# Bridging into Python Ecosystem with Cloud-Native Distributed Machine Learning Pipelines

Yuan Tang, akuity.io
@TerryTangYuan

## Akuity

- https://akuity.io
- Vendor supported enterprise grade distribution of Argo
- Expert support and services from project maintainers

## About me

- Founding Engineer at akuity.io (the enterprise company for Argo)
- Maintainer/PMC/Co-chair
  - ML Frameworks: XGBoost, TensorFlow, metric-learn, Apache MXNet, etc.
  - Infrastructure: Argo Workflows, Kubeflow, etc.
- Books
  - Distributed Machine Learning Patterns (🔔 available on Manning MEAP)
  - TensorFlow in Practice (in Chinese)
  - Dive into Deep Learning (with TensorFlow)
- Contact
  - Twitter/LinkedIn/GitHub: @TerryTangYuan
  - Email: terrytangyuan@gmail.com

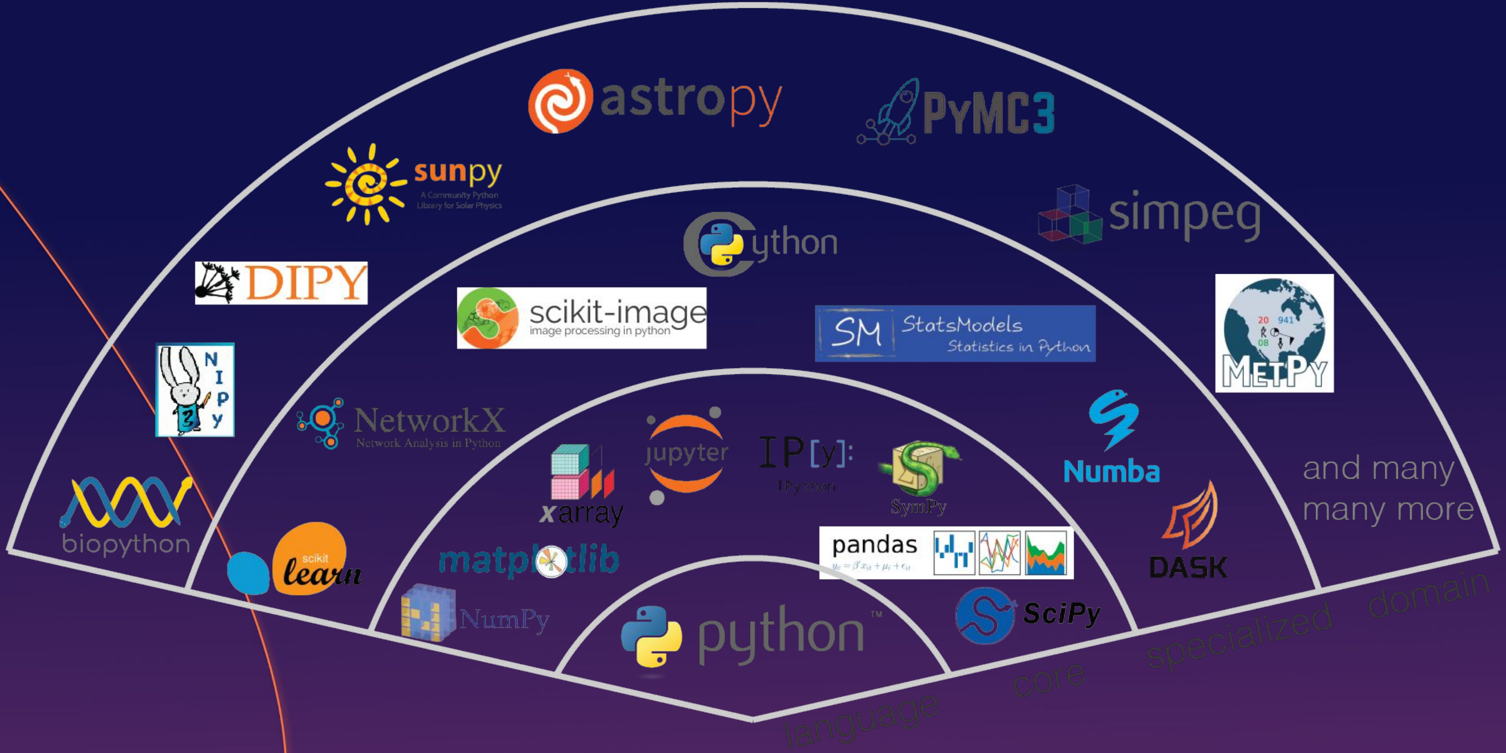
## Agenda

1. Components of distributed ML pipelines
2. Python scientific ecosystem
3. Machine learning frameworks
4. Workflow orchestration tools
5. Cloud-native and Kubernetes
6. Kubernetes-native ML pipelines
7. Stronger together: future outlook

# Components of Distributed ML Pipelines

- Data ingestion and preprocessing
  - Batching/caching/streaming
  - Feature engineering/feature stores
- Distributed model training
  - Hyperparameter tuning
  - Model selection/architecture search
  - Distribute training strategies (PS and allreduce)
  - Scheduling techniques (priority, gang, elastic scheduling, etc.)
- Model serving
  - Replicated services
  - Sharded services
  - Event-driven processing
- Workflow orchestration
- Check out Distributed Machine Learning Patterns for more established patterns

# Python Scientific Ecosystem



https://jupytearth.org/jupyter-resources/introduction/ecosystem.html

TensorFlow

PyTorch

scikit learn

dmlc XGBoost

## Machine Learning Frameworks in Python

Apache Airflow

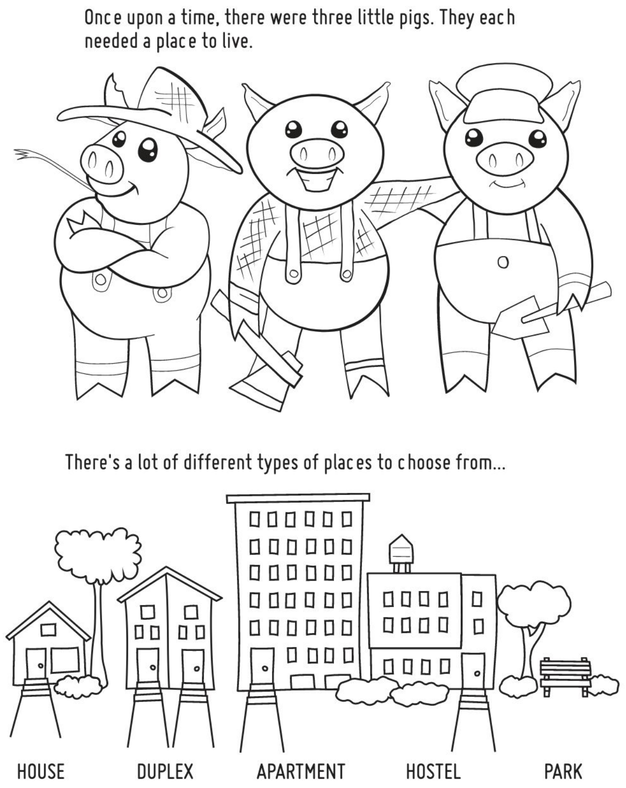
PREFECT

METAFLOW

## Python-native Workflow Orchestration Tools

## What are cloud-native and Kubernetes?

## Cloud-native and Kubernetes (K8s)

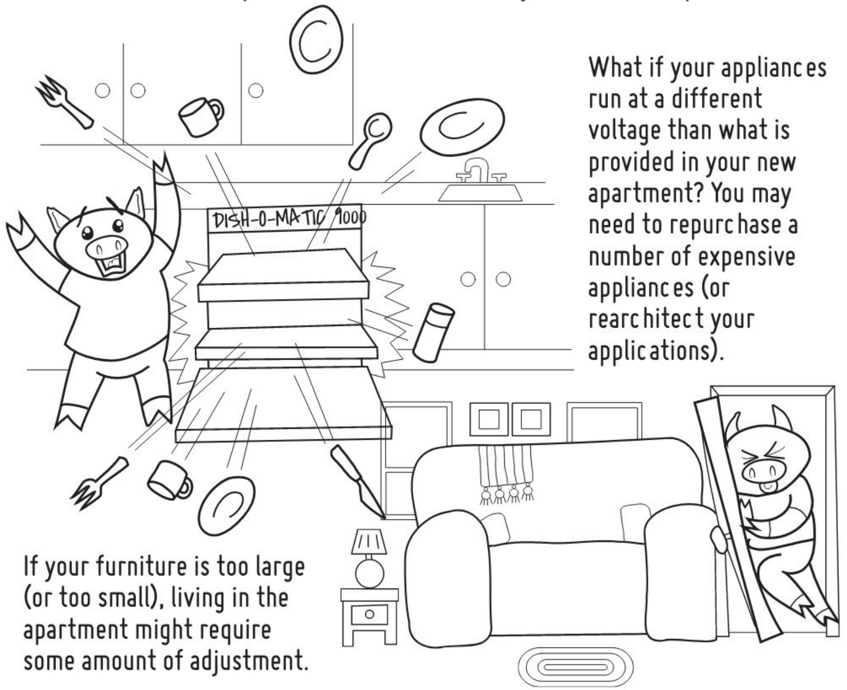

Once upon a time, there were three little pigs. They each needed a place to live.
There's a lot of different types of places to choose from...
HOUSE
DUPLEX
APARTMENT
HOSTEL
PARK

*The Container Coloring Book*
by Dan Walsh and Mairin Duffy from RedHat

Applications live in containers.

# Cloud-native and Kubernetes (K8s)

When selecting a piggy apartment building, it's important to ensure that its infrastructure is compliant with common industry standards and policies.



What if your appliances run at a different voltage than what is provided in your new apartment? You may need to repurchase a number of expensive appliances (or rearchitect your applications).

If your furniture is too large (or too small), living in the apartment might require some amount of adjustment.

*The Container Coloring Book*
by Dan Walsh and Mairin Duffy from RedHat

Kubernetes automates the deployment, scaling, and management of containerized applications.

## What does a Kubernetes-native ML workflow look like?

## Argo Project

A set of Kubernetes-native tools for deploying and running applications and workloads on Kubernetes.

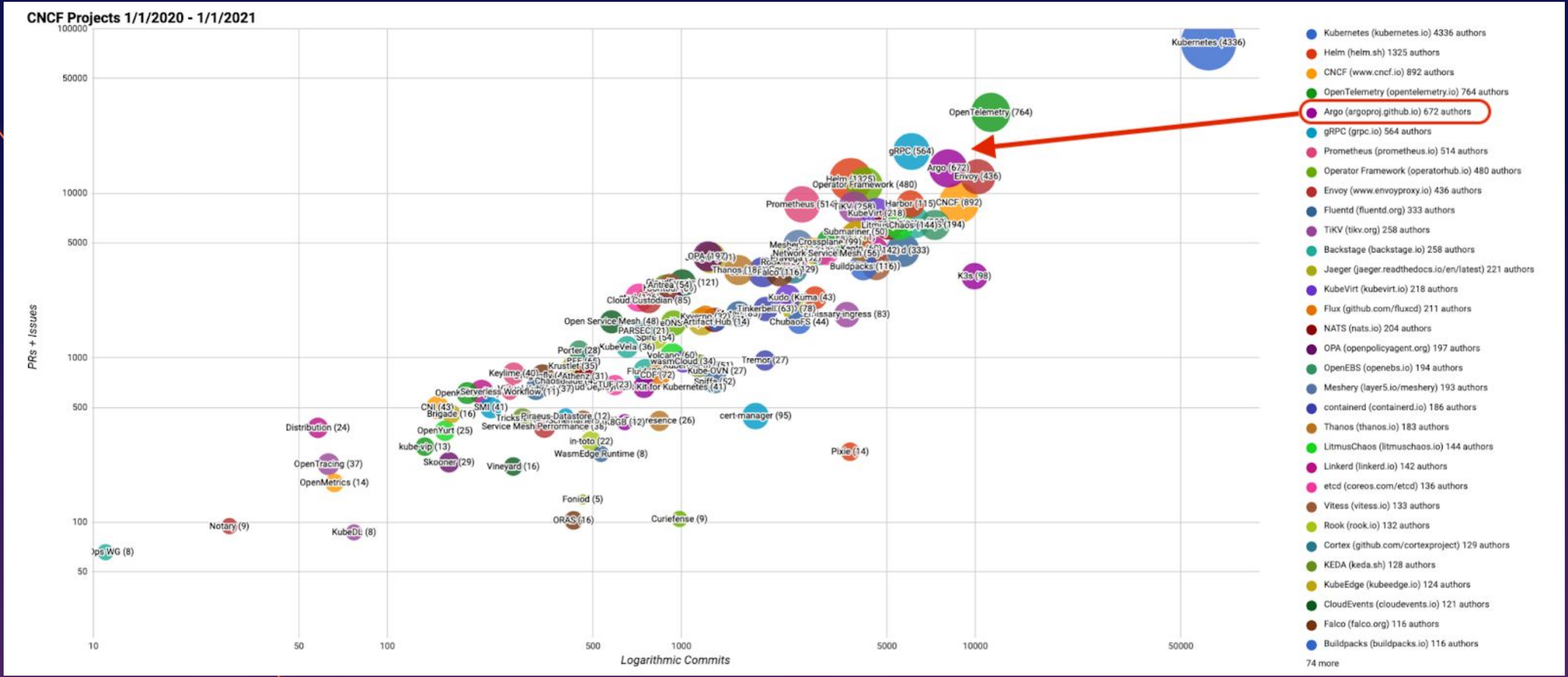
- Argo Workflows: Kubernetes-native workflow engine.
- Argo Events: Event-based dependency management for Kubernetes.
- Argo CD: Declarative continuous delivery with a fully-loaded UI.
- Argo Rollouts: Advanced K8s progressive deployment strategies.

Argo is awesome! https://github.com/terrytangyuan/awesome-argo

# Argo Project

180+ end user companies, 3k+ Slack members, 1k+ contributors, 20k+ GitHub stars

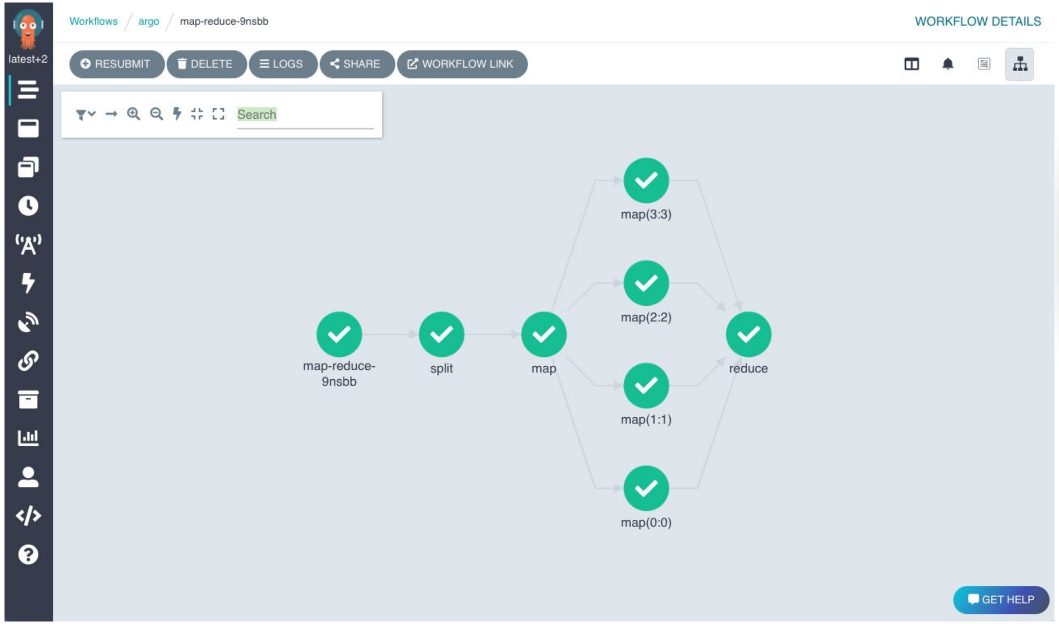
## Argo Project



[CNCF project rankings of developer velocity based on project activity]

# Argo Workflows

## The container-native workflow engine for Kubernetes



- Machine learning pipelines
- Data processing/ETL
- Infrastructure automation
- Continuous delivery/integration

## Argo Workflows

The container-native workflow engine for Kubernetes

CRDs and Controllers

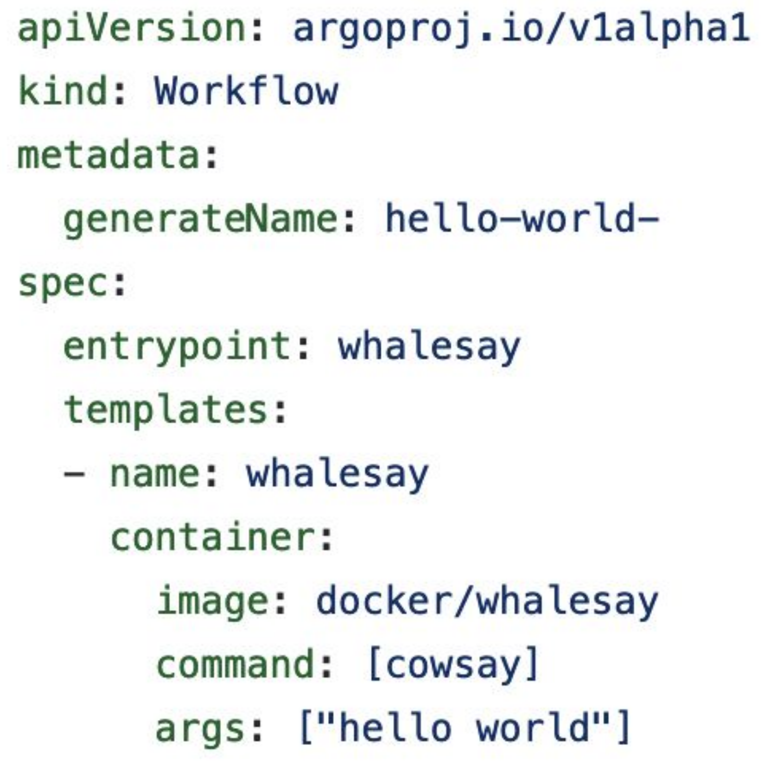
- Kubernetes custom resources that natively integrates with other K8s resources (volumes, secrets, etc.)

Interfaces

- CLI: manage workflows and perform operations (submit, suspend, delete/etc.)
- Server: REST & gRPC interfaces
- UI: manage and visualize workflows, artifacts, logs, resource usages analytics, etc.
- SDKs: Go, Python, and Java

# Argo Workflows

## Example: Hello World



```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-
spec:
  entrypoint: whalesay
  templates:
  - name: whalesay
    container:
      image: docker/whalesay
      command: [cowsay]
      args: ["hello world"]
```

## Argo Workflows

Example: Resource Template

```yaml
- name: k8s-owner-reference
  resource:
    action: create
    manifest: |
      apiVersion: v1
      kind: ConfigMap
      metadata:
        generateName: owned-eg-
      data:
        some: value
```

## Argo Workflows

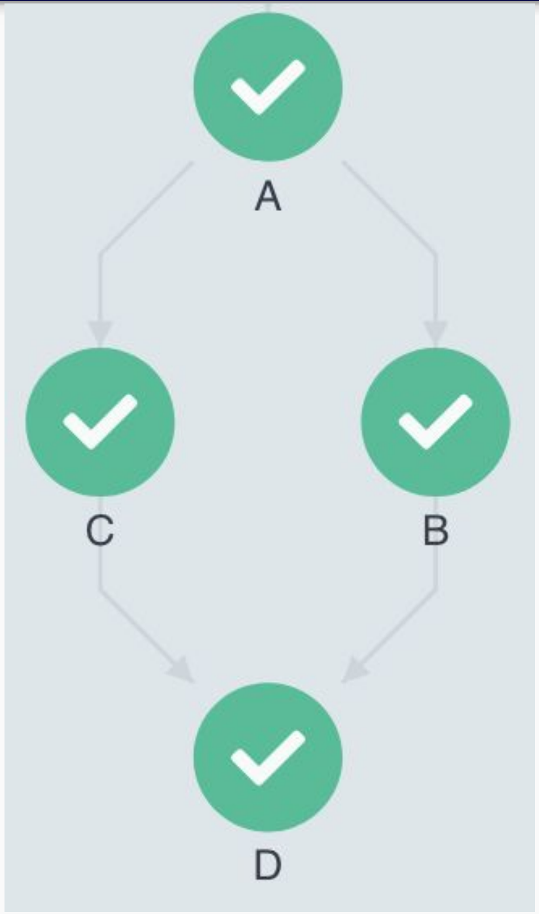
Example: Script Template

```
- name: gen-random-int
  script:
    image: python:alpine3.6
    command: [python]
    source: |
      import random
      i = random.randint(1, 100)
      print(i)
```

# Argo Workflows

## Example: DAG

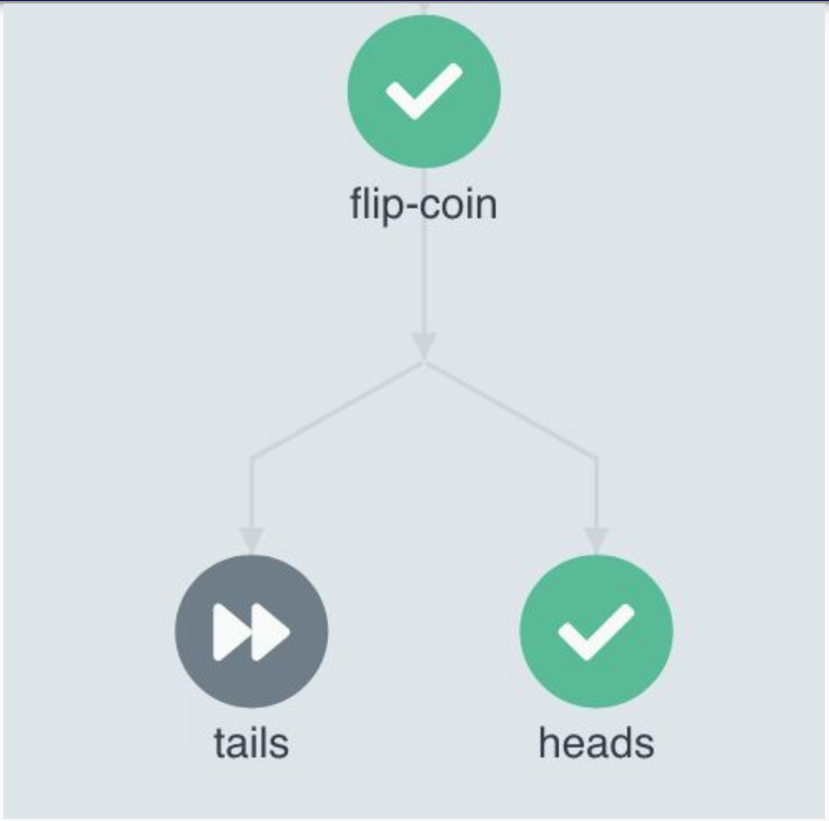
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: dag-diamond-
spec:
  entrypoint: diamond
  templates:
  - name: echo
    inputs:
      parameters:
      - name: message
    container:
      image: alpine:3.7
      command: [echo, "{{inputs.parameters.message}}"]
  - name: diamond
    dag:
      tasks:
      - name: A
        template: echo
        arguments:
          parameters: [{name: message, value: A}]
      - name: B
        dependencies: [A]
        template: echo
        arguments:
          parameters: [{name: message, value: B}]
      - name: C
        dependencies: [A]
        template: echo
        arguments:
          parameters: [{name: message, value: C}]
      - name: D
        dependencies: [B, C]
        template: echo
        arguments:
          parameters: [{name: message, value: D}]
```



# Argo Workflows

## Example: Coin-flip (conditional and step outputs)

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: coinflip-
spec:
  entrypoint: coinflip
  templates:
  - name: coinflip
    steps:
    - - name: flip-coin
        template: flip-coin
    - - name: heads
        template: heads
        when: "{{steps.flip-coin.outputs.result}} == heads"
      - name: tails
        template: tails
        when: "{{steps.flip-coin.outputs.result}} == tails"

  - name: flip-coin
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        result = "heads" if random.randint(0,1) == 0 else "tails"
        print(result)

  - name: heads
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was heads\""]

  - name: tails
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was tails\""]
```



## Can we do everything in Python?

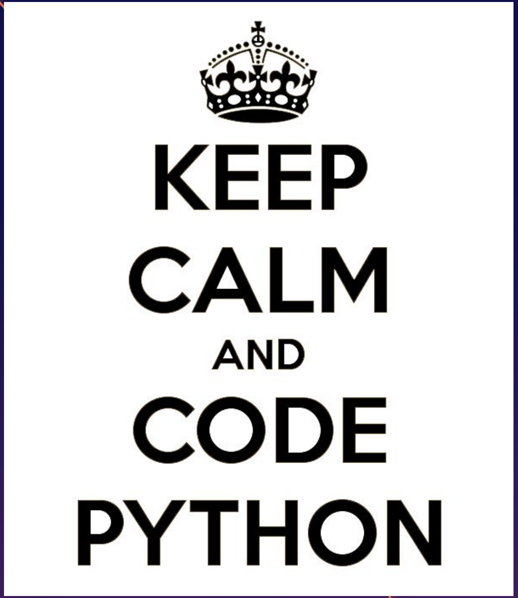


Image source

Kubeflow

Kubeflow Pipelines: Machine Learning Pipelines for Kubeflow

Couler

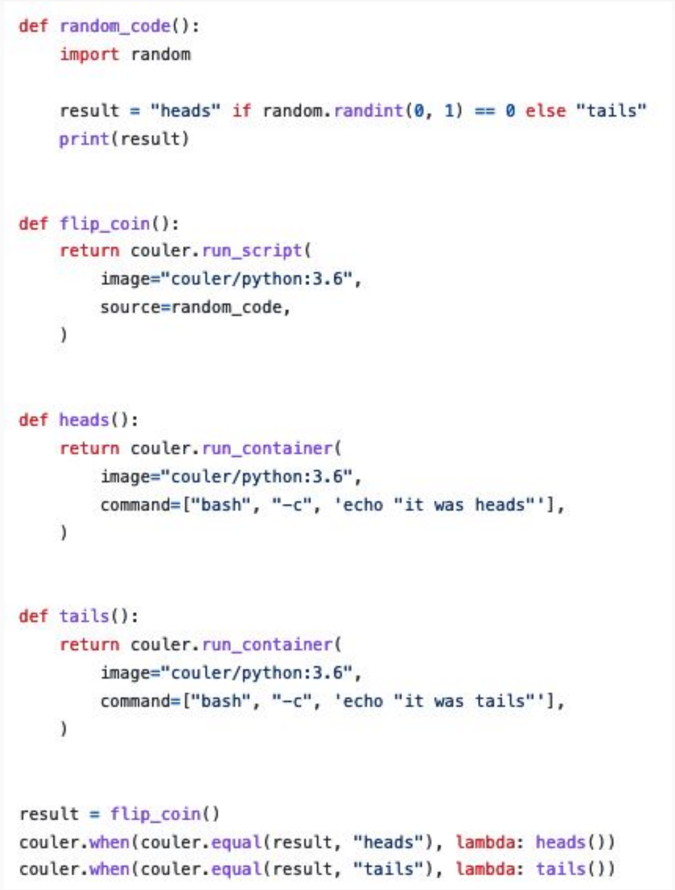
Couler: Unified Interface for Constructing and Managing Workflows

argo

Argo Workflows Officially Maintained Python SDK

Hera: Community Maintained High-level Python SDK

# Example: Coin-flip in Python



```python
def random_code():
    import random

    result = "heads" if random.randint(0, 1) == 0 else "tails"
    print(result)


def flip_coin():
    return couler.run_script(
        image="couler/python:3.6",
        source=random_code,
    )


def heads():
    return couler.run_container(
        image="couler/python:3.6",
        command=["bash", "-c", 'echo "it was heads"'],
    )


def tails():
    return couler.run_container(
        image="couler/python:3.6",
        command=["bash", "-c", 'echo "it was tails"'],
    )


result = flip_coin()
couler.when(couler.equal(result, "heads"), lambda: heads())
couler.when(couler.equal(result, "tails"), lambda: tails())
```

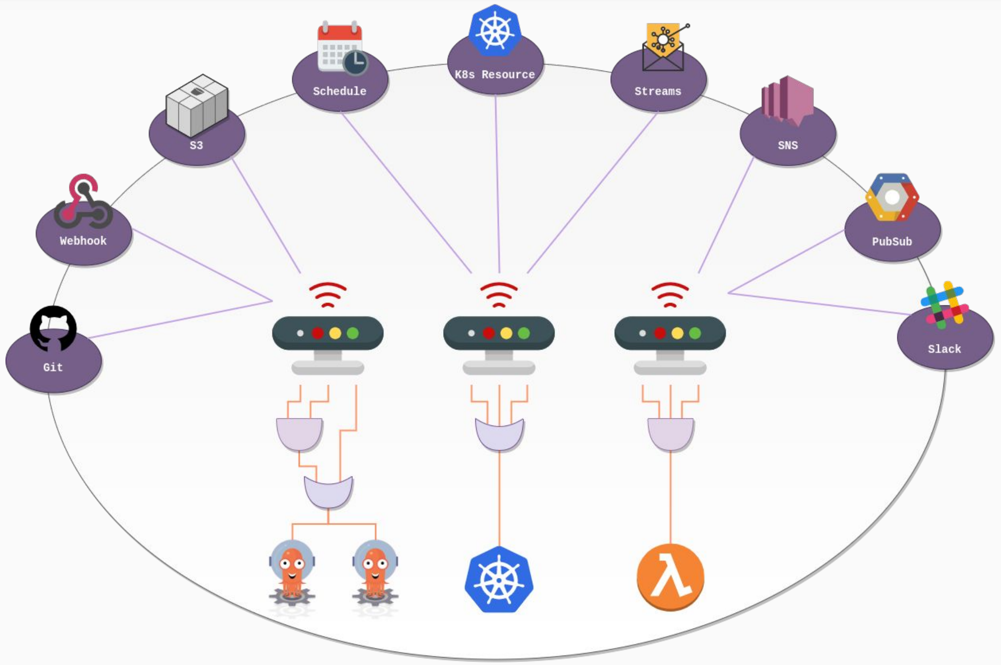
## Example: DAG in Python

```python
def job(name):
    couler.run_container(
        image="docker/whalesay:latest",
        command=["cowsay"],
        args=[name],
        step_name=name,
    )

#     A
#    / \
#   B   C
#  /
# D
def linear():
    couler.set_dependencies(lambda: job(name="A"), dependencies=None)
    couler.set_dependencies(lambda: job(name="B"), dependencies=["A"])
    couler.set_dependencies(lambda: job(name="C"), dependencies=["A"])
    couler.set_dependencies(lambda: job(name="D"), dependencies=["B"])

#   A
#  / \
# B   C
#  \ /
#   D
def diamond():
    couler.dag(
        [
            [lambda: job(name="A")],
            [lambda: job(name="A"), lambda: job(name="B")],  # A -> B
            [lambda: job(name="A"), lambda: job(name="C")],  # A -> C
            [lambda: job(name="B"), lambda: job(name="D")],  # B -> D
            [lambda: job(name="C"), lambda: job(name="D")],  # C -> D
        ]
    )
```

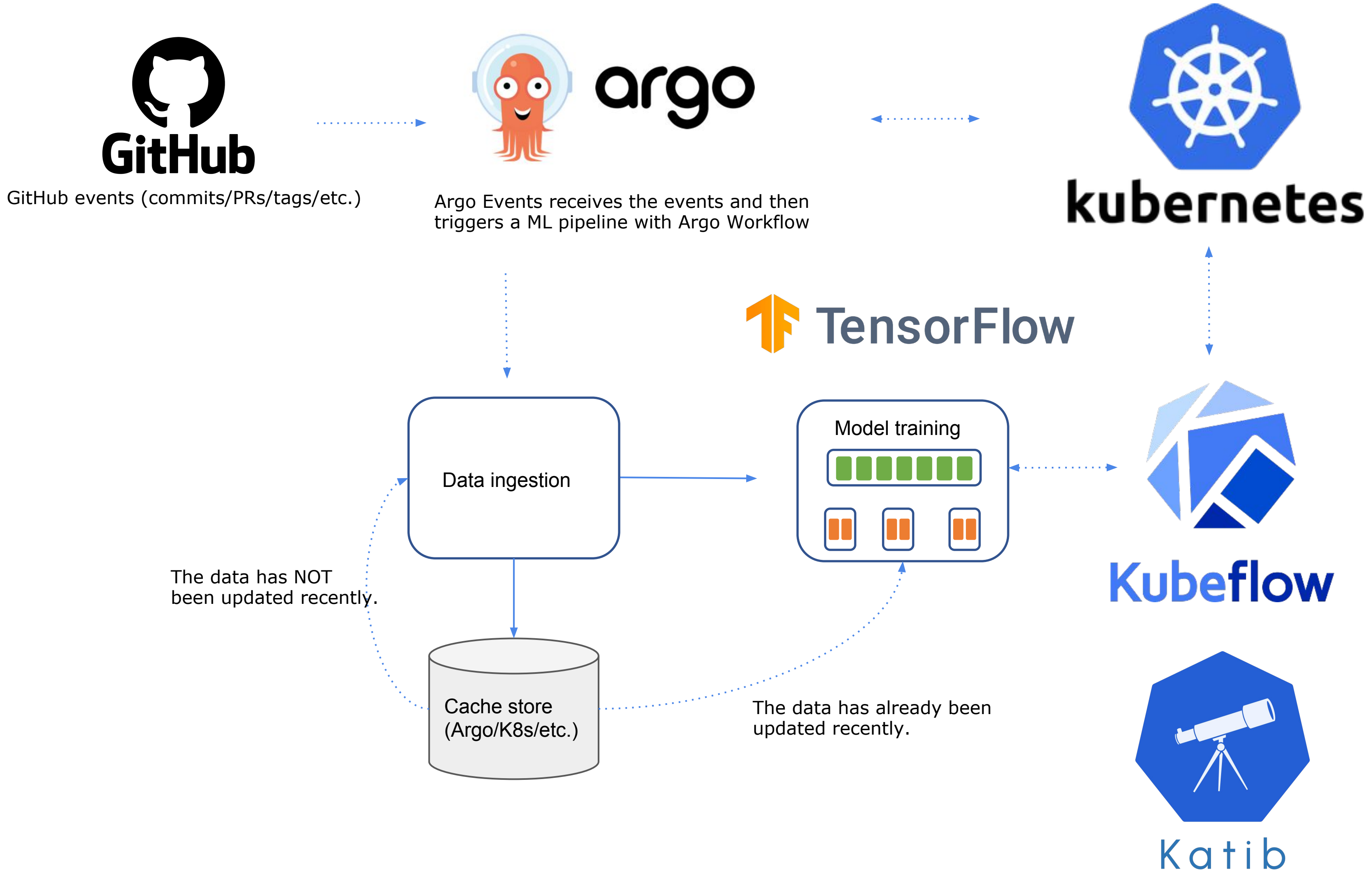
# Argo Events

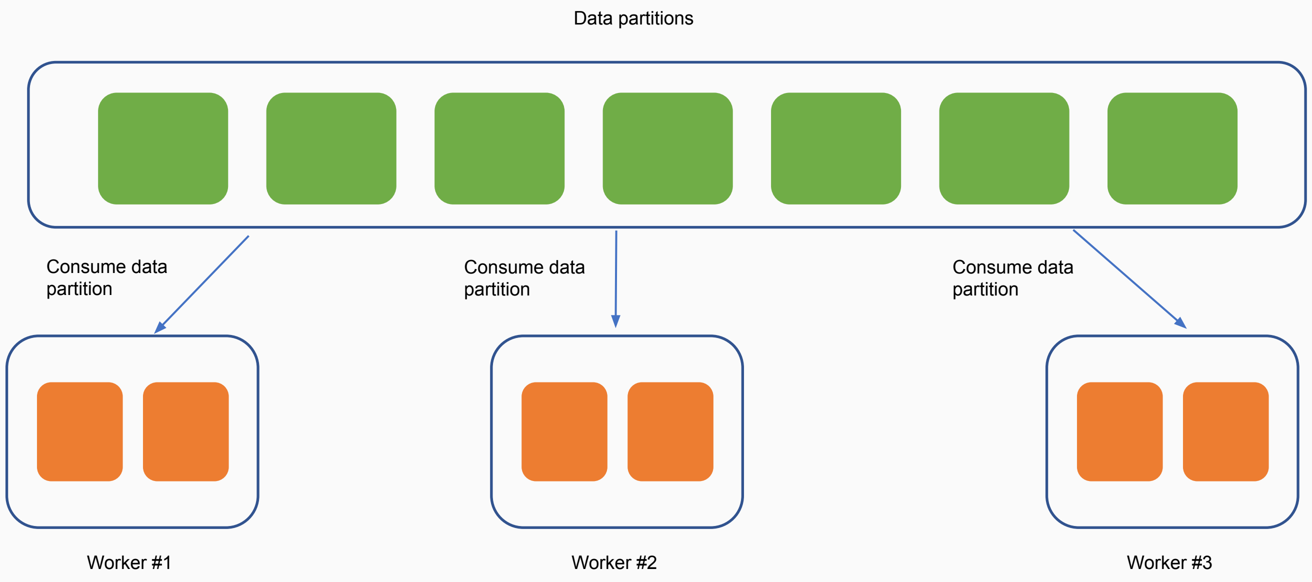
## The Event-driven Workflow Automation Framework

- Supports events from 20+ event sources
    - Webhooks, S3, GCP PubSub, Git, Slack, etc.
- Supports 10+ triggers
    - Kubernetes Objects, Argo Workflow, AWS Lambda, Kafka, Slack, etc.
- Manage everything from simple, linear, real-time to complex, multi-source events
- CloudEvents specification compliant



## What would a typical workflow look like with Argo Workflows + Events?


GitHub
GitHub events (commits/PRs/tags/etc.)
argo
Argo Events receives the events and then triggers a ML pipeline with Argo Workflow
kubernetes
TensorFlow
Data ingestion
Model training
Kubeflow
The data has NOT been updated recently.
Cache store (Argo/K8s/etc.)
The data has already been updated recently.
Katib



Distributed all-reduce model training with multiple workers and data partitions

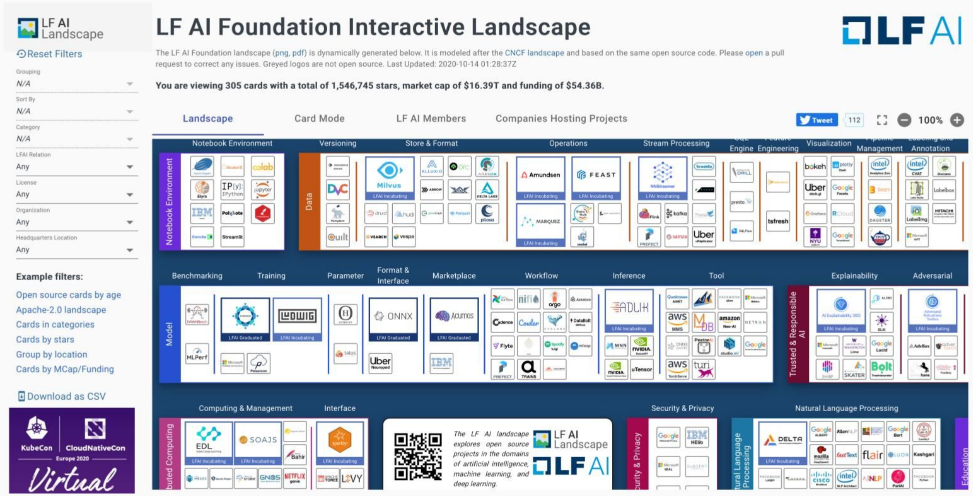
Source: Distributed Machine Learning Patterns

## Stronger Together:

## Cloud-native + Python Ecosystem

## Stronger Together: Future Outlook

- Focusing on developing tools that are most valuable for scientists
- Embracing Kubernetes ecosystem
  - Kubernetes-native operators and custom resources (e.g. Kubeflow, Argo Workflows)
  - Integration with Kubernetes (e.g. Dask/Ray/Spark on Kubernetes)
- Decoupled architecture
  - Infrastructure: MLOps, DevOps, DataOps
  - Frameworks: ML, DL, data visualization, scientific computing



LF AI Landscape

Reset Filters

Grouping: N/A
Sort By: N/A
Category: N/A
LFAI Relation: Any
License: Any
Organization: Any
Headquarters Location: Any

Example filters:
Open source cards by age
Apache-2.0 landscape
Cards in categories
Cards by stars
Group by location
Cards by MCap/Funding

Download as CSV

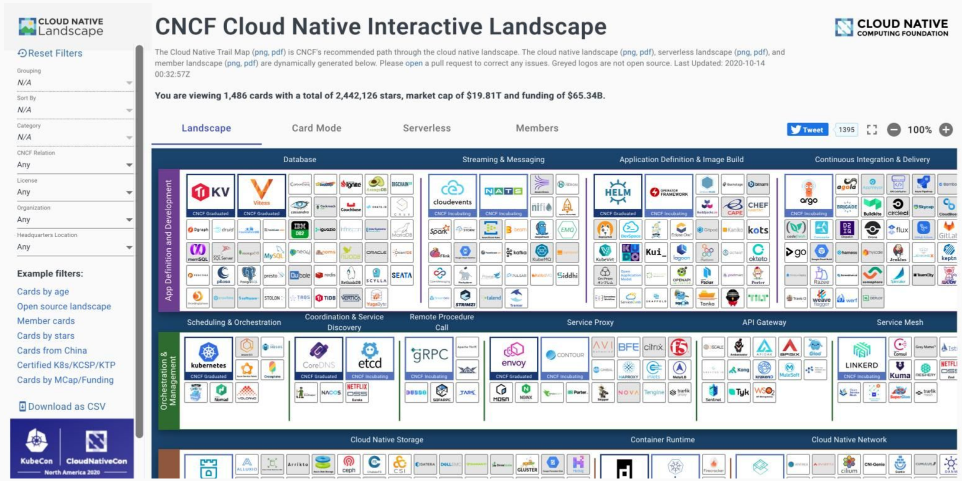
## LF AI Foundation Interactive Landscape

The LF AI Foundation landscape (png, pdf) is dynamically generated below. It is modeled after the CNCF landscape and based on the same open source code. Please open a pull request to correct any issues. Greyed logos are not open source. Last Updated: 2020-10-14 01:28:37Z

**You are viewing 305 cards with a total of 1,546,745 stars, market cap of $16.39T and funding of $54.36B.**

The LF AI landscape explores open source projects in the domains of artificial intelligence, machine learning, and deep learning.

[LF AI & Data Landscape](#)



CLOUD NATIVE Landscape

Reset Filters

Grouping: N/A
Sort By: N/A
Category: N/A
CNCF Relation: Any
License: Any
Organization: Any
Headquarters Location: Any

Example filters:
Cards by age
Open source landscape
Member cards
Cards by stars
Cards from China
Certified K8s/KCSP/KTP
Cards by MCap/Funding

Download as CSV

## CNCF Cloud Native Interactive Landscape

The Cloud Native Trail Map (png, pdf) is CNCF's recommended path through the cloud native landscape. The cloud native landscape (png, pdf), serverless landscape (png, pdf), and member landscape (png, pdf) are dynamically generated below. Please open a pull request to correct any issues. Greyed logos are not open source. Last Updated: 2020-10-14 00:32:57Z

**You are viewing 1,486 cards with a total of 2,442,126 stars, market cap of $19.81T and funding of $65.34B.**

CLOUD NATIVE COMPUTING FOUNDATION
Landscape
Card Mode
Serverless
Members
Tweet
1395
100%
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
App Definition and Development
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Orchestration & Management
Cloud Native Storage
Container Runtime
Cloud Native Network
KubeCon CloudNativeCon North America 2020

[CNCF Cloud Native Interactive Landscape](#)

## Thank you and keep in touch!

- **Email**: terrytangyuan@gmail.com
- **Twitter/LinkedIn/GitHub/Slack**: @TerryTangYuan
- **Argo community**: https://argoproj.github.io/community/join-slack
- **Kubeflow community**: https://www.kubeflow.org/docs/about/community/