Learn what and when to learn: skills for bioinformatics as open science

Tazro Ohta

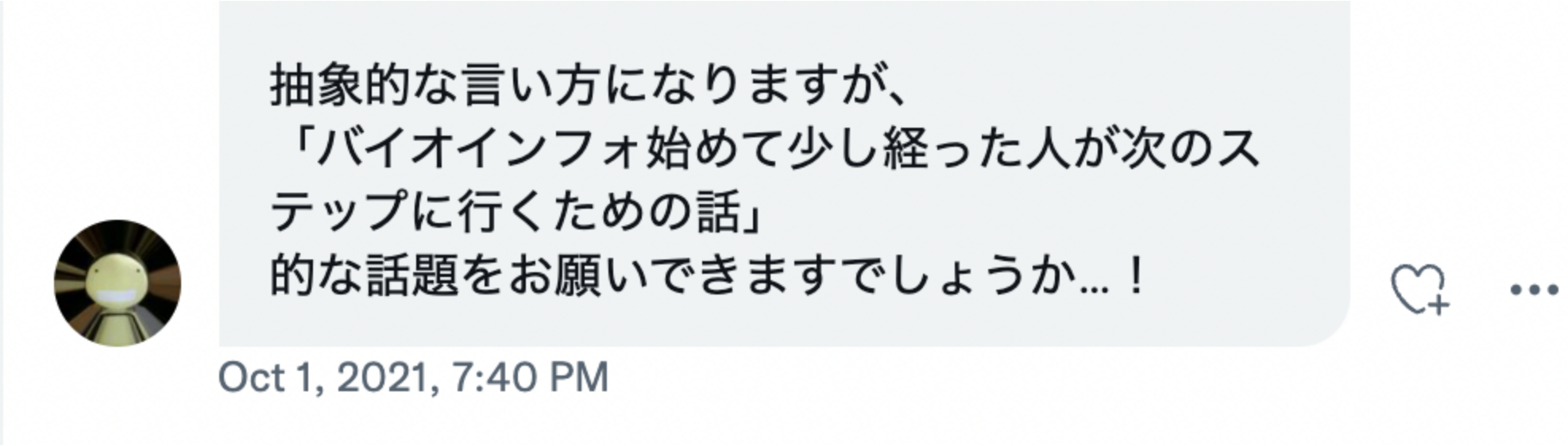
Database Center for Life Science, Research Organization of Information and Systems

Prepared for: Tsukuba Bioinfo Assmebly, 2021-11-04

Who?

- Tazro Ohta, Ph.D. @inutano, Project Assistant Professor in DBCLS
- Works and Collaborations
  - Applications for **public high-throughput sequencing data**
    - ChIP-Atlas https://chip-atlas.org/
  - **Scalable computing** for genomics
    - Sapporo, Workflow Execution Service https://github.com/sapporo-wes
  - **Knowledge graph and ontologies** for Bio-databases
    - TogoDX https://togodx.dbcls.jp/human
    - TogoID https://togoid.dbcls.jp
  - **Community**
    - Pitagora Network https://pitagora-network.org/
    - BioHackathon http://biohackathon.org/

Thank you for inviting me to the TBA community!



Then my talk is for "Those who started Bioinformatics and want to step forward" :)

Agenda

- Introduction: Why we learn informatics skills for Biology
- What and When to learn
- The Open Science skills
    - Documentation
    - Testing
    - Packaging
    - Shipping

Why we learn informatics skills for Biology

Fast and Faultless

- Learning new skills is to move fast and faultless
  - "Fast" for your career
  - "Faultless" for science

Learning: cost and benefit

- New techs will give you a benefit, but you must spend time to learn
- In every moment you need to make a decision:
  - Do with the skills you have
  - Learn new skills and optimize your work

Goal?

- **Make a scientific contribution in a limited time frame**
- Precise estimation is the key
  - How fast can you do with your current skill?
  - How long does it take to learn the new skill?
  - How fast can you do if you got a new skill?

What and When to learn

Bioinformatics as Open Science

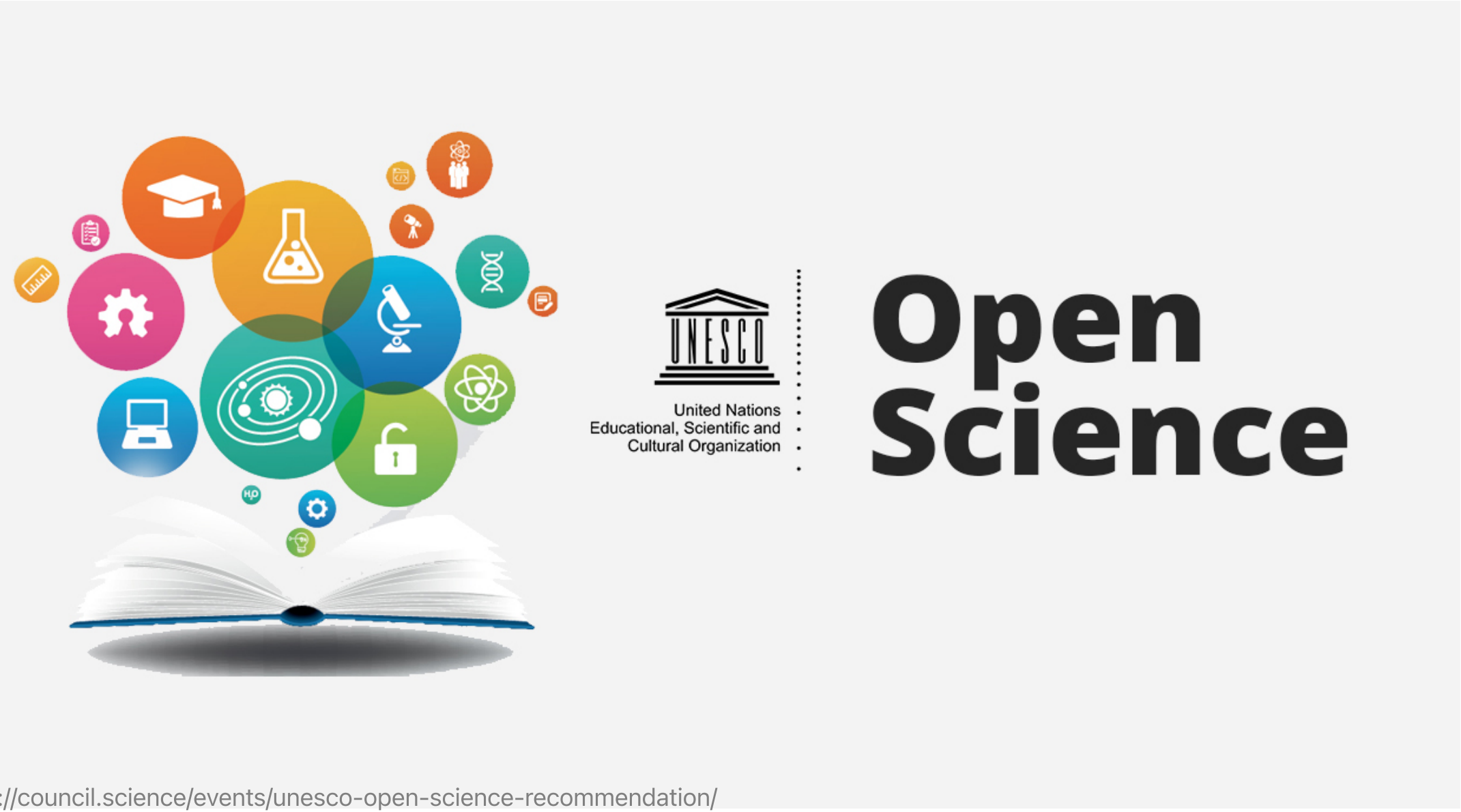
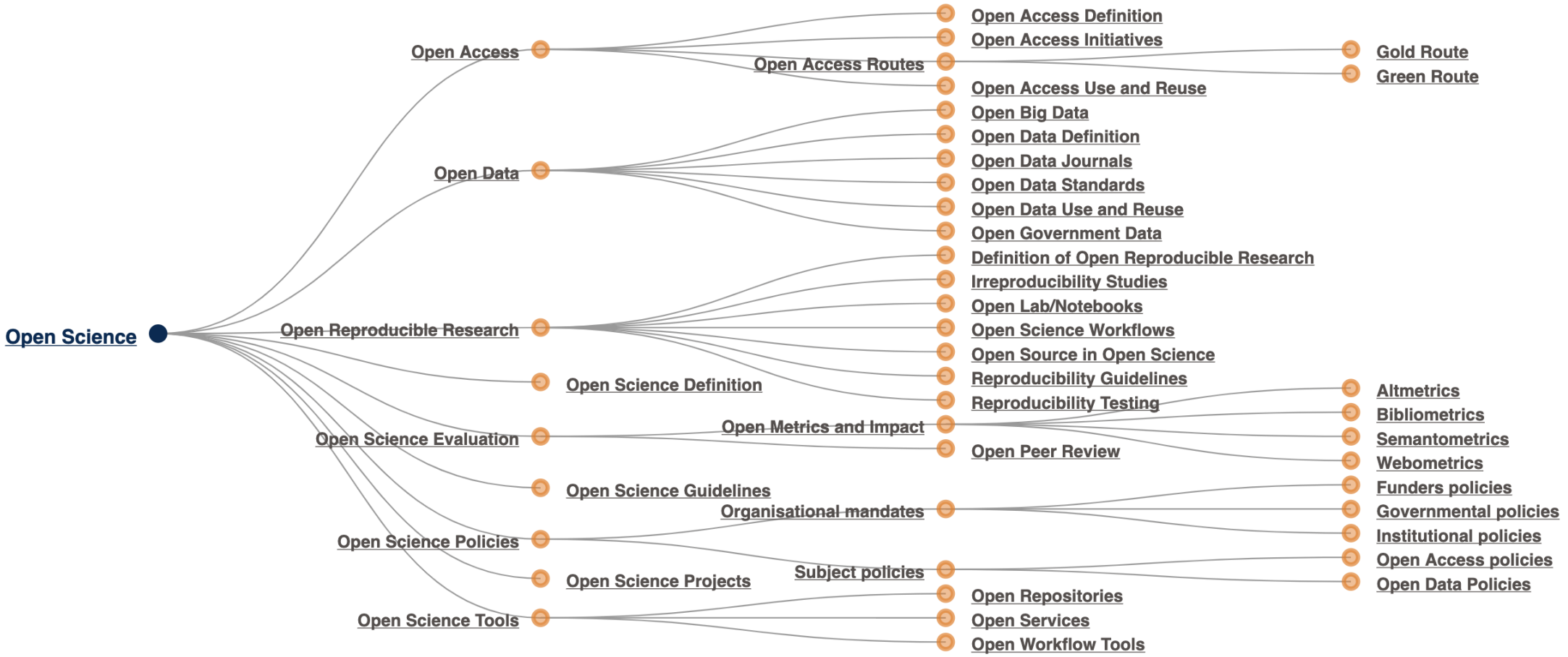


Image: https://council.science/events/unesco-open-science-recommendation/

> Open Science is the practice of science in such a way that others can collaborate and contribute, where research data, lab notes and other research processes are freely available, under terms that enable reuse, redistribution and reproduction of the research and its underlying data and methods.

https://www.fosteropenscience.eu/foster-taxonomy/open-science-definition



https://www.fosteropenscience.eu/foster-taxonomy/open-science-definition

Open Science skills to make your product:

- **Sharable**: for the provenance of your work
- **Repeatable**: for your future work to reuse the materials
- **Reproducible**: for your team and people who want to utilize your work

Not only for "the good thing"!

When is more important than *What*

- Learning is like investment
- Is this the right time to learn this?
  - Your career
  - Your project progress
  - Maturity of the technology
  - Maturity of community

Now or Later?

- Ask people or search about maturity
- Consider **BEFORE** you start your project
  - Every new techonology looks pretty during the project
- Learn when you need it
  - Using is learning
  - "It's trending, everyone is talking about it, then I might need to learn.." => **NO**
- If you'd love to learn, nobody won't stop you from going on vacation to read the books

Skills for Open Science

Skill: Documentation

Describe your product, always

- Write first, then implement
- Write during the implementation
- Write after the implementation
- Write
- Write!!

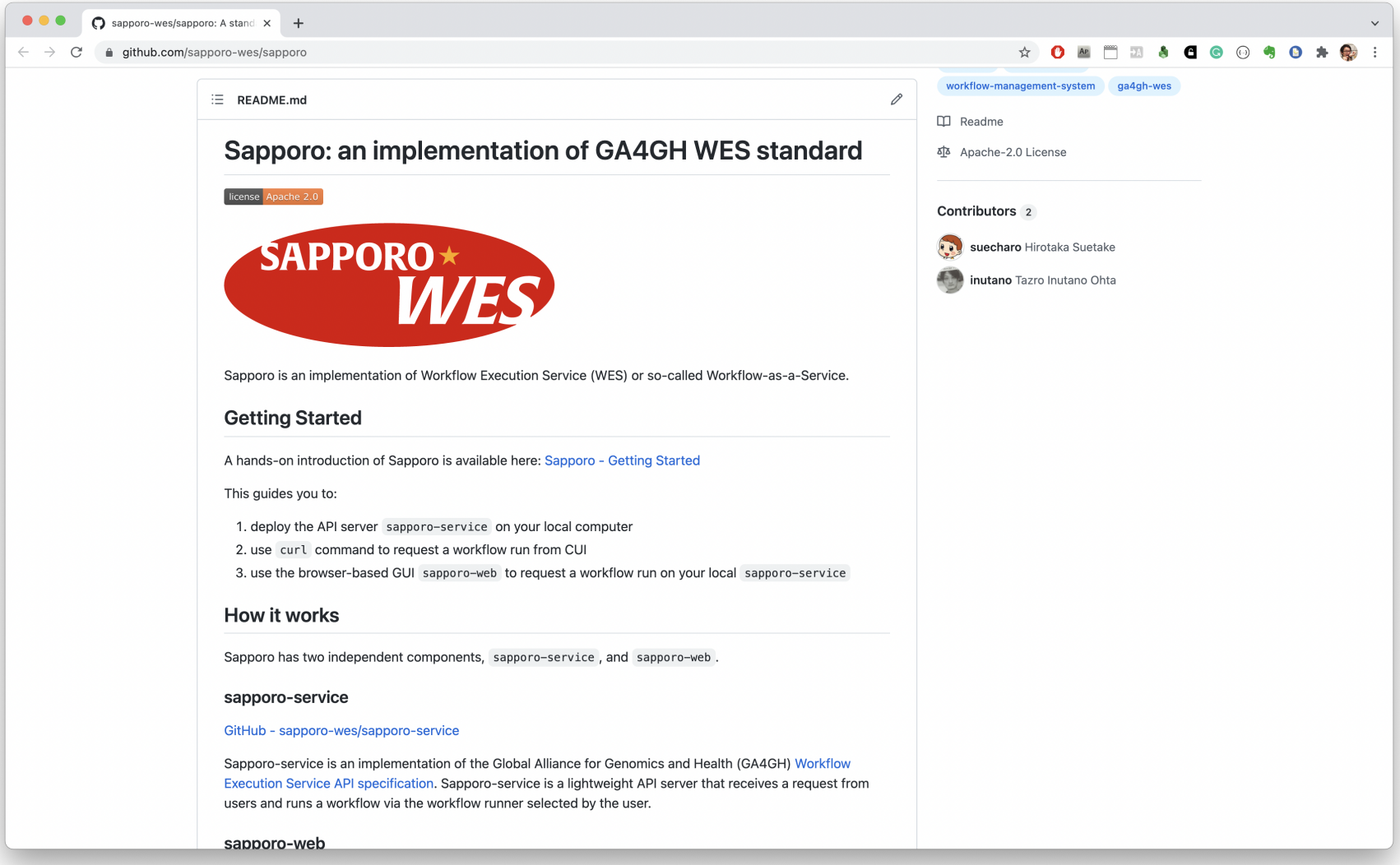
Why does documentation matter?

- You will forget (for sure)
- Save your time to answer the questions from colleagues/collaborators
- Help you to manage your project status
- Help you to **write a paper**
    - Good documentation is a start to writing a good paper

Where to start?

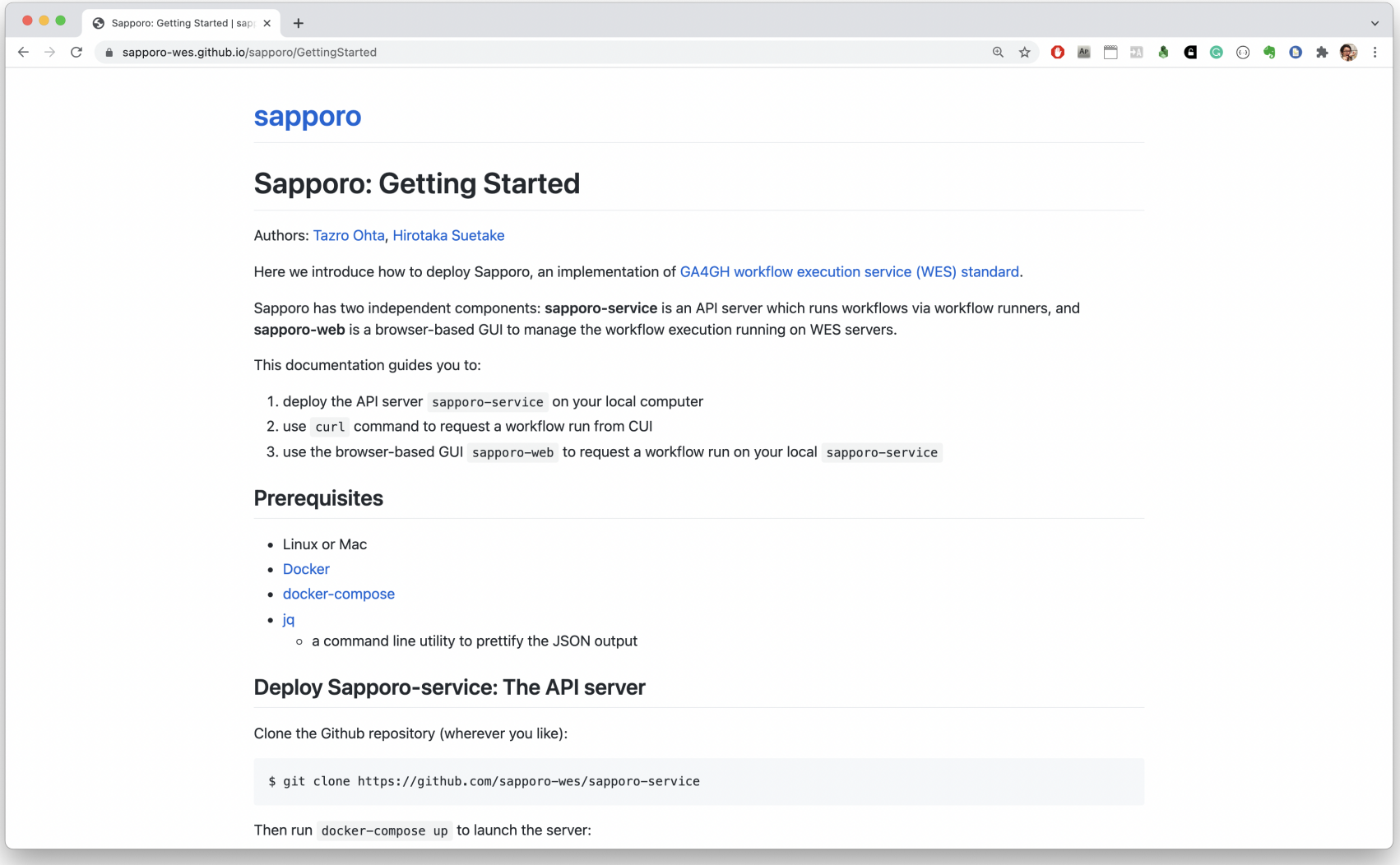
- Lv. 1: Write comments inside your program
- Lv. 2: Write `README` to describe **what it does and how it works**
- Lv. 3: Write markdown `README.md` and push to GitHub pages
- Lv. 4: Write markdown and publish with GitHub pages

Example: Utilize GitHub repository



https://github.com/sapporo-wes/sapporo

Example: Utilize GitHub Pages



https://sapporo-wes.github.io/sapporo/GettingStarted

When to learn

- Start right now
- Don't try to be perfect but always try to be better

Skill: Testing

Testing for Bioinformatics software

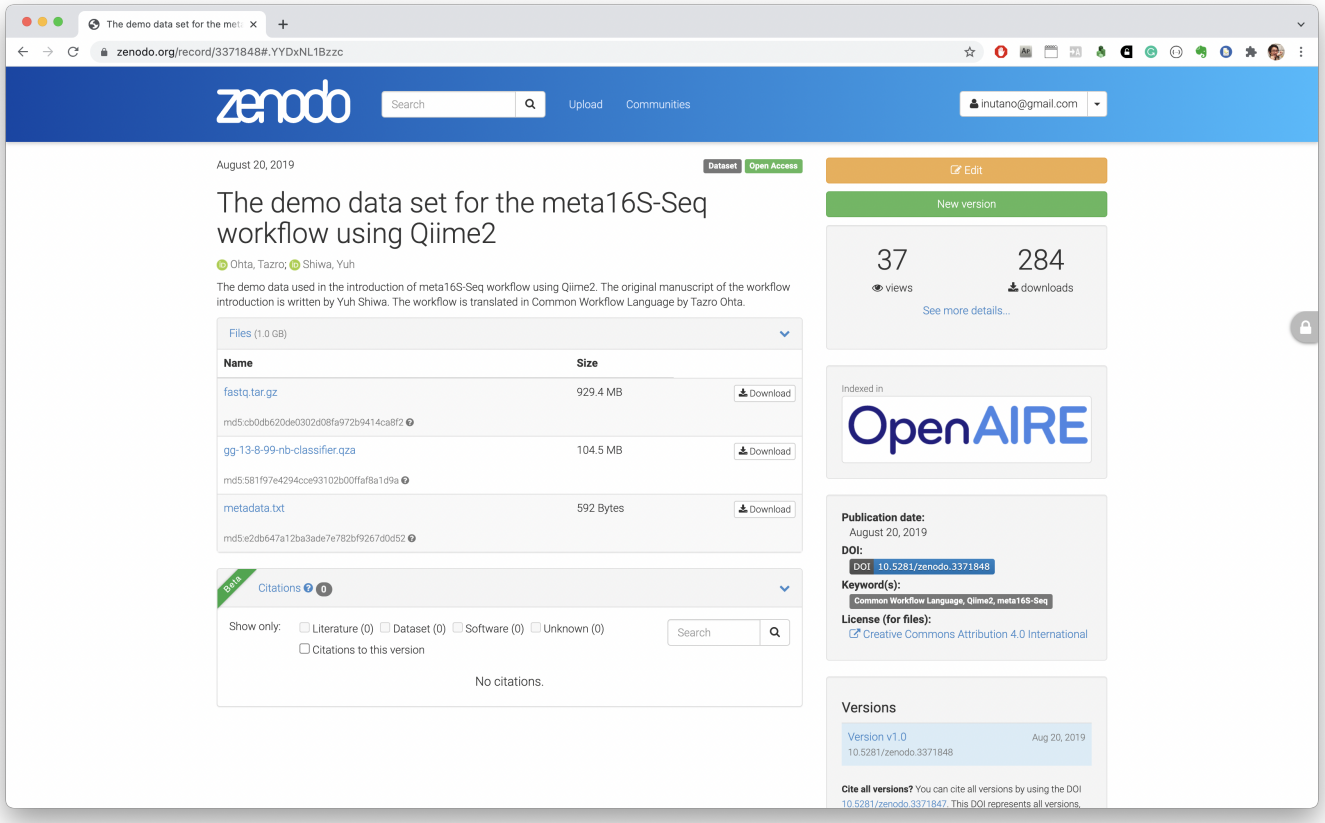
- What people talk about when people talk about testing: **Unit test**
  - Good practice, but it's about **reliability** of code
  - People will change input: need to test **robustness and limitation**
- Testing for scientific software should show:
  - Correct setup: prerequisites and dependencies
  - Expected outcome: the output when the input was ok
  - Known errors: output from incorrect inputs

Where to start?

- Lv. 1: Describe the correct output in the document
- Lv. 2: Share example input data on public storage (e.g. Zenodo)
- Lv. 3: Ship your software with correct input and output
- Lv. 4: Use the testing framework to make testing executable
- Lv. 5: Use CI framework (e.g., GitHub actions) to automate testing

Publish example data with Zenodo

- Max 50GB per dataset with DOI issued



When to learn

- Consider when you feel your project is coming to an end
  - Try to prepare the test data even after the implementation
- Providing resources for testing may give a good impression to the journal reviewers as well

Skill: Packaging

You (will) know the truth of bioinformatics

- You can't run your software six months later.

Why packaging?

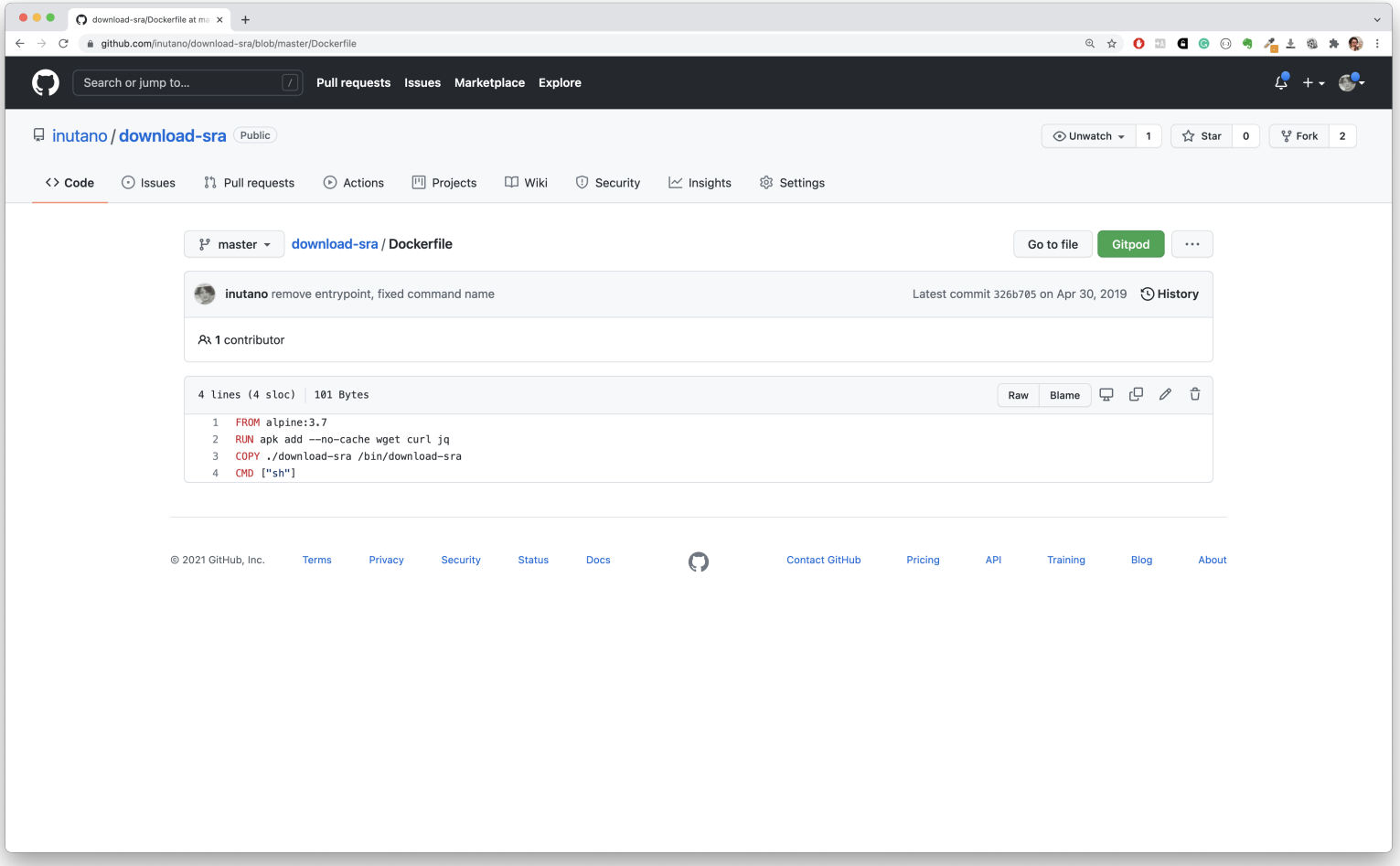
- If things won't change, it won't happen, but things change.
  - **Dependencies (libraries)**
  - **Runtime**
  - **OS**
  - Hardware settings
  - Network settings
  - Remote references

You can package your software with the dependencies, making things much better.

Where to start?

- Level 1. Describe your environment with library versions
- Level 2. Create an installation script to install dependencies and build your product
- Level 3. Use a packaging framework, e.g. debian med, brewsci, bioconda, etc.
- Level 4. Use containers to pack the whole dependencies
- Level 5. Write workflow to pack the combination of containers

Example: packaging in Docker container



https://github.com/inutano/download-sra/blob/master/Dockerfile

Example: run a Docker container

Basic usage:

```
$ download-sra SRR000001 NCBI
```

With Docker:

```
$ docker run \
  --rm \
  -it \
  -v $(pwd):/work \
  -w /work \
  quay.io/inutano/download-sra:0.1.2 \
  download-sra "DRR000001" "NCBI"
```

Tips: Writing Dockerfile

- All-in-one or multiple containers
  - all-in-one: easy to create, but hard to update
  - multiple containers: cost to maintain, but safe for dependency problems
- Tips: Publish both Dockerfile and Image
  - *Container build reproducibility*

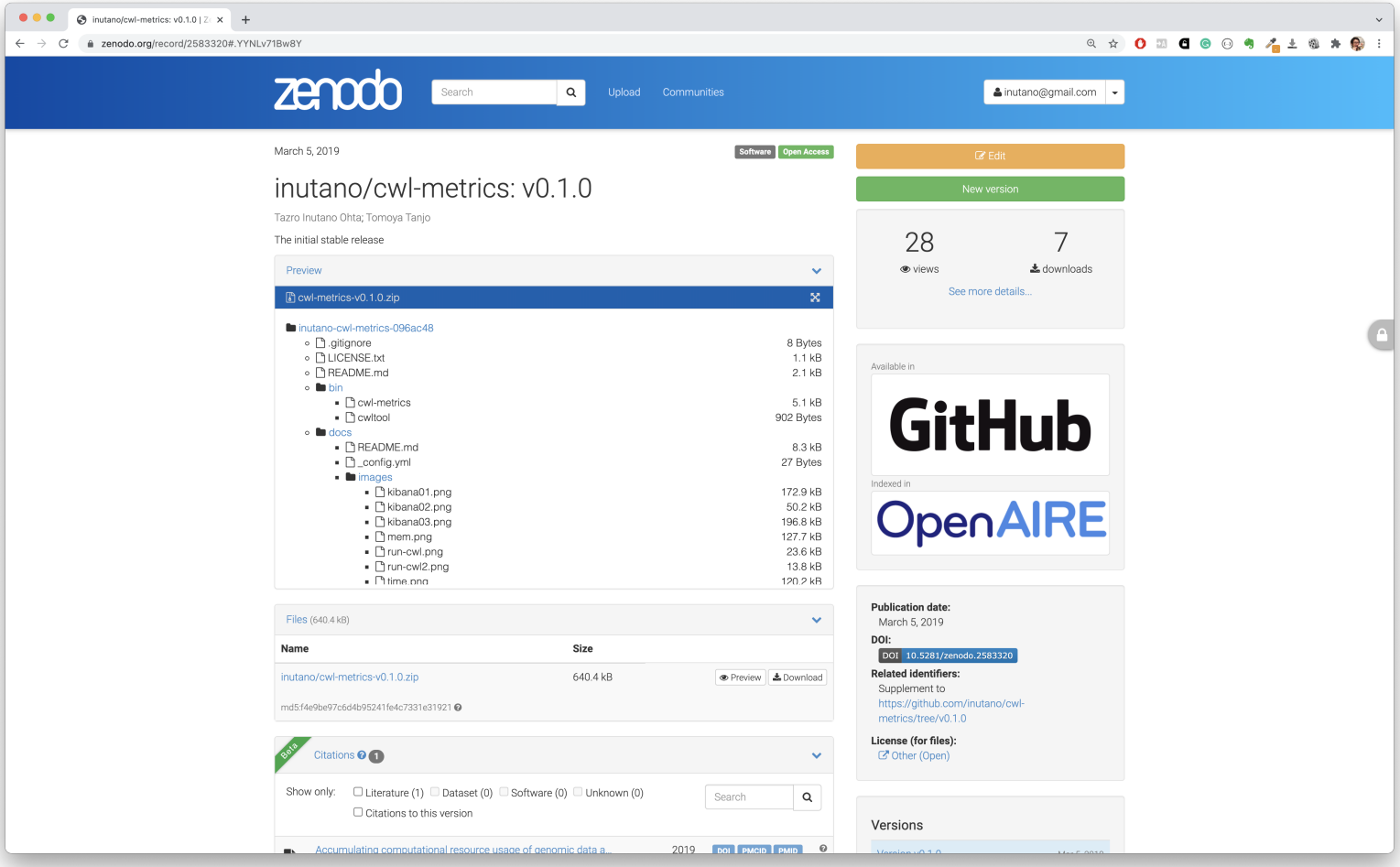
When to learn

- A technology still changing rapidly
  - Will need time to learn the background mechanisms and keep up with the latest trending
- Container is one of the basic concepts of cloud infrastructure
  - Abstraction of computing is an important topic to learn

What to learn: Shipping

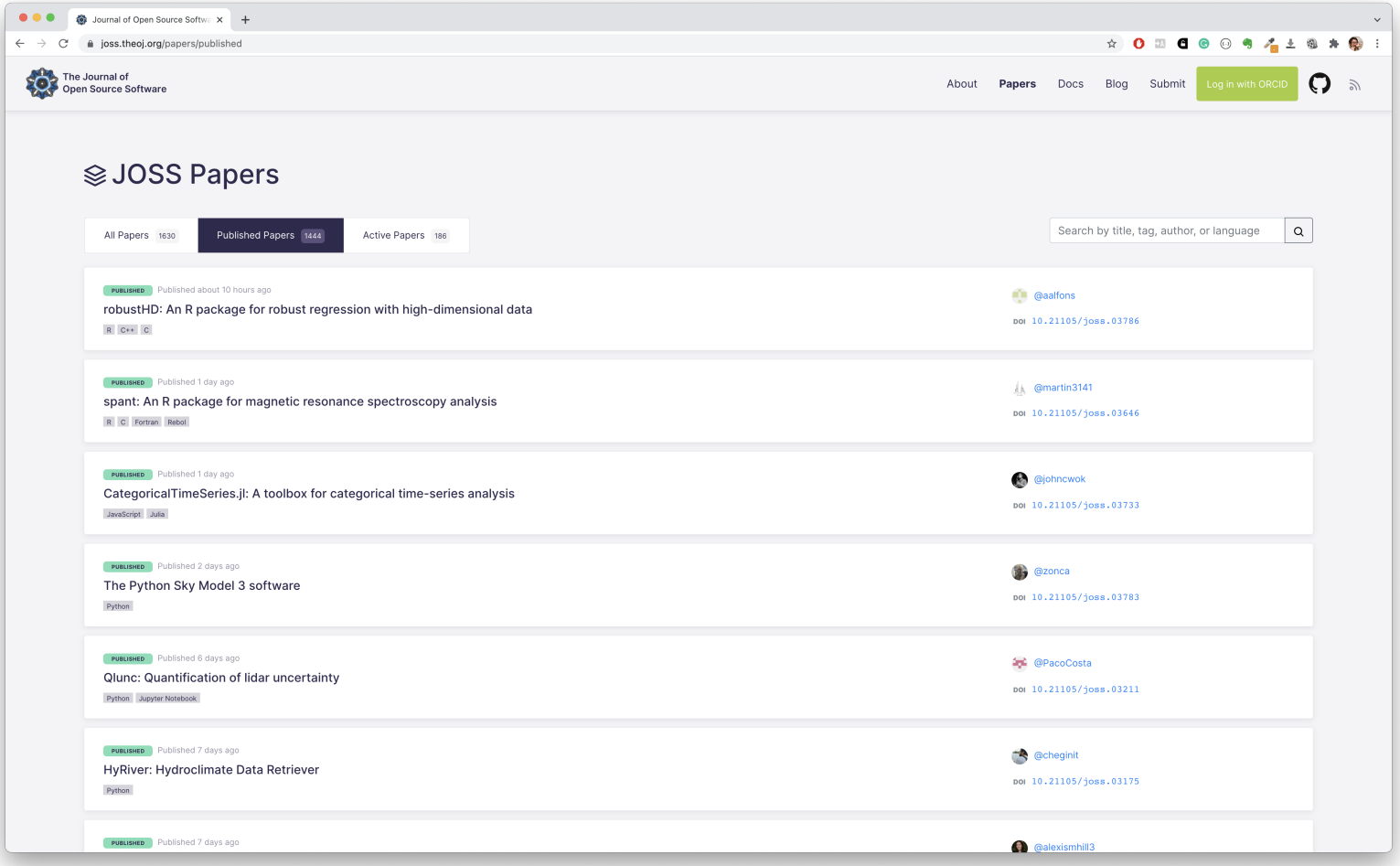
- Publish, then think about who cares
- GitHub driven open-science
  - Tag your software/document version
  - Use issues for questions from users
  - Accept PRs from colleagues/collaborators
  - Cboose a License
  - Assign DOI to the code with Zenodo
  - Publish your repos via JOSS

Example: Assign DOI on GitHub repo



https://zenodo.org/record/2583320#.YYNLv71Bw8Y

Example: JOSS articles



https://joss.theoj.org/papers/published

When to learn

- Shipping software should be a part of a publication
  - If you are writing a paper, do it now
  - If you are writing software, remember to do it later

And more and more skills; Keep learning!

Summary: Get into the community

- **TBA is amazing**
- Community is a learning opportunity
- Working with people is "How" to learn
  - One-day/One-week Hackathon
  - Talk to people
  - Collaborate with people
- Survive. You are not alone