ArgoCon

Comparing Argo Workflows vs. Apache Airflow at scale

J.P. Zivalich - Pipekit

pipekit

INTROS

J.P. Zivalich

CTO & Founder of @ Pipekit

Argo Workflows contributor

JPZ13

jpzivalich

Overview

- Introduction to Argo Workflows
- Introduction to Apache Airflow
- Architecture Comparison
- Integration with Kubernetes
- Scalability
- Ease of Use
- Monitoring and Observability

Introduction to Argo Workflows

What is it?

An open-source, Kubernetes-native tool for managing containerized workflows

What are the key features?

Kubernetes integration, container-centric, YAML-based, great Python SDK, dynamic workflows

Why do people choose Argo Workflows?

Perfect for Kubernetes shops, very generalizable, can use for CI, data, etc.

Introduction to Airflow

What is it?

An open-source workflow automation tool designed for data workflows and task dependencies

What are the key features?

Python scripting, task-based, extensible, broad base of pre-built connectors

Why do people choose Airflow?

Why Apache Airflow: It's a popular choice for organizations managing a wide range of workflows, especially in non-Kubernetes environments.

Architecture Comparison

Argo Workflows

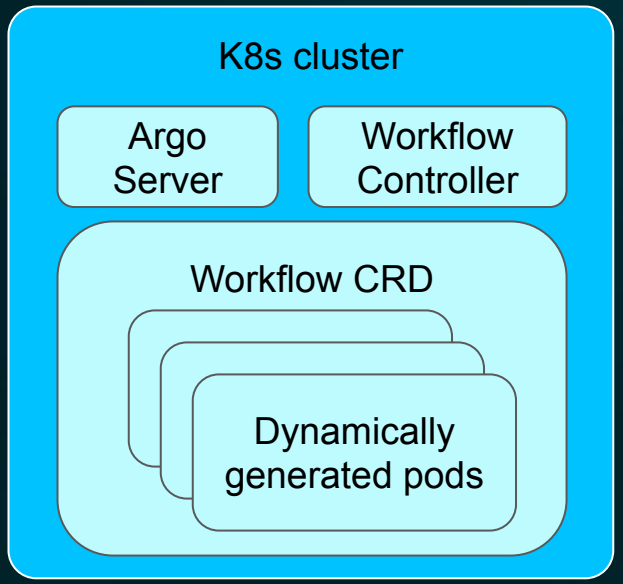
- Uses Custom Resource Definitions (CRDs) to define and manage workflows
- Two main components:
  - Workflow Controller
  - Argo Server
- Optional components:
  - SQL database for offloading and archiving workflows
  - S3 compatible bucket for artifacts and logs
  - Any other k8s primitive (PVCs, etc.)
- Relatively lightweight footprint

Airflow

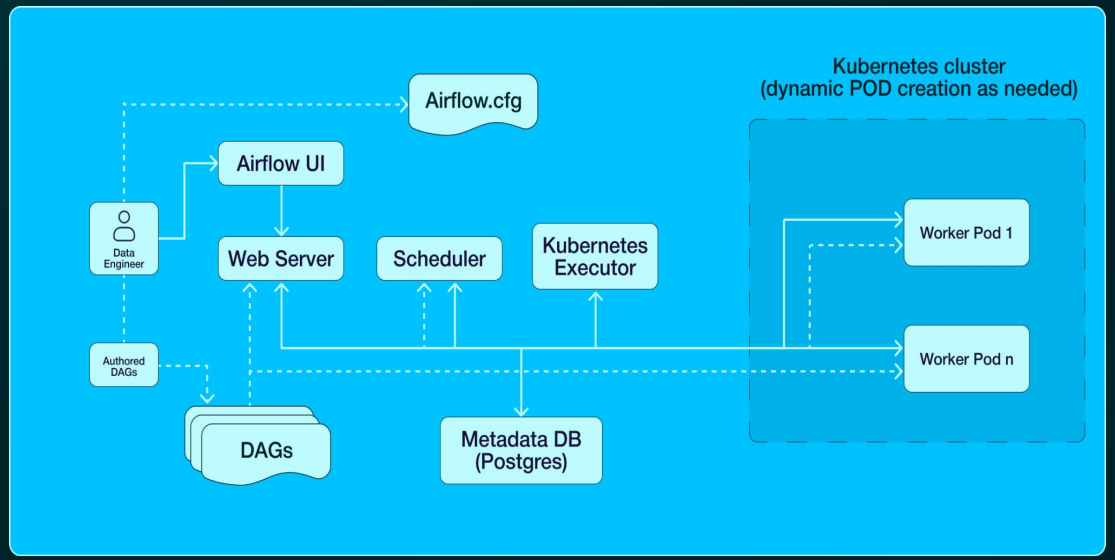
- Main components:
  - Scheduler
  - Executor (optionally part of the scheduler)
  - Web Server
  - DAG folder
  - Metadata database
- Optional components
  - Kubernetes executor
  - Celery executor
  - Etc.
- Can schedule workers as pods on K8s with Kubernetes executor

Architecture Comparison

Argo Workflows



Airflow



Integration with Kubernetes

Argo Workflows

- Seamless integration with K8s
- Can query Workflows with Kube API
- Every workflow node runs in a pod
  - Can group smaller steps into one pod using a ContainerSet
- Can easily specify config like resource requests/limits, service accounts, etc. using either native YAML or python with Hera

Airflow

- Requires configuring the Kubernetes executor
- Can't query Tasks/DAGs with Kube API
  - Can query pods that Airflow uses for scheduling
- Setting config like resource requests/limits, service accounts, etc. is done at a global pod template level in YAML
  - Can specify overrides
- Some users additionally configure Celery to run on K8s and run their smaller steps through Celery and their larger steps on dedicated pods

Scalability

Argo Workflows

- Able to schedule ~10k concurrent pods on a single cluster using EKS K8s 1.27+
- Upper limit is bottlenecked by the K8s API rather than Argo Workflows itself
- Can scale to the upper limits of what Kubernetes can handle
- Could push higher in the future if we implement batch operations on the workflow controller
- Can scale past normal K8s API limits by using vClusters

Airflow

- Requires configuring a dedicated Kubernetes executor
- By default, all resource requests and limits will be set to the same value using a pod template
- Can manually override the pod template, but it's cumbersome
- Bottlenecked by scalability of:
  - Scheduler
  - Metadata DB
- No hard data on scalability given how much Airflow configurations vary

Ease of Use

Argo Workflows

- YAML-based, declarative, easier for Kubernetes users
- Hera Python SDK allows for smoother developer experience
- Good REST API available
- Handles both scheduled and ad hoc workflows
- Default mode is to run individual steps in containers
- Can easily specify resource requests/limits, service accounts, etc. for each pod
- Few pre-built connectors
- Easy to build connectors

Airflow

- Python scripting
- Great library of pre-built connectors to common data sources
- Configuring the default pod configuration is hard - requires storing YAML in your DAG directory
- Overriding the default pod configuration is also hard
- Harder to build and maintain connectors to data sources

Monitoring and Observability

Argo Workflows

- Can bring along whatever existing Kubernetes monitoring and observability you already have
- Web UI that comes with the Argo Server
- Dedicated Prometheus metrics endpoint

Airflow

- Pods scheduled on K8s should play nicely with existing K8s monitoring stack
  - Does not apply to Airflow components outside of K8s
  - Not sure about Celery on K8s
- Web UI that comes with Airflow Web Server
- Can configure either StatsD or OpenTelemetry
  - From either, can send metrics to Prometheus

Conclusion

Argo Workflows

Pros

- Kubernetes-native workflow orchestrator
  - Should promote the Hera Python SDK more
- Scales as far as the K8s API can handle

Cons

- No great 3rd party connector library
- Documentation needs work
- Requires some K8s knowledge
  - Can be mitigated with Hera
- Can’t do multi-cluster

Airflow

Pros

- Mature Python-based workflow orchestrator
- Good connector library to 3rd party data sources

Cons

- Lots of components to manage
- Configuring and customizing the K8s pods in a workflow is challenging
- Have to store config in either a PVC or bake into the container image
- Can’t do multi-cluster

Let’s chat

@J.P. Zivalich on CNCF Slack

@JPZ13 on GitHub

https://s.pipekit.io/chat-argo-airflow