

Why is Argo different from the others? Budapest Data Forum

Riccardo Mocchetti - June 2022



What this presentation is <u>not about</u>

- Is Argo the best workflow manager out there? \bullet
- Pros and Cons of different orchestrators



Context

- Data pipelines with a software engineering mindset
- Everything is code
- Automated processes for testing and releasing to production
- Increased reliability



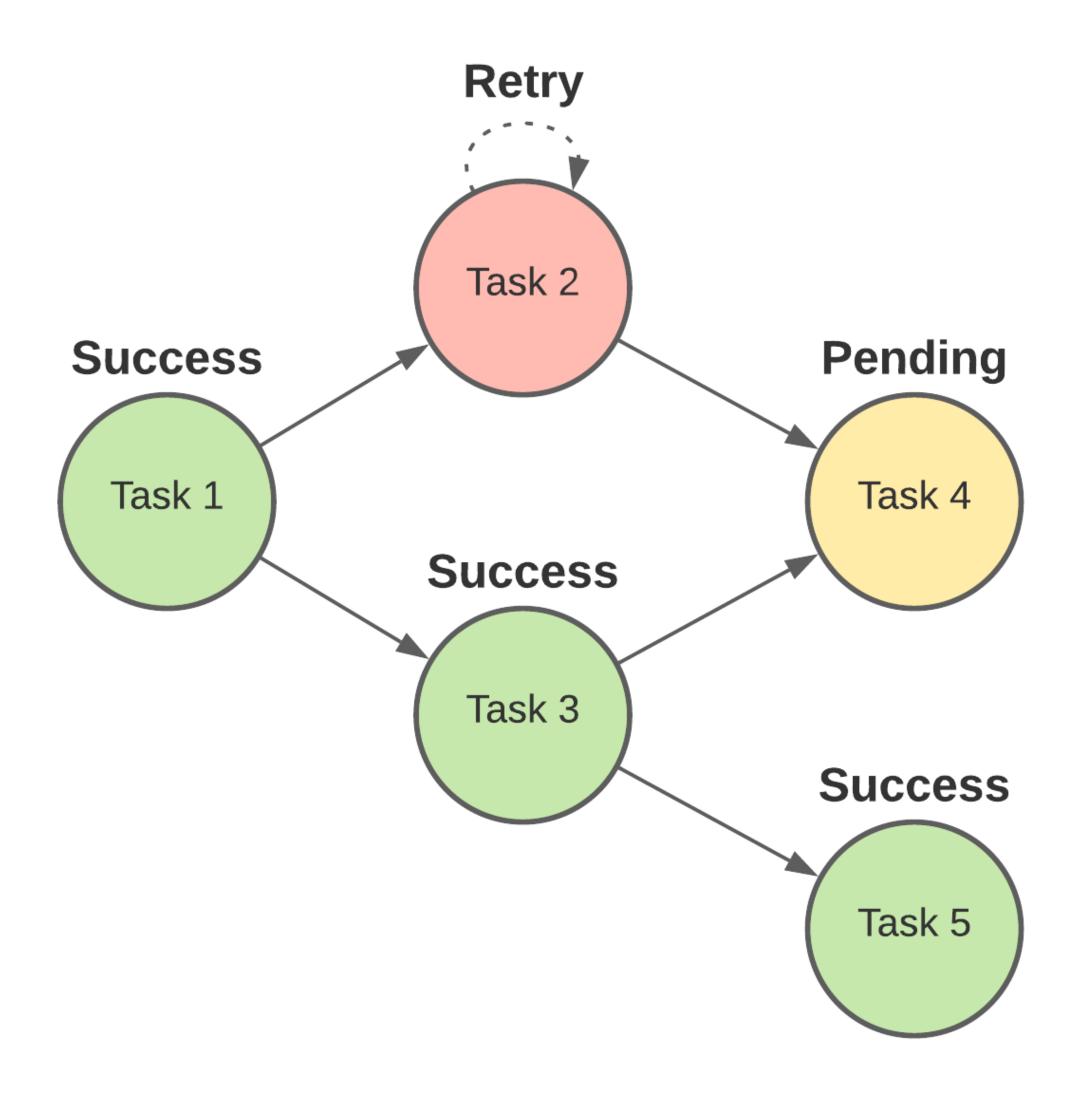
Agenda

- "Workflows as code" orchestrators
- Thinking at different abstraction levels
- Container-Native Orchestration
- Event based workflows











Airflow

•••

```
# Defining a dag
@dag(start_date=pendulum.datetime(2021, 1, 1, tz="UTC"),
    schedule_interval="@daily", catchup=False)
def generate_dag():
    op = EmptyOperator(task_id="task")
dag = generate_dag()
# Defining dependencies
first_task >> [second_task, third_task]
third_task << fourth_task</pre>
```

https://airflow.apache.org/docs/apache-airflow/stable/concepts/dags.html

- Designed for long running tasks
- Everything goes through the scheduler (and the database)
- Airflow workers execute python



Prefect

•••

```
from prefect import task, Task, Flow
import random
```

```
@task
def random_number():
    return random.randint(0, 100)
```

```
@task
def plus_one(x):
    return x + 1
```

```
with Flow('My Functional Flow') as flow:
    r = random_number()
    y = plus_one(x=r)
```

https://docs.prefect.io/core/concepts/flows.html#functional-api

- Designed for short lived tasks
- Passing state between tasks and dynamic task generation
- Flows can be run in isolation



DBT

```
select
orders.id,
orders.status,
sum(case when payments.payment_method = 'bank_transfer' then
payments.amount else 0 end) as bank_transfer_amount,
sum(case when payments.payment_method = 'credit_card' then
payments.amount else 0 end) as credit_card_amount,
sum(case when payments.payment_method = 'gift_card' then
payments.amount else 0 end) as gift_card_amount,
sum(amount) as total_amount
from {{ ref('base_orders') }} as orders
left join {{ ref('base_payments') }} as payments on payments.order_id
= orders.id
```

```
https://docs.getdbt.com/docs/introduction
```

- Each select statement becomes a model (task)
- DAGs are defined implicitly
- Limited by SQL



| | | | |
|---------------------------------------|--|-------------------------------|---|
| stg_asana_projects_companies | | | |
| stg_asana_projects_projects | | | |
| stg_harvest_projects_companies | stg_hubspot_crm_owners | | |
| stg_facebook_ads_ad_activity_snapshot | stg_hubspot_crm_pipeline_stages | stg_hubspot_crm_deals | |
| stg_hubspot_crm_companies | stg_hubspot_crm_pipelines | | |
| stg_jira_projects_companies | | | |
| stg_stripe_payments_companies | int_companies_pre_merged | | |
| stg_google_ads_accounts | | stg_jira_projects_projects | |
| stg_google_ads_campaigns | | | |
| stg_xero_accounting_companies | | | |
| companies_merge_list | | int_companies | |
| | | | |
| | | stg_harvest_projects_projects | |
| | | | |
| | | stg_asana_projects_users | |
| | | stg_harvest_projects_users - | |
| | | stg_jira_projects_users | |
| | | stg_unknown_users | |
| | stg_facebook_ads_ad_performance_snapshot | int_ad_performance_snapshot | |
| stg_facebook_ads_campaigns | int_ad_campaigns | wh_ad_campaigns_dim | |
| stg_facebook_ads_ads | int_ads | wh_ads_dim | |
| stg_intercom_messaging_contacts | | | w |
| stg_intercom_messaging_conversations | | | |
| stg_facebook_ads_adsets | int_adsets | wh_adsets_dim | |
| | stg_mixpanel_events_events | | |
| stg_mixpanel_events_all_events | stg_mixpanel_events_pageviews | | |
| stg_harvest_projects_contacts | stg_segment_events_events | int_web_events | |
| project_merge_list | stg_segment_events_pageviews | | |
| | stg_mailchimp_email_events | int_email_send_outcomes | |
| stg_hubspot_crm_contacts | int_contacts | wh_contacts_dim | |
| stg_mailchimp_email_contacts | | | |
| stg_xero_accounting_contacts | | | |
| stg_stripe_payments_charges | | | |
| stg_xero_accounting_accounts | int_chart_of_accounts | wh_chart_of_accounts_dim | |
| stg_stripe_payments_payouts | | | |
| stg_mailchimp_email_lists | int_email_lists | wh_email_lists_dim | |
| stg_xero_accounting_currencies | int_currencies | wh_currencies_dim | |
| stg_asana_projects_tasks | | | |
| stg_jira_projects_tasks | int_delivery_tasks | wh_delivery_tasks_dim | |
| stg_mailchimp_email_sends | int_email_sends | wh_email_sends_dim | |
| stg_xero_accounting_payments | | stg_harvest_projects_tasks | |
| stg_stripe_payments_transactions | | | |
| audit_dbt_last_results | int_transactions | wh_transactions_fact | |
| stg_xero_accounting_transactions | | | |
| | | | |

https://rittmananalytics.com/blog/2020/5/28/introducing-the-ra-warehouse-dbt-framework-how-rittman-analytics-does-data-centralization

| | referrer_mapping Int_web_events_sessions_initial |
|---------------------------------|--|
| stg_harvest_projects_invoices | int_invoices |
| stg_xero_accounting_invoices | |
| int_deals | |
| | |
| | wh_deals_fact |
| | wh_invoices_fact |
| int_delivery_projects | wh_delivery_projects_dim |
| | |
| wh_companies_dim | |
| stg_harvest_projects_timesheets | Int_timesheets |
| int_timesheet_projects | wh_timesheet_projects_dim int_web_events_sessions_stitched |
| | wh_web_sessions_fact |
| | |
| int_users | wh_users_dim |
| | |
| | |
| | |
| | |
| wh_ad_performance_snapshot_fact | |
| | |
| | |
| | |
| | int_web_events_sessionized wh_timesheets_fact |
| | wh_web_events_fact |
| | |
| | |
| wh_email_send_outcomes_fact | |
| | wh_web_pages_dim |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| int_timesheet_tasks | wh_timesheet_tasks_dim |
| | |
| | |
| int_web_events_user_stitching | |





Thinking at a different abstraction level Difference #1

Argo Workflows

•••

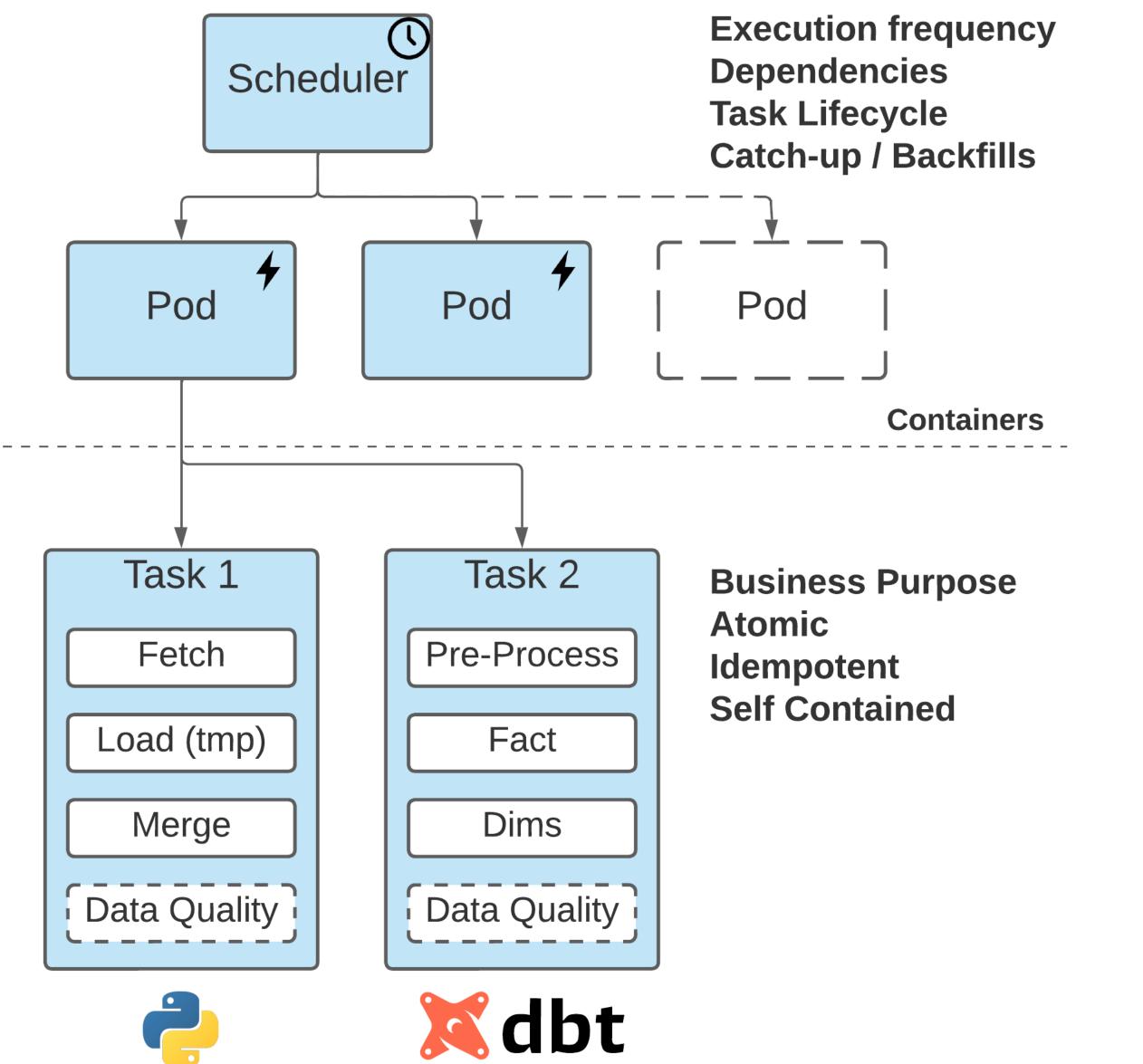
```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
   generateName: data-ingestion-
spec:
   entrypoint: data-ingestion
   templates:
        - name: data-ingestion
        container:
        image: my-company/data-ingestion:1.0.1
        command: [python]
        args: ["run.py"]
```

- Containers are the main building blocks
- Containers can be released in isolation
- Encourages to think about business processes



Orchestrator

Tasks









Container-Native Orchestration Difference #2

Argo Workflows

Argo Workflows is an open source container-native workflow engine for orchestrating parallel jobs on Kubernetes. Argo Workflows is implemented as a Kubernetes CRD (Custom Resource Definition).

https://argoproj.github.io/argo-workflows/#argo-workflows



A little bit about Kubernetes

- Kubernetes is really good at executing long running applications
- Kubernetes is open source, every major cloud supports running Kubernetes as a service
- Kubernetes is highly extensible, one of the reason why it is so popular
- Custom Resource Definitions extend Kubernetes with new resources



Kubernetes Resources

•••

apiVersion: v1 kind: Pod metadata: name: nginx spec: containers:

> - name: nginx image: nginx:1.14.2 ports:

- containerPort: 80





Kubernetes Resources

•••

apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:

- name: nginx image: nginx:1.14.2 ports:

- containerPort: 80

•••

```
apiVersion: argoproj.io/vlalpha1
kind: Workflow
metadata:
   generateName: data-ingestion-
spec:
   entrypoint: data-ingestion
   templates:
   - name: data-ingestion
      container:
      image: my-company/data-ingestion:1.0.1
      command: [python]
      args: ["run.py"]
```





Argo as a CRD

- Argo can leverage all Kubernetes features
 - Install/run
 - High availability and scaling
 - Security and access control
- Can be managed as any other Kubernetes resource
- Cannot be executed outside Kubernetes





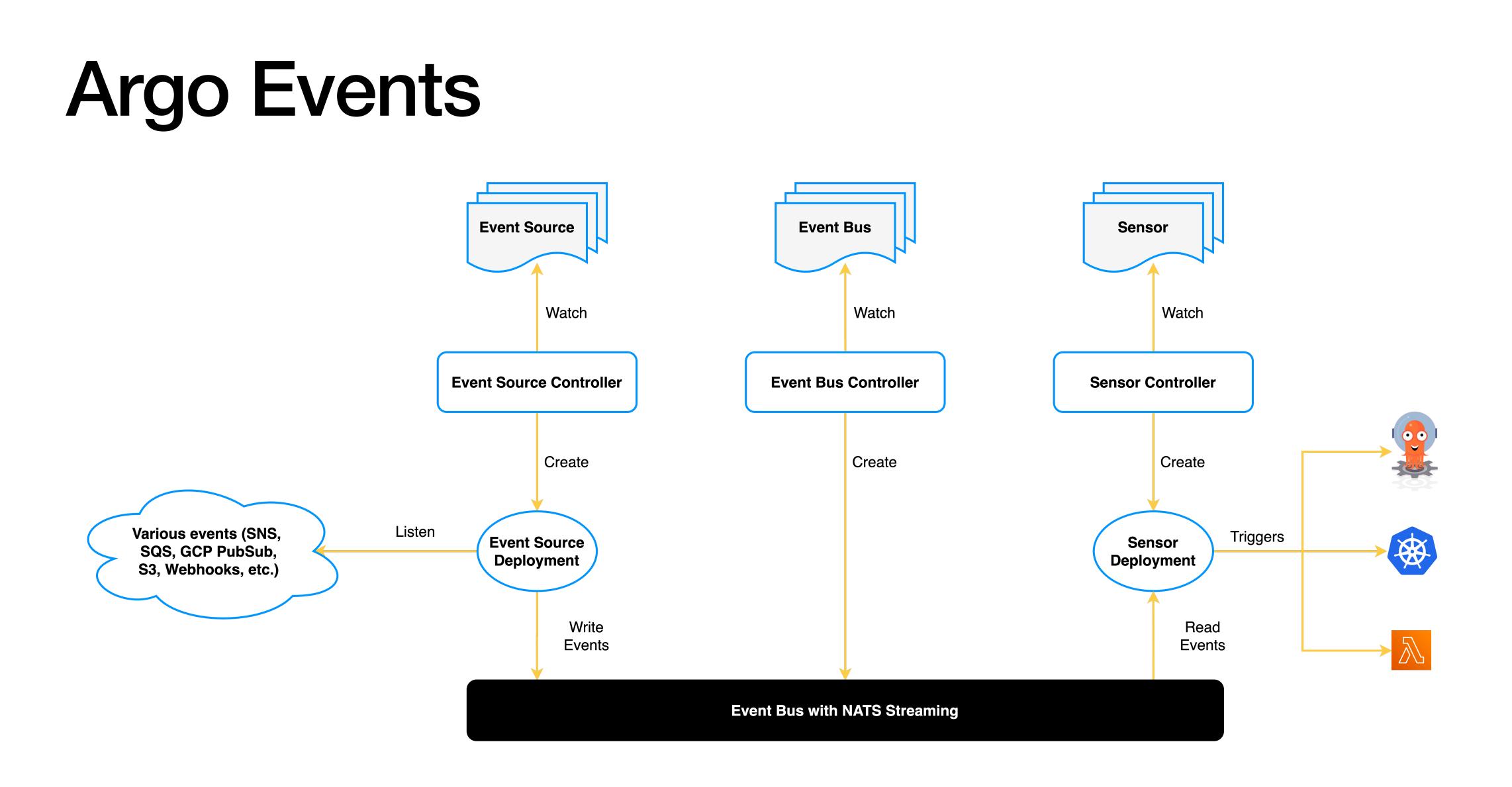
Event Based Workflows Difference #3



DAGs so far

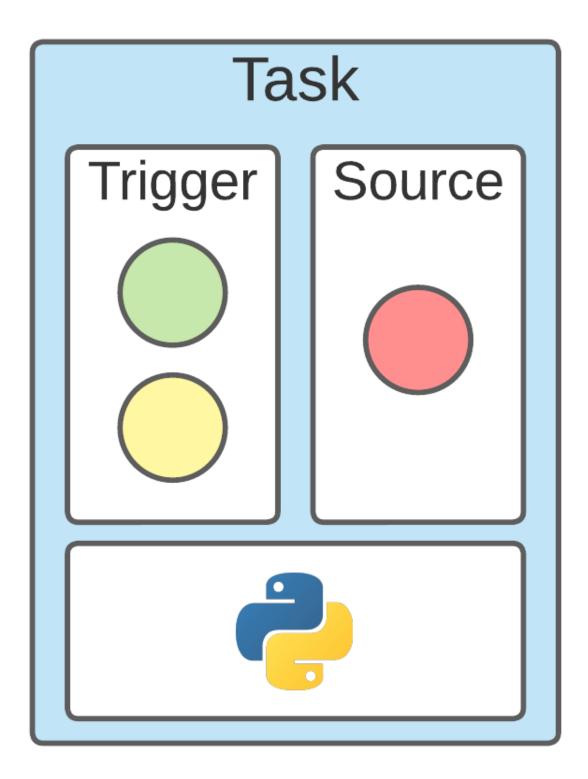
- DAGs are hardcoded in some kind of specification
- Every change requires a new release
- Difficult to model







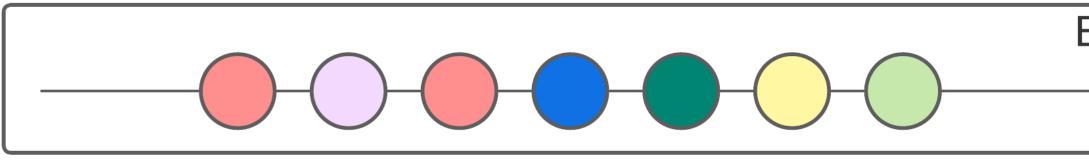
Argo Events - Single Task

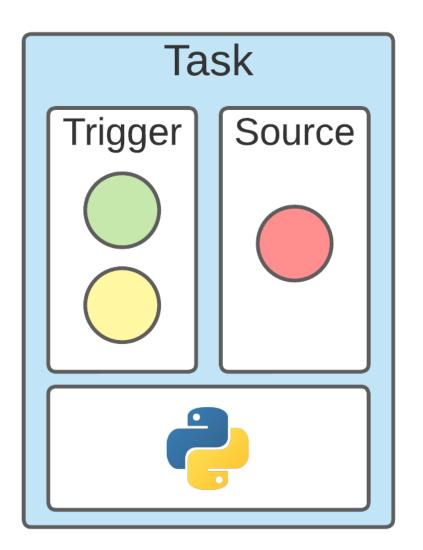


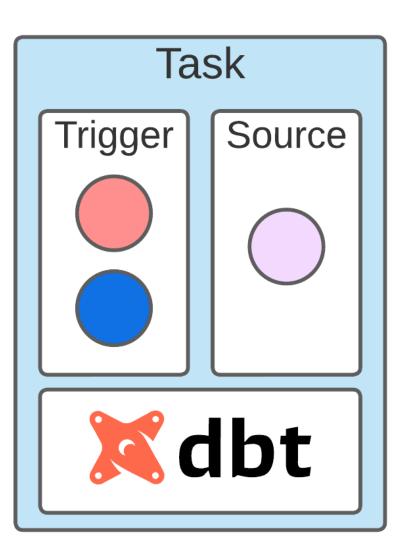
- Business purpose
- Atomic
- Idempotent
- Self-contained
- Triggered by green and yellow messages
- Emits red messages

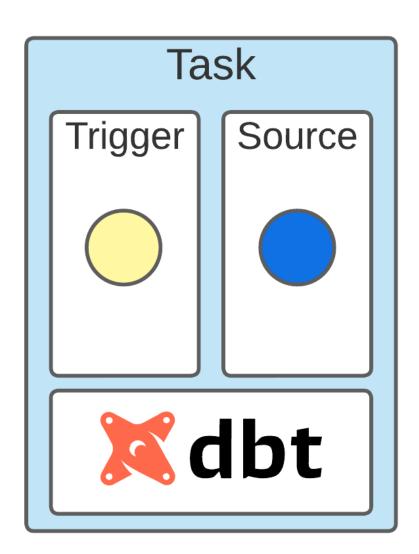


Argo Events - multiple tasks

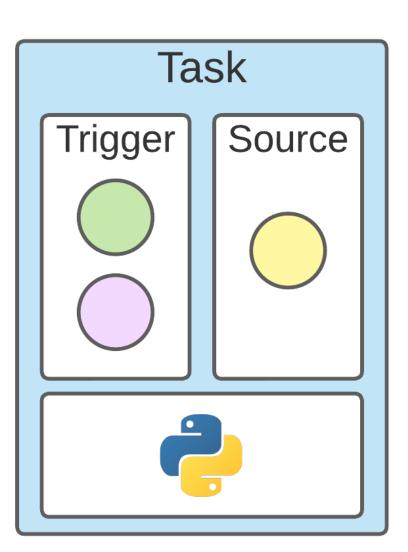


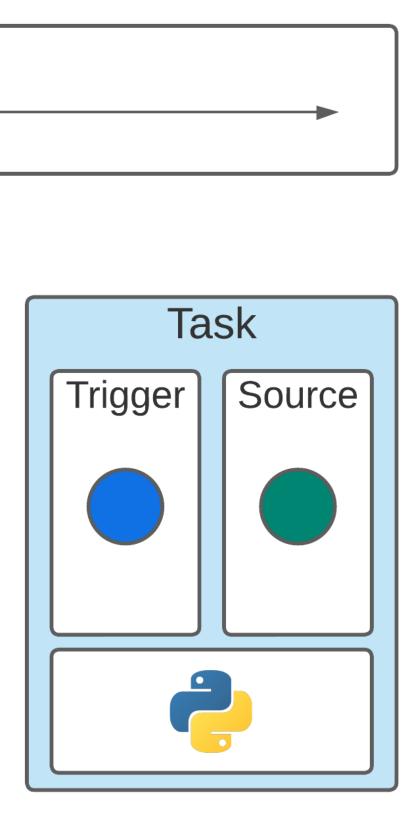






Event Bus







In conclusion

- abstraction
- business processes
- workflows definitions

• "Workflow as code" orchestrators encourage us to think at a low level of

Argo Workflows container-native approach encourages us to think in terms of

Argo Events offers a different approach to tackle tasks dependencies and





Thank you!

