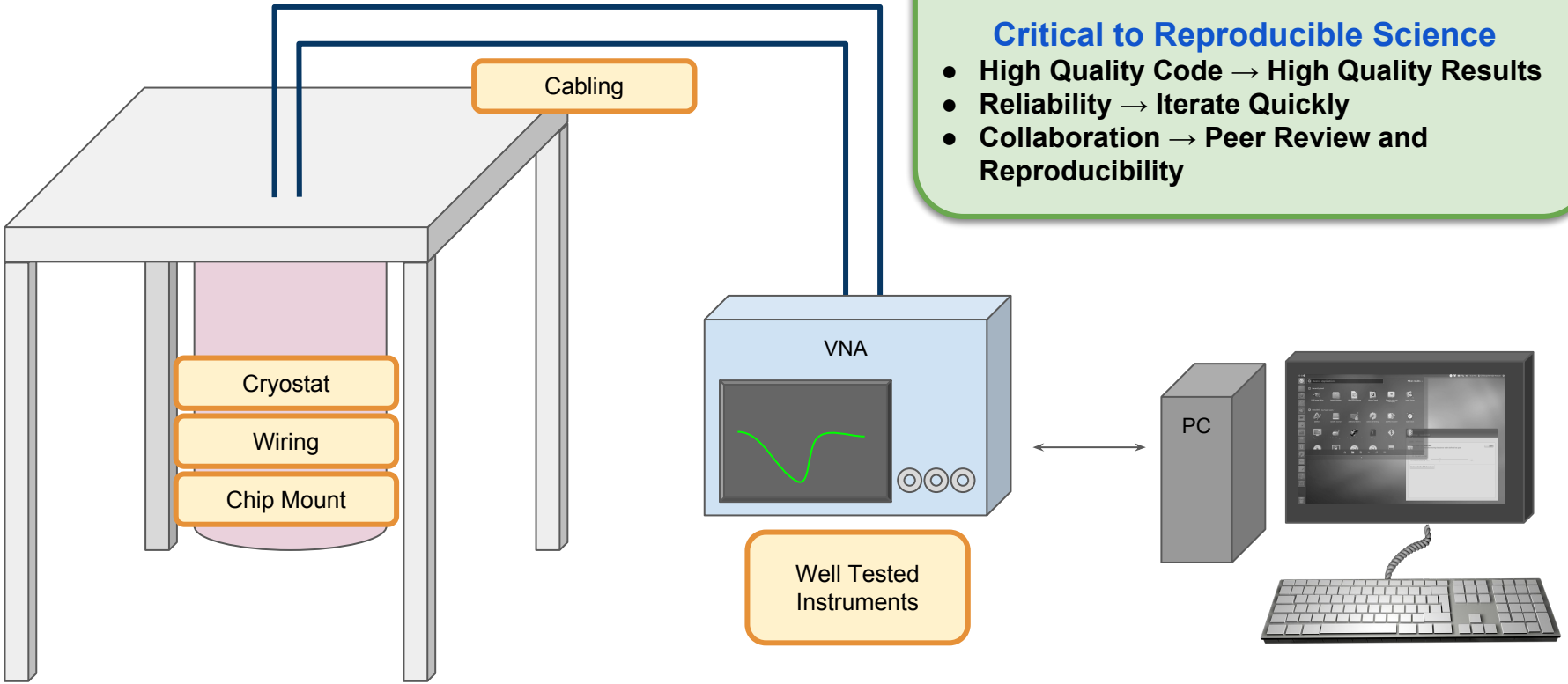
Google AI Quantum

Building Reproducible Experiment Software

Kunal Arya - Google AI Quantum team

February 8, 2019

Experiment Setup



Software

Critical to Reproducible Science

- **High Quality Code → High Quality Results**
- **Reliability → Iterate Quickly**
- **Collaboration → Peer Review and Reproducibility**

Google AI Quantum

Software for Reproducible Science

- Software is a complex challenge
- *Collaborating effectively* on software is even harder!
  - Companies spend millions on collaboration tools

	Lines of Code	**Engineers**
Google	2 Billion[1]	~60,000
Facebook	62 Million[2]	~12,500[3]
Linux Kernel	25 Million[4]	~17,800[4]

- Open Source practices make this easier
- Foster a community to own and drive development

1. https://cacm.acm.org/magazines/2016/7/204032-why-google-stores-billions-of-lines-of-code-in-a-single-repository/fulltext
2. https://www.visualcapitalist.com/millions-lines-of-code/
3. https://www.recode.net/2017/8/4/16090758/facebook-google-profit-per-employee-comparison-chart
4. https://phoronix.com/misc/linux-2017/index.html

Why Open Source Collaboration?

- Lower barrier of entry
- Avoid reinventing the wheel
- Solved problem in big projects
  - Apply them to scientific computing

Open Source Experiments

- A library of instrument interfaces
  - Different labs can evaluate different VNAs
- Spend less time writing instrument code, more time generating results
- Build a community around the code base
  - Place to discuss new physics

3 Components of Reproducible Software

Version Control

- Canonical Code Base
- History of Changes, Author, Date

Collaboration Tools

- Documentation
- Issue Tracking
- Code Review
  (“Peer Review” for Code)

Automated Tests

- Code That Tests Code
- Resilience
- Confidence to Make Changes

Version Control

- “Experiment_Code_2019.zip”

Version Control

- “Experiment_Code_2019_01b.zip”

Version Control

- “Experiment_Code_2019_01b_final.zip”
- Ready to submit March Meeting paper!

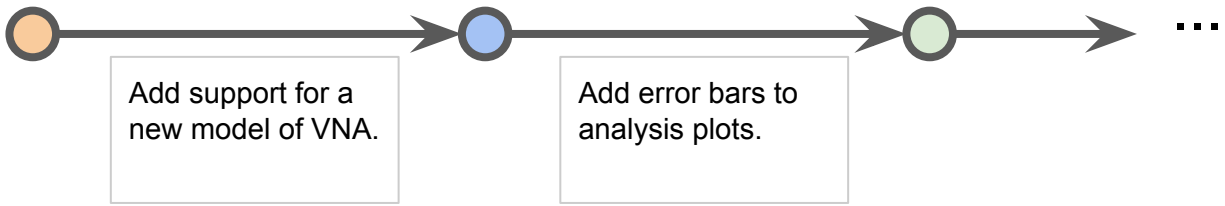
Version Control

- “Experiment_Code_2019_01b_final_final2.zip”

- Tracking history here is cumbersome!
  - How do you dig up that one analysis file you deleted?

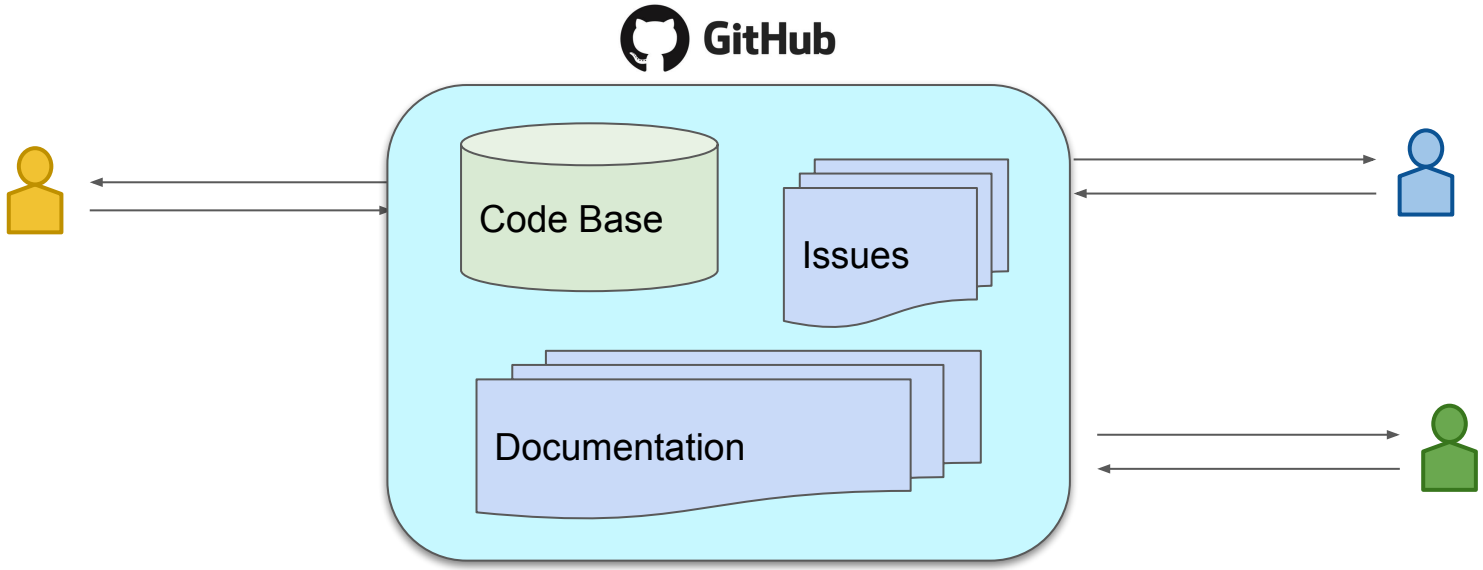
Version Control (Git)

- System for tracking changes in code
- Each change is a “commit”:
  - Snapshot of the whole code base assigned a unique ID.
  - Author and message
- Tooling to navigate history and compare commits



Collaboration Tools (Github)

- Documentation
- Issue Tracking
- Code Review (“Peer Review” for Code)



Automated Tests (Travis CI)

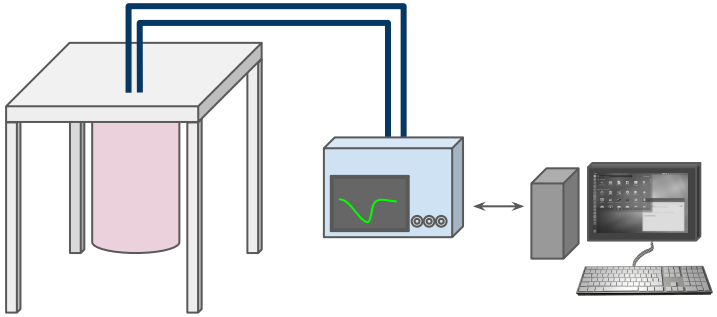
- Code That Tests Code
  - Example:

```
data = generate_lorentzian(peak_GHz=5.5)
results = analyze_data(data)
assert math.isclose(results.peak_GHz, 5.5)
```

- Resilience
- Confidence to Make Changes

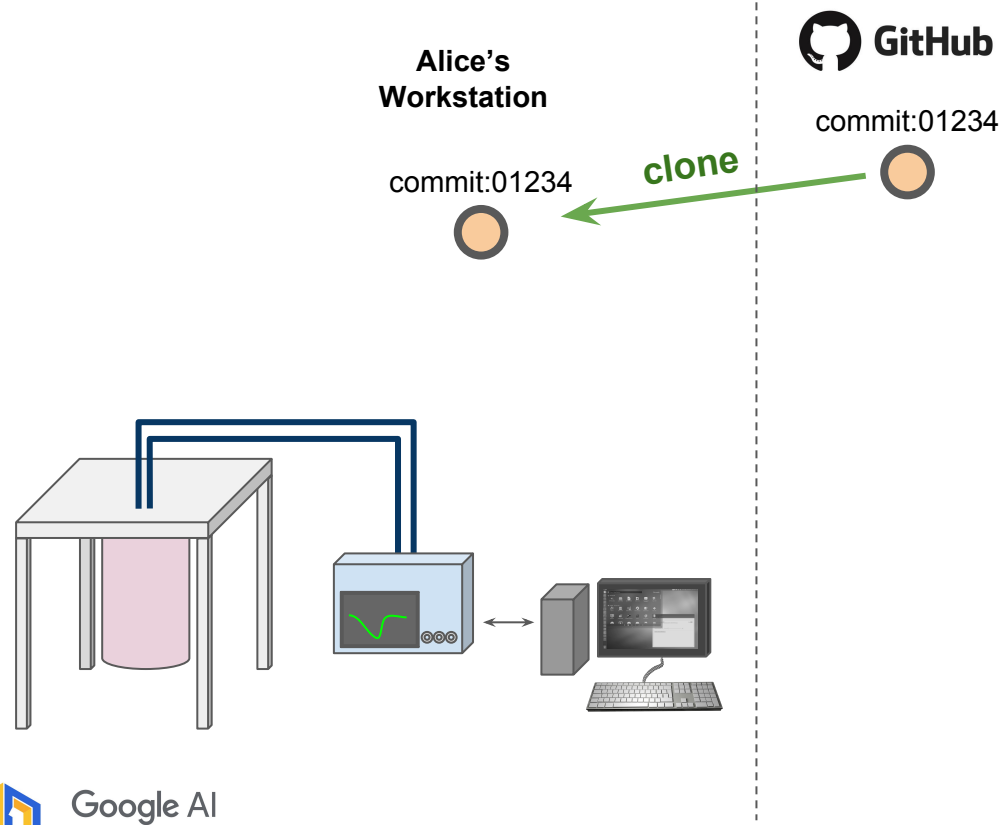
Git & Github Workflow Example

Scenario: Alice wants to take data on a new device.



Google AI Quantum

She first clones the code from Github onto her workstation.

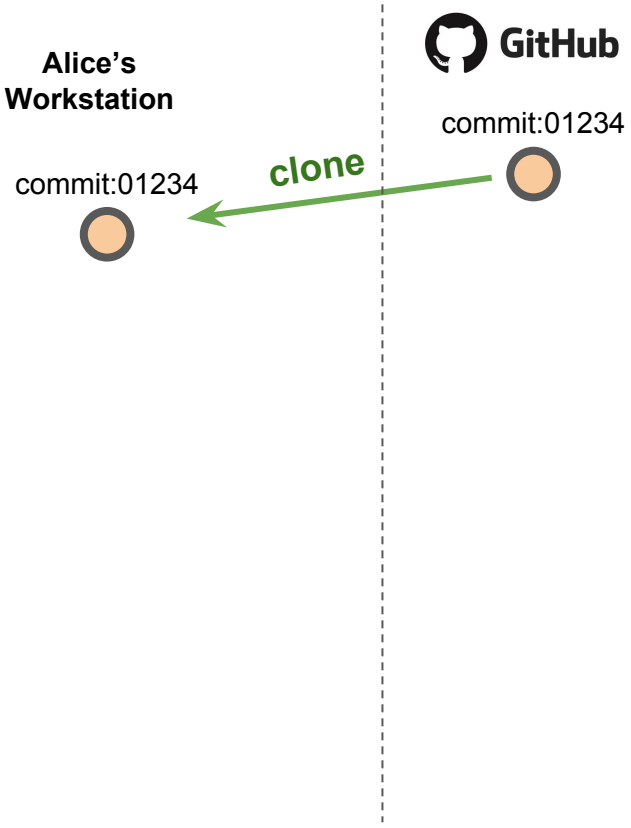


Terminal

```
> git clone http://.../measurement.git
```

Google AI Quantum

She runs the code and discovers a bug in the VNA interface code.

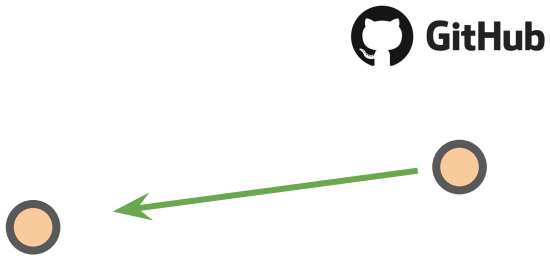


Terminal

```
> git clone http://.../measurement.git

> python3 experiments/run.py
...
IndexError: list index out of range
```

Google AI Quantum

She finds and fixes the issue.



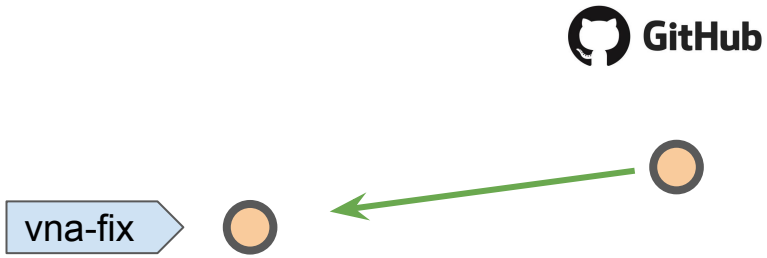

```
# analyze.py
```

```
# agilent_vna.py
...
freq = query.split("/")
freq = query.split(";")
```

Terminal

```
> git clone http://.../measurement.git

> python3 experiments/run.py
...
IndexError: list index out of range

> edit cryores/instruments/agilent_vna.py

> python3 experiments/run.py
Peak frequency: 6.753231 GHz
```

She creates a new branch called “vna_fix”.




```
# analyze.py
```

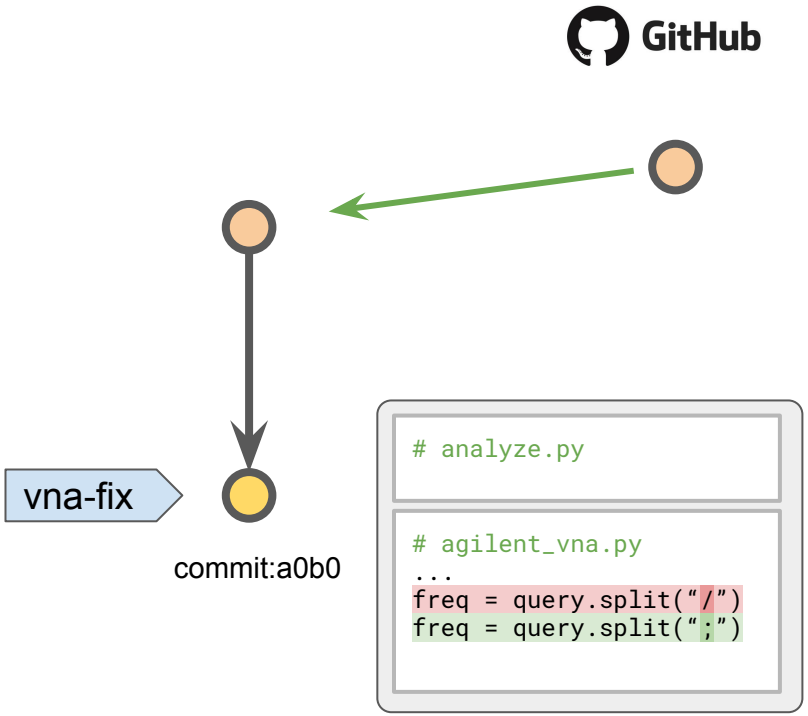
```
# agilent_vna.py
...
freq = query.split(“/”)
freq = query.split(“;”)
```

Terminal

```
> git clone http://.../measurement.git

> python3 experiments/run.py
...
IndexError: list index out of range

> edit cryores/instruments/agilent_vna.py

> python3 experiments/run.py
Peak frequency: 6.753231 GHz

> git checkout -b “vna-fix”
```

Google AI Quantum

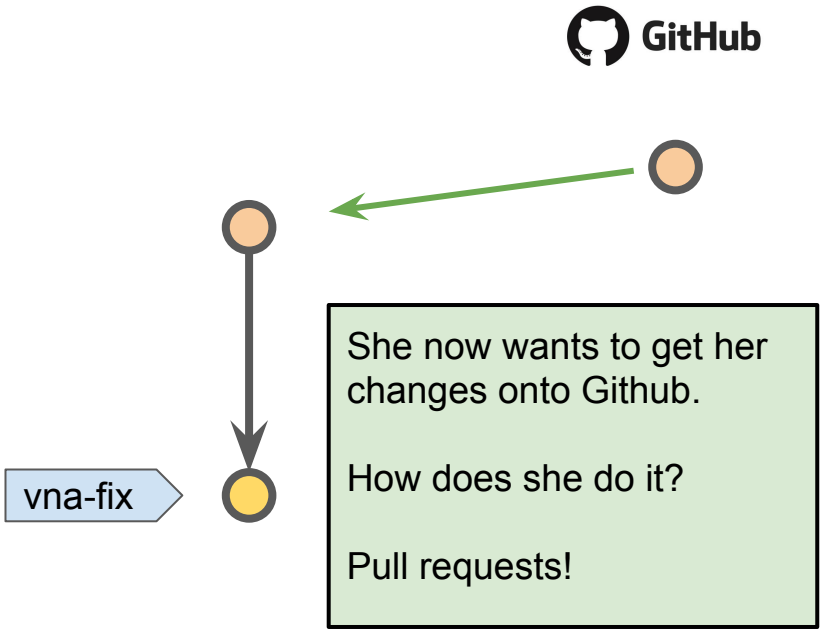
...and adds a commit with her bug fix.



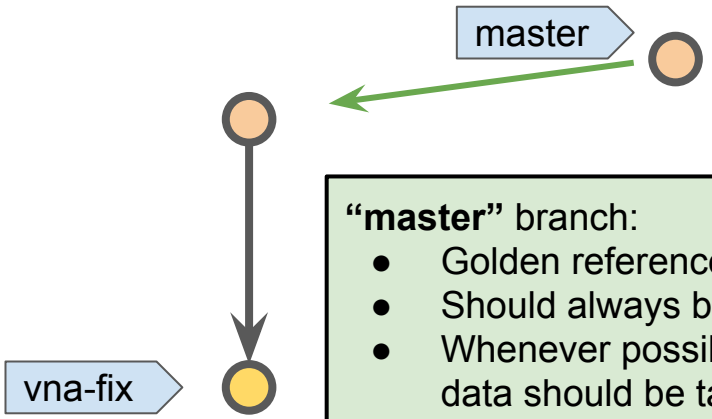
Terminal

```
> git clone http://.../measurement.git

> python3 experiments/run.py
...
IndexError: list index out of range

> edit cryores/instruments/agilent_vna.py

> python3 experiments/run.py
Peak frequency: 6.753231 GHz

> git checkout -b "vna-fix"

> git add cryores/instruments/agilent_vna.py

> git commit -m "Fix frequency message parsing"
```

Google AI Quantum


GitHub
vna-fix
She now wants to get her changes onto Github.
How does she do it?
Pull requests!

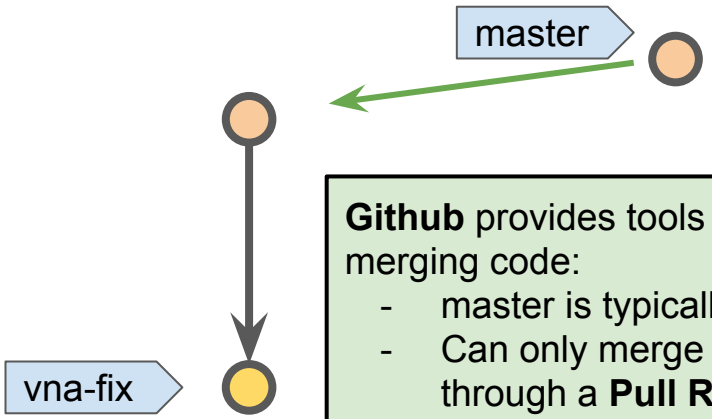
GitHub



“master” branch:

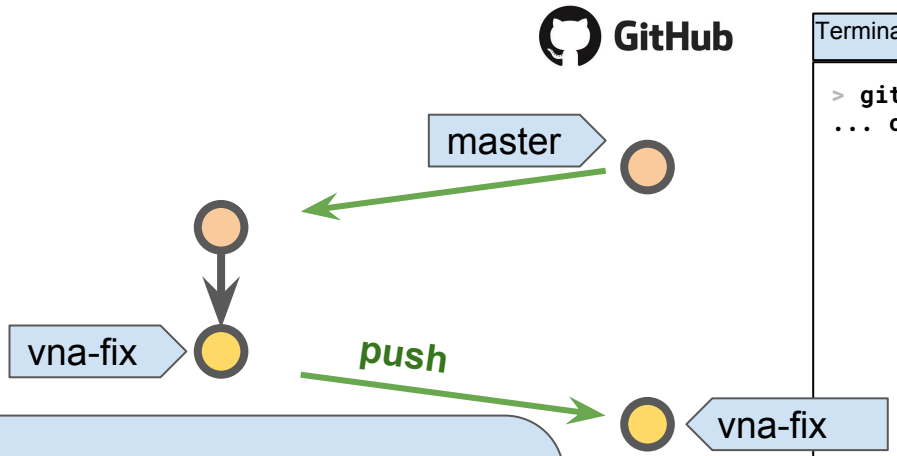
- Golden reference.
- Should always be stable.
- Whenever possible, published data should be taken from code on master.

GitHub



Github provides tools to formalize merging code:

- master is typically protected
- Can only merge into master through a **Pull Request**



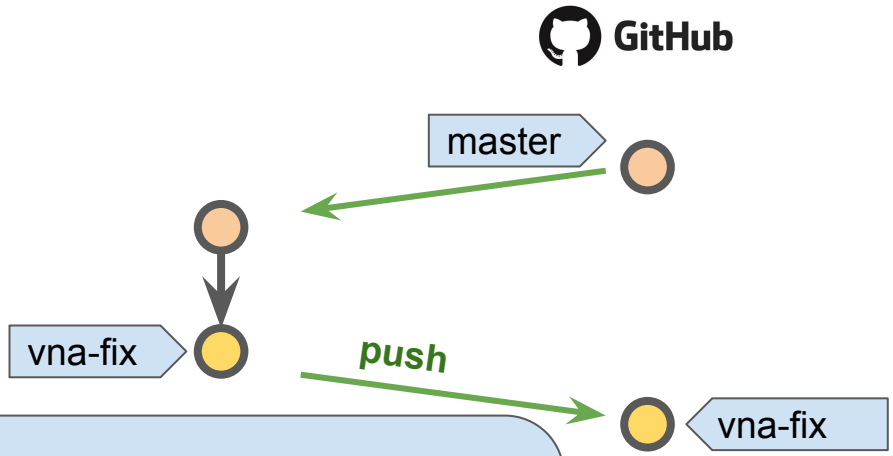
Terminal

```
> git push
... created new branch ...
```

On Github.com:

1. Alice **pushes** “vna-fix”.

Google AI Quantum



GitHub **Pull Request #195**

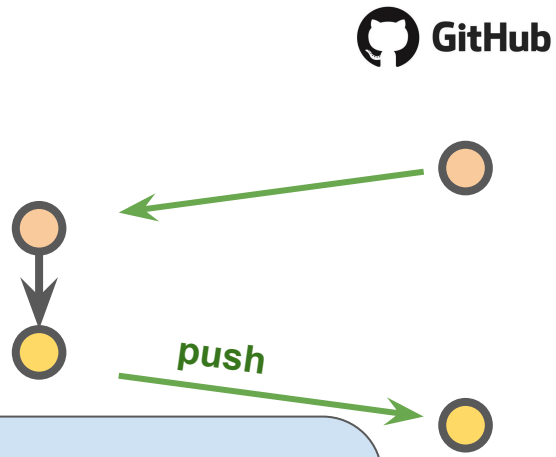
Alice wants to merge “vna-fix” into “master”

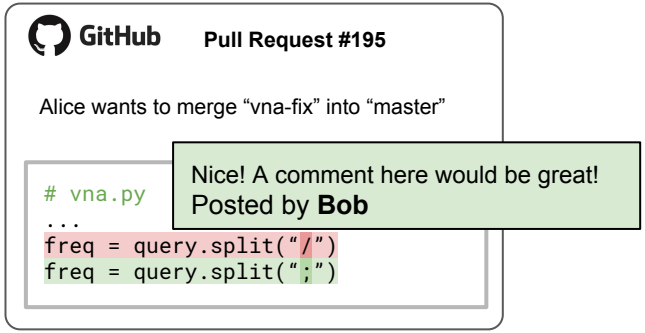
```
# vna.py
...
freq = query.split(“/”)
freq = query.split(“;”)
```

On Github.com:

1. Alice **pushes** “vna-fix”
2. She creates a Pull Request to merge to master.

Google AI Quantum



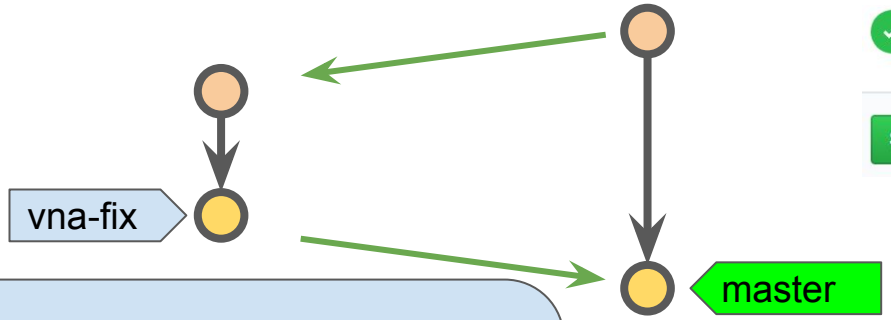


On Github.com:

1. Alice **pushes** “vna-fix”
2. She creates a Pull Request to merge to master.
3. Bob sees the pull request, adds a suggestion then **approves** it.

Google AI Quantum

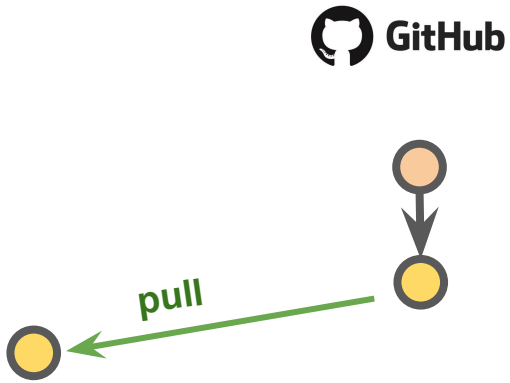
GitHub



On Github.com:

1. Alice **pushes** “vna-fix”
2. She creates a Pull Request to merge to master.
3. Bob sees the pull request, adds a suggestion then **approves** it.
4. Alice **merges** it into master in Github.

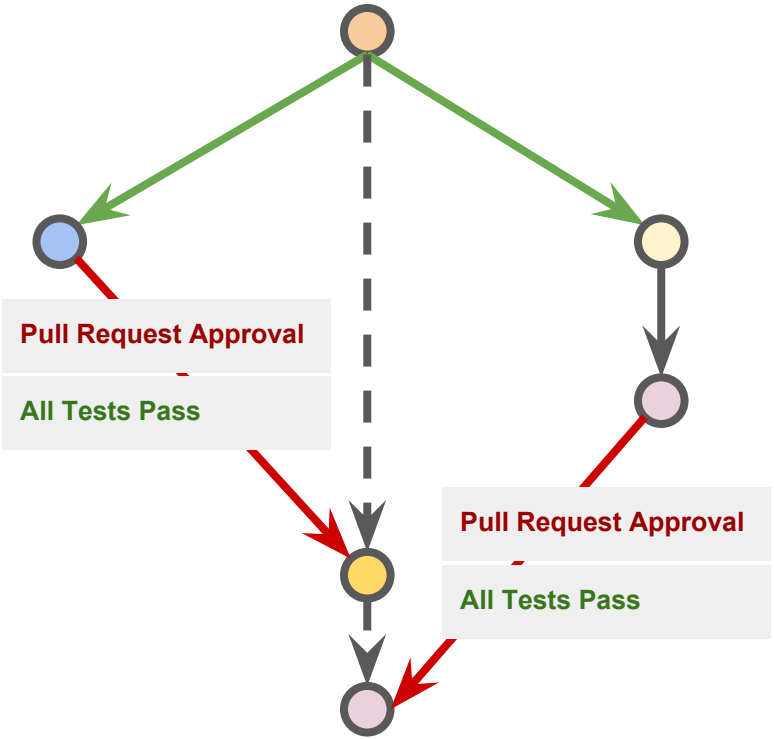
Google AI Quantum



On her local machine, Alice switches back to master, **pulls** the changes she just merged in, and takes more data, **resulting in reproducible results**.

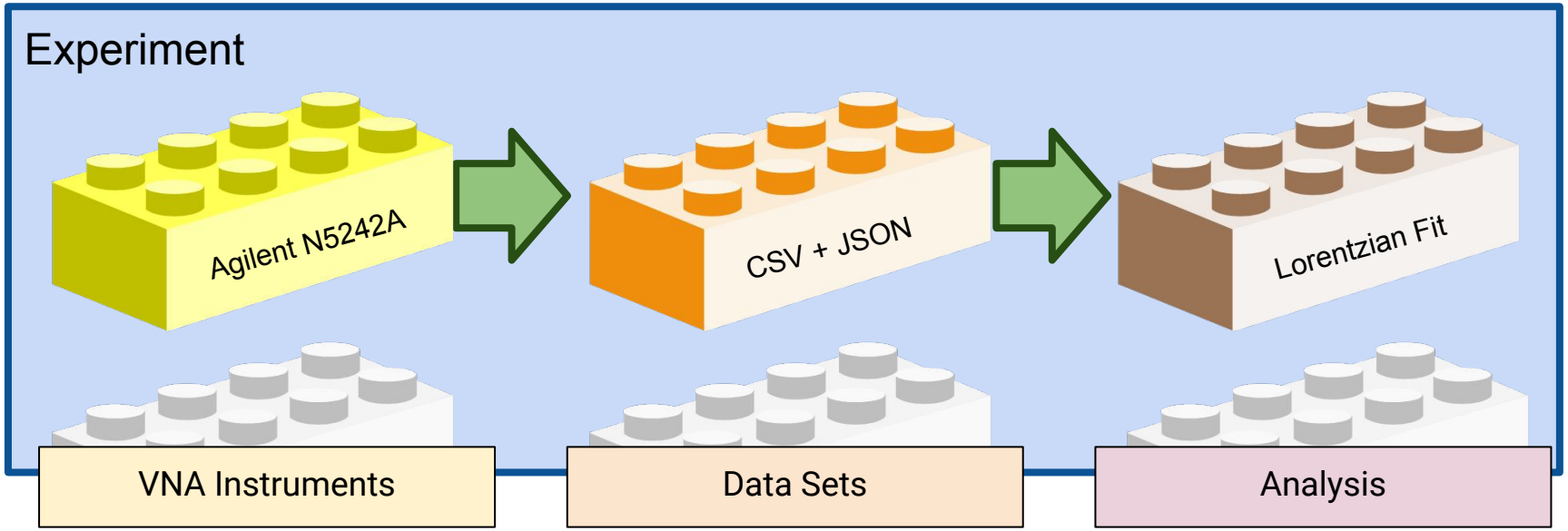
Automated Testing

- Code that tests code
- Wherever possible, all code paths should execute during tests
- Integrate with Github (**Continuous Integration**)



Experiment Code Modularity

Overview of Code



> Library of different VNA interfaces

> Well-tested, well-documented

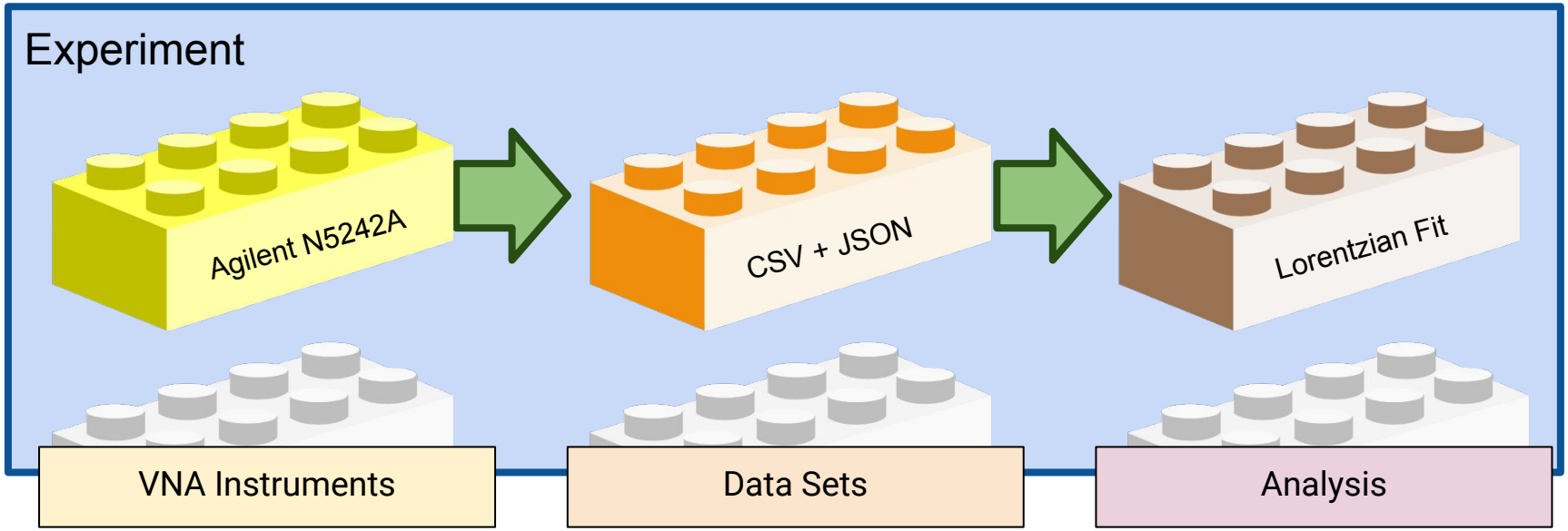
> Support CSV & Metadata

> Allow for many data sources (Excel, HDF5, database)

> Different approaches as appropriate

> Plotting tools for publication

Overview of Code



- Can isolate any part of this
  - e.g. Create interface to existing data and try different analyses

Governance/Code of Conduct

Code of Conduct

- Encourage people from all walks of life to contribute
- Positive, constructive feedback in Pull Request reviews
- See: https://github.com/Boulder-Cryogenic-Resonator-Testbed/measurement/blob/master/CODE_OF_CONDUCT.md

Governance

- How should the software organization be run?
- Handling conflicts, feature requests, legacy platforms, etc.

Potential Uses for These Tools

- Layout scripts - generating CPW geometries with materials-specific parameters
  - Open source resonator chip
- Publications - latex works well with git
- Sonnet, COMSOL models
- Solidworks (or other CAD) models
- Supply management
  - Shared BOMs

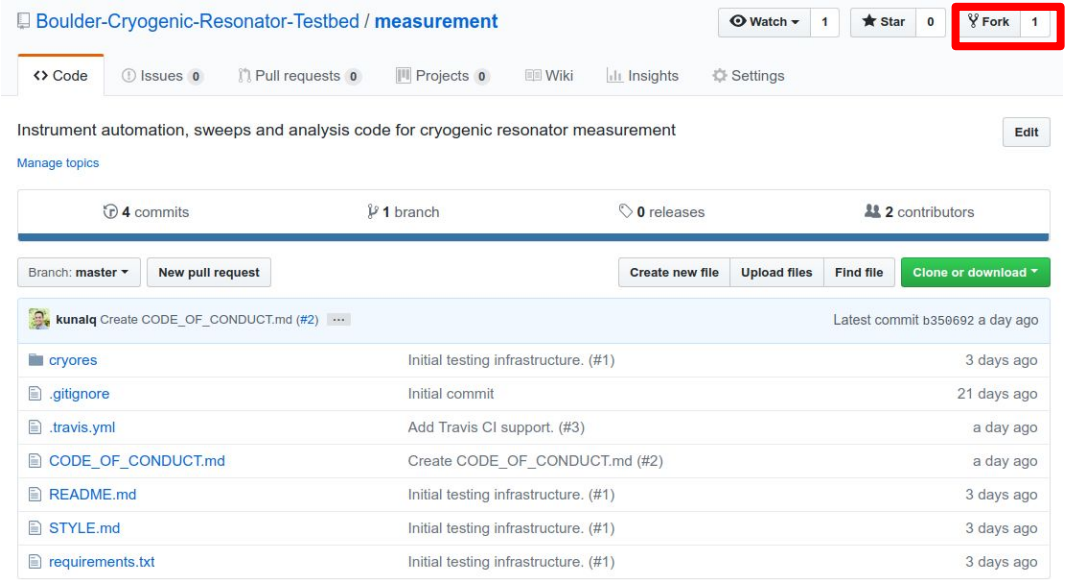
Questions/Comments?

kunalarya@google.com

Activities/Supplemental Slides

Pull Request Workflow

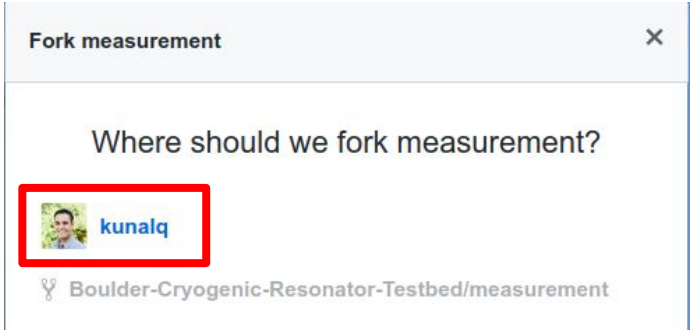
1. Sign up for an account on Github.com
2. Visit https://github.com/Boulder-Cryogenic-Resonator-Testbed/measurement
3. Click on the Fork button to fork the repo into your account:



Google AI Quantum

Pull Request Workflow

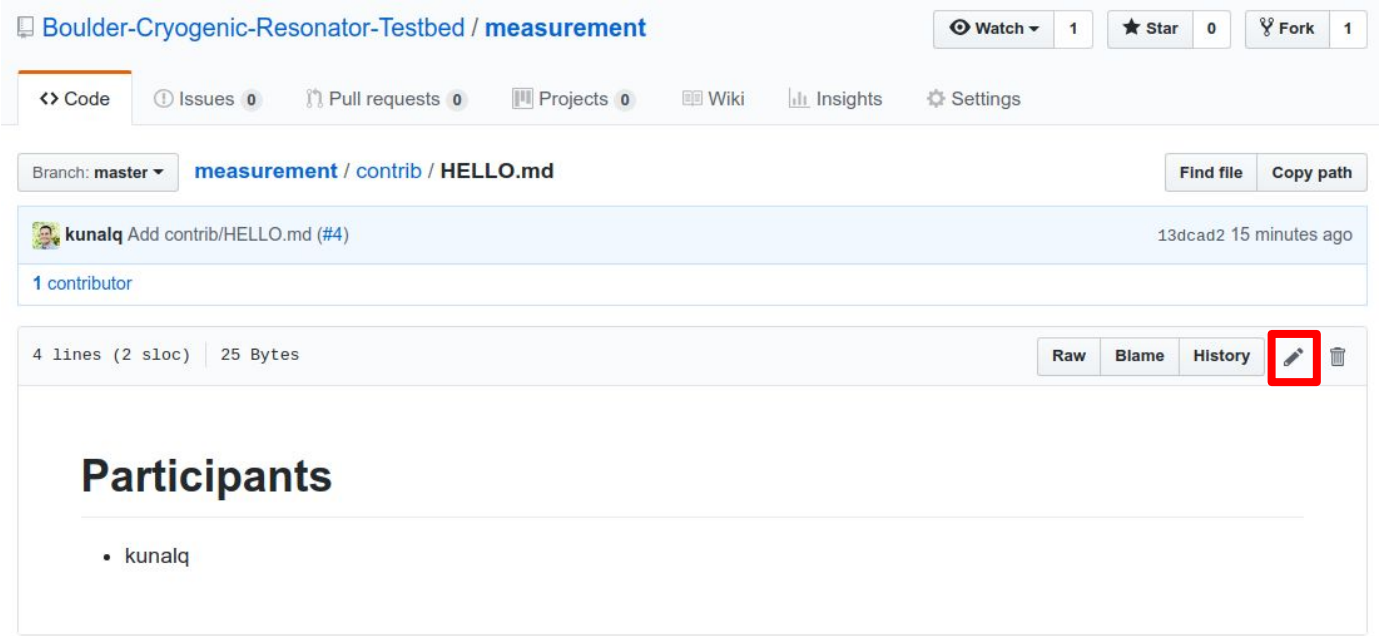
4. Select your username to fork into **[your-user-name]/measurement**



5. Click on the “contrib” folder, and the “HELLO.md” file.

Pull Request Workflow

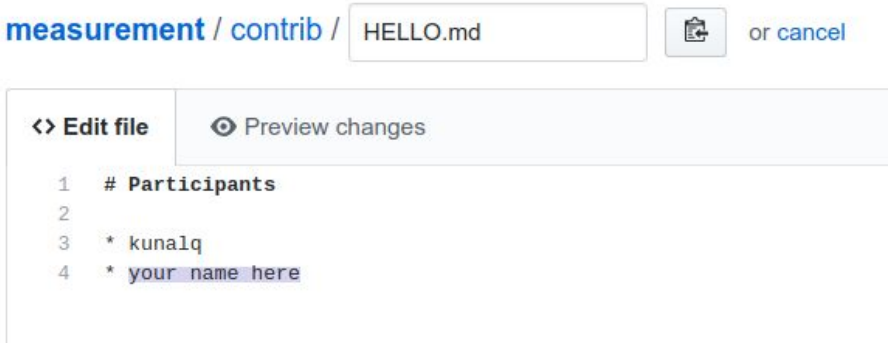
6. Edit the file and add your name:



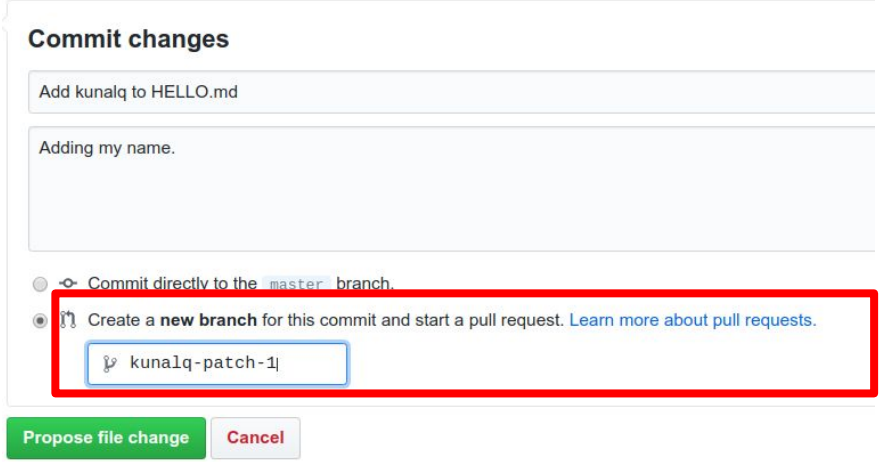
Google AI Quantum

Pull Request Workflow

6. Add your name:



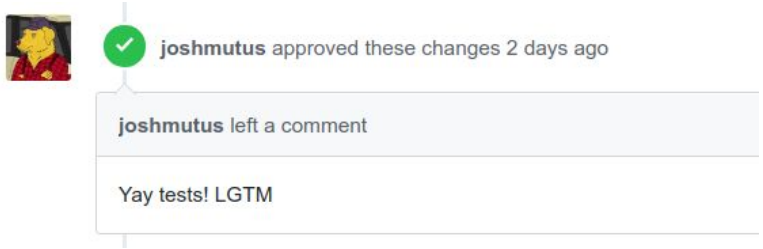
7. Fill in the commit summary and more information in the body. Add it to a **new branch**, then propose the change.



Google AI Quantum

Pull Request Workflow

8. Your PR is now ready! Wait for approval:



9. Merge the PR :)

Git and Github Resources:

- Software Carpentry
  - https://software-carpentry.org/
  - Can set up a software carpentry workshop at your institute!
- Installing git:
  - https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- Understanding git:
  - https://rachelcarmena.github.io/2018/12/12/how-to-teach-git.html
- Syncing your repo to the main repo:
  - First add the “upstream” remote: https://help.github.com/articles/configuring-a-remote-for-a-fork/
  - Fetch from “upstream” and merge: https://help.github.com/articles/syncing-a-fork/

Python Resources

- Creating a Python 3 virtualenv:
  - Linux (Ubuntu): https://gist.github.com/Geoyi/d9fab4f609e9f75941946be45000632b
  - Mac OS: https://gist.github.com/pandafulmanda/730a9355e088a9970b18275cb9eadef3
  - Windows: https://programwithus.com/learn-to-code/Pip-and-virtualenv-on-Windows/