PyCon APAC 2016

Easy contributable internationalization process with Sphinx

Takayuki Shimizukawa

Sphinx co-maintainer

Sphinx-users.jp

안녕하세요.

annyeonghaseyo.

나는시미즈카와

입니다.

naneun shimizukawa ibnida.

こんにちは

Kon-Nichi-Wa

Who am I

@shimizukawa

- Python developer since 2004

- Sphinx co-maintainer
- Sphinx-users.jp organizer
- PyCon JP committee

- BePROUD co, ltd. member

Respect, Diversity

of languages at PyCon APAC 2016

Agenda

1. Sphinx introduction & Setup for i18n
2. A easy way and Non easy ways to translate
3. Sphinx i18n feature
4. Automated translation process with several services
5. Tips

Question

Question1

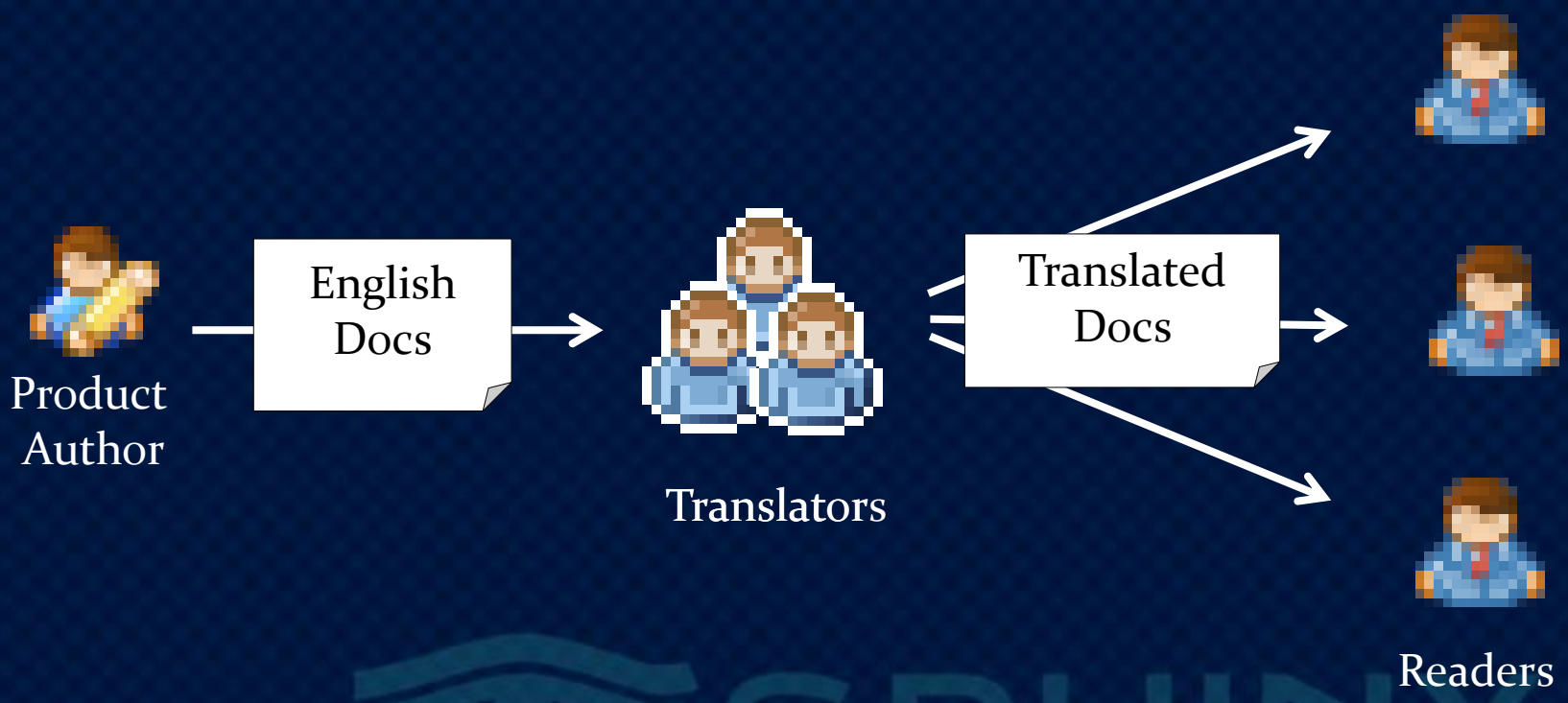
How many people do you use Open Source Software?

Question2

How many people may have contributed to the OSS?

Question3

Does the translation into familiar language contribute to other developers?


Product Author
English Docs
Translators
Translated Docs
Readers

Tools & Services

Tools

1. sphinx - documentation generator
2. docutils - base of sphinx
3. sphinx-intl - support utility for sphinx i18n
4. transifex-client - file transfer tool for Transifex

Services

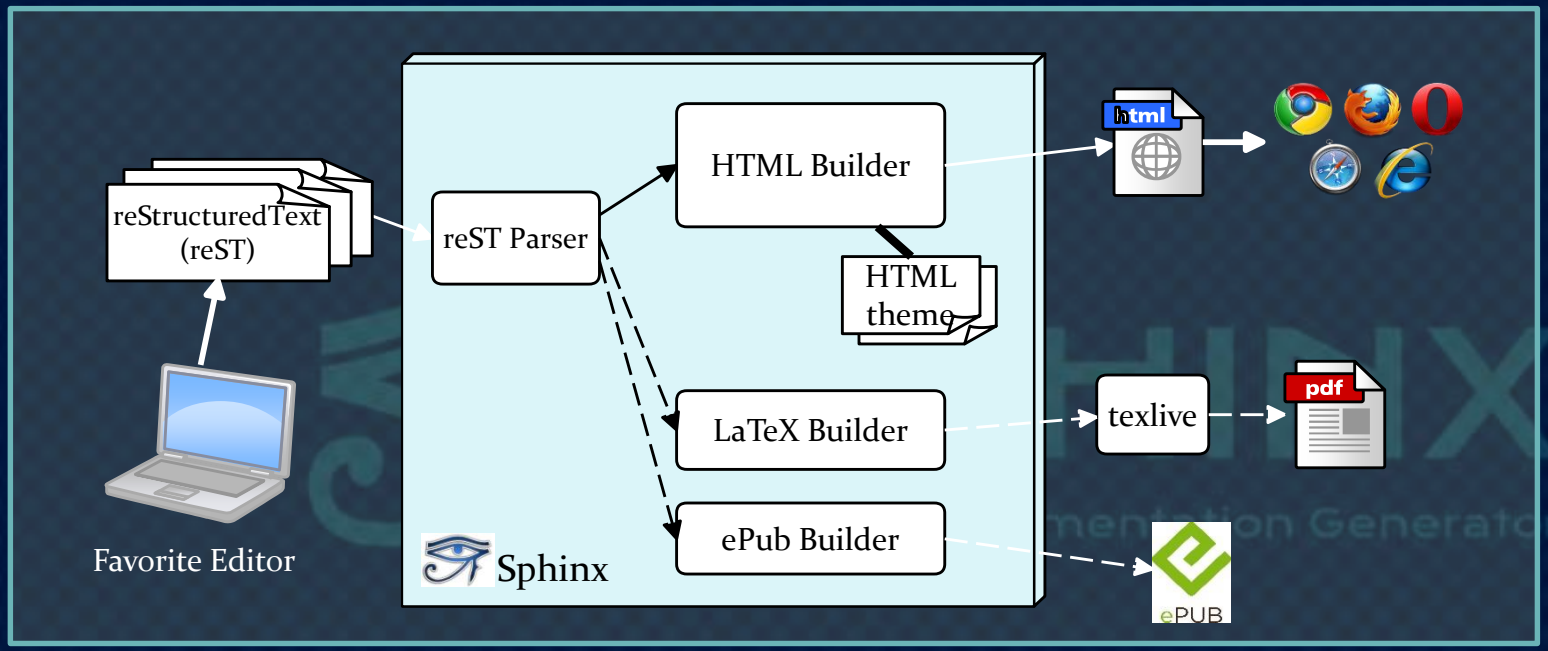
1. Transifex - translation support service
2. Drone.io - continuous integration service

1. Sphinx introduction & Setup for i18n

What is Sphinx?

Sphinx is a documentation generator

Sphinx generates doc as several output formats from the reST text markup



The history of Sphinx (short ver.)

The father of Sphinx

Too hard to maintenance

Easy to write
Easy to maintenance

~2007

python™ + LaTeX

2007~

python™ + SPHINX Python Documentation Generator

Sphinx Before and After

Before

- There was no standard ways to write documents
- Sometime, we need converting markups into other formats

After

- Generate multiple output format from single source
- Integrated html themes make read docs easier
- API references can be integrated into narrative docs
- Automated doc building and hosting by ReadTheDocs

Many docs are written by Sphinx

For examples

- Python libraries/tools:

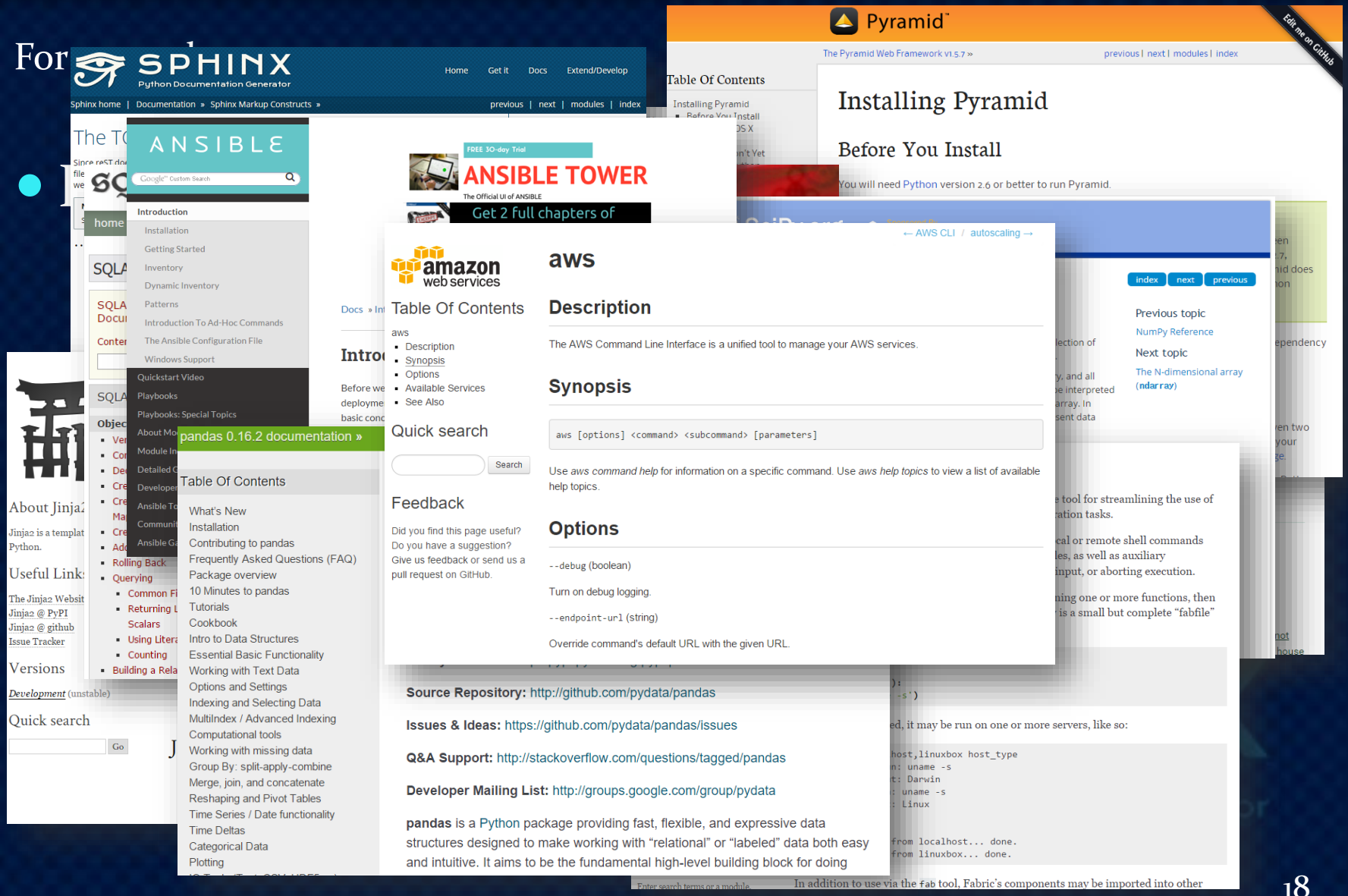
  Python, Sphinx, Flask, Jinja2, Django, Pyramid, SQLAlchemy, Numpy, SciPy, scikit-learn, pandas, fabric, ansible, awscli, ...

- Non python libraries/tools:

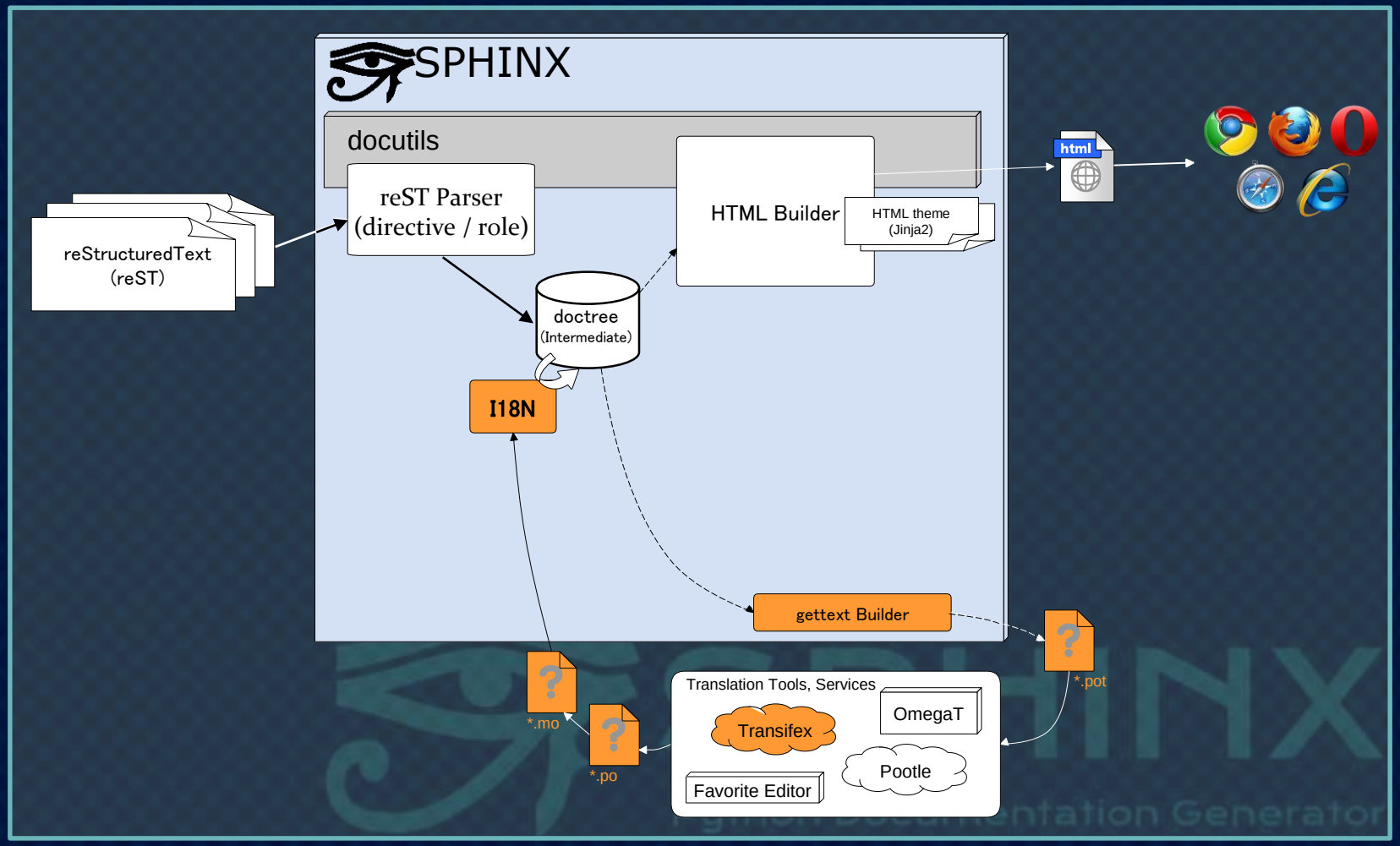
  Chef, CakePHP(2.x), MathJax, Selenium, Varnish

Many docs are written by Sphinx

For



Sphinx i18n feature (built-in)



How to install Sphinx with i18n tools

To build a sphinx document

```
$ pip install sphinx
```

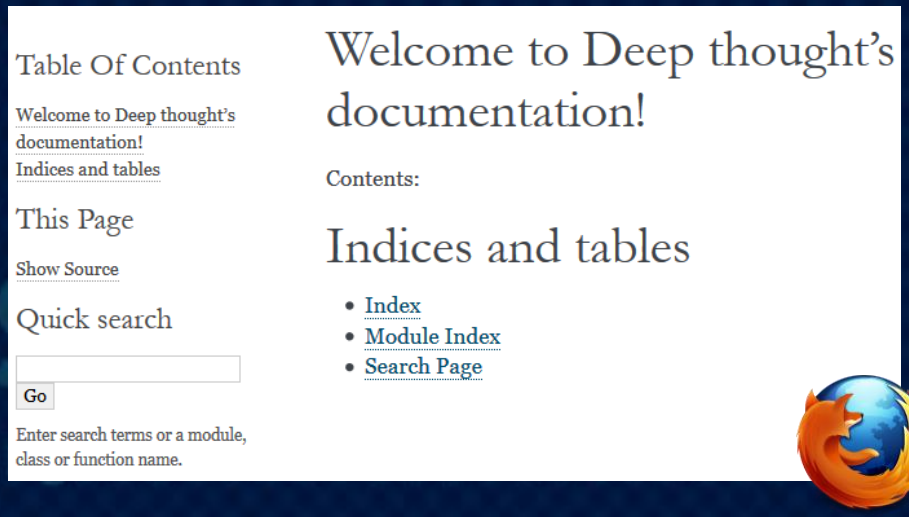
Support tools for translation

```
$ pip install sphinx-intl
$ pip install transifex-client
```

make html once

```
$ git clone https://github.com/shimizukawa/deepthought.git
$ cd deepthought/doc
$ make html
...
Build finished. The HTML pages are in _build/html.
```

"make html" command generates html files into _build/html.



Files & setup conf.py

```
$ tree /path/to/deepthought
+- deep_thought
|    +- __init__.py
|    +- api.py
|    +- calc.py
|    +- utils.py
+- doc
|    +- _build/
|    |     +- html
|    +- _static/
|    +- _templates/
|    +- conf.py
|    +- index.py
|    +- make.bat
|    +- Makefile
+- setup.py
```

doc/conf.py

```
1.  ...
2.
3.  language = 'ko'
4.  locale_dirs = ['locale']
5.
```

Document source

2. Easy contributable internationalization process with Sphinx

Easy contributable
internationalization
process with Sphinx

What is internationalization?

I18N

Easy contributable internationalization process with Sphinx

What is easy contributable?

Not a easy way (example 1/3)

The manual are provided only in the HTML files

- You can rewrite HTML files
- How to follow the original?

Not a easy way (example 2/3)

The HTML manual are generated from
reST files and
docstrings in the source files

- You can rewrite reST files and docstrings
- How to follow the upstream?

Not a easy way (example 3/3)

OK, we are engineers. we can use GIT!

- Learn git
- Learn GitHub
- git clone to get the code
- Translation (Yes! This is what I want to do!)
- git commit
- git pull
- Sometimes you must resolve conflict
- git push

Not easy vs easy

Not a easy way

1. Learn git
2. Learn GitHub
3. git clone to get the code
4. Translation
5. git commit, pull, push
6. Resolve conflict
7. Build html your self

Easy way

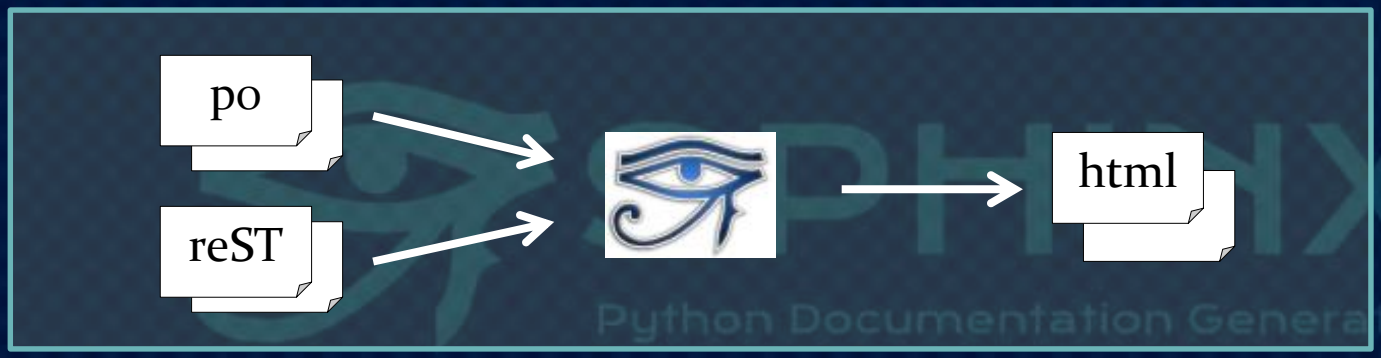
1. No git
2. No Github
3. No file
4. Translation
5. Update Automatically
6. No conflict
7. You can get a HTML output w/o hand-build.

3. Sphinx i18n feature

Two i18n features of Sphinx

- Output pot files:
  from reST

- Input po files:
  to generate translated HTML



Translation flow

- Generate pot

te pot into po

- Generate Translated HTML

Output pot files


```
$ make gettext
...
Build finished. The message catalogs are in _build/gettext.

$ ls _build/gettext
api.pot  examples.pot  generated.pot  index.pot
```

generated.pot

```
#: ../../../deep_thought/utils.py:docstring of
deep_thought.utils.dumps:1
msgid "Serialize ``obj`` to a JSON formatted :class:`str`."
msgstr ""

msgid "For example:"
msgstr ""
```

Preparing po files to translate


```
doc
+- _build/
|   +- gettext/
|      +- api.pot
|      +- examples.pot
|      +- generated.pot
|      +- index.pot
+- locale/
```

copy and rename

for each langs

```
doc
+- _build/
+- locale/
   +- ko/
   |   +- LC_MESSAGES/
   |      +- api.po
   |      +- examples.po
   |      +- generated.po
   |      +- index.po
   +- ja/
```

Translate them

Preparing po files to translate

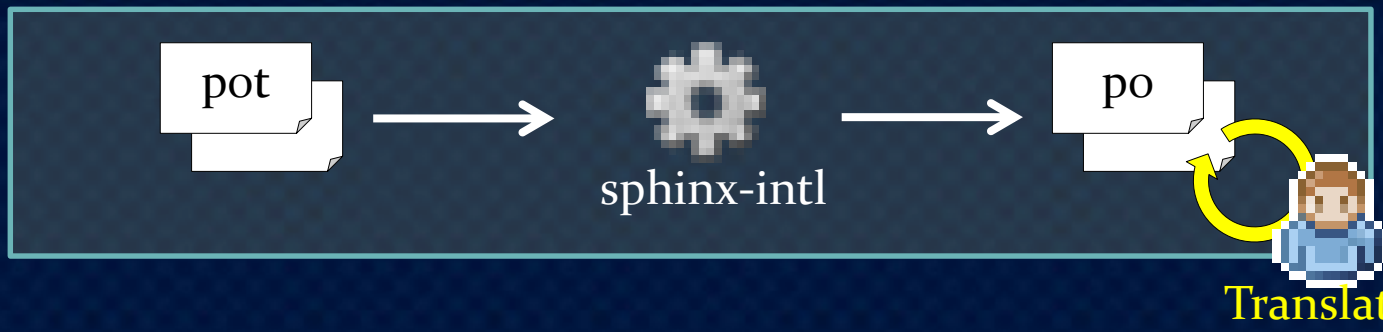
At first, sphinx-intl copy pot files and rename them

```
$ sphinx-intl update -p _build/gettext -l ko
Create: locale/ko/LC_MESSAGES/api.po
Create: locale/ko/LC_MESSAGES/examples.po
Create: locale/ko/LC_MESSAGES/generated.po
Create: locale/ko/LC_MESSAGES/index.po
```

After change the document, sphinx-intl update differences

```
$ make gettext
$ sphinx-intl update -p _build/gettext -l ko
Not Changed: locale/ko/LC_MESSAGES/api.po
Updated: locale/ko/LC_MESSAGES/examples.po +3, -1
```

Translate po files



generated.po

```
#: ../../../deep_thought/utils.py:docstring of
deep_thought.utils.dumps:1
msgid "Serialize ``obj`` to a JSON formatted :class:`str`."
msgstr ""
```

Translate ↓ by using Vim, Emacs, OmegaT, ...

generated.po

```
#: ../../../deep_thought/utils.py:docstring of
deep_thought.utils.dumps:1
msgid "Serialize ``obj`` to a JSON formatted :class:`str`."
msgstr "JSON 형식의 :class:`str` 유형에 ``obj`` 를 직렬화."
```

Input po files

유틸리티 함수

깊은 생각은 여러 가지 유틸리티 기능을 제공합니다.

deep_thought.utils.**dumps**(*obj*, *ensure_ascii=True*) [소스]

JSON 형식의 **str** 유형에 obj 를 직렬화.

예를 들면:

```
>>> from deep_thought.utils import dumps
>>> data = dict(spam=1, ham='egg')
>>> dumps(data)
'{spam: 1, ham: "egg"}'
```

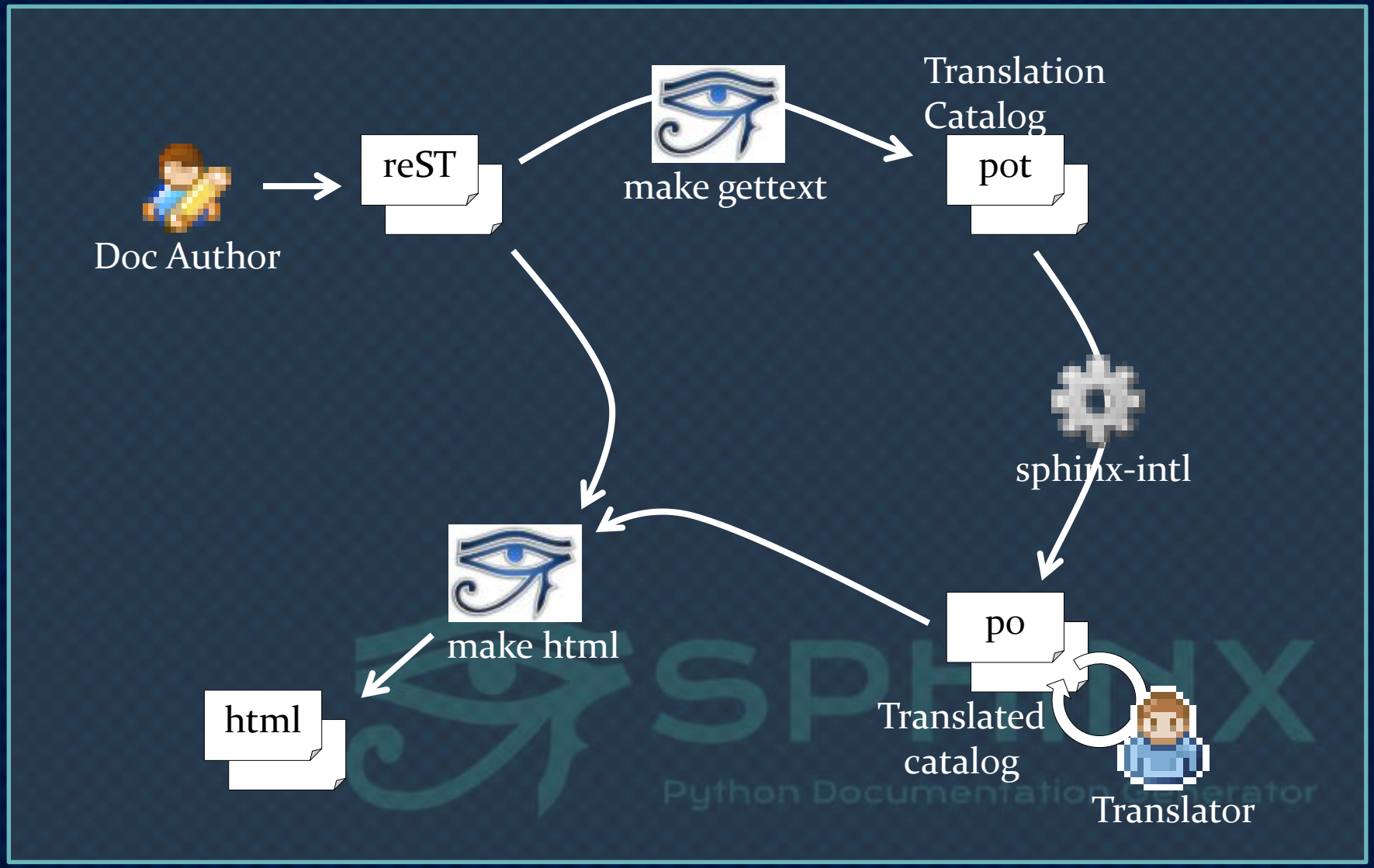
매개 변수:
- **obj** (*dict*) – 객체 직렬화 DICT 유형입니다.
- **ensure_ascii** (*bool*) – 기본값은 True입니다. 거짓 경우, 모든 비 ASCII 문자가 아닌 ...

반환: JSON 형식의 문자열

반환 형식: str

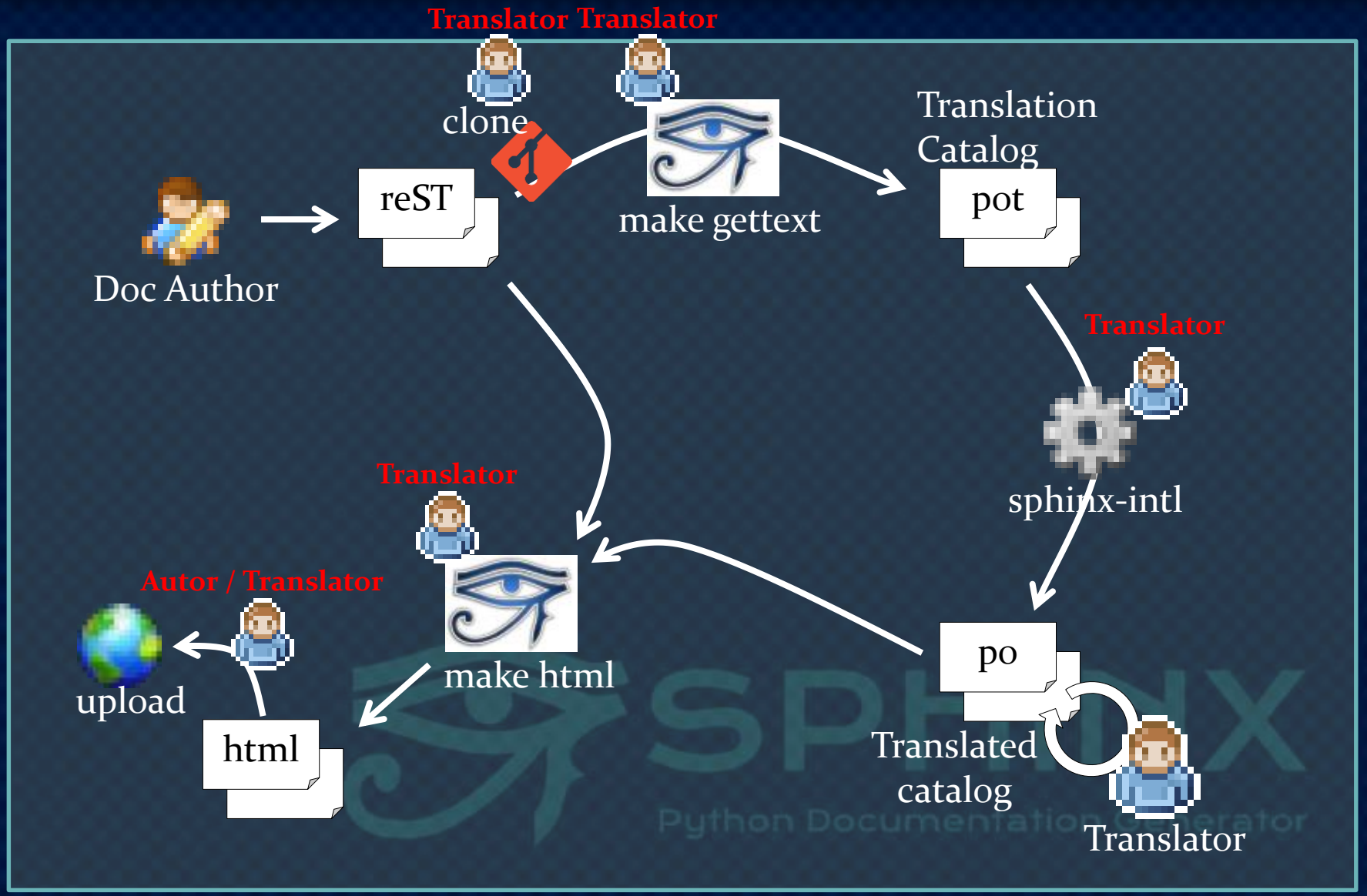
```
$ make html
...
Build finished. The HTML pages are in _build/html.
```

Big picture

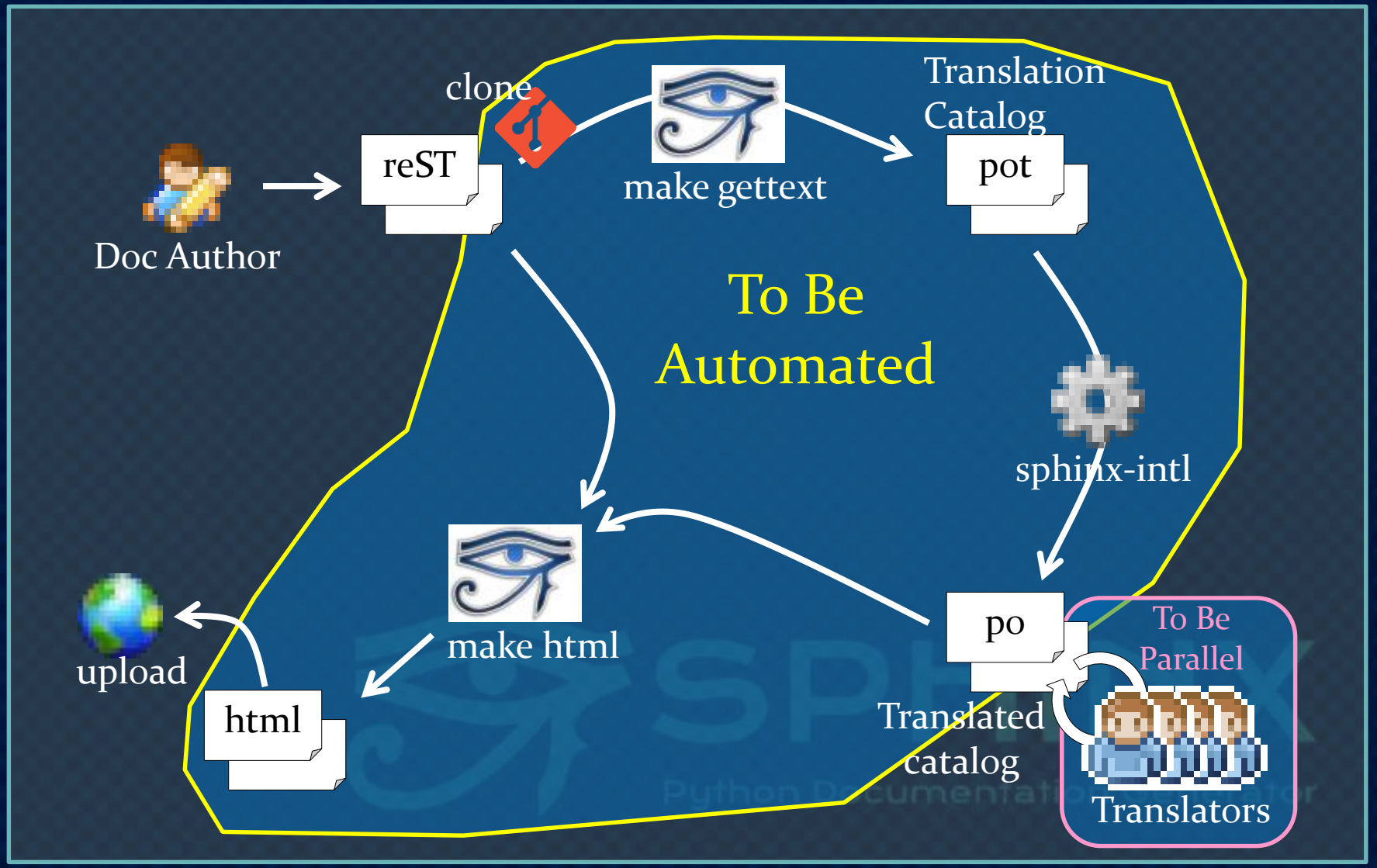

Translation
Catalog
reST
make gettext
pot
Doc Author
sphinx-intl
make html
po
html
Translated
catalog
Translator

4. Automated translation process with several services

Entire process to translate sphinx docs



To Be

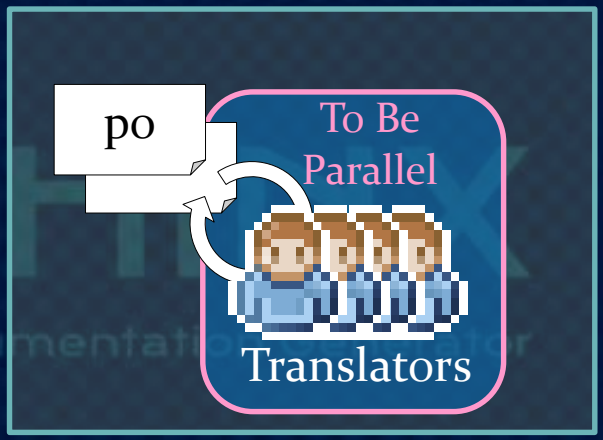

clone
Translation
Catalog
reST
make gettext
pot
Doc Author
To Be
Automated
sphinx-intl
make html
po
To Be
Parallel
upload
html
Translated
catalog
Translators

Translation tool types

- Vim / Emacs (Editors)
  - Edit local files
  - Translation support plugins are available.

- OmegaT (Translation Tools)
  - Edit local files.
  - Translation support features.

- Transifex (Services)
  - Edit online
  - Translation support features.



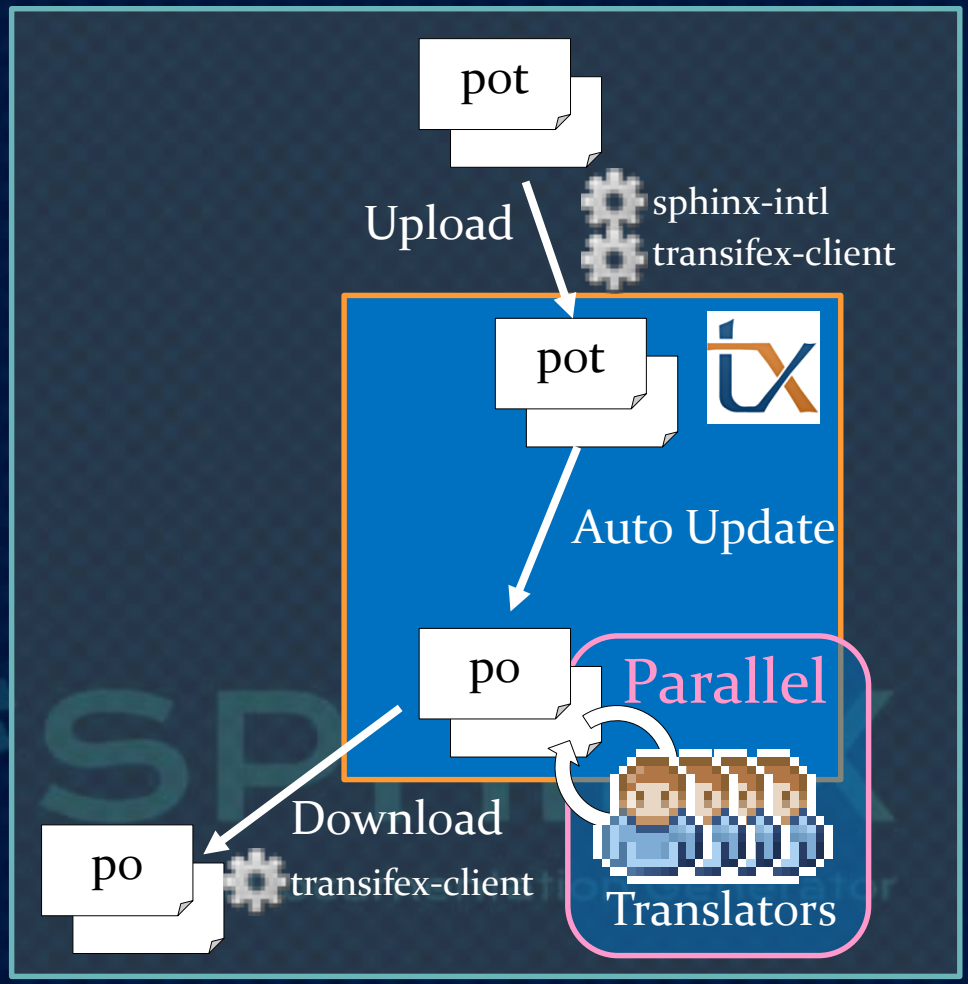
Be Parallel by using Transifex

Transifex provides...
As API:

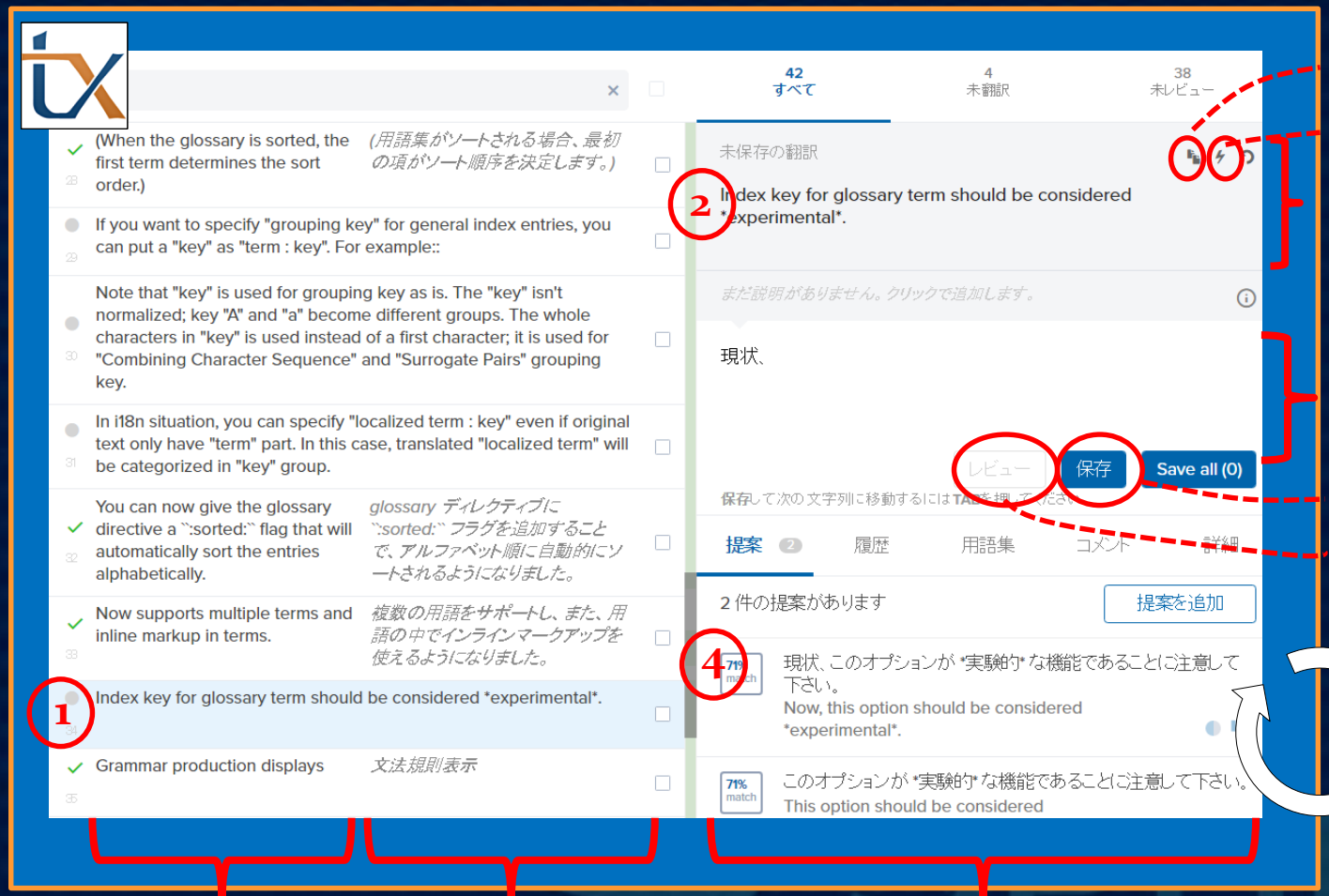
- Upload pot
- Download po

As Web console:

- Glossary
- Translation memory
- Recommendation
- Auto-translation



Translation on Transifex web console



3 Copy orig to translation

Machine translation

Original Text

Translated Text (you should keep reST syntax)

Save 5

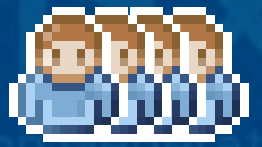
Review (if needed) 6

Original (pot)

Translation (po)

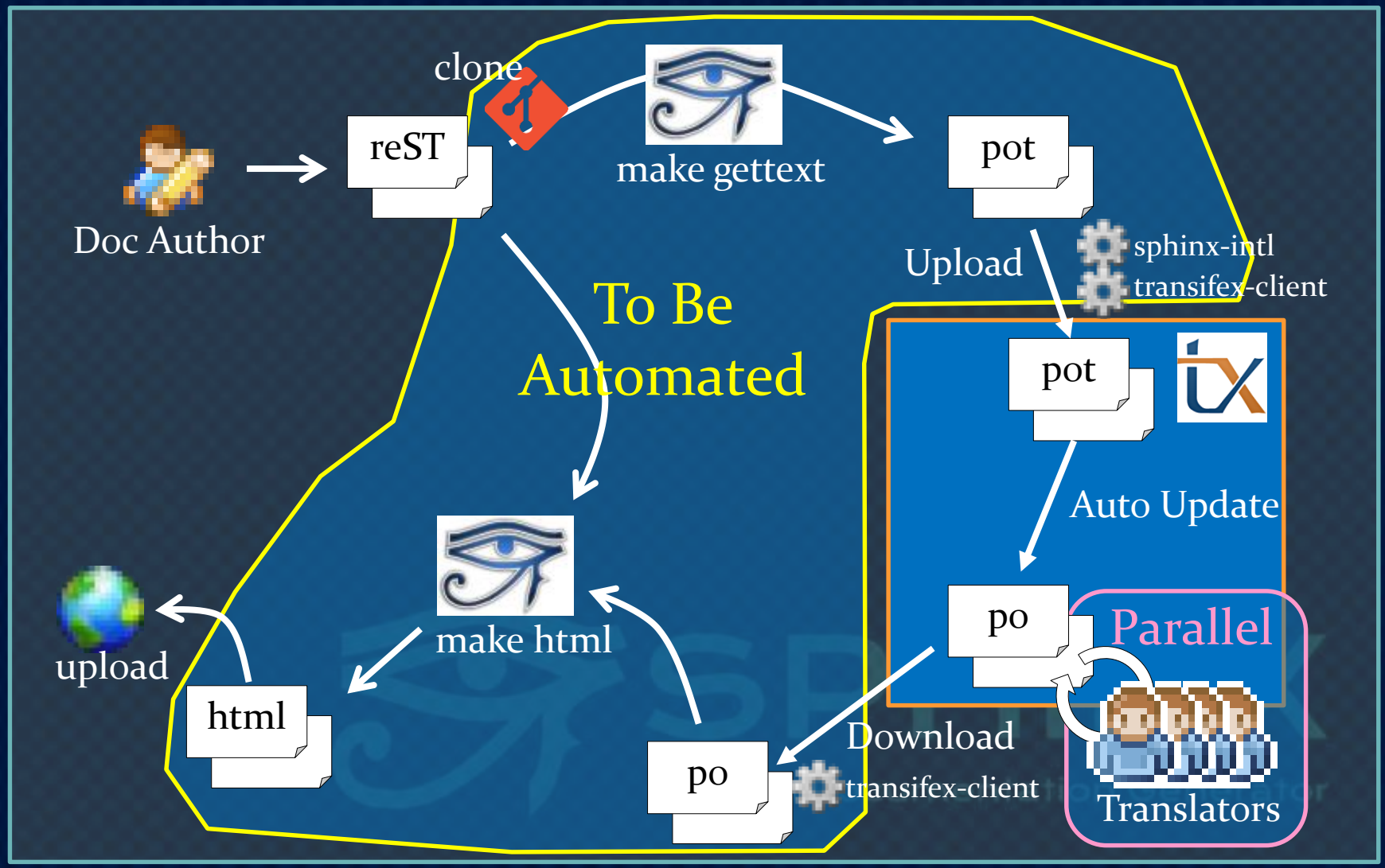
Suggestions from Translation Memory (TM)

Parallel

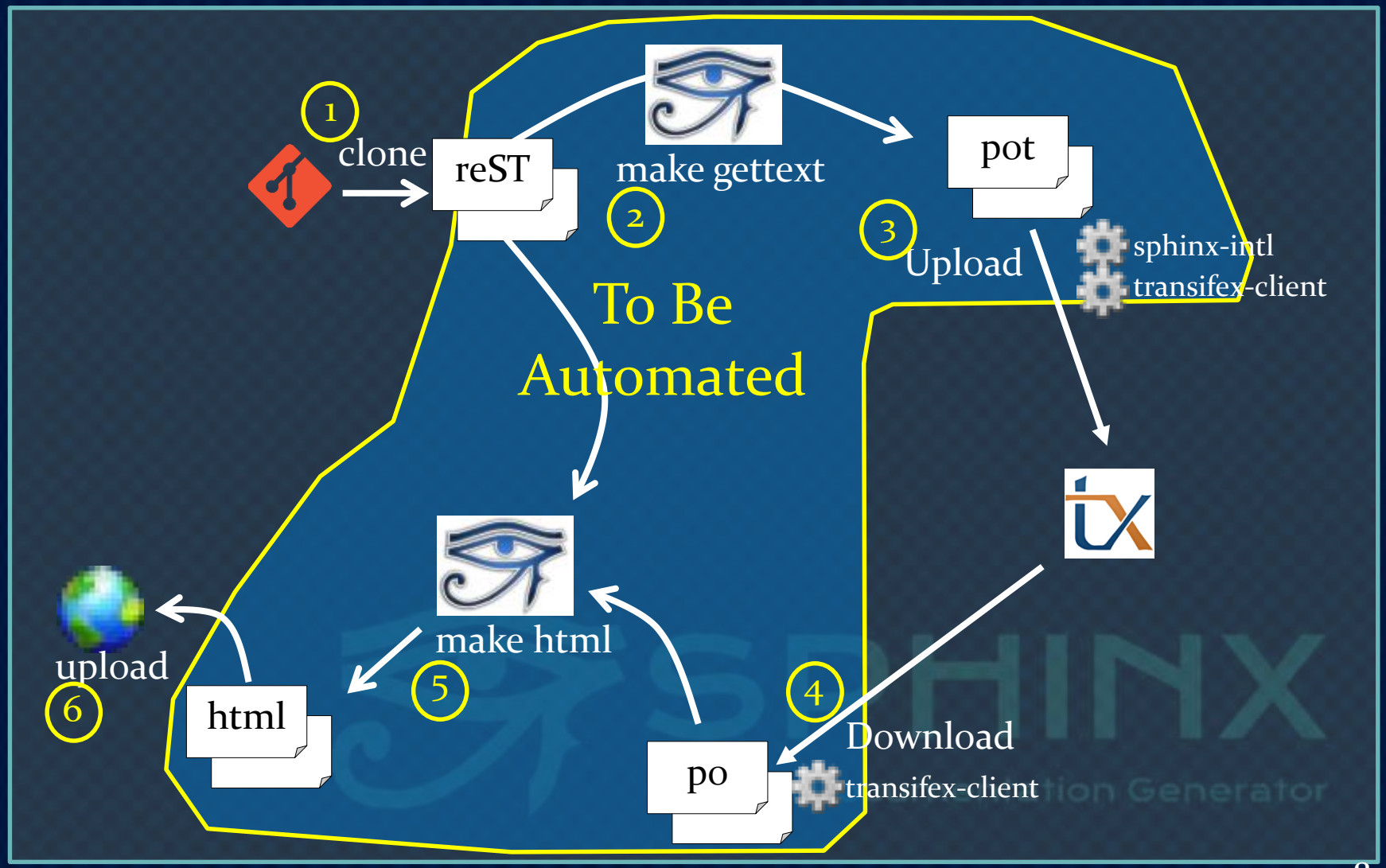


Translators

To Be Automated

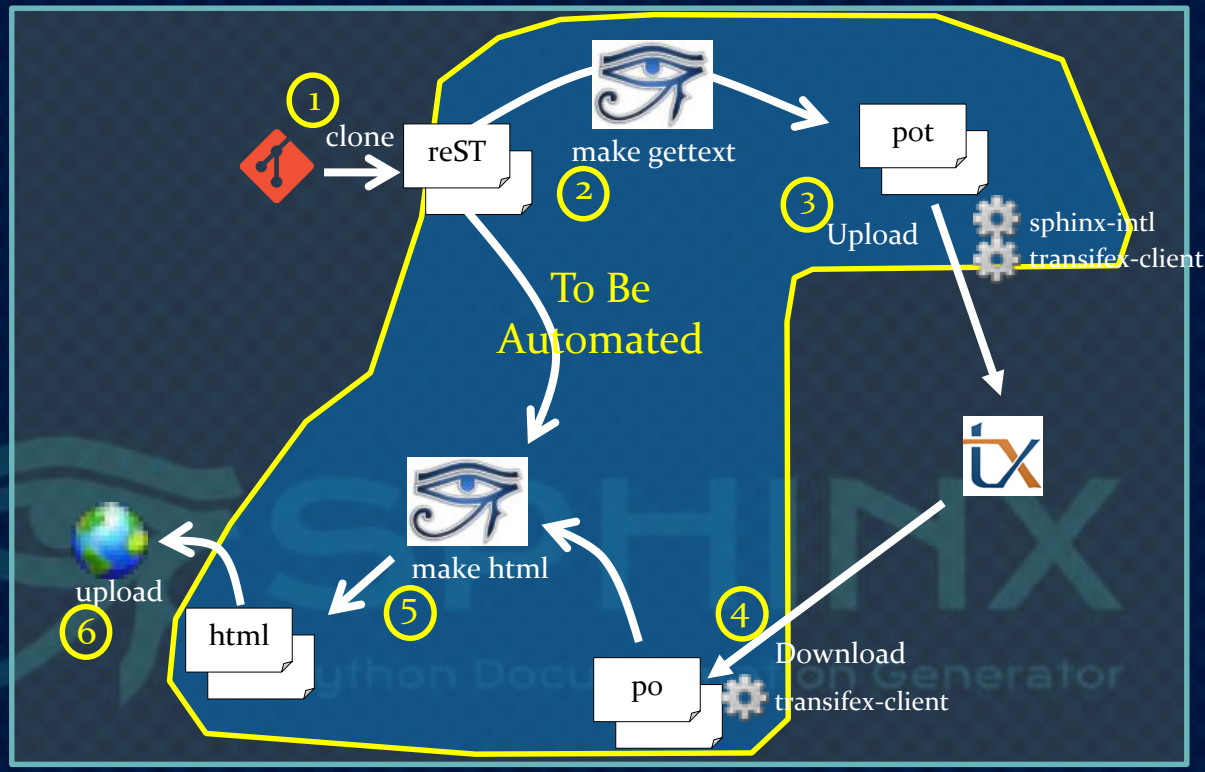


To Be Automated



The procedure for automation

1. clone repository
2. make gettext
3. Upload pot
4. Download po
5. make html
6. Upload html



Shell commands for automation

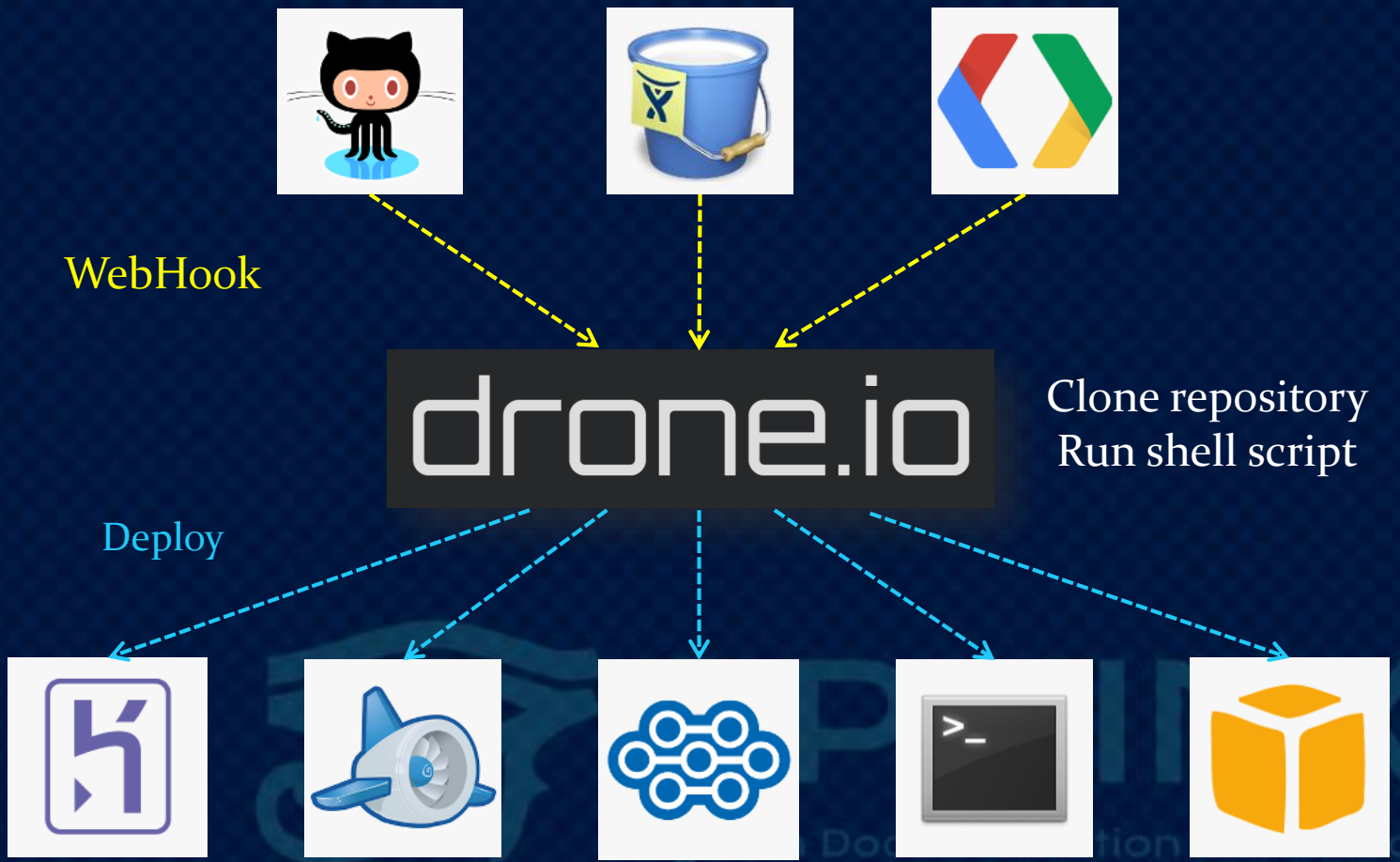
run.sh

```
1.  pip install sphinx sphinx-intl transifex-client
2.  git clone https://github.com/shimizukawa/deepthought.git
3.  cd deepthought/doc
4.  sphinx-intl create-transifexrc        # create ~/.transifexrc
5.  sphinx-intl create-txconfig           # create .tx/config
6.  make gettext
7.  sphinx-intl -p _build/gettext update-txconfig-resources
                                          # update .tx/config
8.  tx push -s                            # push pot files to transifex
9.  tx pull -l ko                         # pull po files from transifex
10. make html SPHINXOPTS="-D language=ko"
```

```
$ export SPHINXINTL_TRANSIFEX_USERNAME=mice
$ export SPHINXINTL_TRANSIFEX_PASSWORD=42
$ export SPHINXINTL_TRANSIFEX_PROJECT_NAME=deepthought-0_7
$ export SPHINXINTL_POT_DIR=_build/gettext
$ run.sh
```

The drone.io

The drone.io is a continuous integration service.



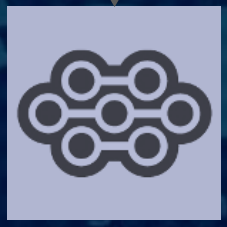
GitHub + drone.io + S3

GitHub

Transifex

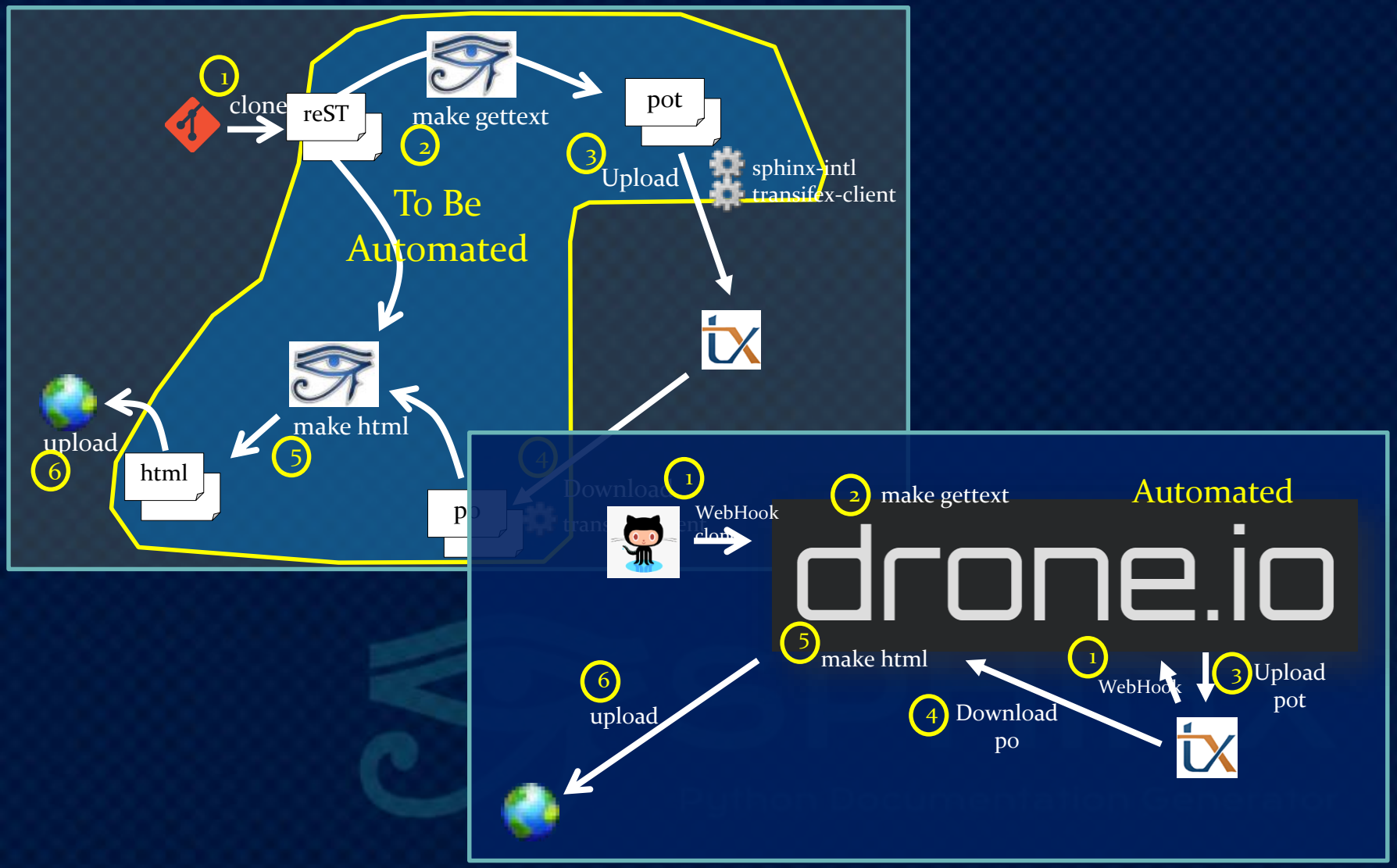
drone.io

1. Clone repository
2. make gettext.
3. Upload pot
4. Download po
5. make html
6. Upload html

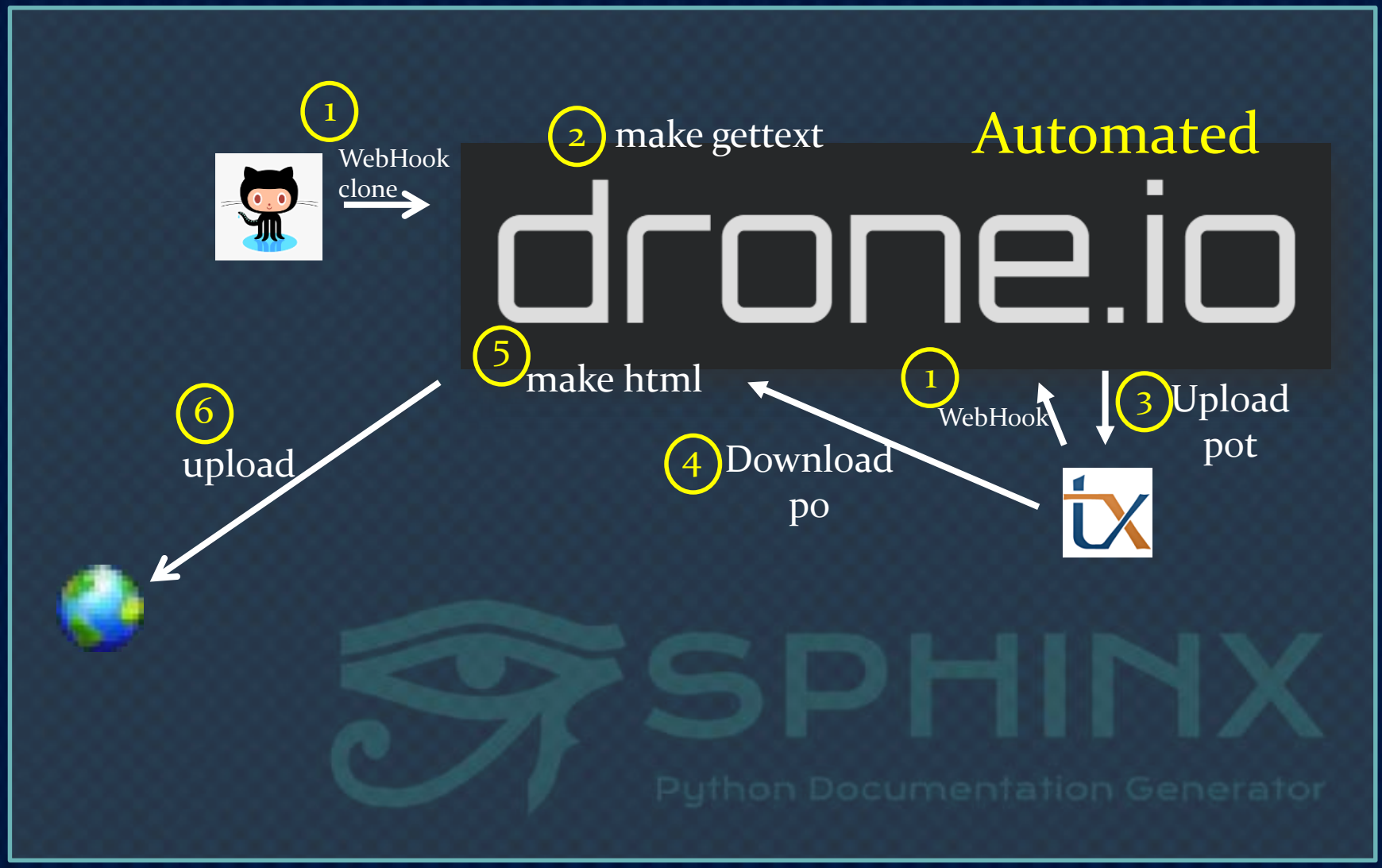


Amazon S3

Be Automated



Automated by drone.io

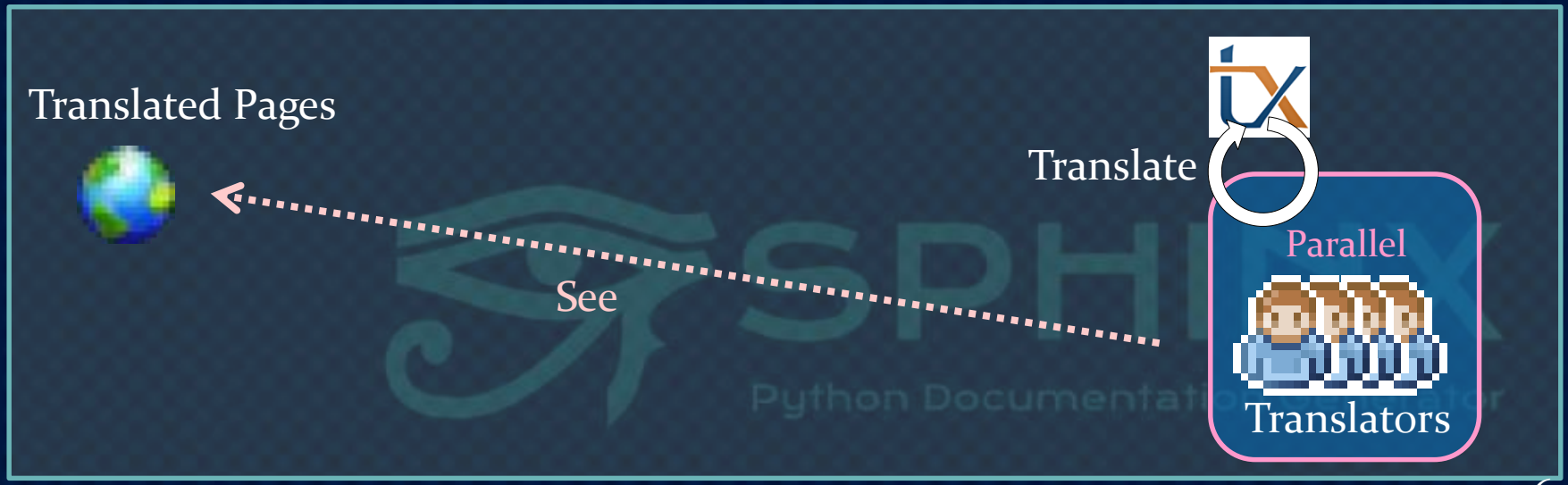


View from doc author

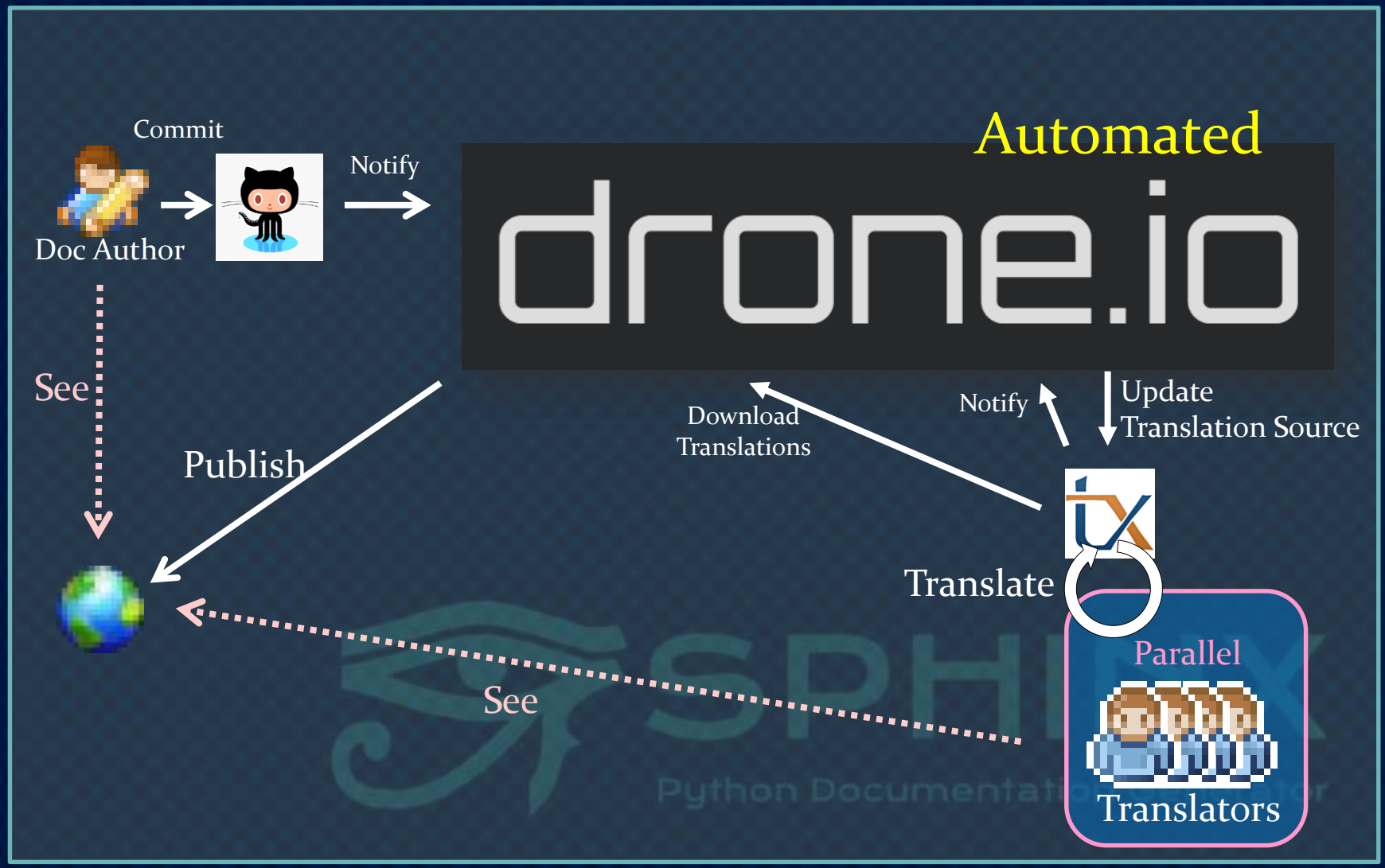
- Doc Author doesn't require annoying procedure.

View from doc translators

- No git
- No file
- No conflict
- Update Automatically
- They can get a translated HTML output w/o hand-build.



The entire automated process



Summary

Tools

1. sphinx - documentation generator
2. docutils - base of sphinx
3. sphinx-intl - support utility for sphinx i18n
4. transifex-client - file transfer tool for Transifex

Services

1. Transifex - translation support service
2. Drone.io - continuous integration service

5. Tips

How to handle URLs

*.rst

```
Sphinx translates a set of reStructuredText_ source files into
various output formats.

.. _reStructuredText: http://docutils.sourceforge.net/rst.html
```

Hyperlink Target

*.po

```
msgid "Sphinx translates a set of reStructuredText_ source files
into various output formats."
msgstr "Sphinx translates a set of “
“`reStructuredText <http://docutils.sphinx-users.jp/web/rst.html>`_ “
“source files into various output formats."
```

- Hyperlink Target doesn't appear in POT files.
- If you want to change URLs into translation version, you can use **Embedded URLs** notation.

How to change language w/o rewriting conf.py

```
$ make html SPHINXOPTS="-D language=ko"
```

- You can use SPHINXOPTS make option
- It also works with other targets not only 'html'.
  - linkcheck target is useful if your translation contains URLs. It will find typos in translation text.

```
$ make linkcheck SPHINXOPTS="-D language=ko"
```

You can control pot file resolution

- Use gettext_compat

doc/conf.py

```
language = 'ko'
locale_dirs = ['locale']
gettext_compat = False  # generate .pot for each .rst
```

- By default 'gettext_compat = True'.
- reST files under sub directories are collected into single POT file.

Translatable sphinx html templete

- Sphinx HTML template is **Jinja2.**
- You can use "{% trans %}" template tag.

_templates/index.html

```
<p>
  <em>{%trans%}What users say:{%endtrans%}</em>
</p>
<p>
  {%trans%}&#8220;Cheers for a great tool that actually
makes programmers <b>want</b>
to write documentation!&#8220;{%endtrans%}
</p>
```

- "make gettext" exports "**trans**" contents into sphinx.pot file files.

Example projects

- Sphinx-1.4 doc for "ja" translation
  https://drone.io/bitbucket.org/shimizukawa/sphinx-doc15/admin

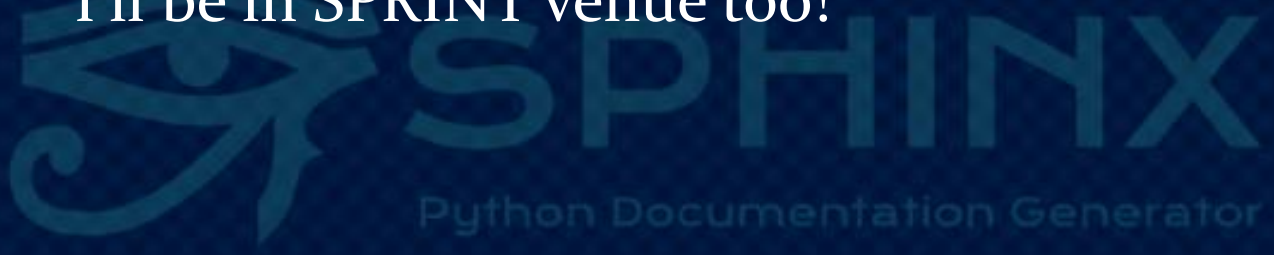
- Docutils doc for "ja" translation
  https://drone.io/bitbucket.org/sphinxjp/docutils-translation/admin

Questions?

@shimizukawa

Grab me after the session :)

I'll be in SPRINT venue too!



Thanks :)