Meeting of the LF AI & Data Technical Advisory Council (TAC)

January 27, 2022

LF AI & DATA

Antitrust Policy

Recording of Calls

Reminder:

TAC calls are recorded and available for viewing on the TAC Wiki

LF AI & DATA

Reminder: LF AI & Data Useful Links

- Web site: lfaidata.foundation
- Wiki: wiki.lfaidata.foundation
- GitHub: github.com/lfaidata
- Landscape: https://landscape.lfaidata.foundation or https://l.lfaidata.foundation
- Mail Lists: https://lists.lfaidata.foundation
- Slack: https://slack.lfaidata.foundation
- Youtube: https://www.youtube.com/channel/UCfasaeqXJBCAJMNO9HcHfbA
- LF AI Logos: https://github.com/lfaidata/artwork/tree/master/lfaidata
- LF AI Presentation Template: https://drive.google.com/file/d/1eiDNJvXCqSZHT4Zk_-czASlz2GTBRZk2/view?usp=sharing

- Events Page on LF AI Website: https://lfaidata.foundation/events/
- Events Calendar on LF AI Wiki (subscribe available): https://wiki.lfaidata.foundation/pages/viewpage.action?pageId=12091544
- Event Wiki Pages: https://wiki.lfaidata.foundation/display/DL/LF+AI+Data+Foundation+Events

LF AI & DATA

Agenda

- Roll Call (2 mins)
- Reminder about attending meetings and quorum (2 minutes)
- Reminder about Security Badge requirement (5 minutes)
- Artigraph Incubation Proposal (40 minutes)
- Approval of Minutes from previous meetings (2 mins)
- Reminder about Google Summer of Code interest (2 minutes)
- LF AI General Updates (2 min)
- Open Discussion (2 min)

TAC Voting Members - Please note

Please ensure that you do the following to facilitate smooth procedural quorum and voting processes:

- Change your Zoom display name to include your First/Last Name, Company/Project Represented
  - example: Nancy Rausch, SAS
- State your First/Last Name and Company/Project when submitting a motion
  - example: First motion, Nancy Rausch/SAS

Challenge with TAC Quorum

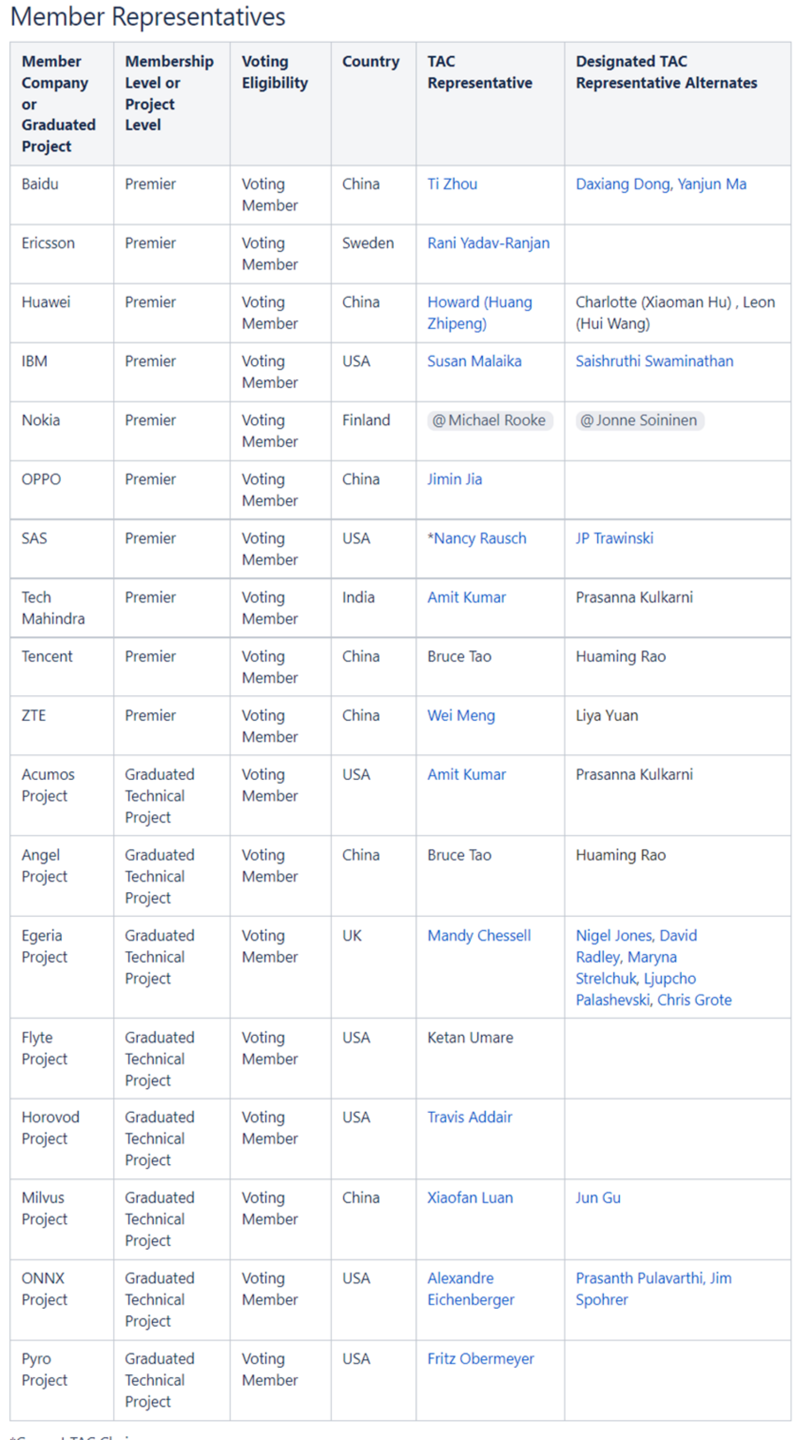
- 18 voting members requiring 10 voting members to achieve quorum

- Proposing updating charter to reflect the following changes:
  - A TAC voting member who misses 2 TAC meetings in a row will lose their voting seat until they attend twice in a row.

- Process: Socialize with GB and TAC. Propose amendment to the Charter and have the GB vote on it.

TAC Voting Members

* = still need backup specified on wiki



Member Representatives

Member Company or Graduated Project	Membership Level or Project Level	Voting Eligibility	Country	TAC Representative	Designated TAC Representative Alternates
Baidu	Premier	Voting Member	China	Ti Zhou	Daxiang Dong, Yanjun Ma
Ericsson	Premier	Voting Member	Sweden	Rani Yadav-Ranjan	
Huawei	Premier	Voting Member	China	Howard (Huang Zhipeng)	Charlotte (Xiaoman Hu) , Leon (Hui Wang)
IBM	Premier	Voting Member	USA	Susan Malaika	Saishruthi Swaminathan
Nokia	Premier	Voting Member	Finland	@Michael Rooke	@Jonne Soininen
OPPO	Premier	Voting Member	China	Jimin Jia	
SAS	Premier	Voting Member	USA	*Nancy Rausch	JP Trawinski
Tech Mahindra	Premier	Voting Member	India	Amit Kumar	Prasanna Kulkarni
Tencent	Premier	Voting Member	China	Bruce Tao	Huaming Rao
ZTE	Premier	Voting Member	China	Wei Meng	Liya Yuan
Acumos Project	Graduated Technical Project	Voting Member	USA	Amit Kumar	Prasanna Kulkarni
Angel Project	Graduated Technical Project	Voting Member	China	Bruce Tao	Huaming Rao
Egeria Project	Graduated Technical Project	Voting Member	UK	Mandy Chessell	Nigel Jones, David Radley, Maryna Strelchuk, Ljupcho Palashevski, Chris Grote
Flyte Project	Graduated Technical Project	Voting Member	USA	Ketan Umare	
Horovod Project	Graduated Technical Project	Voting Member	USA	Travis Addair	
Milvus Project	Graduated Technical Project	Voting Member	China	Xiaofan Luan	Jun Gu
ONNX Project	Graduated Technical Project	Voting Member	USA	Alexandre Eichenberger	Prasanth Pulavarthi, Jim Spohrer
Pyro Project	Graduated Technical Project	Voting Member	USA	Fritz Obermeyer	

LF AI & DATA

Security Badge Requirement

- All project maintainers are reminded that the OpenSSF Best Practices Badge (formerly known as CII Best Practices badge) is a requirement for all hosted projects.
- Currently only a handful of projects show that they've earned the badge: https://landscape.lfai.foundation/card-mode?bestpractices=yes&project=hosted
- Please as project leader take the initiative to enroll the project in the program and get it badged and show your badge on your GH org

Please direct questions to Ibrahim Haddad <ibrahim@linuxfoundation.org>

Artigraph

Declarative Data Production

Artifact + Graph = Artigraph

Artigraph is a tool to improve the authorship, management, and quality of data. It emphasizes that the core deliverable of a data pipeline or workflow is the data, not the tasks.

Language: Python
License: Apache 2.0
Project Lead: Jacob Hayes
Replica Contributors: Brett Naul, Joyce Xu, Kael Greco,
Marco Palmeri, Robert Regue, Steven Soojin Kim
Source: https://github.com/artigraph/artigraph

Overview and Vision

Motivations

Artigraph is motivated by experience:

- collaborating with data engineers to ingest, maintain, and QA data
- enabling data operators to run pipelines with self-service tooling
- helping application developers design environments, publishing, and versioning
- interfacing with customer teams to release data timely and safely
- supporting data scientists and modellers in shipping production-ready code

Challenges

- Ad hoc experimentation
  - Lose track of inputs/parameters
  - Parallel experiments may conflict
- Failed jobs requiring human intervention
  - Identify the kind of failure
  - Cleanup partial data
  - Try to restart from a good spot
- I/O is tricky
  - Changing format/storage needs
  - Ingestion quirks (buffered, flakey, rate-limited, etc)
  - Types may not be one:one

Challenges

- Shared / mutable data ("the prod table")
  - Accidents or failures expose bad data
  - Hard or manual rollback (*if* backed up!)
  - Unknown history
- Missing, late, or partial vendor deliveries
  - Manual recovery
  - Subtle downstream impacts
- "What data is available?" / "Is this good to use?"
  - Engineers in the loop
  - Hard to discover latest data or version
  - Lack of trust in data quality

Use Cases

Automatic data lineage, provenance, and quality tracking

Features

- Arbitrary partitioning and mapping
- Automatic backfills and rebuilds
- Decoupled I/O
- Definition-time validation
- Intelligent target-based checkpointing with (relatively) friendly paths/names
- Native dependency tracking
- Pluggable types, formats, storage, views, etc
- Schema checking and quality gating
- Self-healing failure recovery
- Tag / publish a cohesive collection of data, atomically

Use Cases

Build resilient data graphs easily

Features

- Arbitrary partitioning and mapping
- Automatic backfills and rebuilds
- Decoupled I/O
- Definition-time validation
- Intelligent target-based checkpointing with (relatively) friendly paths/names
- Native dependency tracking
- Pluggable types, formats, storage, views, etc
- Schema checking and quality gating
- Self-healing failure recovery
- Tag / publish a cohesive collection of data, atomically

Use Cases

Work and experiment independently, in parallel

Features

- Arbitrary partitioning and mapping
- Automatic backfills and rebuilds
- Decoupled I/O
- Definition-time validation
- Intelligent target-based checkpointing with (relatively) friendly paths/names
- Native dependency tracking
- Pluggable types, formats, storage, views, etc
- Schema checking and quality gating
- Self-healing failure recovery
- Tag / publish a cohesive collection of data, atomically

Use Cases

Zero-downtime data-application updates with easy, quick, and safe rollback

Features

- Arbitrary partitioning and mapping
- Automatic backfills and rebuilds
- Decoupled I/O
- Definition-time validation
- Intelligent target-based checkpointing with (relatively) friendly paths/names
- Native dependency tracking
- Pluggable types, formats, storage, views, etc
- Schema checking and quality gating
- Self-healing failure recovery
- Tag / publish a cohesive collection of data, atomically

Target Roles - Data Engineers

Data Engineers will likely be the primary users, initiating experimentation and driving adoption within a project or organization.

The declarative structure (eg: I/O, partitioning, schema, validation) will support fast iteration and reduce the marginal development cost. Checkpointing and partition based parallelization will speed execution and simplify failure recovery.

Over time, clear and discoverable schemas and dependencies will ease refactoring and impact analysis. Additionally, Artifact- (schema) and Graph- (inputs, intermediates, and outputs) based interfaces and publishing support collaboration and evolution.

Target Roles - Data Scientists and ML Engineers

Data Science or ML Eng may discover the project when "productionizing" analysis or models, prompting experimentation.

Definition-time validation, managed I/O, and automatic rebuilds will hasten development. Schema definitions and partition visibility will promote discovery and reuse.

Schema standardization and the ability to track lineage will support reproducible analysis and running many experiments in parallel without losing track of inputs/parameters. Associated Statistics enable tracking and comparing quality over time or across experiments.

Target Roles - Data Operations

Data Operations will benefit from Artigraph's data-level management and visibility.

Self-service visibility into data availability and health will reduce communication friction and delays. Native data management tooling will simplify failure recovery.

Lineage and quality metadata will allow comparisons over time and support root cause analysis of quality issues (which can often present downstream). Additionally, operators can identify at-risk pipelines or products downstream from flakey vendors/providers, supporting proactive redundancy planning.

Target Roles - Business

The larger business, while not often direct users, will benefit from improved data quality and developer efficiency and efficacy.

With improved development and quality controls, product timelines should firm and shorten. Additionally, with increased and incremental quality control, quality and importantly *trust* in data deliverables should increase. With atomic data publishing and native versioning, product updates become a much lighter, safer process.

As tooling improves and barriers lower, certain business roles can use self-service dashboards over data availability, lineage, and quality to support customers and other internal functions.

Why Artigraph?

Work with first class Artifacts, not return values

- Access data interactively, outside a workflow, by accessing the Graph
- Manage the entire data lifecycle: sourcing, garbage collection, versioning, publishing, etc
- Schema and quality checks are associated with Artifacts, not tasks
- Set Artifact level metadata, such as "vendor", "data owner", "vintage", etc
- Underlying data is discoverable with meaningful and contextual paths/names

Simplify logic with declarative metadata

- Automatically validate schema, compute statistics, and apply quality gating
- Decouple data format+storage from preferred in-memory data structure with automatic I/O
- Leverage metadata, such as custom annotations, derived statistics, and partition keys, to define dependencies
- Use standard python type hints to define parameters and returns, supporting static analysis tools such as `mypy`

Project Structure

Overview

Artigraph is written in Python as a collection of small, well defined interfaces with pluggable implementations. The project is distributed as a namespace package, allowing for progressive addition and enhancement. Most data structures are immutable with state encapsulated in Backends. A majority of the interfaces are defined, but need implementations contributed.

There are 3 primary components: Artifacts, Producers, and Graphs.

Primary Components - Artifact

Artifacts are first class representations of *data* comprised of:

› type: structure of the data - data type, fields, nullability, partitioning, etc.
› format: serialized format - CSV, Parquet, database native, etc.
› storage: persistent storage system - blob storage, database native, etc.

Example:

```
class TotalSpend(Artifact):
    """Aggregate spend over all time."""
    type = Float64()
    format = JSON()
    storage = LocalFile()
```

Primary Components - Artifact

Artigraph has a robust hub-and-spoke type system supporting format and framework type adaptation.

Artifacts serve as a data contract, easing collaboration, mocking, testing, and validation. The configurable format and storage can be swapped to tailor for tests, local dev, or remote builds.

Friendly, contextual paths/names are generated for all Artifacts, easing interactivity and debuggability. Arbitrary partitioning and data discovery supports flexible granularity and self-healing.

Primary Components - Producer

Producers are *tasks* taking and producing Artifact(s) comprised of:

- version: versioning strategy - GitCommit, SemVer, Source, etc
- map: logic defining partition dependencies - 1:1, rolling window, geo buffer, etc
- build: logic to generate an output partition - python, pandas, SQL query, etc

Example:

```
@producer(version=SemVer(major=1, minor=0, patch=0))
def aggregate_transactions(
    transactions: Annotated[list[dict], Transactions]
) -> Annotated[float, TotalSpend]:
    return sum(txn["amount"] for txn in transactions)
```

Primary Components - Producer

Producers operate on in-memory Views (native, pandas, database, etc) defined in standard type hints, which are validated against the Artifact Type *at Producer definition*. Dependencies are defined by Artifact *kind*, not name, easing linting, refactoring, and reuse.

Each output partition, as defined by *map*, gets a unique "input fingerprint" based on the producer name, version, and fingerprint of input partition *content*.

Producers should aim to be deterministic and side-effect free.

Primary Components - Graph

Graphs define a DAG comprised of:

- artifacts: arbitrarily nested named collection of *raw* and *produced* artifacts
- backend: state storage for all metadata - memory, file, database, API, etc
- executor: adaptor to a workflow execution tool - Airflow, Argo, Prefect, etc

Example:

```
with Graph(name="test") as g:
    g.artifacts.transactions = Transactions(
        annotations=[Vendor(name="Acme")],
        format=JSON(),
        storage=LocalFile(path=str(DIR / "transactions" / "{date.iso}.json")),
    )
    g.artifacts.spend = aggregate_transactions(transactions=g.artifacts.transactions)
```

Primary Components - Graph

A Graph is named and may materialize to one of many *snapshots* defined by the Graph *structure*, Producer *name* and *version*, and *raw* Artifact *contents*. Graph Snapshots can be tagged/published for future lookup.

Metadata, such as partitions, lineage, and tags, is stored in the Backend.

Graphs provide a convenient handle to interact with Artifact data for cross-graph references and interactive or application use.

Example

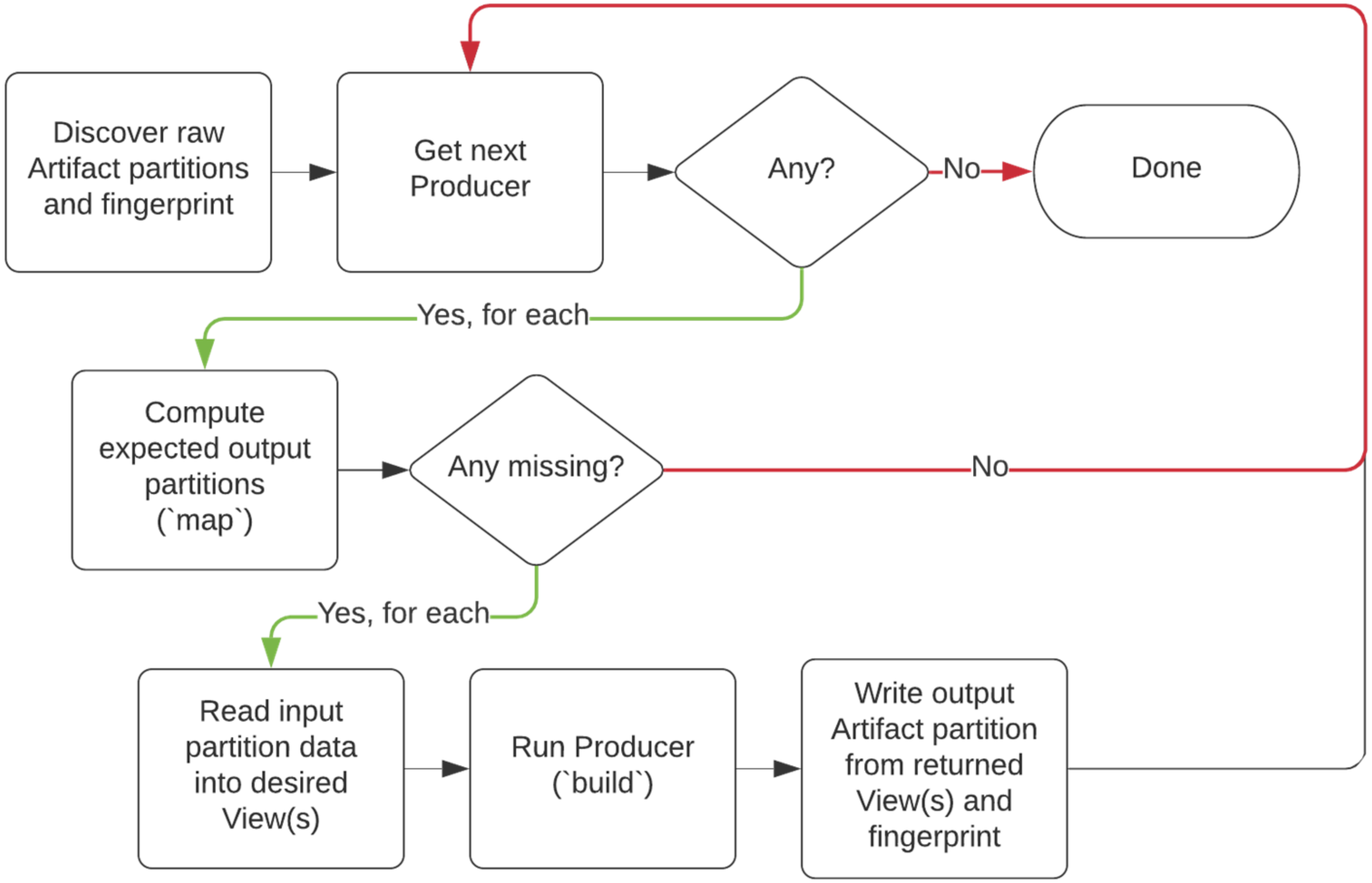
```python
class Transactions(Artifact):
    """Transactions partitioned by day."""
    type = Collection(
        element=Struct(fields={"date": Date(), "amount": Float64()}),
        partition_by=("date",),
    )

class TotalSpend(Artifact):
    """Aggregate spend over all time."""
    type = Float64()
    format = JSON()
    storage = LocalFile()

@producer(version=SemVer(major=1, minor=0, patch=0))
def aggregate_transactions(
    transactions: Annotated[list[dict], Transactions]
) -> Annotated[float, TotalSpend]:
    return sum(txn["amount"] for txn in transactions)

with Graph(name="test-graph") as g:
    g.artifacts.vendor.transactions = Transactions(
        annotations=[Vendor(name="Acme")],
        format=JSON(),
        storage=LocalFile(path=str(DIR / "transactions" / "{date.iso}.json")),
    )
    g.artifacts.spend = aggregate_transactions(transactions=g.artifacts.vendor.transactions)
```

g.build()


Discover raw Artifact partitions and fingerprint
Get next Producer
Any?
No
Done
Yes, for each
Compute expected output partitions (`map`)
Any missing?
No
Yes, for each
Read input partition data into desired View(s)
Run Producer (`build`)
Write output Artifact partition from returned View(s) and fingerprint

Planned Features

Short Term

- Dashboard
- Statistics
- Thresholds

Medium Term

- Hooks
- Metrics
- Resources

Long Term

- Cost Attribution
- Data Access / IAM
- Garbage Collection
- Row- and Field- Slicing

Very Long Term

- DaaS Platform / Data Hub

Incubation

Target Contributors

Data Engineers are expected to be the primary contributors, adding support for their preferred tools. As the project grows, companies and OSS communities developing complementary tools may directly contribute support or maintenance.

The main contributions will be:

- Formats
- Storage
- Statistics
- Views
- I/O
- Resources
- Backends
- Executors

Why Incubate?

- Adopt governance best practices developed by LF AI & Data and projects
- Collaborate with other LF AI & Data projects
- Increase project visibility to gather contributors and users
- Neutral holding ground

Possible Collaborations

- AI Explainability 360 / AI Fairness 360: explain and assess bias in model outputs
- Amundsen: data discovery and visualization
- Flyte / Kedro: pipeline execution/integration
- ONNX: serialize model outputs
- OpenLineage / Egeria / Marquez: metadata backend
- RosaeNLG: user-friendly content in dashboards

Incubation

We're formally requesting the incubation of Artigrah in LF AI & Data at the Sandbox level. Looking forward to a positive vote and great collaborations to follow.

TAC Vote on Artigraph Incubation at the Sandbox Level Project Proposal

Proposed Resolution:

The TAC approves the Artigraph proposal as an incubation at the sandbox level project of the LF AI & Data Foundation

Minutes approval

Approval of January 13, 2022 Minutes

Draft minutes from the January 13 TAC call were previously distributed to the TAC members via the mailing list

Proposed Resolution:

- That the minutes of the January 13th meeting of the Technical Advisory Council of the LF AI & Data Foundation are hereby approved.

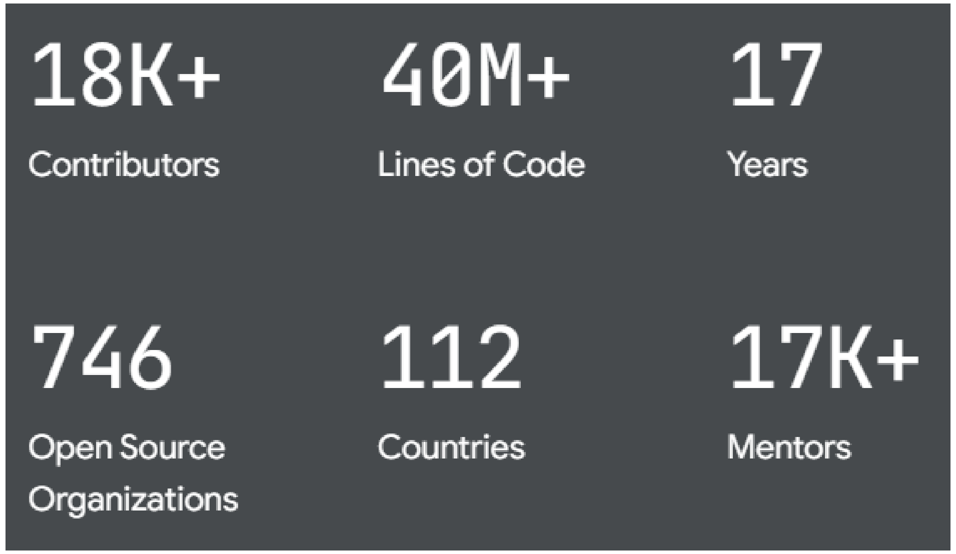
Reminder: Google Summer of Code - Interest in LFAI Mentorship?

Please reach out to Jun Gu jun.gu@zilliz.com if interested

LF AI & DATA

Apply Google Summer of Code 2022

A global, online program focused on bringing new contributors into open source software development.



18K+ Contributors	40M+ Lines of Code	17 Years
746 Open Source Organizations	112 Countries	17K+ Mentors

More details:
https://summerofcode.withgoogle.com/

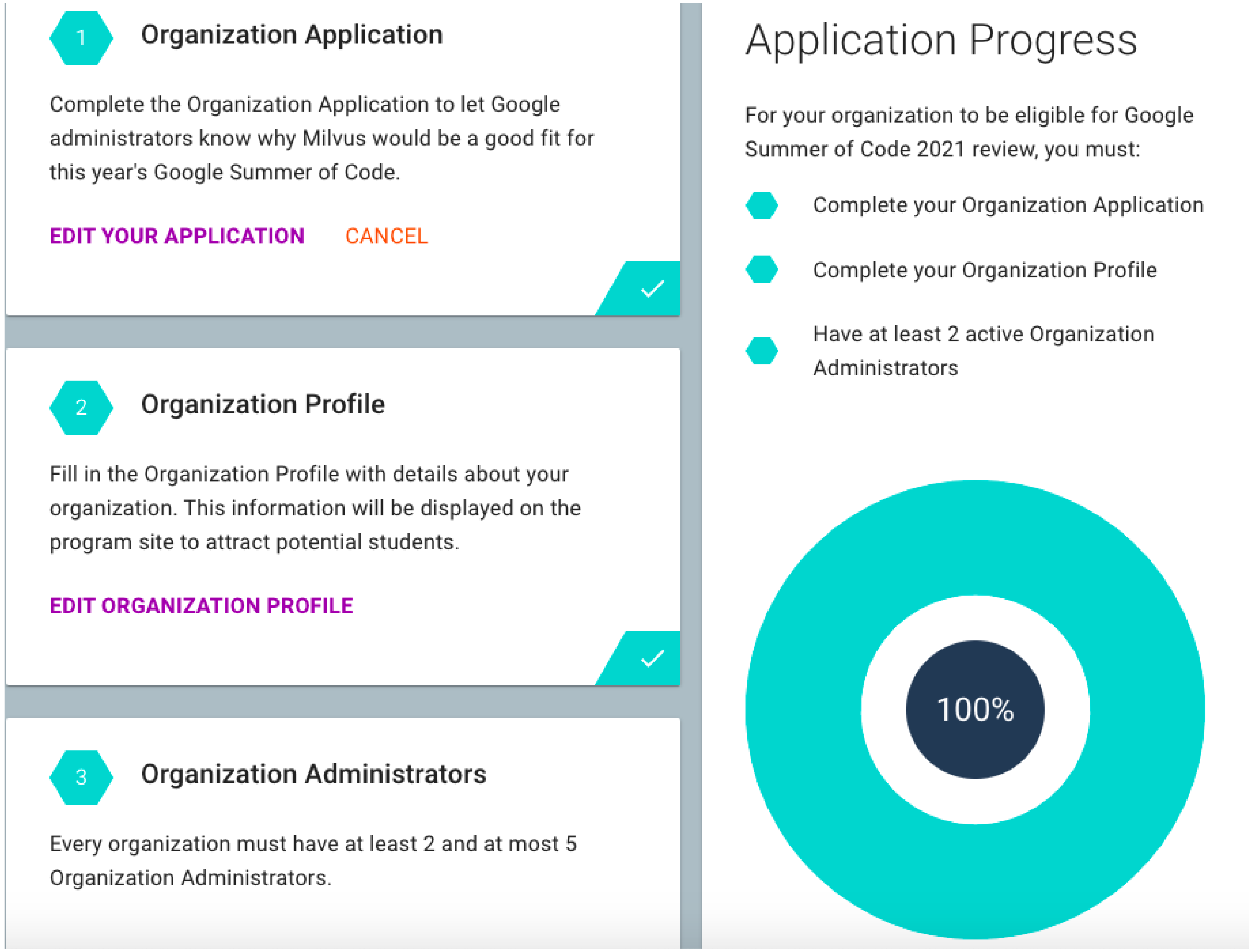
[Expanding Google Summer of Code in 2022]()

The organizations apply in **Feb. 2022**, what we need to do?

- Register the organization with GSoC
  - 2-5 org administrators needed

- List the program ideas/projects, e.g.,
  - The idea/project in detail
  - The mentors for the idea
  - Skills needed for the idea
  - The expected outcome of the idea

Reference from GSoC 2021: Organization Application



1 **Organization Application**

Complete the Organization Application to let Google administrators know why Milvus would be a good fit for this year's Google Summer of Code.

EDIT YOUR APPLICATION CANCEL

2 **Organization Profile**

Fill in the Organization Profile with details about your organization. This information will be displayed on the program site to attract potential students.

EDIT ORGANIZATION PROFILE

3 **Organization Administrators**

Every organization must have at least 2 and at most 5 Organization Administrators.

Application Progress

For your organization to be eligible for Google Summer of Code 2021 review, you must:

- Complete your Organization Application
- Complete your Organization Profile
- Have at least 2 active Organization Administrators

100%

Upcoming TAC Meetings

LF AI & DATA

Upcoming TAC Meetings

- February 10, 2022: Kompute moving from Sandbox to Incubation
- February 24, 2022: Project review TBD

Please send agenda topic requests to tac-general@lists.lfaidata.foundation

Open Discussion

LF AI & DATA

TAC Meeting Details

- To subscribe to the TAC Group Calendar, visit the wiki: https://wiki.lfaidata.foundation/x/cQB2 ______
- Join from PC, Mac, Linux, iOS or Android: https://zoom.us/j/430697670
- Or iPhone one-tap:
  - US: +16465588656,,430697670# or +16699006833,,430697670#
- Or Telephone:
  - Dial(for higher quality, dial a number based on your current location):
  - US: +1 646 558 8656 or +1 669 900 6833 or +1 855 880 1246 (Toll Free) or +1 877 369 0926 (Toll Free)
- Meeting ID: 430 697 670
- International numbers available: https://zoom.us/u/achYtcw7uN

Legal Notice

LF AI & DATA

27JAN2022