KubeCon CloudNativeCon Europe 2026

pipekit

Artifact Driver Plugins

Extending Argo Workflows with Custom Storage Backends

Alan Clucas

Intro

Scale Argo Workflows with Pipekit

- Direct support from Argo Workflows maintainers
- Add 3 Argo maintainers and 8 Argo contributors to your team
- Save time debugging Argo Workflows with Pipekit's Multi-cluster Control Plane UI
- Serving startups & Fortune 500 enterprises since 2021:

> **Enterprise Support for Argo**
> Ideal for Platform Eng & Data/ML teams scaling with Argo

> **Control Plane for Argo Workflows**
> Unlock granular RBAC, multi-cluster architectures, and observability

What is an Artifact?

A file or directory passed into or out of a workflow step

Input artifacts

Downloaded from storage into the container before it runs

Output artifacts

Uploaded from the container to storage after it completes

Compressed and archived by default (tgz)

Distinct from parameters, which are string values

Argo logs are also output artifacts

How Artifacts Flow

KubeCon CloudNativeCon Europe 2026

pipekit

Step A
container

Produces output artifact
compressed / archived

Upload

Artifact Repository

S3 / GCS / MinIO
Azure / HTTP...

Not direct pod-to-pod

Download

Step B
container

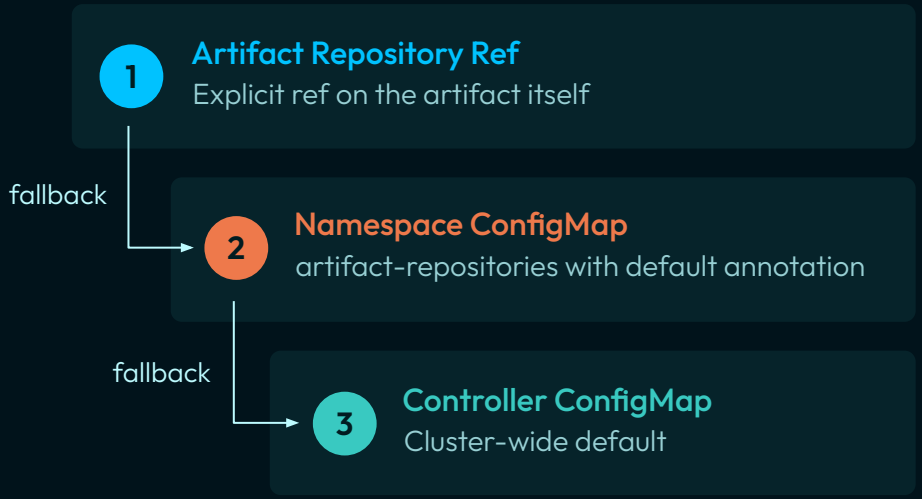
Receives input artifact
already decompressed

Artifact Repository Configuration

KubeCon CloudNativeCon Europe 2026

pipekit

Resolution Order



Built-in Drivers

S3 AWS, GCS, MinIO	**Azure Blob**
Artifactory	**OSS** Alibaba Cloud
HTTP	**Git** input only
Raw input only	**HDFS**

Artifact Driver Plugins

The Problem

Before v4.0, custom storage meant **no artifact declaration**

If your storage wasn't in the built-in set, you lost:

1. Artifact passing syntax between steps
2. Artifact garbage collection
3. Artifact access in the UI
4. Archive/compression handling

You were on your own with scripts in containers

The Solution: Artifact Driver Plugins

KubeCon CloudNativeCon Europe 2026

pipekit

- Implements the ArtifactService interface over a Unix socket
- Packaged as container image
- Runs alongside workflow pods as sidecar (outputs) and init container (inputs)

A plugin is a

GRPC server

- Also runs alongside argo-server for UI streaming/download
- Same artifact GC support
- Same artifact declaration syntax in your workflow YAML

Not to be confused with executor plugins

What You Get

Same as built-in

Artifact declaration syntax in workflows

Artifact GC support

Artifact access in UI

Compression/archive handling

Different

Configuration is a free-form string (not a typed schema) — the plugin parses it

You own the image, release cycle, and dependencies

You need

A GRPC server binary

A Docker image containing it

Registration in two places

For maintainers

No Argo fork, no CRD changes

Registering plugins

workflow-controller-configmap
→ artifactDrivers section

argo-server
→ manual sidecar

Each entry specifies:

`name`	the plugin identifier used in workflow specs
`image`	the Docker image to inject as a container

Building an Artifact Driver Plugin

The GRPC Contract

Core Operations

- Load
- Save
- Delete

Supporting Operations

- ListObjects
- IsDirectory
- OpenStream

Defined in artifact.proto

The Request Structure

Every request includes an Artifact message with a PluginArtifact sub-message:

`name`
matches the registered plugin name

`key`
path/key in the plugin's storage system

`configuration`
the free-form string from the configuration

Load and **Save** also include a path

The configuration field is the same string for both input and output artifacts. Your plugin parses it however you like.

YAML is recommended for consistency.

Generating Stubs

Download artifact.proto + googleapis dependencies

Go

```
protoc --go_out=. --go-grpc_out=. artifact.proto
```

Python

```
python -m grpc_tools.protoc -I. --python_out=. --grpc_python_out=. artifact.proto
```

C# / Java / Rust

```
Anything with GRPC support works
```

Pick whatever matches your storage backend's SDK

Packaging as a Docker Image

KubeCon CloudNativeCon Europe 2026

pipekit

Requirements

1. /tmp must exist and be writable by the container user
2. Run as non-root (argo-server runs non-root)
3. Keep the image minimal

Testing and Workflow Usage

Local Testing

- Start your plugin binary pointed at a local socket file
- Use a GRPC test client to call Load/Save/Delete
- Verify before deploying into a cluster

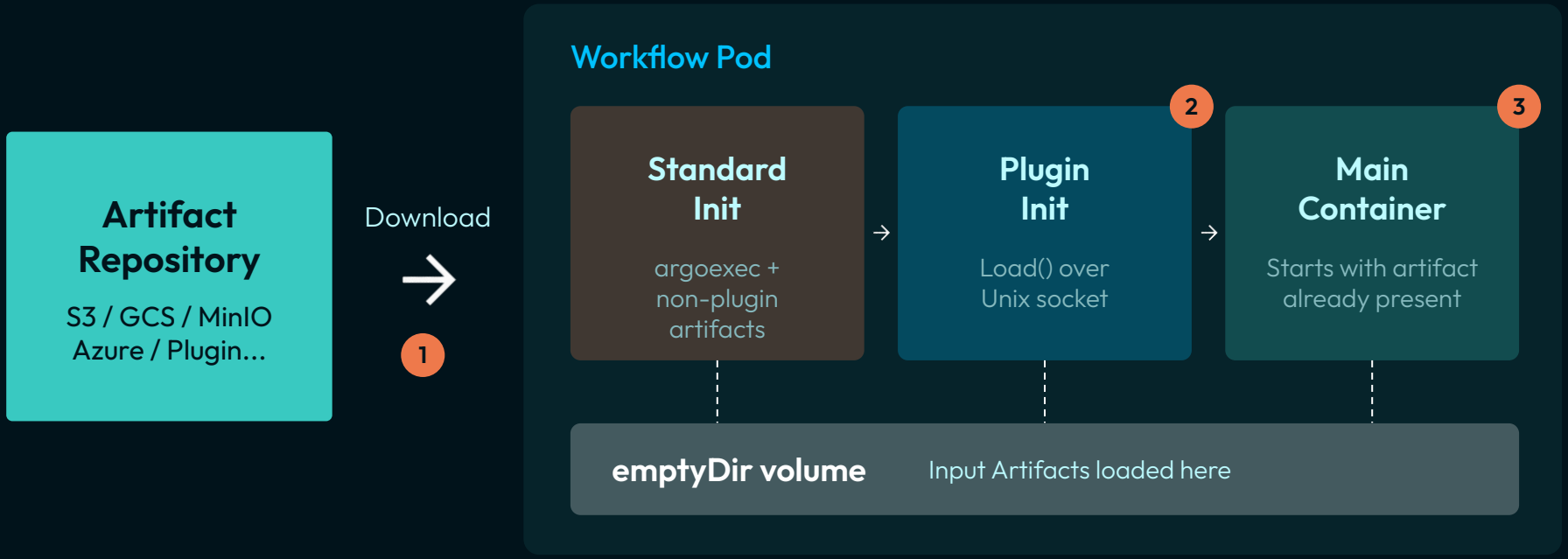
In Workflow YAML

- Use **plugin:** (instead of s3:, oss:, etc.)
- name must match the registered plugin name
- key and configuration are yours to define

Implementation Details

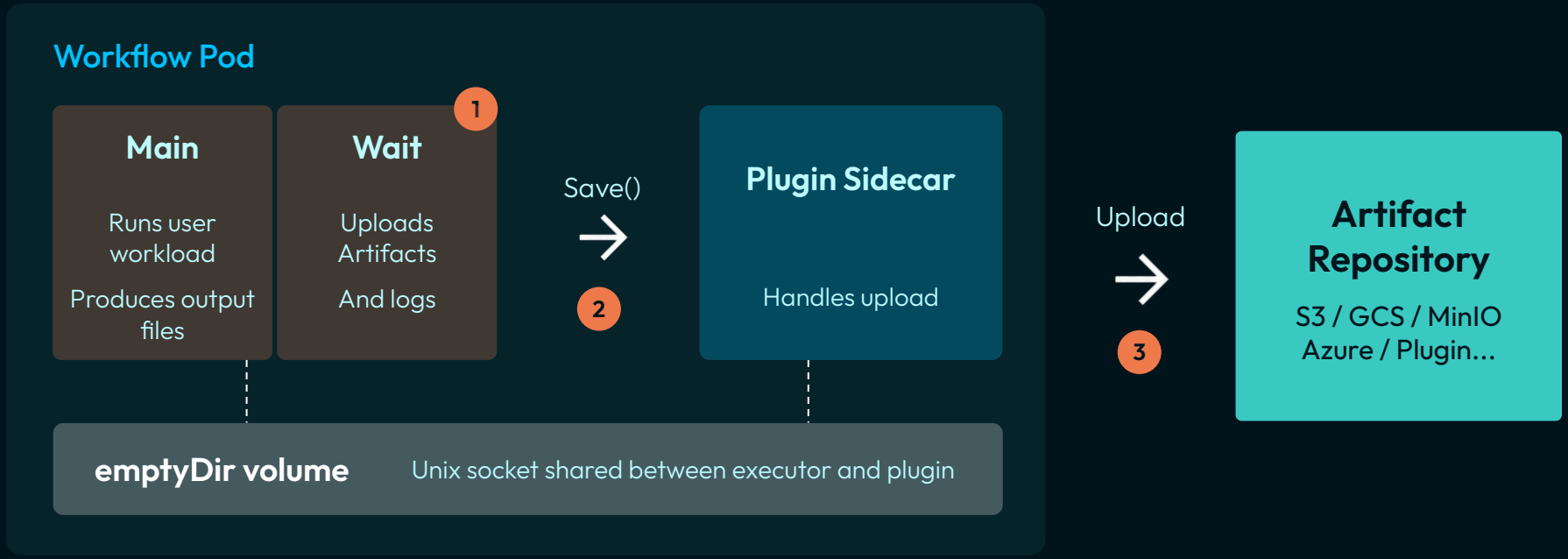
How It Actually Works in Argo

Injection: Init Containers (Inputs)



Volume mount sharing is handled automatically by the controller

Injection: Sidecars (Outputs)



The Full Picture

ConfigMap
artifactDrivers

reads →

Workflow Controller
Constructs pod specs

↓ injects plugin containers

Workflow Pod

Standard Init
argoexec + non-plugin artifacts

→

Plugin Init
Load() over Unix socket

→

Main Container
User workload runs here

→

Plugin Sidecar
Save() over Unix socket

emptyDir volume
Unix socket shared between plugin ↔ executor, and shared disk

argo-server

Plugin Sidecar
OpenStream() for UI downloads

emptyDir volume
Same Unix socket pattern

↑ Streams artifacts to browser UI

Init for inputs, sidecar for outputs. You don't need to care.

Argo-Server

KubeCon CloudNativeCon Europe 2026

pipekit

This is a manual step — you edit the argo-server Deployment yourself

1. Add the plugin container as a sidecar to the argo-server Deployment
2. Create an emptyDir volume, mount it in both the argo-server container and the plugin container at the same path
3. The socket filename must be socket (hardcoded expectation)
4. Plugin's first CLI arg is the full socket path

Why argo-server?

It needs the plugin to serve artifact downloads and streaming in the UI.

If you forget this step:
the argo-server checks expectations against the configmap at startup and will fail.

Current State and Resources

KubeCon CloudNativeCon Europe 2026

pipekit

Status

New in v4.0.0 (February) so expect rough edges

No known community plugins published yet

https://github.com/argoproj-labs/artifact-plugin-s3

Not supported on Windows

Future possibilities

Community index

Sometimes bypass two init containers using ImageVolumes

Links

Plugin Development Guide:
argo-workflows.readthedocs.io/en/latest/artifact-plugin/

Configure Artifact Repository
argo-workflows.readthedocs.io/en/latest/configure-artifact-repository/

Issue #5862
github.com/argoproj/argo-workflows/issues/5862

v4.0 Release Notes
argo-workflows.readthedocs.io/en/latest/new-features/

Thank You!
Let's chat at our booth.

KubeCon CloudNativeCon Europe 2026

pipekit

This talk & examples:
github.com/pipekit/talk-demos

Come and talk to us:

pipekit

Booth 494