Data Science in the Cloud-Native Era

Yuan Tang (@TerryTangYuan), Akuity Inc. (akuity.io)

About me

- Founding Engineer at akuity.io (the enterprise company for Argo)
- Maintainer/Committer:
  - ML Frameworks: XGBoost, TensorFlow, Apache MXNet, metric-learn, 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/GitHub/LinkedIn: @TerryTangYuan
  - Open source and collaboration: https://calendly.com/chat-with-terry/

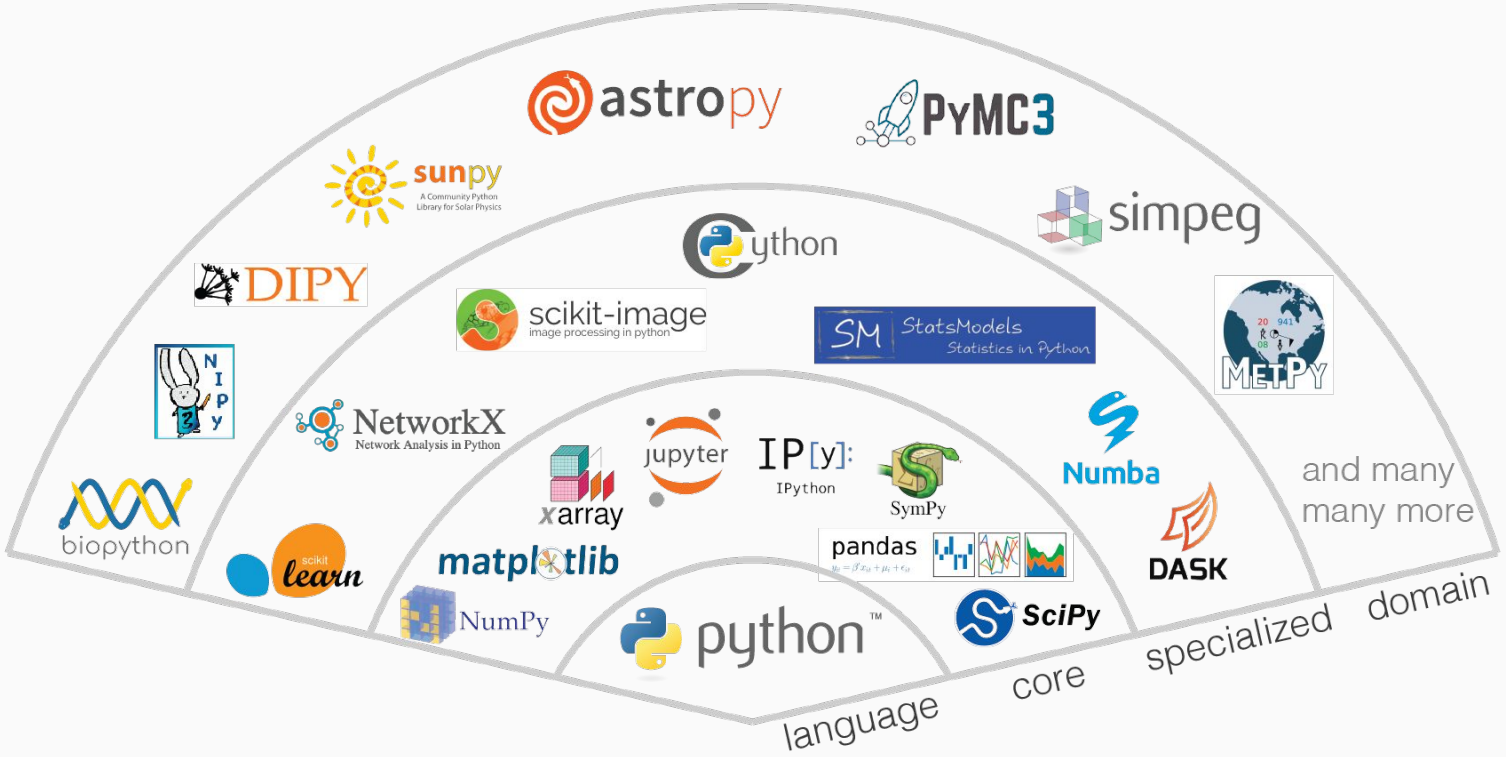
Agenda

- Components of distributed ML pipelines
- Python scientific ecosystem
- Machine learning frameworks
- Workflow orchestration tools
- Cloud-native and Kubernetes
- Kubernetes-native ML pipelines
- 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

Machine Learning Frameworks

Workflow Orchestration Tools (Python)

Apache Airflow

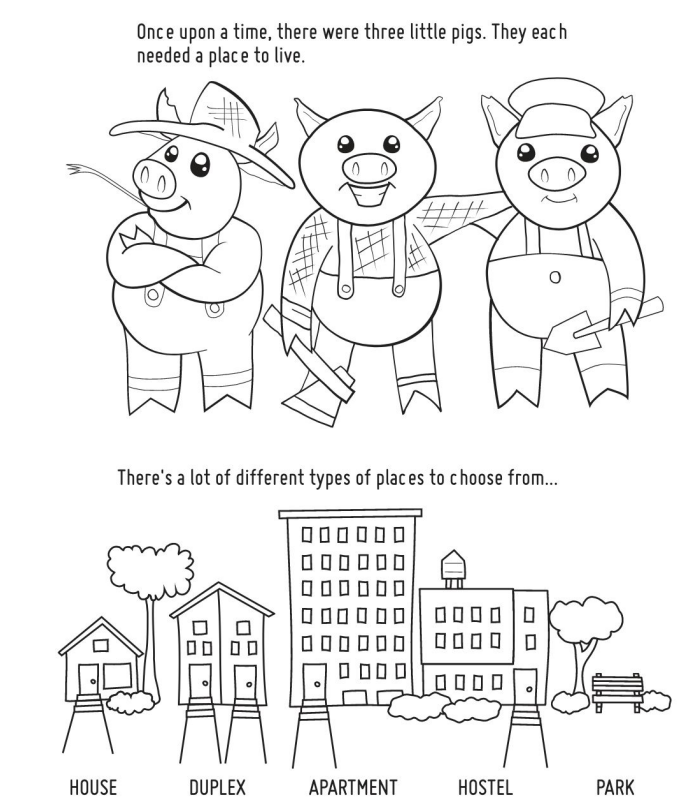
PREFECT

METAFLOW

argo

What is cloud-native and Kubernetes?

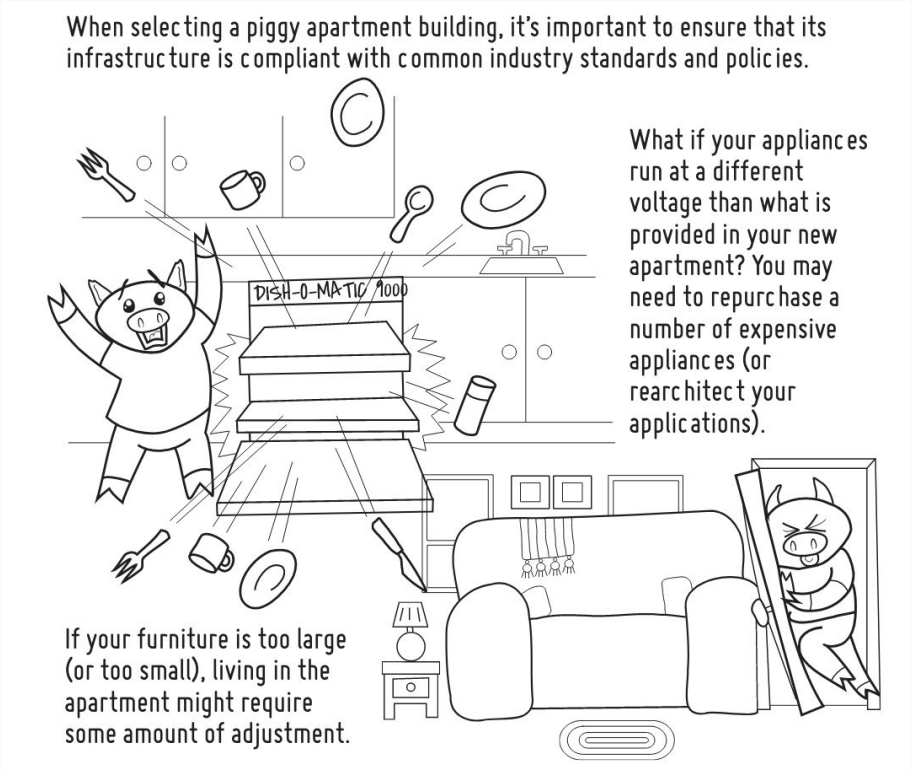
Cloud-native and Kubernetes



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

Applications live in containers.

Cloud-native and Kubernetes



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, managing clusters, and do GitOps right.

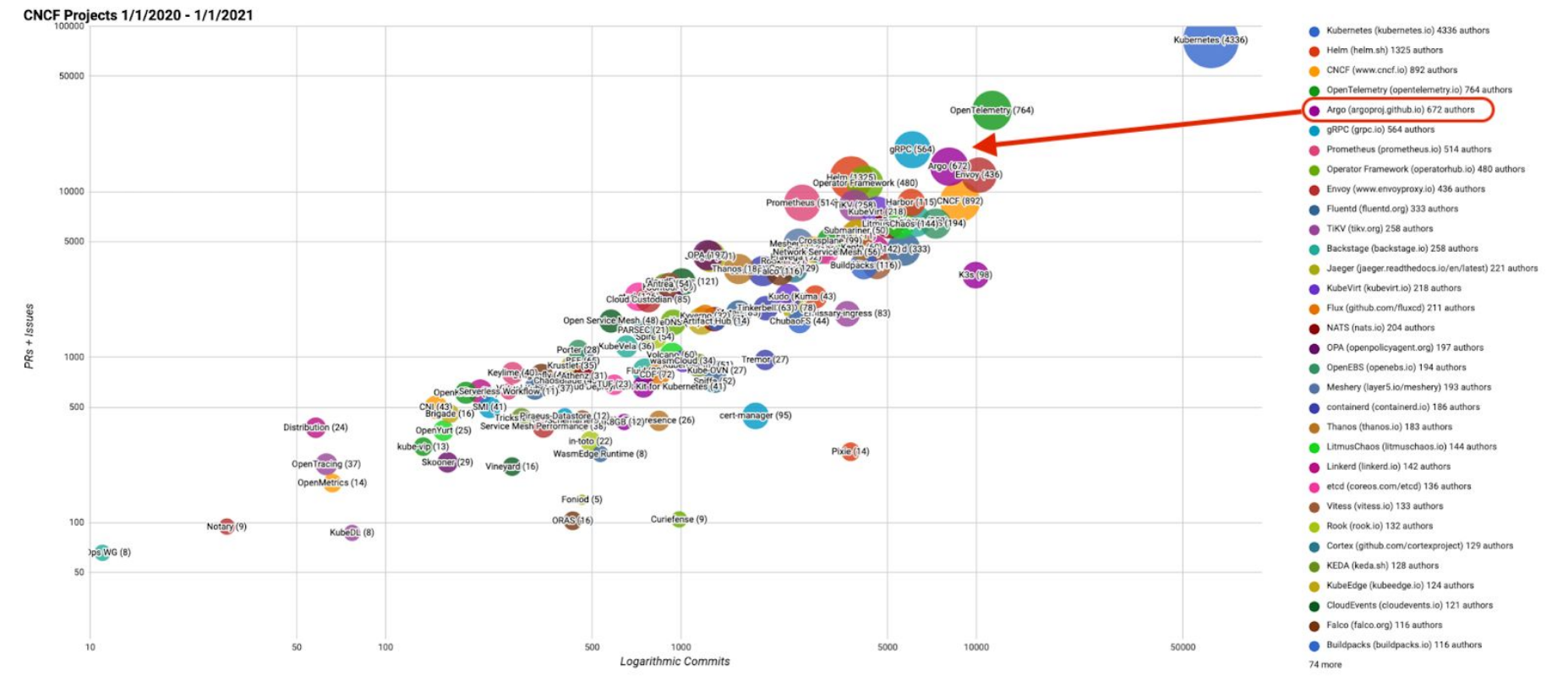
- Argo Workflows: Kubernetes-native workflow engine.
- Argo Events: Event-based dependency management for Kubernetes.
- Argo CD: Declarative continuous delivery with a fully-loaded UI.
- Argo Rollouts: Advanced K8s progressive deployment strategies.

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

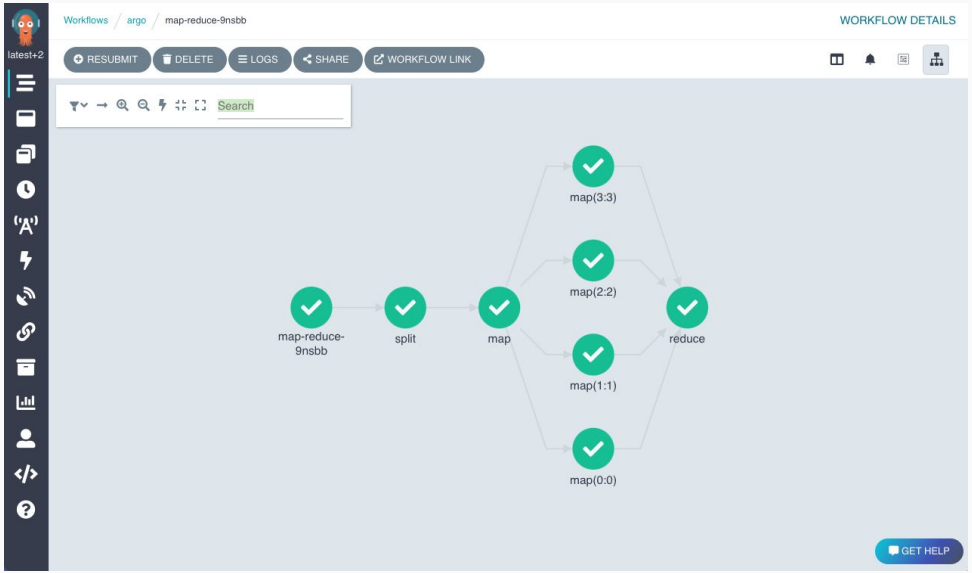


[CNCF project rankings of developer velocity based on project activity]

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.
- Python and Java SDKs

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"]
```

Example: Resource Template

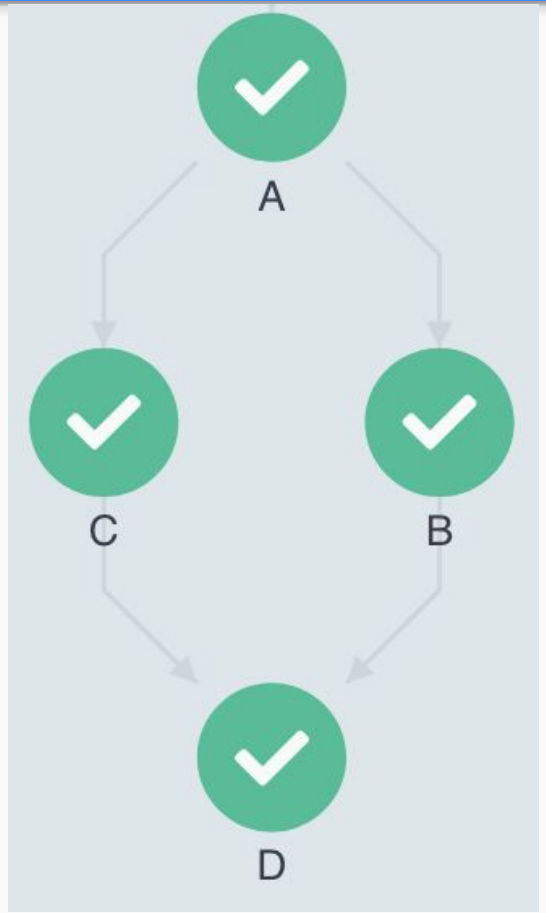
```
- name: k8s-owner-reference
  resource:
    action: create
    manifest: |
      apiVersion: v1
      kind: ConfigMap
      metadata:
        generateName: owned-eg-
      data:
        some: value
```

Example: Script Template

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

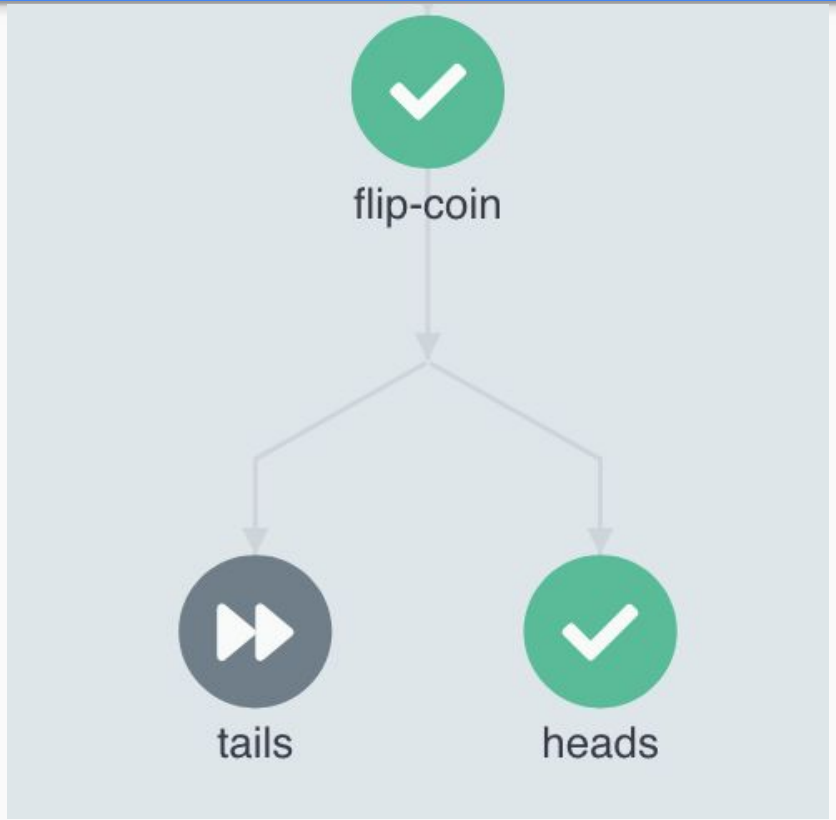
Example: DAG

```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}]
```

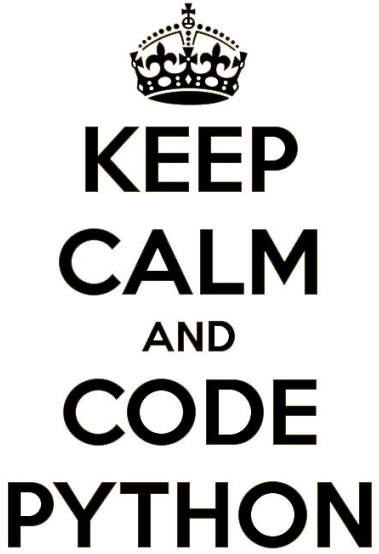


Example: Coin-flip (conditional and step outputs)

```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?



Kubeflow

kubeflow/pipelines: Machine Learning Pipelines for Kubeflow

Couler

couler-proj/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

```
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())
```

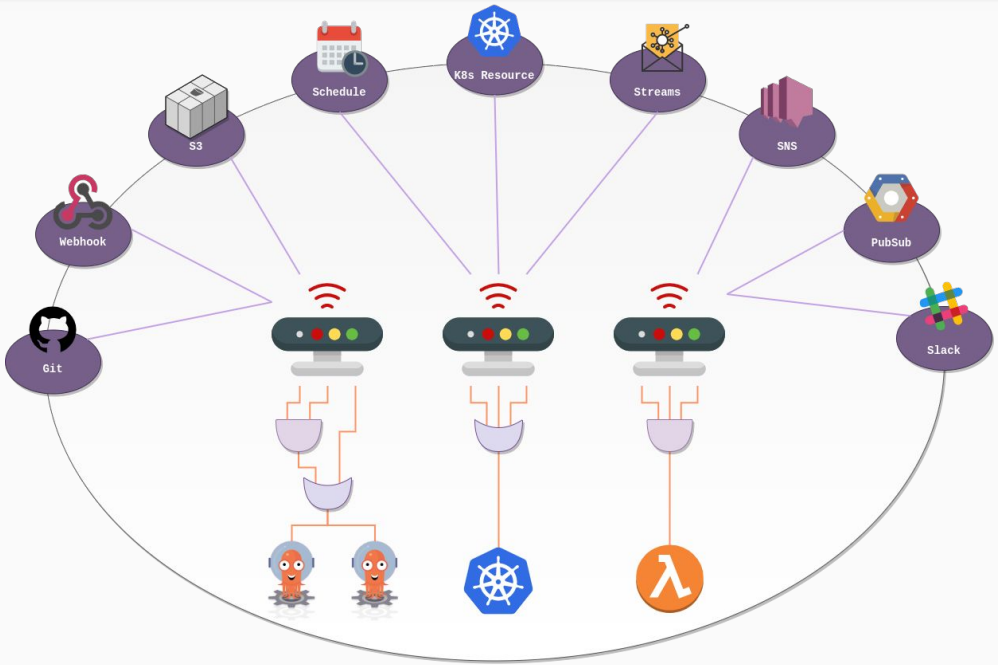
Example: DAG in Python

```
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
        ]
    )
```

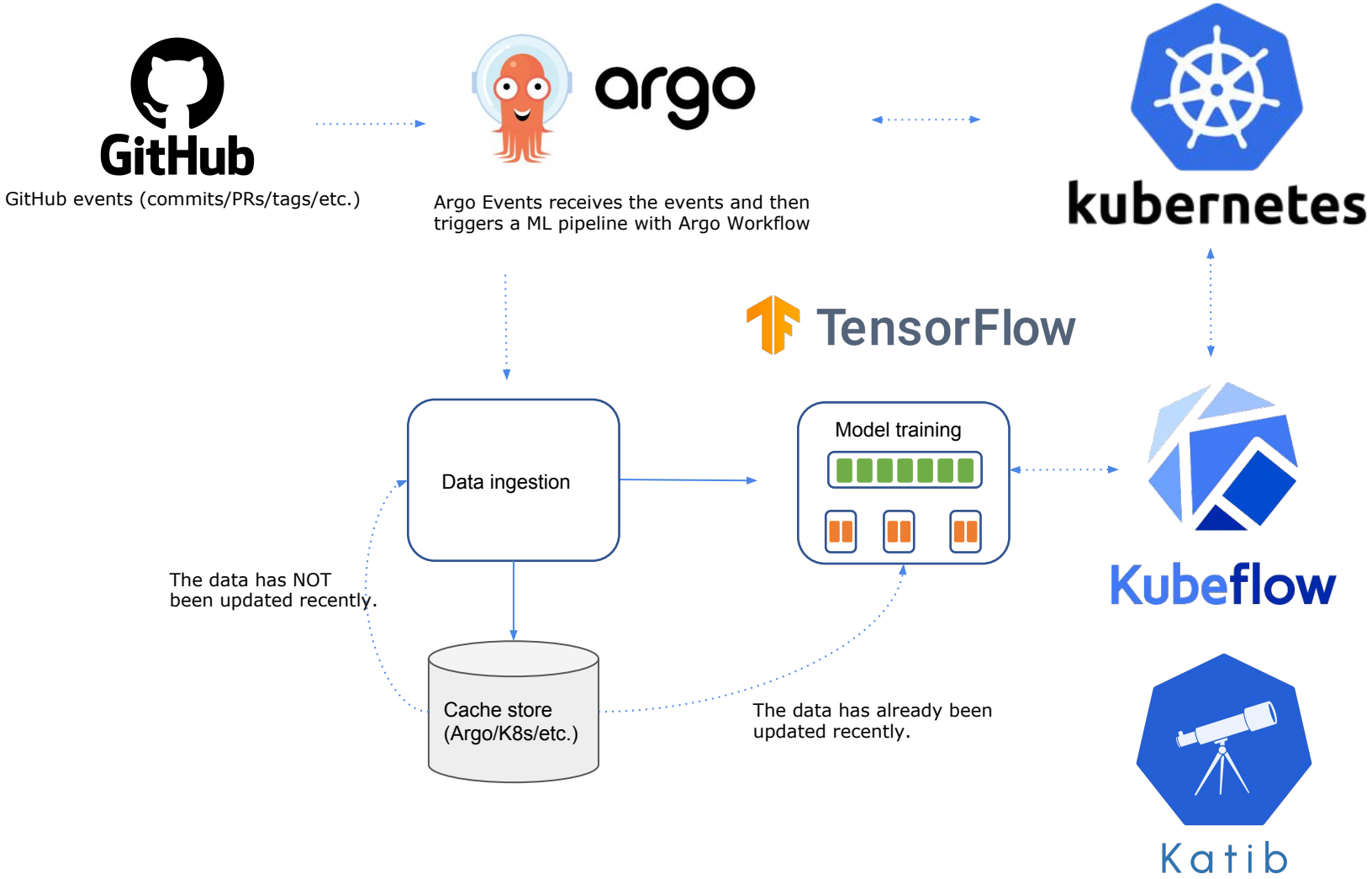
Argo Events

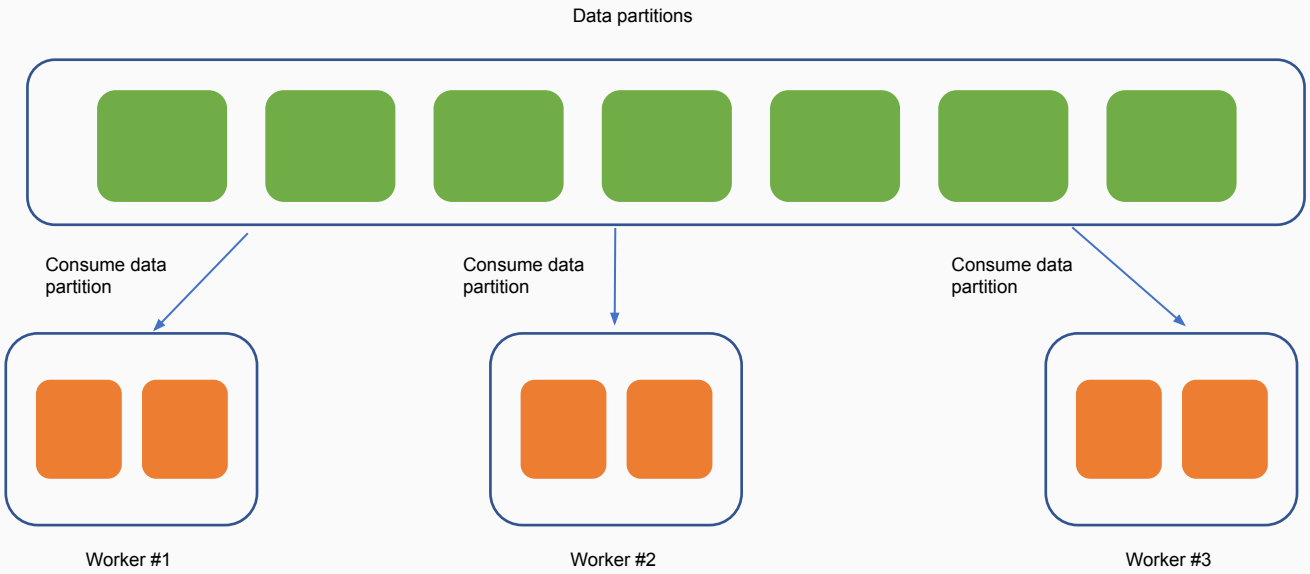
The Event-driven Workflow Automation Framework

- 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

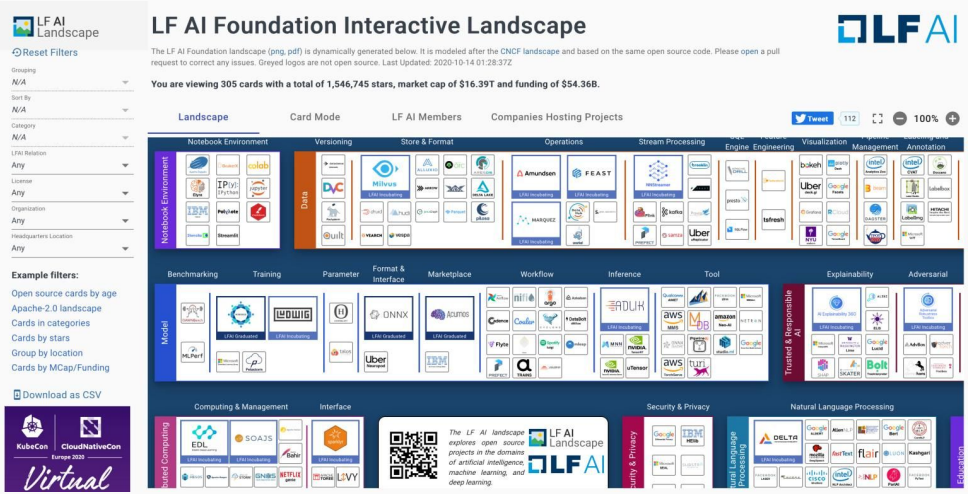


Distributed model training with multiple workers and data partitions

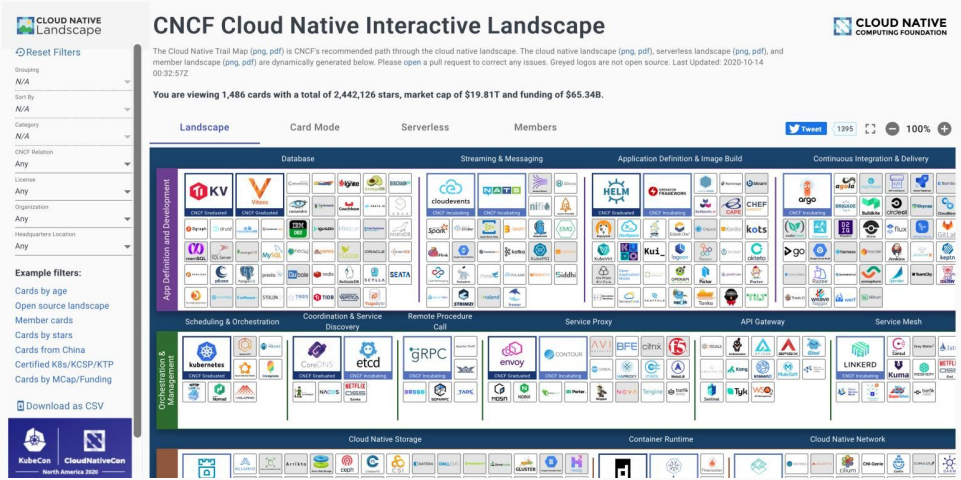
Source: Distributed Machine Learning Patterns

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 & Data Landscape]



[CNCF Cloud Native Interactive Landscape]

Contact

- Email: terrytangyuan@gmail.com
- Twitter/LinkedIn/GitHub/Slack: @TerryTangYuan
- Open source and collaboration: https://calendly.com/chat-with-terry/
- Argo community: https://argoproj.github.io/community/join-slack
- Kubeflow community: https://www.kubeflow.org/docs/about/community/