible for your device project.

- ensure that your *conf.py* includes the *sphinx.ext.autodoc* extension.
- ensure that your sources are in the system path, e.g. by adding *sys.path.append('../../src/')* to *conf.py*.
- include in your index.rst's toctree the name of the file, for puffin this is puffin_code.rst, defining which source files should be searched for doc strings to render. Puffin currently renders only strings from Puffin.py, which makes puffin_code very short and avoids any complication. If you need complication, then do it and update this documentation to share your knowledge. The directives in puffin_code are sufficient to render class and class method doc strings.
- searching for doc strings in your project's source requires importing Karabo framework files and these have to be made available to the virtual environment used by RTD. The *Framework.git* directive, see puffin's requirements.rst below, must be inserted as-is into your project's *requirements.rst* to make framework files available. Note that the robotDevice.git directive is required by puffin as Puffin.py inherits from karaboDevices/RobotDevice, this file must also be made available and your project may have similar dependencies.

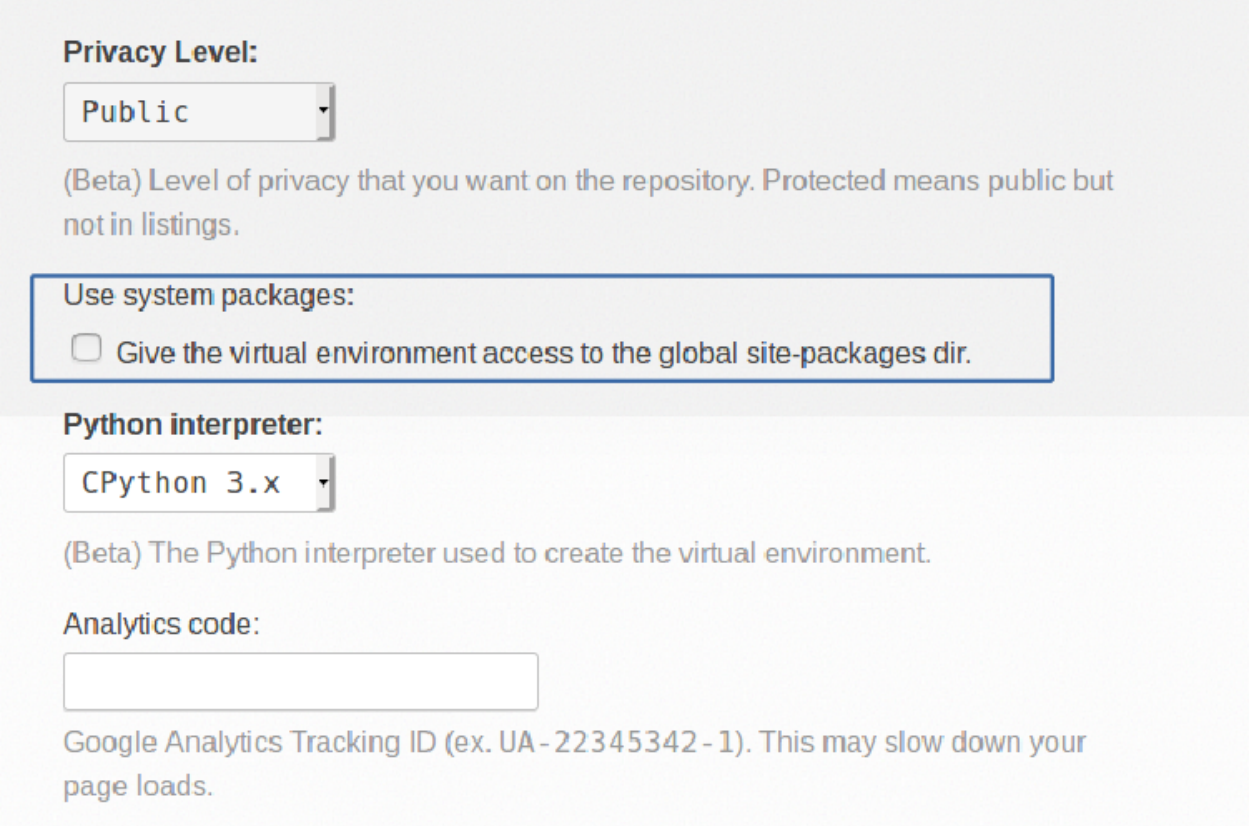
```
sphinx==1.4.5
pint
git+https://git.xfel.eu/gitlab/Karabo/rtd-conf-py.git
slumber
breathe
numpy
matplotlib
git+https://github.com/enthought/traits.git
git+https://xfel.redmine:K9Sss5CmWtQ_nbx3gyt2@git.xfel.eu/gitlab/Karabo/Framework.
↪git@master#egg=karabo&subdirectory=src/pythonKarabo
```

(continues on next page)

(continued from previous page)

```
git+https://xfel.redmine:K9Sss5CmWtQ_nbx3gyt2@git.xfel.eu/gitlab/karaboDevices/
↪robotDevice.git
lxml
```

It is not recommended to select the "Use system packages" option in (*Admin* > *Advanced Settings*), if a different version for one of the (internal) dependencies is required. e.g. sphinx 1.4.5 or sphinx-rtd-theme 0.4.3.



CHAPTER 8

How To Document

Note: Always start a file with reference to itself. E.g the first lines of this file look like this:

```
.. _project/howto_document:

***************
How To Document
***************
```

In order to allow proper cross-referencing later, use one of the following pre-fixes:

- *framework/* - For all framework related documenation
- *devices/* - For all device related documenation
- *projects/* - For documentation about karabo projects or setups

8.1 Headers

Only use one top header (section header) per file.

Top headers use stars (“*”) as surrounding. Sub-headers are underlined by “=”, “+”, “_” and “^” in this order.

8.1.1 Sub-Header

Sub-Sub Header

Sub-Sub-Sub Header

```
Headers
=======

Sub-Header
----------

Sub-Sub Header
++++++++++++++

Sub-Sub-Sub Header
```

8.2 Paragraphs

This text reflects a regular paragraph. Use the same indentation and separate text by adding one or more blank lines.

Do not use more than 80 characters per line.

8.3 Inline Markup

Inline markup should go like this: *emphasis*, **strong emaphasis**, `code related`. Escaping is sometimes needed, use a backslash if you want to show e.g. *, \, ‘.

Always use double backticks if you want to refer to a class or function, like: `void foo()`

```
Inline markup should go like this: *emphasis*, **strong emaphasis**,
``code related``. Escaping is sometimes needed, use a backslash if you want
to show e.g. \*, \\, \`.

Always use double backticks if you want to refer to a class or function, like:
``void foo()``
```

8.4 Lists

- Bulleted item
- Bulleted item
  - Nested bulleted item (watch the blank line)
  - Nested bulleted item
- Bulleted item

1. Numbered item
2. Numbered item
   1. Nested numbered item (watch the blank line)
   2. Nested numbered item
3. Numbered item 3

1. Explicit item
2. Explicit item
    a. Nested explicit item
    b. Nested explicit item
3. Explicit item

```
* Bulleted item
* Bulleted item

  * Nested bulleted item (watch the blank line)
  * Nested bulleted item

* Bulleted item


#. Numbered item
#. Numbered item

   #. Nested numbered item (watch the blank line)
   #. Nested numbered item

#. Numbered item 3


1. Explicit item
2. Explicit item

   a. Nested explicit item
   b. Nested explicit item

3. Explicit item
```

8.5 Links

Use Link text for inline web links.

Internal links should look like this framework/howto_document, which refers to the own section.

```
Use `Link text <http://xfel.eu>`_ for inline web links.

Internal links should look like this :ref:`framework/howto_document`,
which refers to the own section.
```

8.6 Special Directives

Topic title

This is a topic. Something that highlights in a box.

See also:

This is a seealso.

Note: This is a note.

Warning: This is a warning.

```
.. topic:: Topic title

   This is a topic. Something that highlights in a box.

.. seealso::

   This is a seealso.

.. note::

   This is a note.

.. warning::

   This is a warning.

.. todo::

   This is a todo note.

.. ifconfig:: includeDevInfo is True

   This is information that is very detailed and can be switched off during
   rendering.
```

8.7 Tables

- Simple tables are formatted like so:

A	B	A and B
False	False	False
True	False	False
False	True	False
True	True	True

```
=====  =====  =======
  A      B    A and B
=====  =====  =======
False  False  False
True   False  False
False  True   False
True   True   True
=====  =====  =======
```

- Complex tables are formatted like so:

Title 1	Title 2
Some entry	Single row
Some other entry	Split row
	Split row

```
+------------------+-----------------------------------------------------------+
|**Title 1**       |**Title 2**                                                |
+------------------+-----------------------------------------------------------+
|Some entry        |Single row                                                 |
+------------------+-----------------------------------------------------------+
|Some other entry  |Split row                                                  |
|                  +-----------------------------------------------------------+
|                  |Split row                                                  |
+------------------+-----------------------------------------------------------+
```

- Tables from csv format can be imported like so:

```
.. csv-table:: Table Title
   :file: myfile.csv
```

8.8 Code

Code blocks are initiated by

```
@Slot
def foo(self):
    """Does nothing"""
    pass
```

```
KARABO_REGISTER_SLOT(foo);
void foo() {
    // Does nothing
}
```

```
 .. code-block:: Python

@Slot
def foo(self):
    """Does nothing"""
    pass
```

```
KARABO_REGISTER_SLOT(foo);
void foo() {
    // Does nothing
}
```

8.9 Images and Figures

To add an image use:

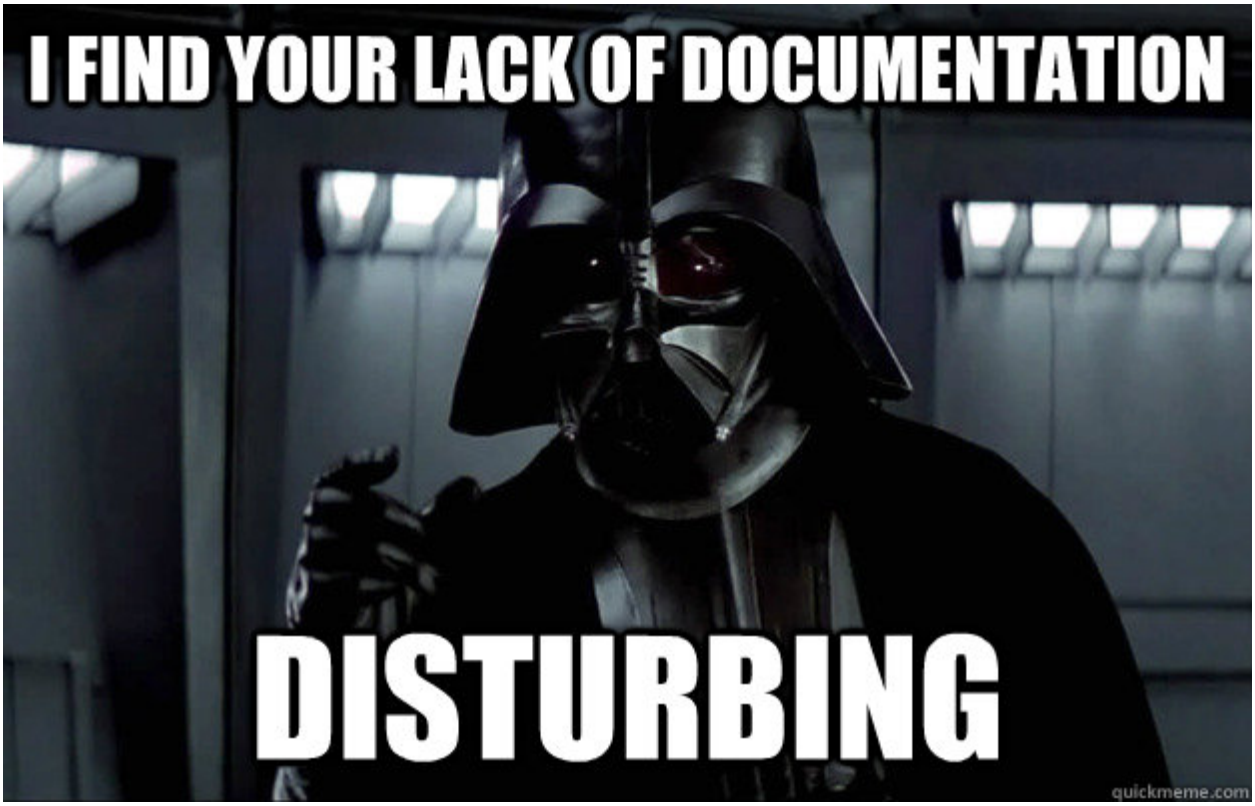


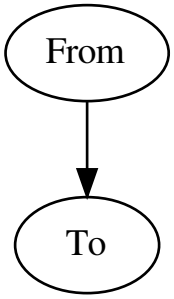
Fig. 1: Figures are like images but with a caption

```
.. image:: _images/darthvader.jpg

.. figure:: _images/xfel.jpg
    :align: center
    :height: 100px
    :alt: alternate text
    :figclass: align-center

    Figures are like images but with a caption
```

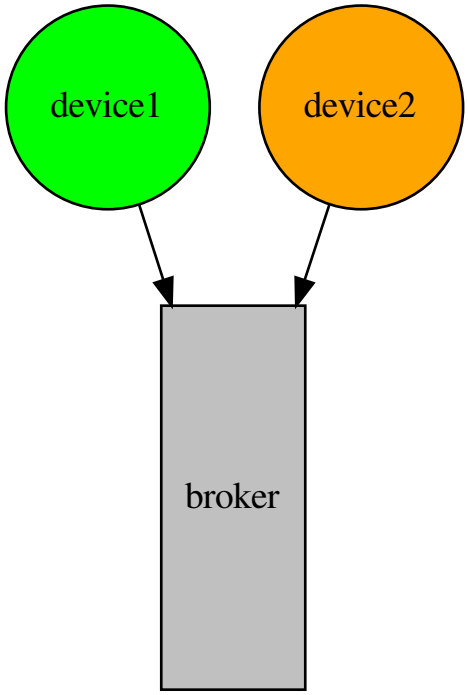
8.10 Graphs

Drawing of graphs is also supported:

8.10.1 Examples




```
.. graphviz::

   digraph {
      "From" -> "To";
   }
```



```
.. digraph:: example

   "device1" [shape=circle, style=filled, fillcolor=green]
   "device2" [shape=circle, style=filled, fillcolor=orange]
   "broker"  [shape=box, height=2, style=filled, fillcolor=gray]

   "device1" -> "broker"
   "device2" -> "broker"
```

8.11 Math

$$n_{\text{offset}} = \sum_{k=0}^{N-1} s_k n_k$$

```
.. math::

    n_{\mathrm{offset}} = \sum_{k=0}^{N-1} s_k n_k
```

8.12 Footnotes

Some text that requires a footnote[1] .

```
Some text that requires a footnote [#f1]_ .

.. rubric:: Footnotes

.. [#f1] Text of the first footnote.
```

[1] Text of the first footnote.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search