ArgoCon

EUROPE

ArgoCon
EUROPE

One Engine to Rule Them All

Unifying Cloud Workloads with Argo Workflows

Sebastian Beyvers, Giessen University

ArgoCon EUROPE

1 April 2025

London, England

Sebastian Beyvers

Distributed Systems Researcher

Giessen University

Agenda

ArgoCon EUROPE

Introduction

Background: where are we coming from

Our Problem

A heterogenous ecosystem of workflow engines and compute tools

Solution

Argo workflows as universal scheduling engine ?

Implementation

Migration and Integration patterns

Results

What are the benefits ? What are the drawbacks ?

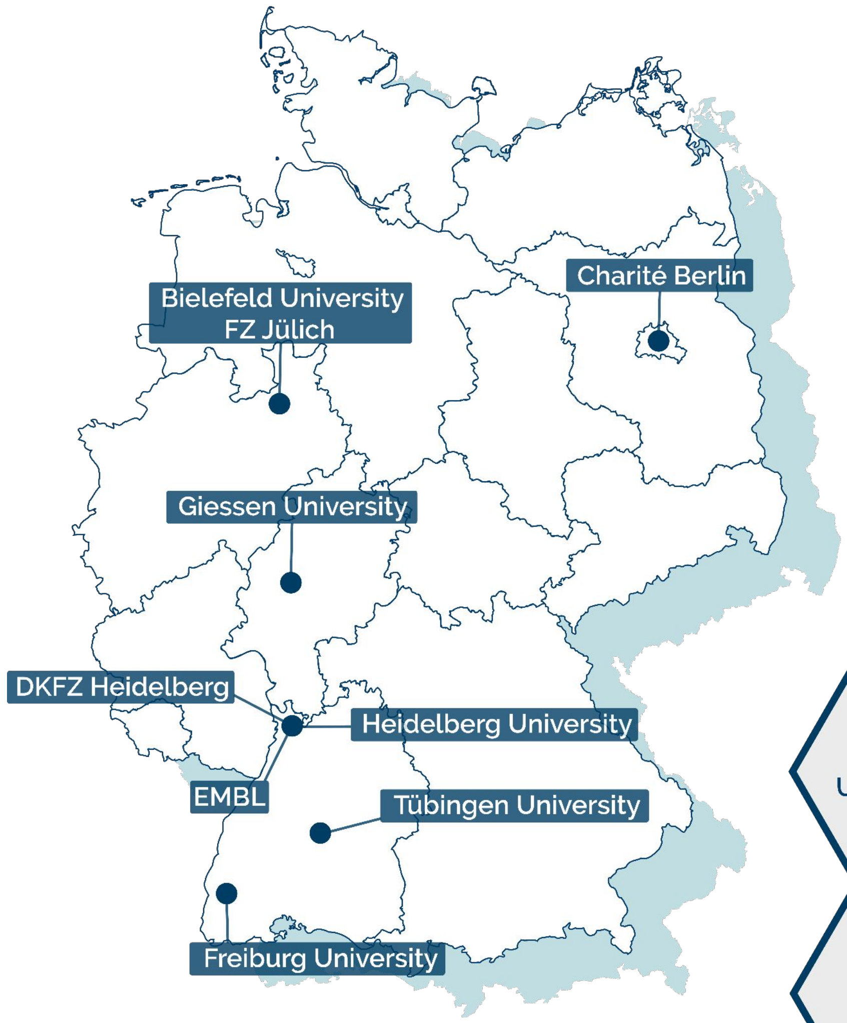
Background

ArgoCon EUROPE

de.NBI GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

elixir GERMANY

- Largest community cloud for life-science in Germany
- Free to use for the research community
- OpenStack
- Managed Kubernetes
- SLURM
- VMs



56.000 CPU cores

720TB RAM with up to 4TB per node

100 PB of storage capacity with 720TB high-speed SSD storage

520 GPUs

SimpleVM

Kubernetes

OpenStack

Galaxy Europe

Annual User Meetings

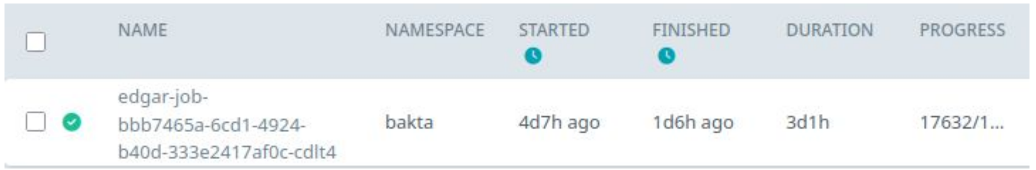
3500+ Users registered in 1200+ projects

1000+ publications with reference to the de.NBI Cloud

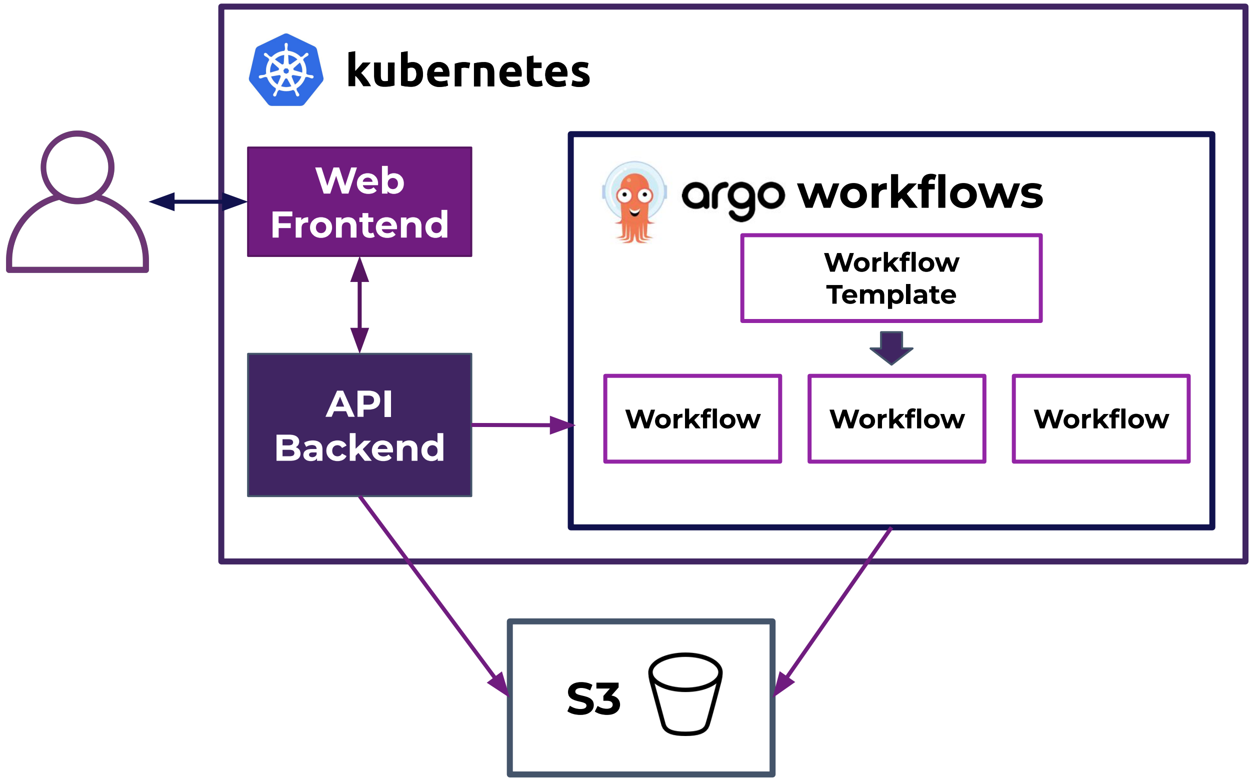
Our Argo Workflows journey

ArgoCon EUROPE

- Started ~**5** years ago
- Currently ~**10** clusters:
  - **200 - 5000** vCPUs
  - Including some high-mem nodes > **2 TB RAM**
- High throughput with **100-1000**s workflows per day
- Long running workflows (Days - Weeks)
- Primarily analysis workflows, almost **no streaming / real time analysis**



	NAME	NAMESPACE	STARTED	FINISHED	DURATION	PROGRESS
✓	edgar-job-bbb7465a-6cd1-4924-b40d-333e2417af0c-cdlt4	bakta	4d7h ago	1d6h ago	3d1h	17632/1...



Why Argo Workflows ?

ArgoCon EUROPE

Flexibility	You can do almost everything with Argo workflows
Extensibility	Plugin system, Workflow Templates and easy API access allows for scheduling of pre-configured workflows
Events	Argo Events provides additional automation capabilities and integration with message queues like Kafka, RabbitMQ etc.
Kubernetes native	A K8s native system enables easy handling of Kubernetes resources and optimal use of containers
Ecosystem	The Argo Ecosystem provides many additional useful tools like the **Hera** python library, integration with **Argo CD**, delivery with **Argo Rollouts**

The Problem

ArgoCon EUROPE

Cloud

- KubeFlow
- Argo workflows
- Apache Airflow
- Nextflow
- Custom Jobs
- Spark
- Snakemake

HPC / SLURM

- Snakemake
- Nextflow
- Conveyor
- CWL
- Custom Jobs
- Spark

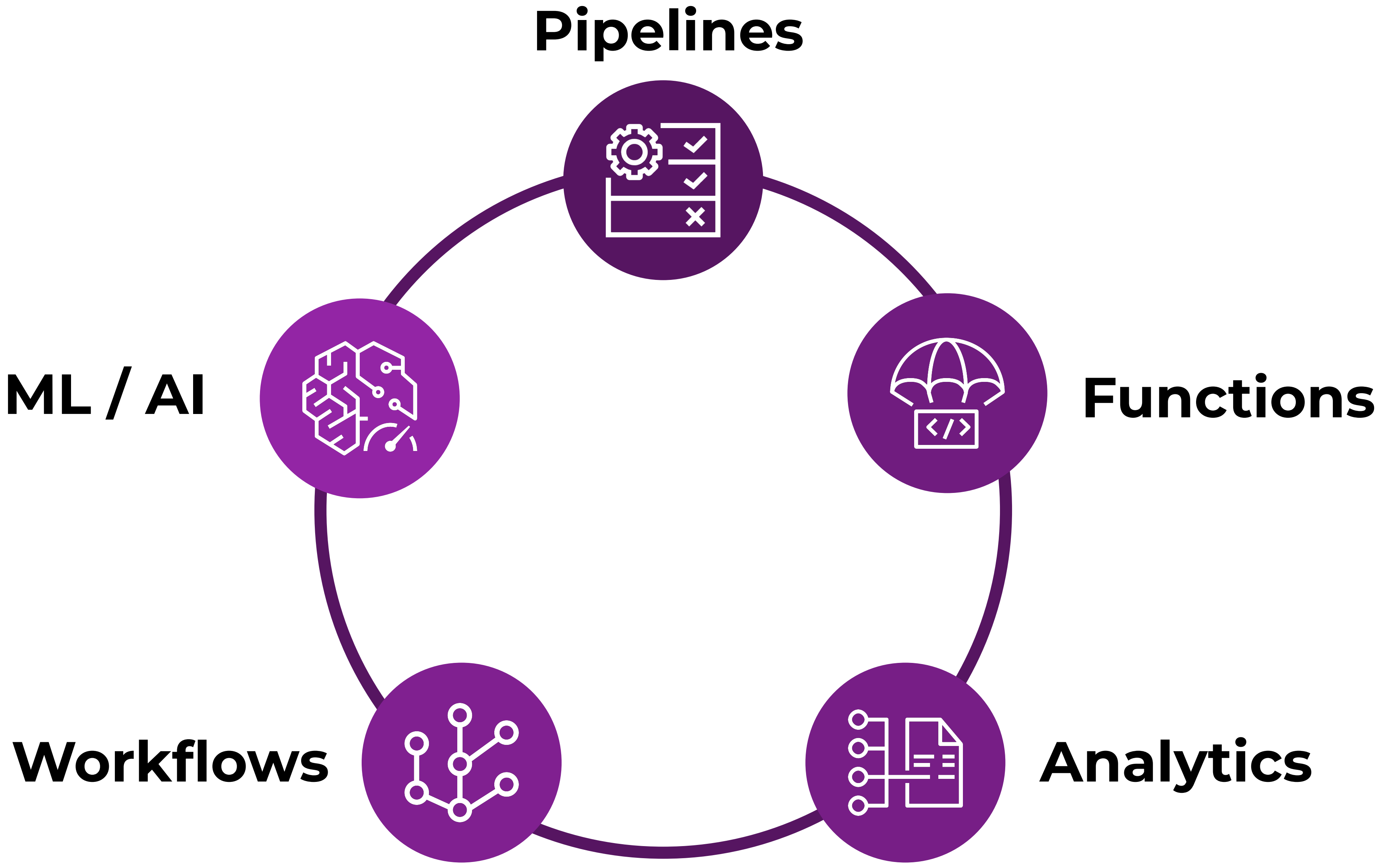
VMs

- Snakemake
- Custom Jobs
- Nextflow

A wide range of workflow tools and environments

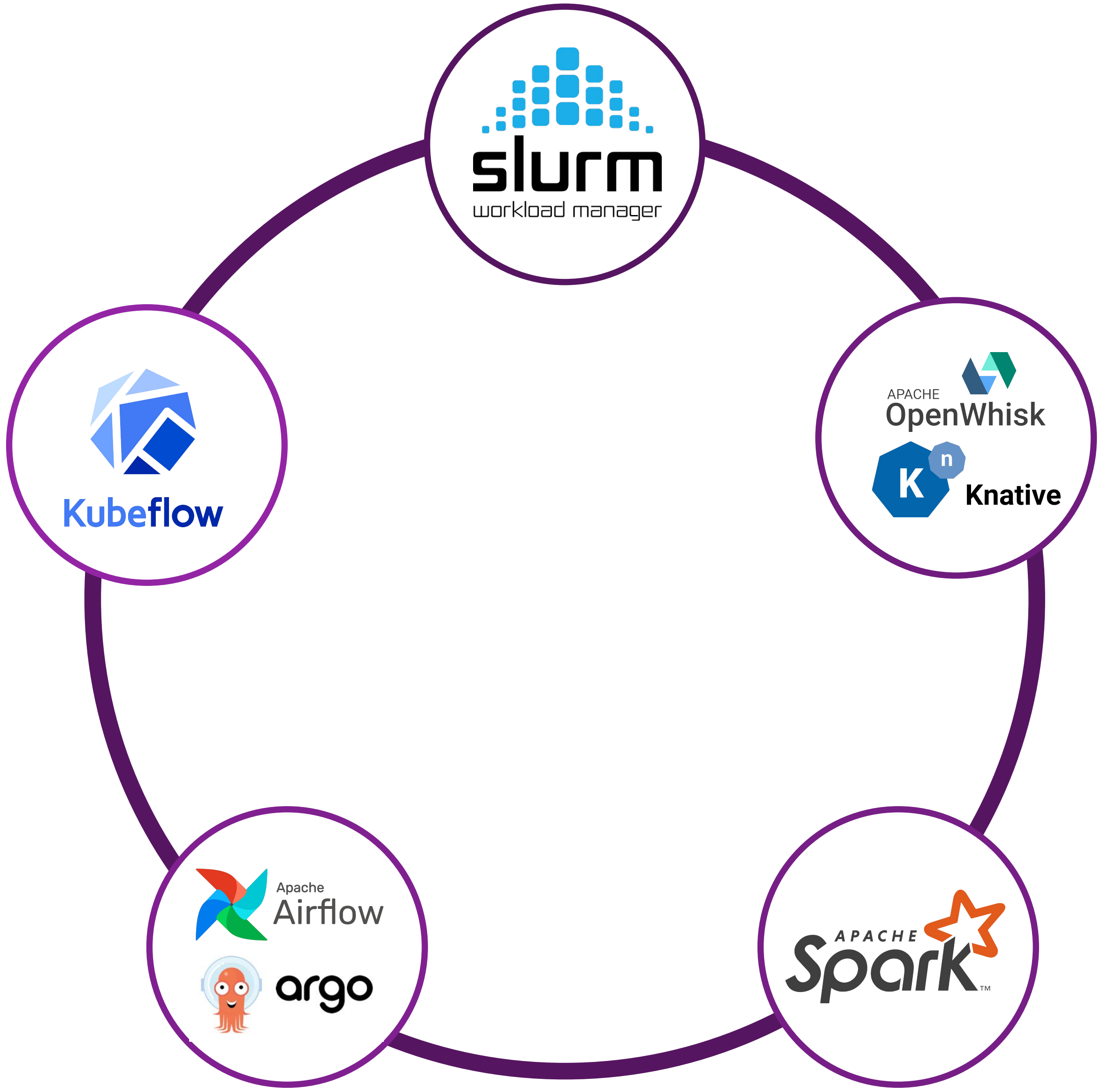
Promises of a public cloud

ArgoCon EUROPE



DIY: Collection of tools

ArgoCon EUROPE



DIY: Pros & Cons

Pro

- Sovereignty & Independence
- Specialized solution
- Data protection
- Flexibility and cost effectiveness

Con

- Mental overhead
- Larger human resource requirements
- Drastically increased onboarding time

Argo Workflow as coordinator

ArgoCon EUROPE

Can Argo workflows be a universal scheduling engine ?

Argo Workflow as coordinator

How can Argo Workflows coordinate my cloud workloads ?

Argo Workflow as coordinator

ArgoCon EUROPE

How can Argo Workflows coordinate my cloud workloads ?

A: Migration

Argo Workflow as coordinator

ArgoCon EUROPE

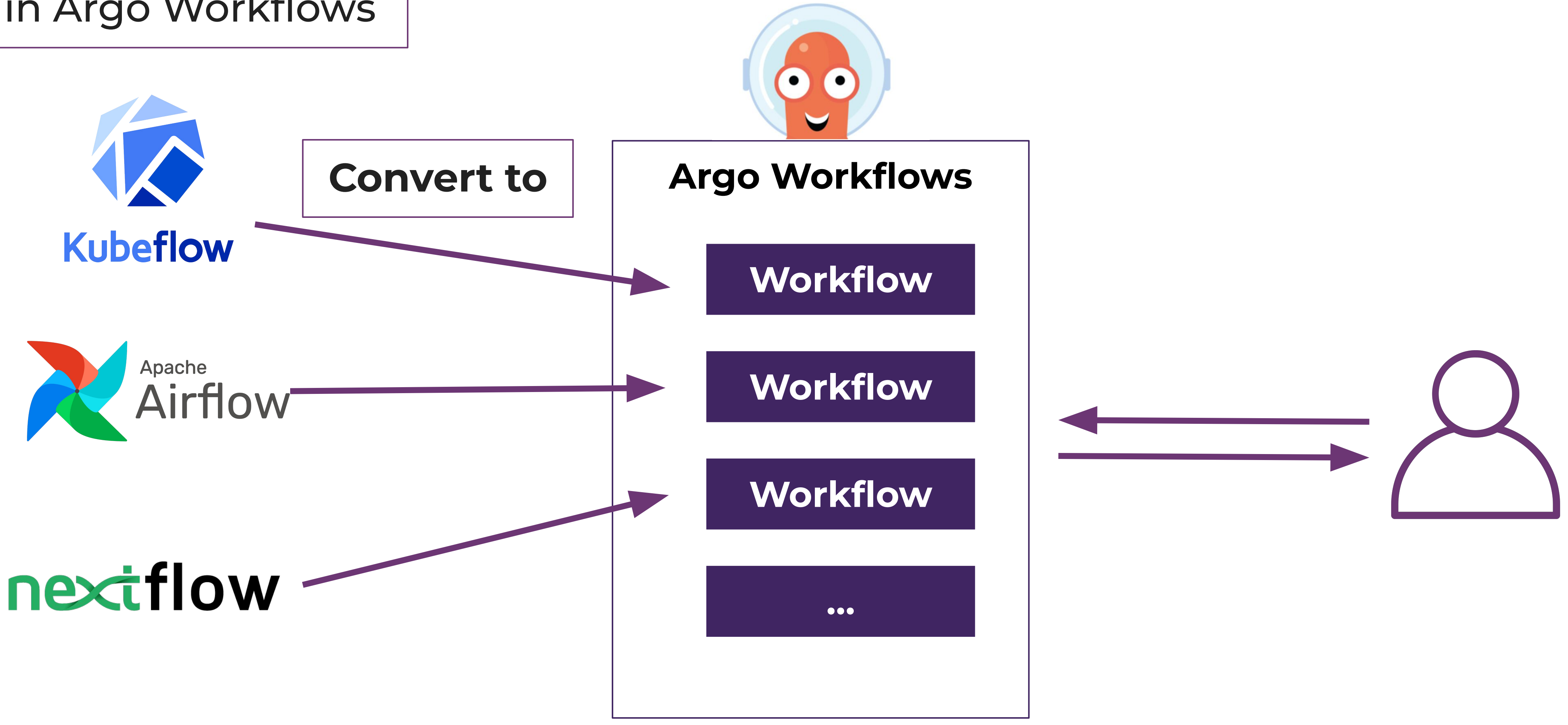
How can Argo Workflows coordinate my cloud workloads ?

A: Migration

B: Integration

Strategy: Migration

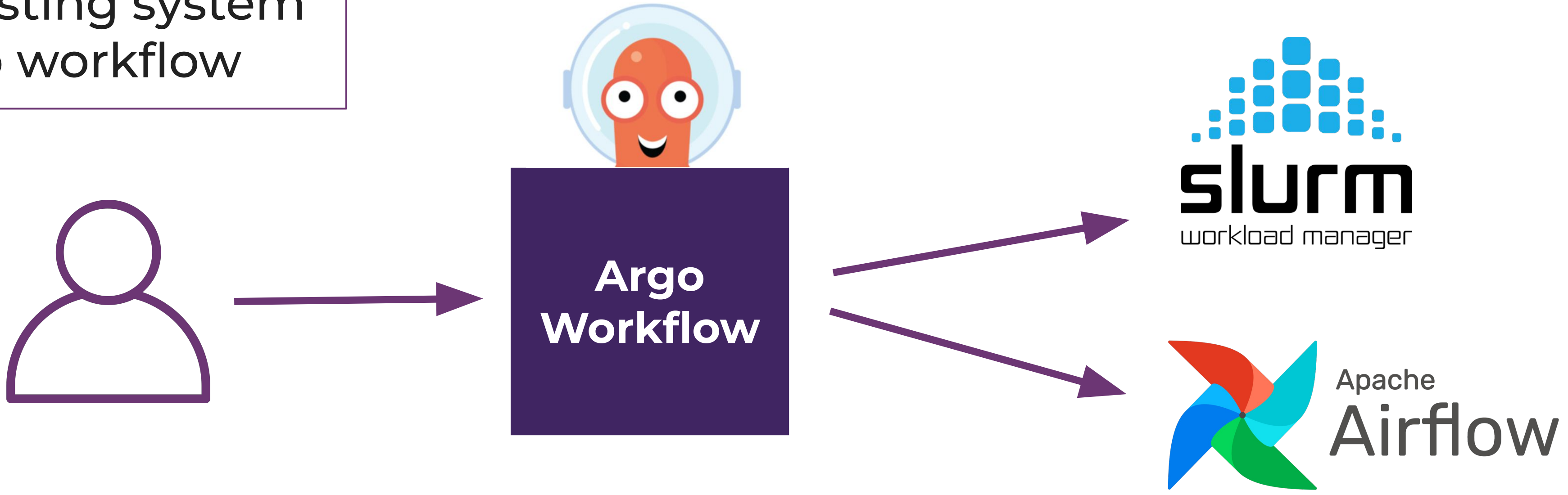
Reimplement the workload directly in Argo Workflows



Strategy: Integration

ArgoCon EUROPE

Wrap the existing system in an Argo workflow



Init

- Initializes data storage
- Upload data to a shared storage
- Pre-configure variables and secrets

Main

- Start the external run
- Monitor the status
- Mirror the external status in the argo workflow status

Finalize

- Accumulate results
- Process outcome
- (optional) Upload results to cloud storage

Strategy: Migration vs. Integration

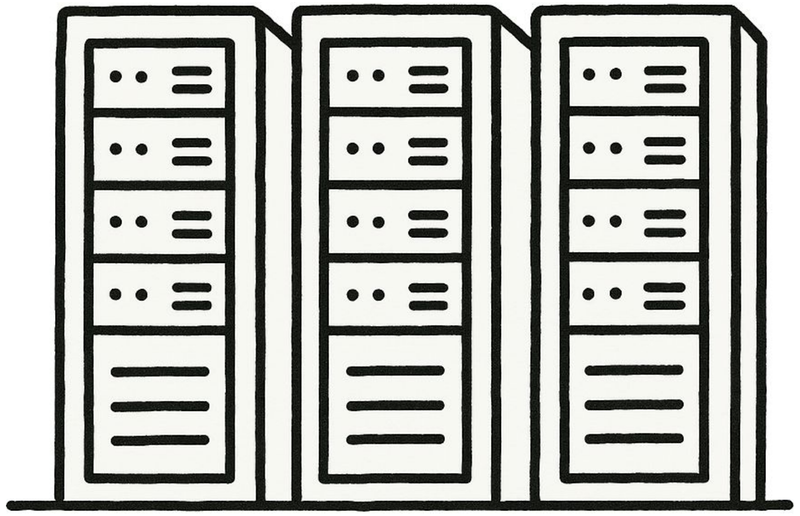
ArgoCon EUROPE

		Integration	Migration
Hard requirement to use an existing workflow engine?	yes	✓	✗
	no	✗	✓
Is the existing workflow very complex ?	yes	✓	(✓)
	no	✗	✓
Sufficient team expertise for the existing solution ?	yes	✓	✗
	no	✗	✓
Has the workload a varying degree of resource requirements ?	yes	✓	(✓)
	no	✗	✓
Is your workload already containerized or easily containerizable ?	yes	✗	✓
	no	✓	✗

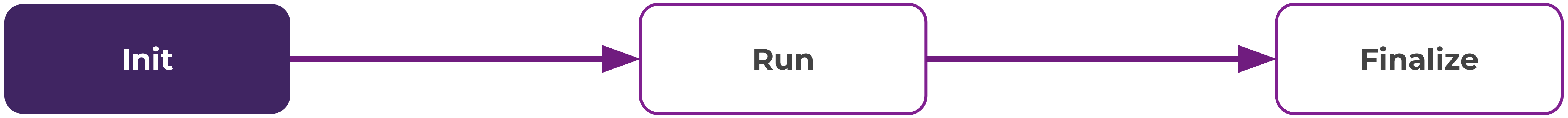
HPC Workloads: Migration

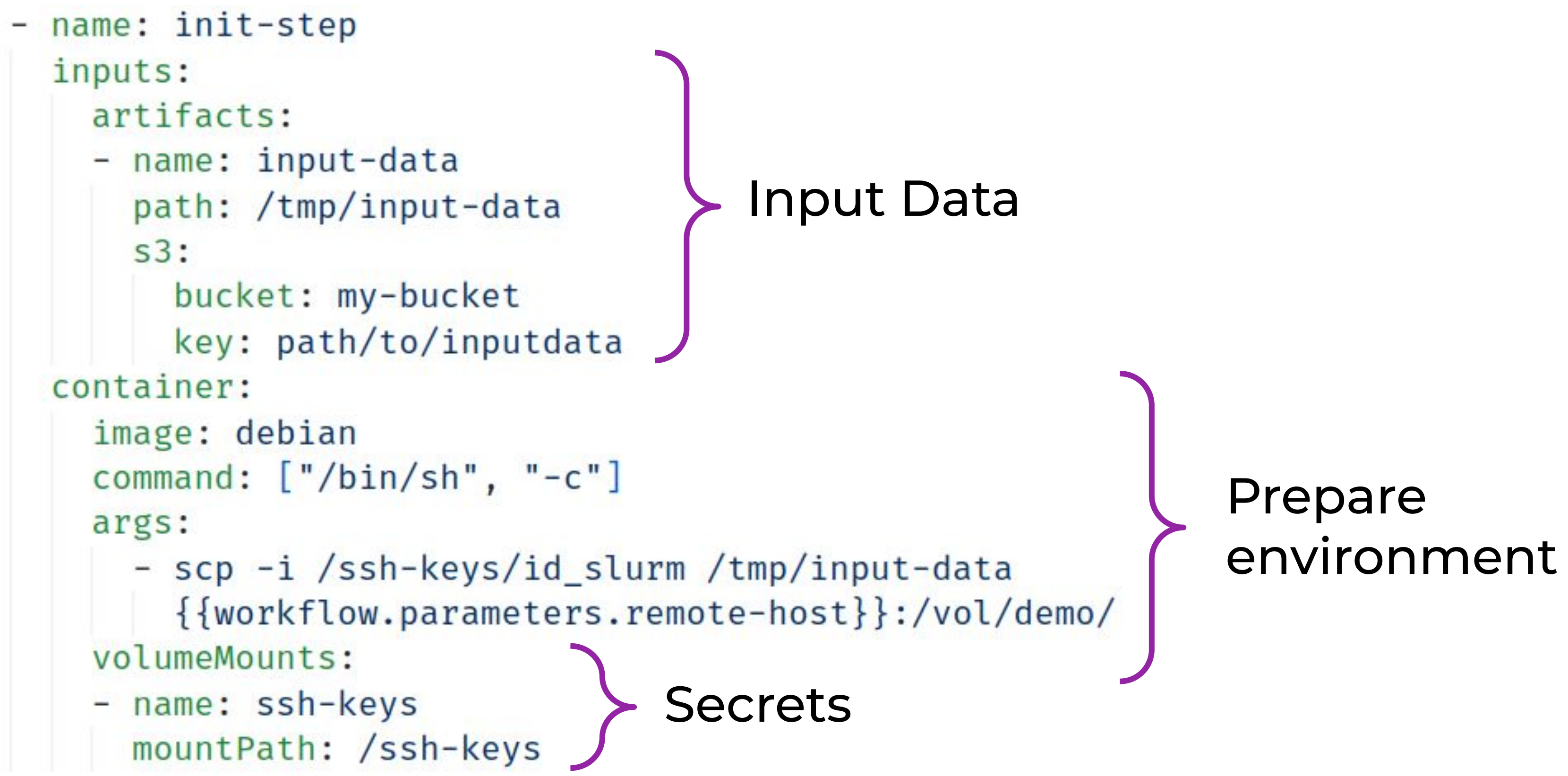
ArgoCon EUROPE

- Rewrite the analysis workflow as Argo workflow
- **Challenges**:
    - Shared file system
    - Shared permissions
    - Internal databases
    - Specialized Hardware (FPGA, GPUs etc.)



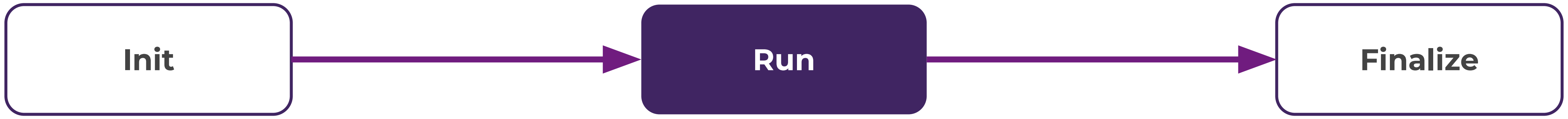
HPC Workloads: Integration

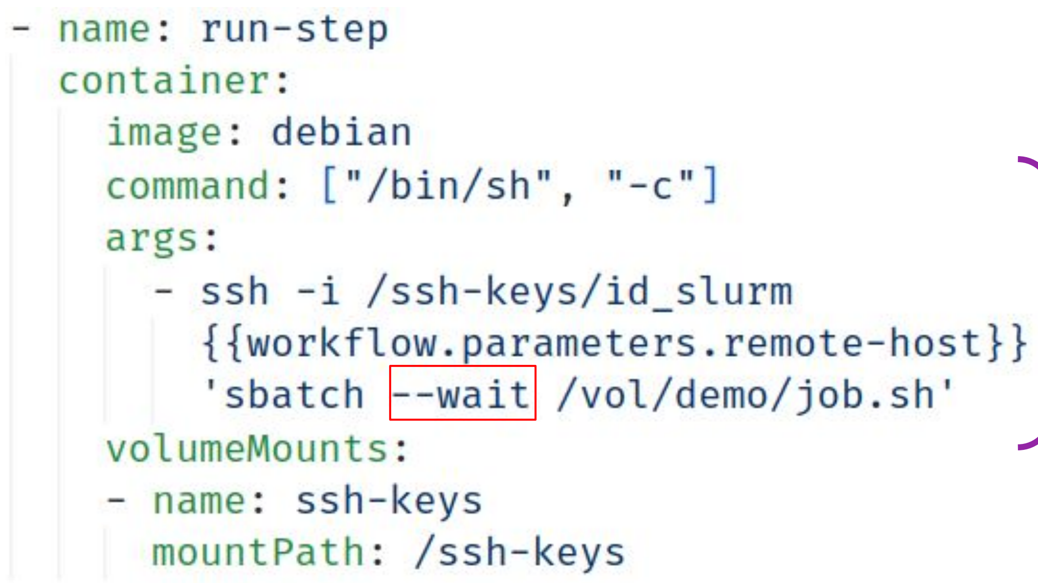




```yaml
- name: init-step
  inputs:
    artifacts:
    - name: input-data
      path: /tmp/input-data
      s3:
        bucket: my-bucket
        key: path/to/inputdata
  container:
    image: debian
    command: ["/bin/sh", "-c"]
    args:
      - scp -i /ssh-keys/id_slurm /tmp/input-data
        {{workflow.parameters.remote-host}}:/vol/demo/
    volumeMounts:
    - name: ssh-keys
      mountPath: /ssh-keys
```

HPC Workloads: Integration





```
- name: run-step
  container:
    image: debian
    command: ["/bin/sh", "-c"]
    args:
      - ssh -i /ssh-keys/id_slurm
        {{workflow.parameters.remote-host}}
        'sbatch --wait /vol/demo/job.sh'
    volumeMounts:
    - name: ssh-keys
      mountPath: /ssh-keys
```

Run SLURM workload in HPC environment

HPC Workloads: Integration

$$\begin{tikzcd}\text{Init} \arrow[r] & \text{Run} \arrow[r] & \text{Finalize}\end{tikzcd}$$

```
- name: finalize-step
  container:
    image: debian
    command: ["/bin/sh", "-c"]
    args:
      - scp -i /ssh-keys/id_rsa
        {{workflow.parameters.remote-host}}:/vol/demo/results
        /tmp/results
    volumeMounts:
    - name: ssh-keys
      mountPath: /ssh-keys
  outputs:
    artifacts:
    - name: results
      path: /tmp/results
      s3:
        bucket: my-bucket
        key: path/to/results
```

Retrieve remote data from HPC environment

Create output artifact from remote data

Data Pipelines: Migration

ArgoCon EUROPE

Apache Airflow

```python
from airflow.decorators import dag, task

@task()
def hello(name: str):
    print(f"Hello {name}!")

@dag(
    dag_id = "dag_example",
)
def dag_example():
    t1 = hello(name="hello-1")
    t2 = hello(name="hello-2")
    t1 >> t2

dag_instance = dag_example()
```

argo

```python
from hera.workflows import DAG, Parameter, Workflow, script

@script()
def hello(name: str):
    print(f"Hello {name}!")

with Workflow(
    generate_name = "dag-example",
    entrypoint = "dag",
) as w:
    with DAG(name = "dag") as d:
        t1 = hello(name="hello-1", arguments = {"name": "hello-1"})
        t2 = hello(name="hello-2", arguments = {"name": "hello-2"})
        t1 >> t2
```

Data Pipelines: Integration

ArgoCon EUROPE

Apache Airflow

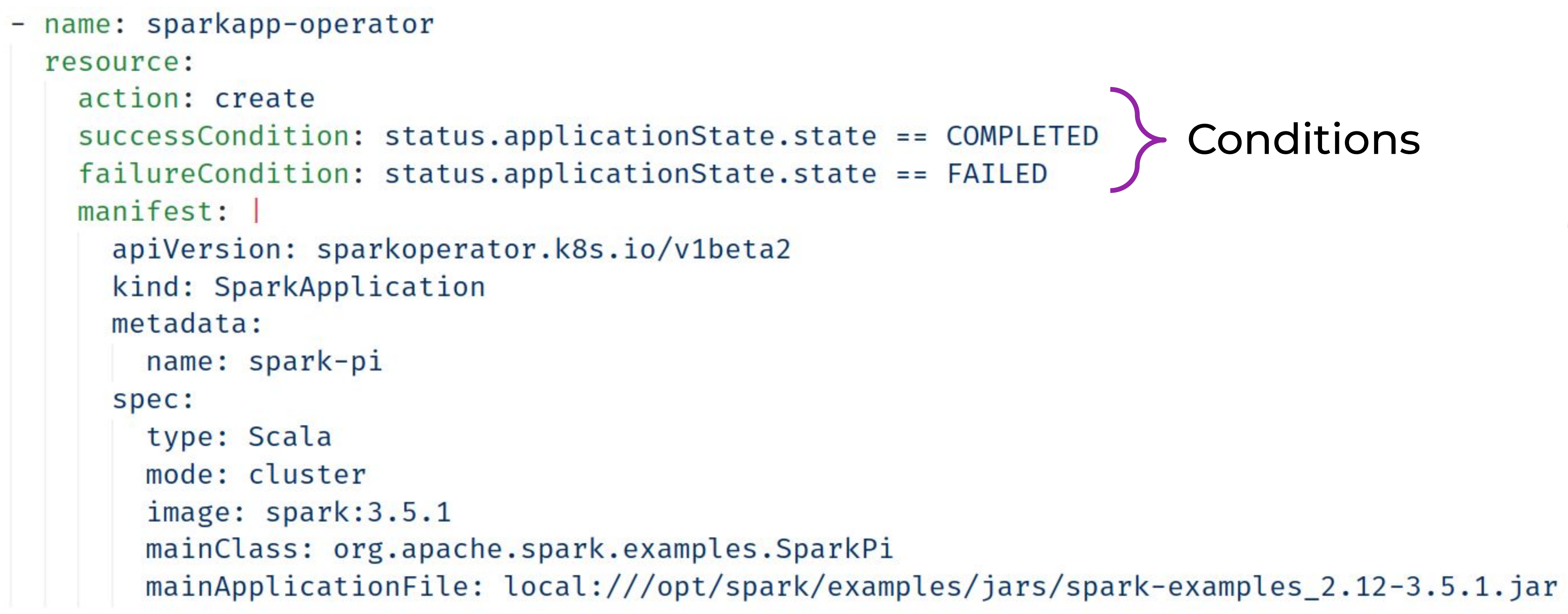
```
- name: trigger-airflow
  script:
    image: python:3.14
    command: [python]
    source: |
      import requests, time, sys

      airflow_url = "{{workflows.parameters.airflow_url}}"
      dag_id = "{{workflows.parameters.dag_id}}"
      run_id = "{{workflows.parameters.run_id}}"
      headers = "{{workflows.parameters.headers}}"
      payload = {"dag_run_id": run_id}

      response = requests.post(f"{airflow_url}/api/v1/dags/{dag_id}/dagRuns",
      json=payload, headers=headers)
      if response.status_code != 200:
          print(f"Failed to trigger DAG: {response.status_code}")
          sys.exit(1)

      status_endpoint = f"{airflow_url}/api/v1/dags/{dag_id}/dagRuns/{run_id}"
      while True:
          status_response = requests.get(status_endpoint, headers=headers)
          state = status_response.json().get('state')
          if state in ['success', 'failed']:
              sys.exit(0 if state == 'success' else 1)

          time.sleep(5)
```

Setup parameters

Run DAG

Wait for completion

Analytics: Integration



```yaml
- name: sparkapp-operator
  resource:
    action: create
    successCondition: status.applicationState.state == COMPLETED
    failureCondition: status.applicationState.state == FAILED
    manifest: |
      apiVersion: sparkoperator.k8s.io/v1beta2
      kind: SparkApplication
      metadata:
        name: spark-pi
      spec:
        type: Scala
        mode: cluster
        image: spark:3.5.1
        mainClass: org.apache.spark.examples.SparkPi
        mainApplicationFile: local:///opt/spark/examples/jars/spark-examples_2.12-3.5.1.jar
```

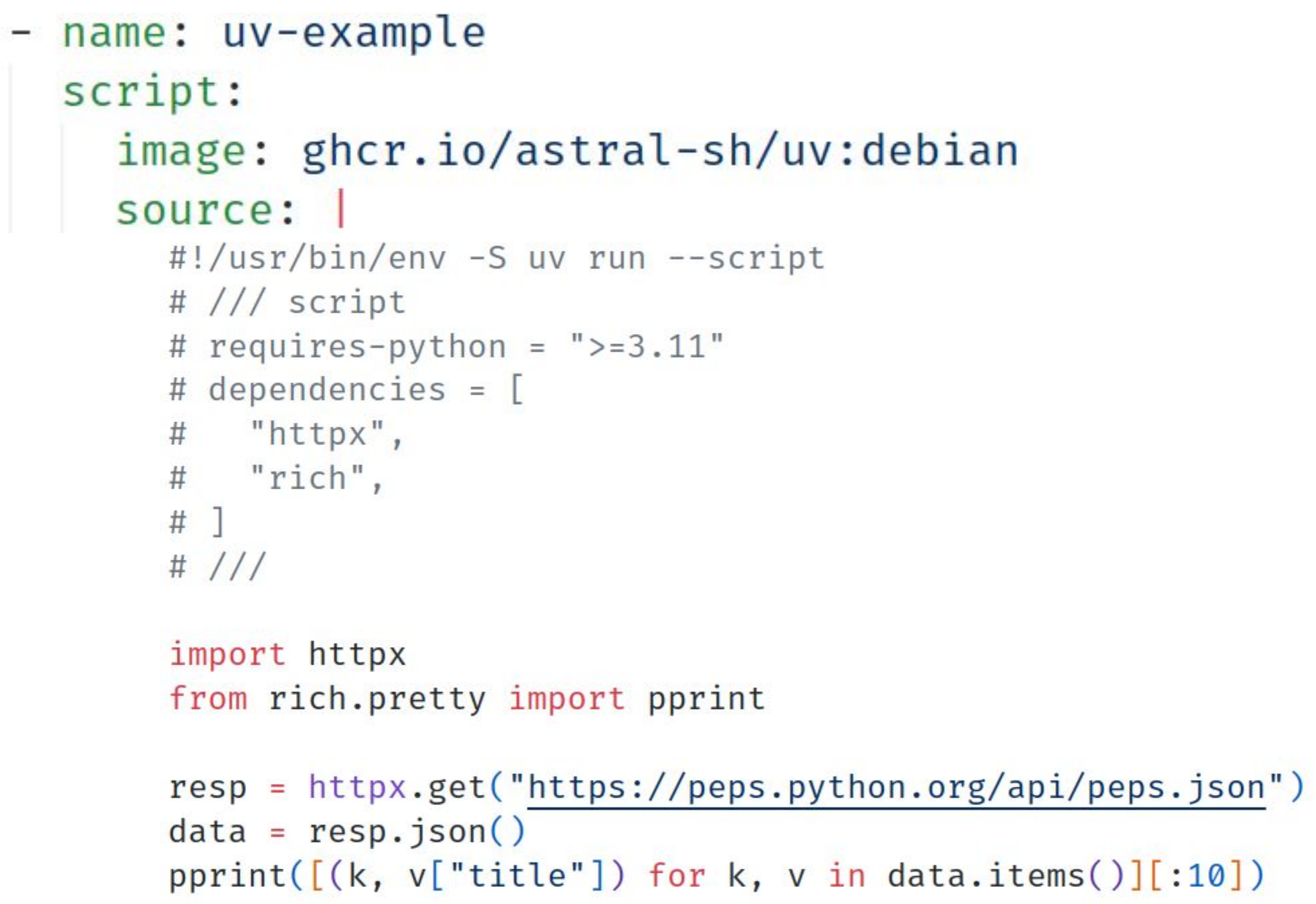
$\}$ Conditions

$\}$ Kubernetes resource

- Using KubeFlow Spark operator:
  - https://github.com/kubeflow/spark-operator
- https://pipekit.io/blog/argo-workflows-spark

Functions: Migration

- Use different templates for programming languages
- Use Python, Javascript, Rust etc. for function
- Possible strategy: One **WorkflowTemplate** per programming language

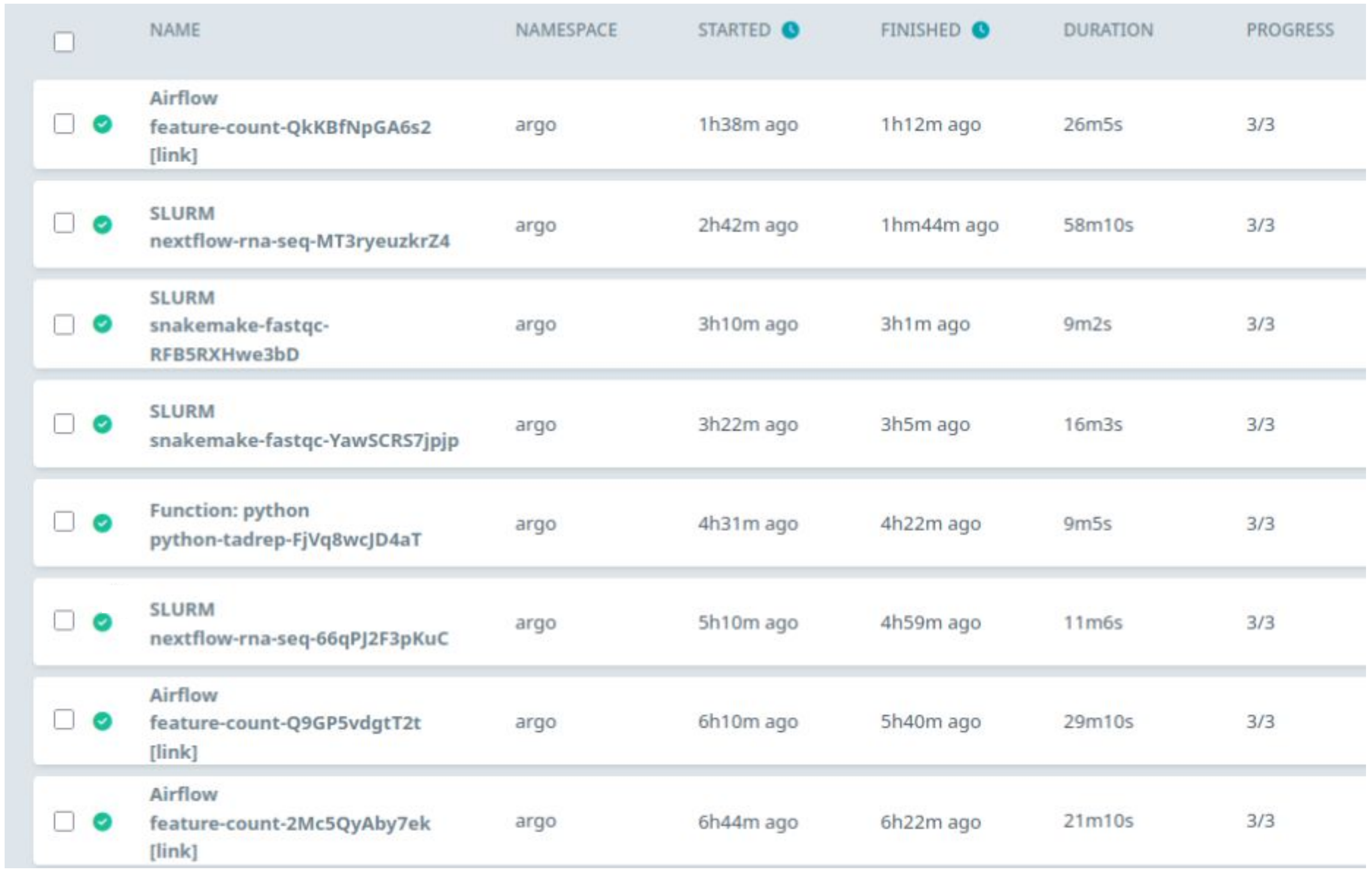


```
- name: uv-example
  script:
    image: ghcr.io/astral-sh/uv:debian
    source: |
      #!/usr/bin/env -S uv run --script
      # /// script
      # requires-python = ">=3.11"
      # dependencies = [
      #   "httpx",
      #   "rich",
      # ]
      # ///

      import httpx
      from rich.pretty import pprint

      resp = httpx.get("https://peps.python.org/api/peps.json")
      data = resp.json()
      pprint([(k, v["title"]) for k, v in data.items()][:10])
```

Unified Developer Experience

- single entry point for all workflows
- centralized API
- KPI & metrics friendly
- Pre-configured workflows using **WorkflowTemplates**
- Fast onboarding time

"All under one roof"



	NAME	NAMESPACE	STARTED	FINISHED	DURATION	PROGRESS
✓	Airflow feature-count-QkKBfNpGA6s2 [link]	argo	1h38m ago	1h12m ago	26m5s	3/3
✓	SLURM nextflow-rna-seq-MT3ryeuzkrZ4	argo	2h42m ago	1hm44m ago	58m10s	3/3
✓	SLURM snakemake-fastqc-RFB5RXHwe3bD	argo	3h10m ago	3h1m ago	9m2s	3/3
✓	SLURM snakemake-fastqc-YawSCRS7jpjp	argo	3h22m ago	3h5m ago	16m3s	3/3
✓	Function: python python-tadrep-FjVq8wcJD4aT	argo	4h31m ago	4h22m ago	9m5s	3/3
✓	SLURM nextflow-rna-seq-66qPJ2F3pKuC	argo	5h10m ago	4h59m ago	11m6s	3/3
✓	Airflow feature-count-Q9GP5vdgtT2t [link]	argo	6h10m ago	5h40m ago	29m10s	3/3
✓	Airflow feature-count-2Mc5QyAby7ek [link]	argo	6h44m ago	6h22m ago	21m10s	3/3

Limitations & Challenges

ArgoCon EUROPE

- Low latency
- Stream processing
- Performance overhead
- Specialized hardware (GPUs etc.)
- Multiple Argo Workflow instances

Lessons

ArgoCon EUROPE

Argo workflows can be used as a unified scheduling engine

WorkflowTemplates can help to drastically reduce the onboarding time

Kubernetes **can replace** many traditional HPC workloads !

Outlook

Argo workflows wishlist:

- Better **synchronization** between different Argo workflow clusters
- Smarter resource **request / limit** configurability
- **Kueue** integration
- Better data management capabilities e.g. **FUSE**

Personal wishlist:

- Successful **migration** of most cloud workloads to Argo workflows
- Automated migration tools
- Integrate our data management solutions into workflows

ArgoCon EUROPE

Share your feedback

Questions ? Contact me:

sebastian.beyvers@cb.jlug.de
@St4NNi
St4NNi

Join our poster session

KubeCon | CloudNativeCon Europe 2025

Thursday April 3, 13:15 - 14:15 BST
@Level 1 N8-N9 | Poster Pavilion

Thanks to our partners:

de.NBI GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE
elixir GERMANY
NFDI4 MICROBIOTA
NFDI 4 BIODIVERSITY