

# Growing your contributors base



Hacking the Community Growth

# About us

---



Ismaël Mejía

Apache Avro/Beam committer & PMC  
Microsoft  
Twitter: [@iemejia](https://twitter.com/iemejia)



Jarek Potiuk

Apache Airflow Committer & PMC member  
Independent Open-Source Contributor and Advisor  
Twitter: [@jarekpotiuk](https://twitter.com/jarekpotiuk)

# Main theme for this talk

**Conscious** thinking  
about your contributor's base growth

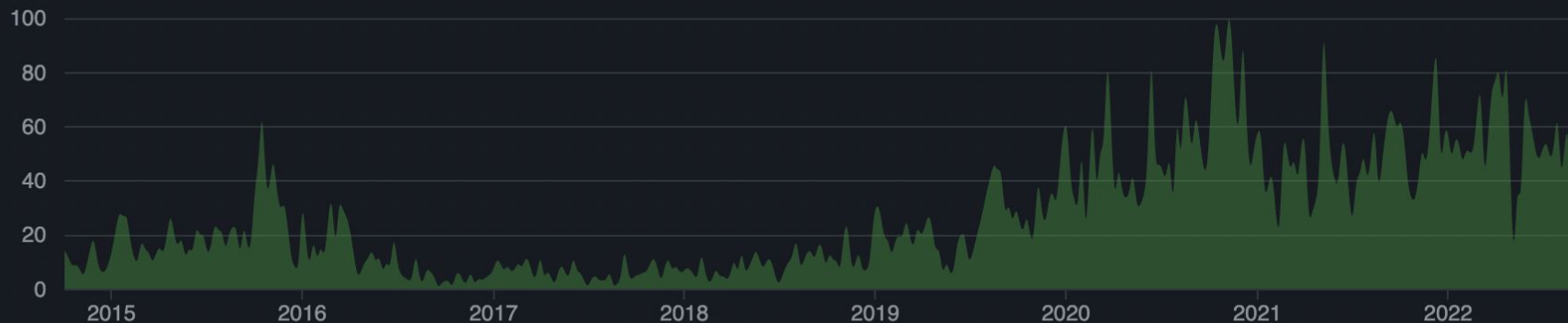
# Projects / Context

# Project evolution - Apache Airflow

Oct 5, 2014 – Aug 31, 2022

Contributions: Commits ▾

Contributions to main, excluding merge commits and bot accounts

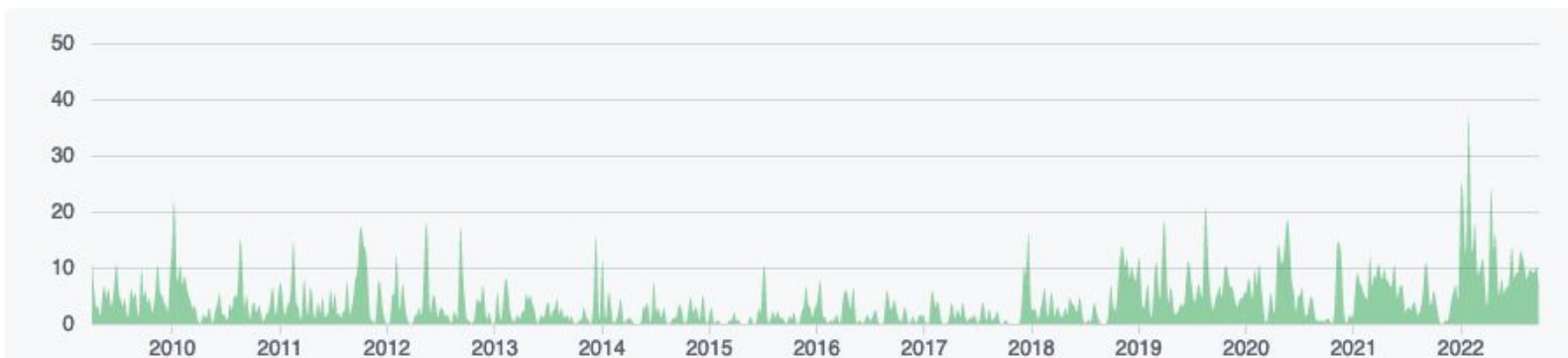


# Project evolution - Apache Avro

Apr 5, 2009 – Sep 28, 2022

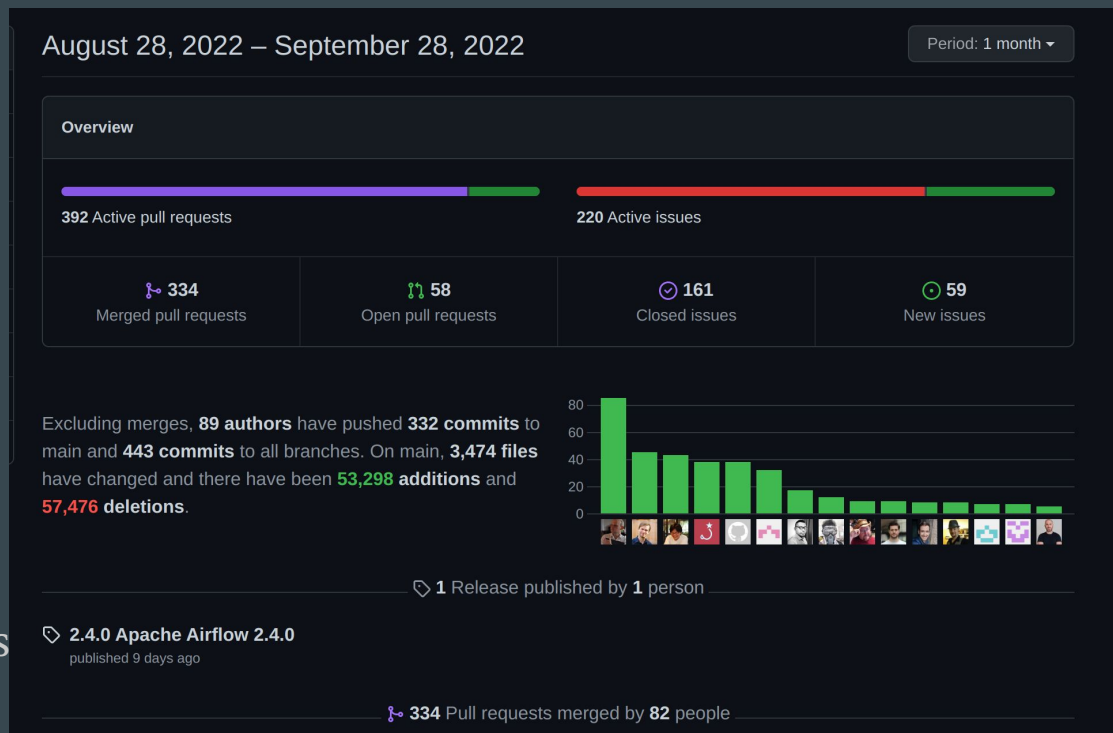
Contributions: Commits ▾

Contributions to master, excluding merge commits and bot accounts



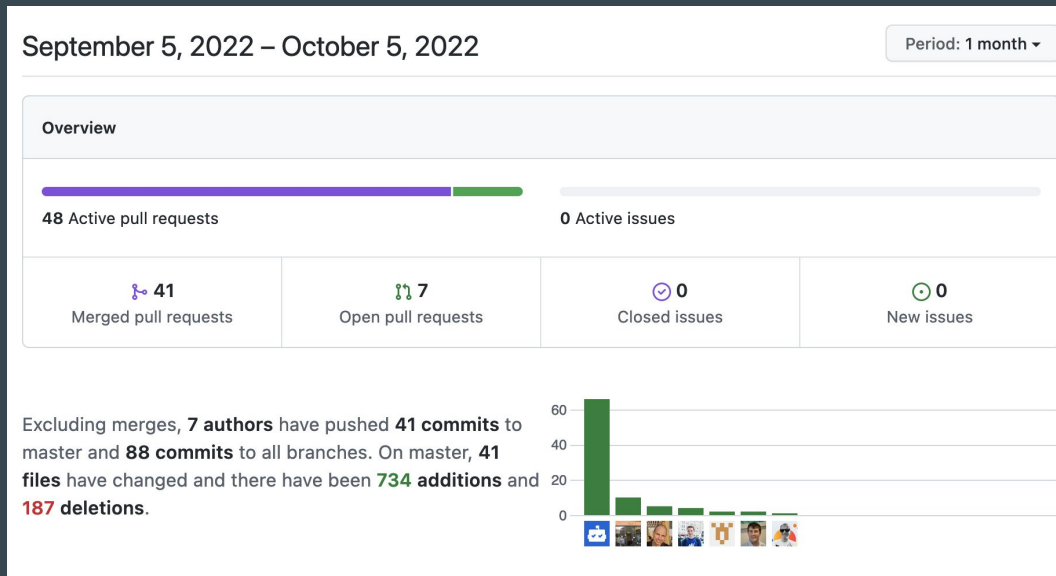
# Pulse - Apache Airflow

- >10 PR merged/day
- >80 active contributors
- >2 issues open/day
- >5 issues closed/day
- Strong/small group of strongly vested maintainers
- Bigger group of active contributors
- Long tail of casual contributors



# Pulse - Apache Avro

- 13 year old project
- Mature and stable
- ~2 PR merged/day
- ~6 active contributors
- Small group of casual maintainers (no one is paid)
- ~5 casual contributors per month





# Contribution landscape

# Types of (code) contributors

PMC members

Seasoned maintainers

Maintainers / committers

Frequent contributors/power users

One-time contributors - usually frequent users

People improving documentation - usually users that struggle

New contributors with programming experience usually frequent users of the project

New contributors with little programming experience but using the project occasionally or frequently

# Types of (non code) contributors

- Power Users
- Issue triagers
- Community animators
- Project advocates
- Local community animators
- Event organizers / speakers
- Stakeholders and their animators / advocates

# Why diversity matters?

- All contributions matter
- Getting to see different perspectives
- Users do not have same assumptions as vested committers
- Everyone can learn from seeing different perspectives
- Documentation might be unclear to new users
- Typos/spelling mistakes/grammar issues are annoying
- Inexperienced contributors might grow

# Conscious growing

# Why you might want to grow contributors?

- Increase the velocity of the project development
- Avoiding Single Points of Failure
- Getting more interest from existing users
- Improving project's diversity
- Reach out to different groups of users
- Opening up to new technologies

# Why you might want to NOT grow contributors?

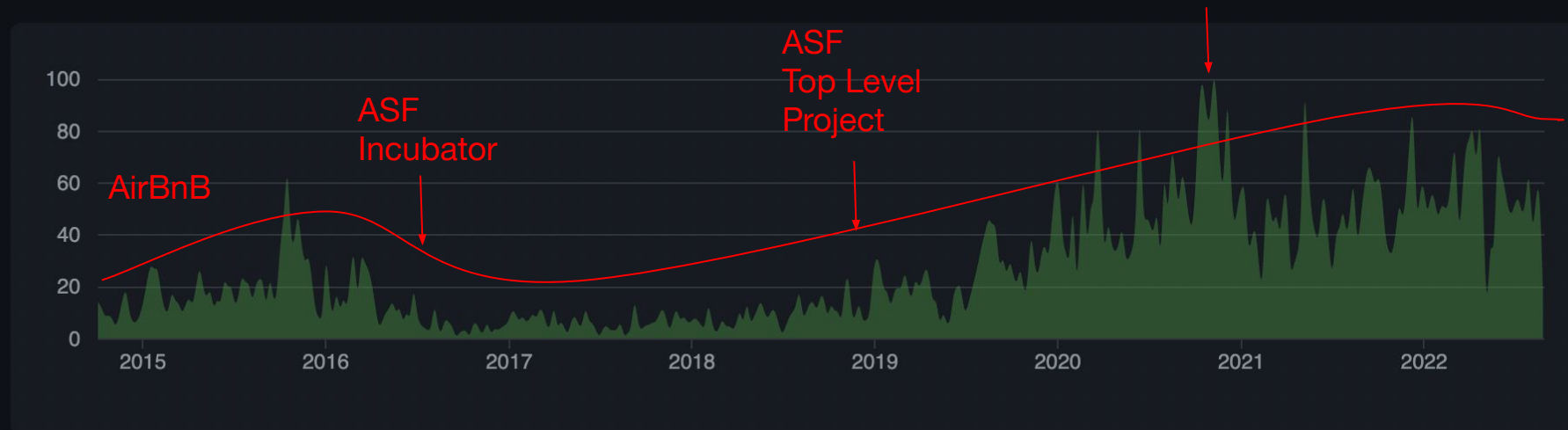
- Cost of on-boarding
- Dragging out maintainers
- Level of noise
- Too much discussion vs. job done
- Lack of focus
- Lack of guidelines to follow
- Lack of experienced contributors who can help newcomers
- Project in a wind-down phase

# Apache Airflow - growth needs

Oct 5, 2014 – Aug 31, 2022

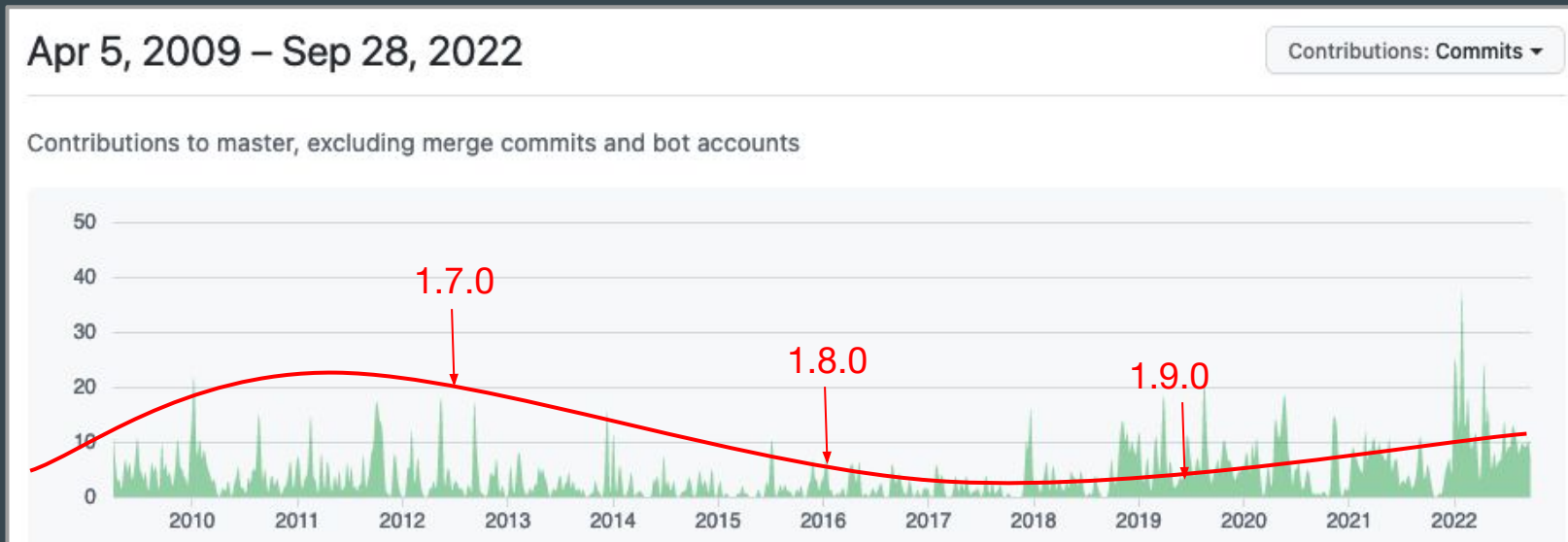
Contributions: Commits ▾

Contributions to main, excluding merge commits and bot accounts





# Apache Avro - Growth needs



# Growth Hacking Mindset

# Traps you can fall in (automated thoughts)

- **[Why promoting?]** We have great product - people will come by themselves
- **[I want to only code]** I just want to code and do not want to “sell”
- **[Others are better to promote]** Others can promote the project better than me
- **[Why to make it easy?]** People are smart and will find how to do stuff, go figure
- **[I am lost what to do]** I have no idea how to do “growth”, so I will do nothing
- **[We are not ready]** We are not ready yet to invite others, we need this and that
- **[We can be faster]** We can do it faster and better as we know the project
- **[They will not stay]** The contributors you help will not stay in the project

# Avoiding the traps (change your mindset)

- **[Why promoting?]** If no-one hears about perfect code, it's perfectly forgotten
- **[I want to only code]** Talking about the project you love is actually cool
- **[Others are better to promote]** People will look at you as an example
- **[Why to make it easy?]** Make it easy for others to learn - exercise empathy
- **[I am lost what to do]** Start spotting opportunities. Once you do it's easy
- **[We are not ready]** Perfect and not done is worse than crap and done
- **[We can be faster]** The outcome is also the learning/potential for the future
- **[They will not stay]** If only 1 person stay - you have 1 more contributor

Certain thing:

**If you do nothing - nothing will happen**

# Individuals and interactions

# Contributing to OSS is not only doing “good”

- Don't expect from people to give time for “free”
- Incentives: Everyone must get something
- Intrinsic motivations of people and finding them
- “Fairness” is super important
- Give people what they want
- Some people get paid for contributing

# Why Contributors contribute ?

- Experienced
  - They know the project well
  - They are tech enthusiasts
  - They are paid to do the job
- Occasional
  - Users want to get their problems fixed
  - Consultants want to contribute changes they got paid for
- Fresh
  - They want to learn from experienced ones
  - They expect some guidance and assistance/assurance
  - They want to learn good practices
  - They want to participate in popular OSS project and gain experience
  - They are obsessed with typos :)



# How to approach people

- Lead by example
- Pay forward
- Explain “free” nature of the projects - set people’s expectations
- Express expectations to give back
- Simply ask people to return favours
- Give the rod not the fish
- Be open but assertive

# Helping people: be helpful but assertive

- Be helpful for newcomers
- Set the boundaries
- Be clear about expectations
- Do not allow private troubleshooting - not good for community
- Be careful about burning out

# Example Tools

# Why do you need tools?

- Contributors
  - easy quick start
  - do not be afraid of breaking changes
  - guidance in contributions
  - avoid unnecessary discussions
  - tools have limits
- Maintainers:
  - consistency of the project (quality checks)
  - do not be afraid to merge
  - scale their efforts
  - avoid useless discussions
  - keep the sanity
  - have time for project direction

# Documentation - important tool

- Communication tool
- Cater to different users
  - Novices - step-by-step tutorials
  - Experienced - quickly find what they need
- Engineer the user experience - treat documentation as a tool
- Contributing documentation by users
  - Maintainers are bad documentation writers
  - Informed users are best to write the documentation
  - If you help the user and make them informed - ask them to contribute back

Time Zones — Airflow x +

airflow.apache.org/docs/apache-airflow/stable/timezone.html

Apache Airflow

Community Meetups Documentation Use-cases Announcements Blog Ecosystem

Version: 2.4.0 ▾

Search docs 🔍

**CONTENT**

- Overview
- Project
- License
- Quick Start
- Installation
- Upgrading from 1.10 to 2
- Tutorials
- How-to Guides
- UI / Screenshots
- Concepts
- Executor
- DAG Runs
- Plugins
- Security
- Logging & Monitoring
- Time Zones**
  - Web UI
  - Concepts
  - Time zone aware DAGs
- Using the CLI
- Integration
- Kubernetes
- Lineage
- Listeners
- DAG Serialization
- Modules Management
- Release Policies

Home / Time Zones

# Time Zones

Support for time zones is enabled by default. Airflow stores datetime information in UTC internally and in the database. It allows you to run your DAGs with time zone dependent schedules. At the moment, Airflow does not convert them to the end user's time zone in the user interface. It will always be displayed in UTC there. Also, templates used in Operators are not converted. Time zone information is exposed and it is up to the writer of DAG to decide what do with it.

This is handy if your users live in more than one time zone and you want to display datetime information according to each user's wall clock.

Even if you are running Airflow in only one time zone, it is still good practice to store data in UTC in your database (also before Airflow became time zone aware this was also the recommended or even required setup). The main reason is that many countries use Daylight Saving Time (DST), where clocks are moved forward in spring and backward in autumn. If you're working in local time, you're likely to encounter errors twice a year, when the transitions happen. (The pendulum and pytz documentation discuss these issues in greater detail.) This probably doesn't matter for a simple DAG, but it's a problem if you are in, for example, financial services where you have end of day deadlines to meet.


The time zone is set in `airflow.cfg`. By default it is set to UTC, but you change it to use the system's settings or an arbitrary IANA time zone, e.g. `Europe/Amsterdam`. It is dependent on `pendulum`, which is more accurate than `pytz`. Pendulum is installed when you install Airflow.

## Web UI

By default the Web UI will show times in UTC. It is possible to change the timezone shown by using the menu in the top right (click on the clock to activate it):

Time Zones

- Web UI
- Concepts
  - Naive and aware datetime objects
  - Interpretation of naive datetime objects
  - Default time zone
- Time zone aware DAGs
- Templates
- Cron schedules
- Time deltas

 Suggest a change on this page

# CI

- **Absolute must**
- Github Actions - fantastic integration with repos
- Gives contributors the confidence
- Guide them to fix problems / understand project practices
- Automate automatable commiter's tasks (chores)
- Keeps project consistency
- Allows to release with little overhead



# Development environment

- Help with all the basic development tasks
- Get it up and running in 10 minutes
  - Clone & “Codespace” - for novices
  - More sophisticated for experienced users
  - Freedom for “power users”
- Enable teamwork
  - Avoid “works for me” syndrome
  - Easy to reproduce CI failures
  - Easy to get help from others



# “It’s a Breeze to develop Airflow”

```
Breeze commands

Usage: breeze [OPTIONS] COMMAND [ARGS]...

Basic flags
--python          -p Python major/minor version used in Airflow image for images. (>3.7< | 3.8 | 3.9 | 3.10)
                  [default: 3.7]
--backend         -b Database backend to use. (>sqlite< | mysql | postgres | mssql) [default: sqlite]
--postgres-version -P Version of Postgres used. (>10< | 11 | 12 | 13 | 14) [default: 10]
--mysql-version   -M Version of MySQL used. (>5.7< | 8) [default: 5.7]
--mssql-version   -S Version of MsSQL used. (>2017-latest< | 2019-latest) [default: 2017-latest]
--integration     -I Integration(s) to enable when running (can be more than one).
                  (cassandra | kerberos | mongo | openldap | pinot | rabbitmq | redis | statsd | trino |
                  all)
--forward-credentials -f Forward local credentials to container when running.
--db-reset        -d Reset DB when entering the container.

Common options
--verbose         -v Print verbose information about performed steps.
--dry-run         -D If dry-run is set, commands are only printed, not executed.
--github-repository -g GitHub repository used to pull, push run images. (TEXT) [default: apache/airflow]
--answer         -a Force answer to questions. (y | n | q | yes | no | quit)
--max-time        -m Maximum time that the command should take - if it takes longer, the command will fail.
                  (INTEGER RANGE)
--help           -h Show this message and exit.

Basic developer commands
start-airflow    Enter breeze environment and starts all Airflow components in the tmux session. Compile assets
                 if contents of www directory changed.
static-checks    Run static checks.
build-docs       Build documentation in the container.
stop             Stop running breeze environment.
shell            Enter breeze environment. this is the default command use when no other is selected.
exec             Joins the interactive shell of running airflow container.
compile-www-assets  Compiles www assets.
cleanup          Cleans the cache of parameters, docker cache and optionally built CI/PROD images.

Advanced command groups
testing          Tools that developers can use to run tests
ci-image         Tools that developers can use to manually manage CI images
k8s              Tools that developers use to run Kubernetes tests
prod-image       Tools that developers can use to manually manage PROD images
setup            Tools that developers can use to configure Breeze
release-management  Tools that release managers can use to prepare and manage Airflow releases
ci               Tools that CI workflows use to cleanup/manage CI environment
```

**What else can I do?**

# Workshops

- Design first-time contributor workshops
- Explain how community works
- Build a base of good first issues
- Point to good first issues
- REALLY small issues
- Ideally ends up with merged change

# Measure and learn

- We do not have tools for it yet
- Apache Beam and Apache Airflow teams started to work on it
- Possible approach
  - Data driven approach
  - Looking at contributions/discussions
  - Spot opportunities better
  - Flag people to reach out to
  - At the right time
  - We are just starting ...

## Growing your contributors base - summary

- Consciously think about growth
- Adapt to the needs/timeline/stage/people
- Actively push it forward