ArgoCon

Lightning Talk: Configuring Volumes for Parallel Workflow Reads and Writes

Lukonde Mwila, Amazon Web Services

Tim Collins, Pipekit

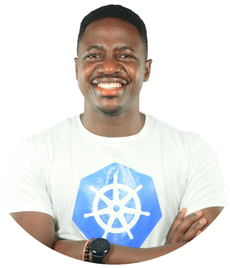
Intro's

Tim Collins

Senior Infrastructure Engineer @ Pipekit.io

pipekit

- Lives in the Argo Workflows Slack
- Keen (but bad) kiteboarder



Lukonde Mwila

Senior Developer Advocate, Amazon EKS team @ AWS and a CNCF Ambassador

aws

- Application development, solution architecture, cloud engineering, and DevOps workflows
- Cloud-native ecosystem contributor

The Problems

- How do I efficiently get from Workflows Step A to Workflow Step B?
- How do I efficiently share the same data with multiple Steps in parallel?
- How can I use semi-persistent data across many Workflows in parallel?

Persistent | Semi-Persistent | Transient

Storage Options

	Pros	Cons
S3	• Easy setup • Artifacts visible in the UI	• Files are tarred by Workflows before uploading - takes time and resources. • Copies are downloaded in the init container of each workflow task (so not RWX). • Only useful if you're in the cloud.
Minio	• Easy Setup. • Artifacts visible in the UI • Useful on prem	• Needs a disk behind it. • Small maintenance overhead. • Nuanced featureset differences with S3. • Files are tarred by Workflows before uploading - takes time and resources. • Copies are downloaded in the init container of each workflow task (so not RWX).
EBS	• Easy setup • Some performance advantages over S3 • No tarring required	• Not RWX
EFS	• RWX • No tarring required	• Slow with parallel read/writes at scale • Only useful if you're in the cloud.

NFS-Server Provisioner

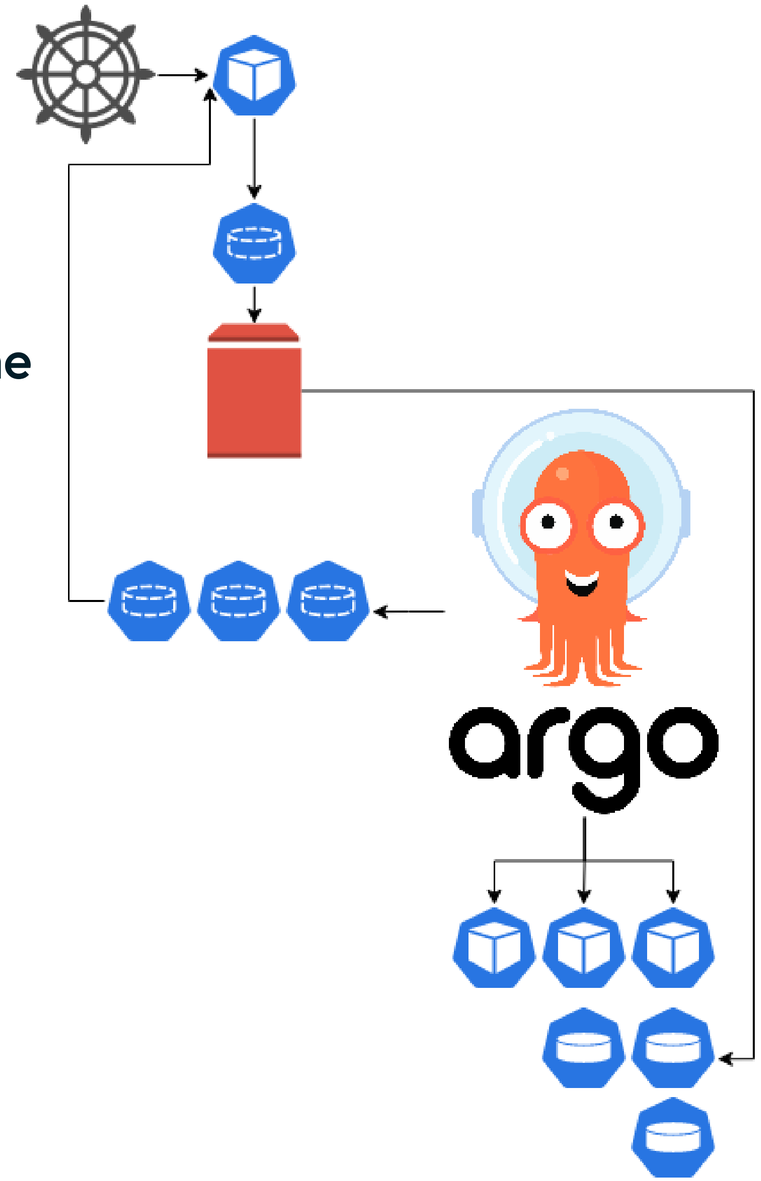
What is it?
https://github.com/kubernetes-sigs/nfs-ganesha-server-and-external-provisioner

	Pros	Cons
NFS-server-provisioner	• As EFS • Can be provisioned with a PVC • Useful on prem • Easy setup	• Slow with parallel read/writes at scale. • Small maintenance overhead.

Full setup including ArgoCD deployment and a CI example using it can be found via
https://github.com/pipekit/talk-demos

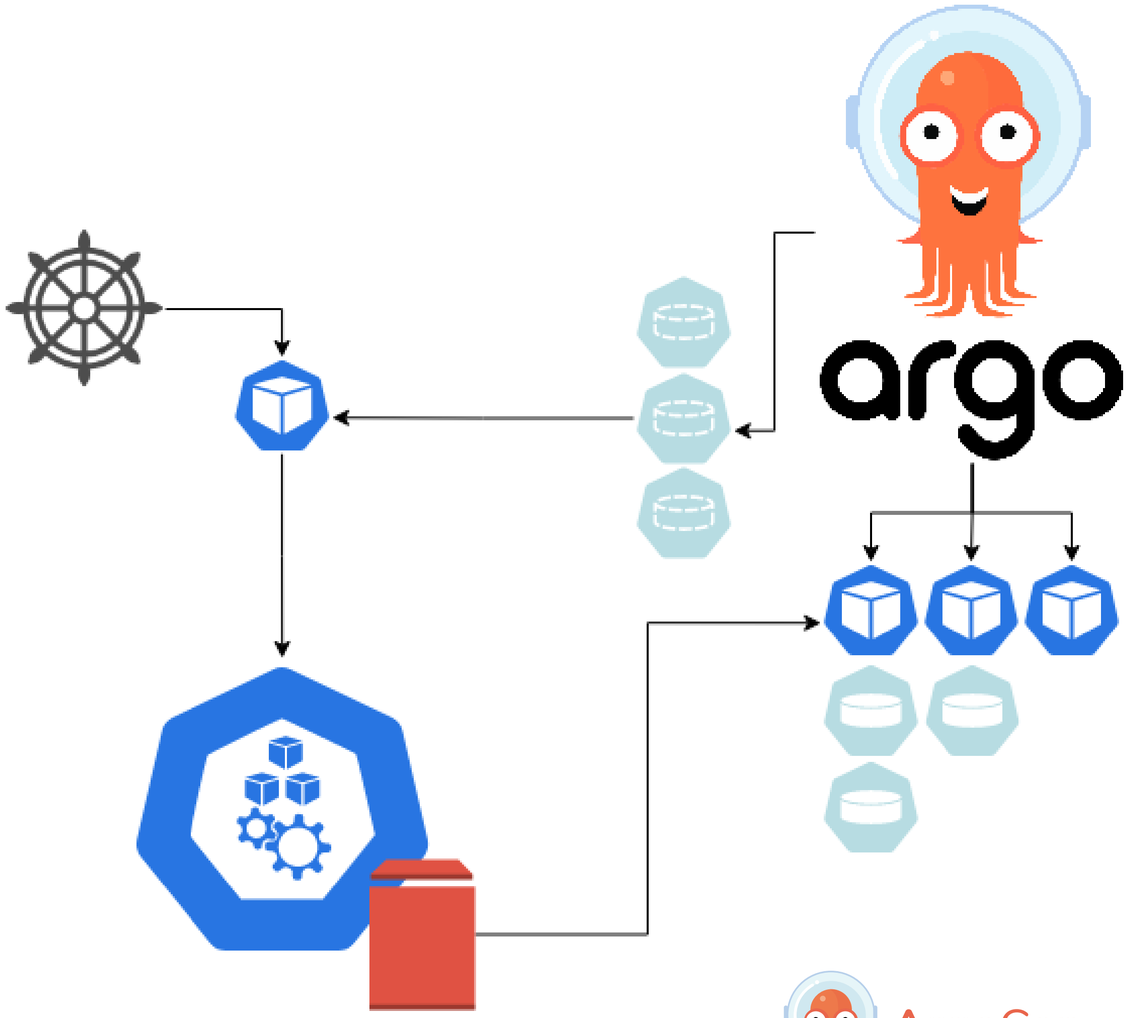
How does it work for semi-persistent data?

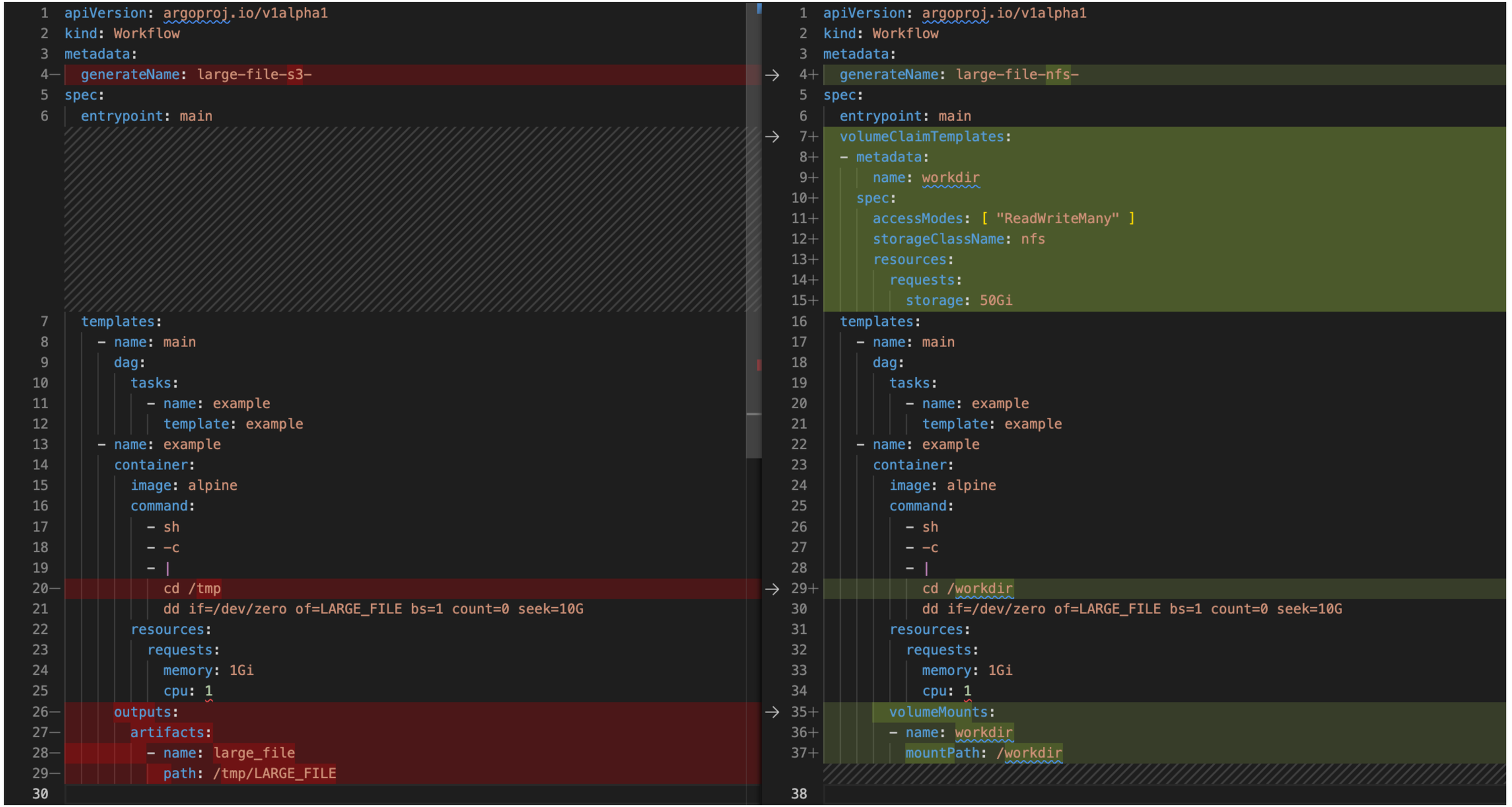
- Deployment creates an NFS Server Provisioner pod, PVC and storageClass.
- PVC provisions an EBS (RWO) disk and mounts it to the NFS Server Provisioner pod.
- We create our own PVCs using the new storageclass
- We are allocated Read-write-many persistent volumes, backed by EBS.
- In a Workflow, we mount these new volumes as required.



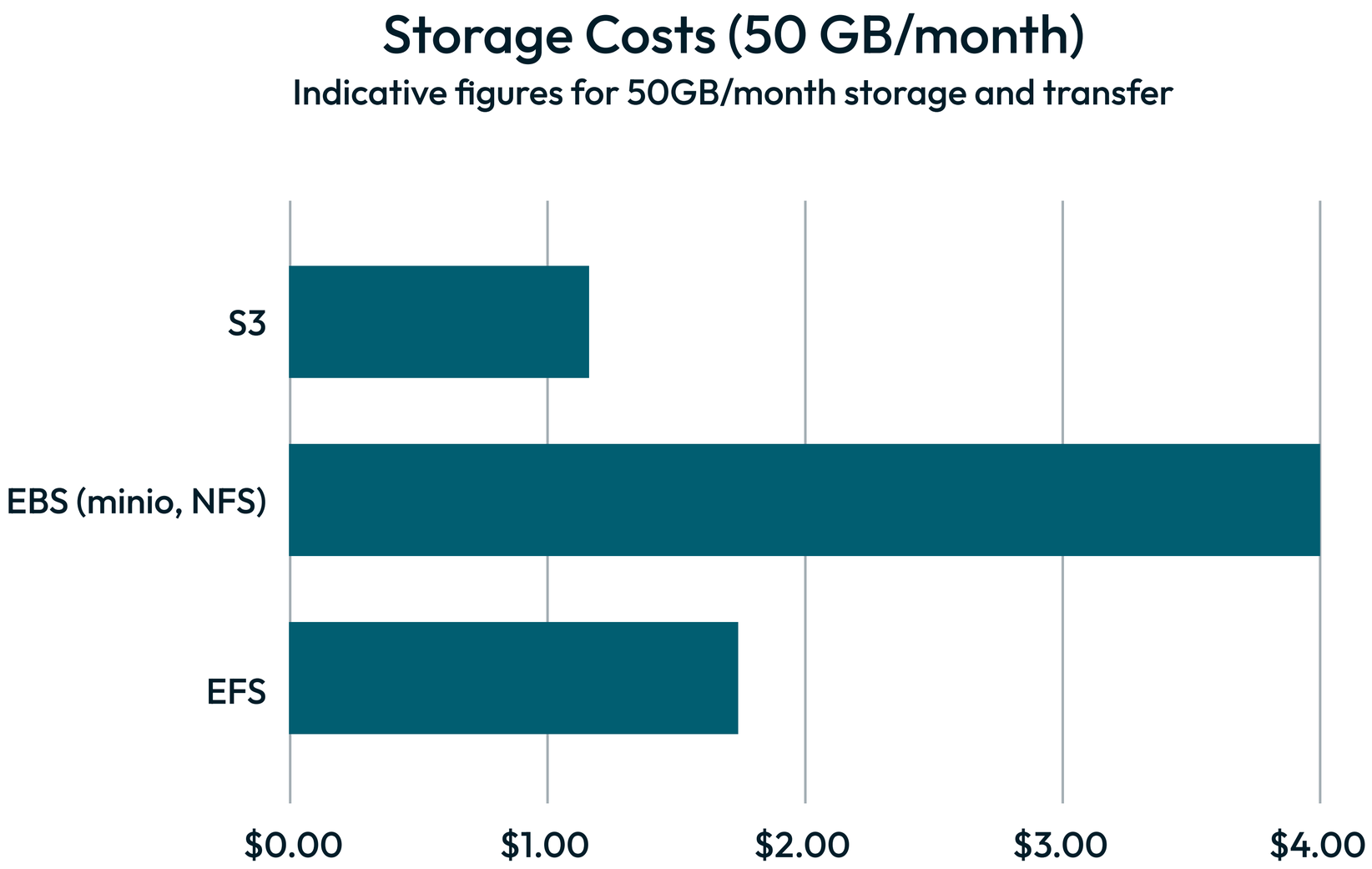
How does it work for transient data?

- Deployment creates an NFS Server Provisioner pod and storageClass.
- When we start a Workflow, we create volumeClaimTemplates to create PVCs that exist for the life of the Workflow.

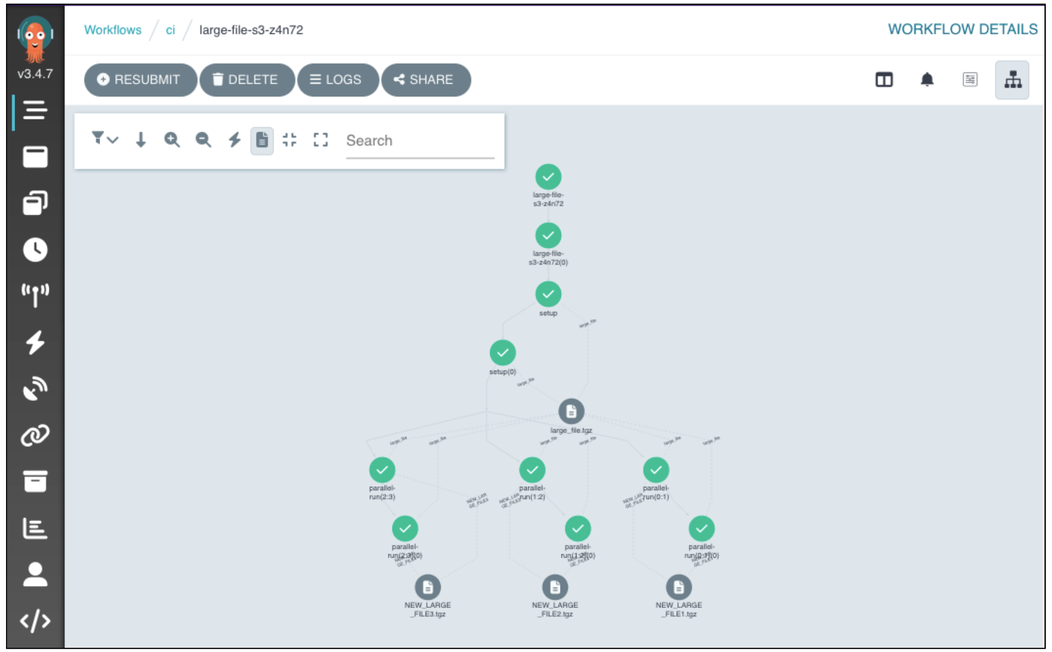


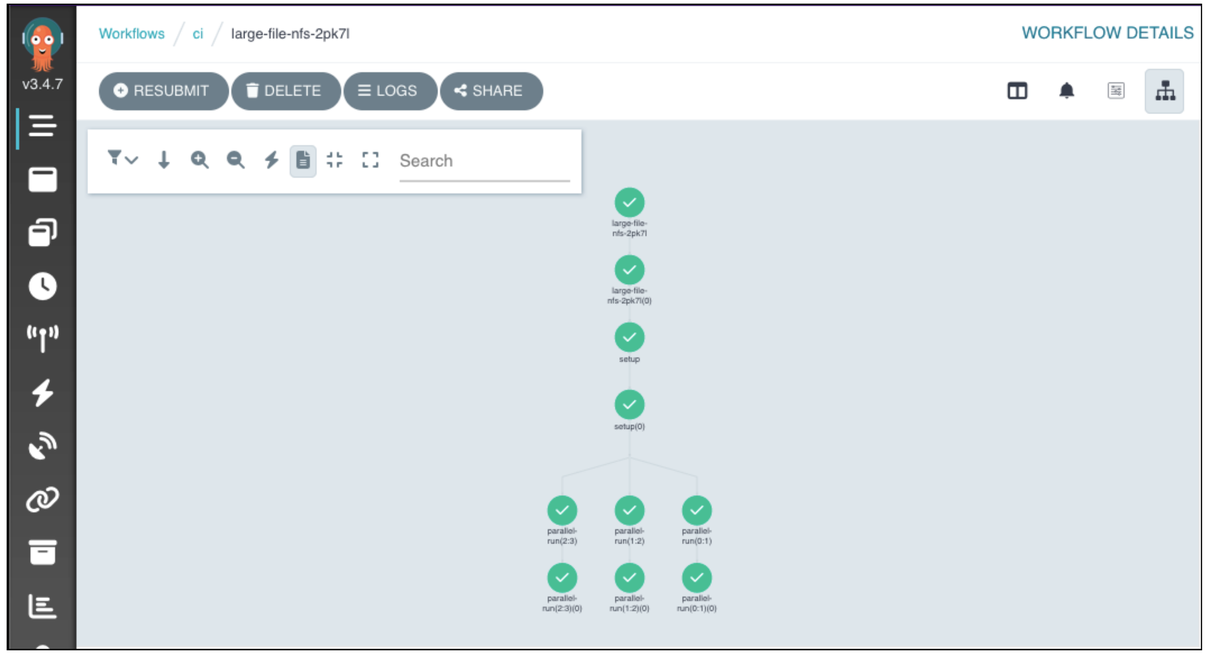

apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
generateName: large-file-s3-
spec:
entrypoint: main
templates:
- name: main
dag:
tasks:
- name: example
template: example
- name: example
container:
image: alpine
command:
- sh
- -c
- |
cd /tmp
dd if=/dev/zero of=LARGE_FILE bs=1 count=0 seek=10G
resources:
requests:
memory: 1Gi
cpu: 1
outputs:
artifacts:
- name: large_file
path: /tmp/LARGE_FILE
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
generateName: large-file-nfs-
spec:
entrypoint: main
volumeClaimTemplates:
- metadata:
name: workdir
spec:
accessModes: ["ReadWriteMany"]
storageClassName: nfs
resources:
requests:
storage: 50Gi
templates:
- name: main
dag:
tasks:
- name: example
template: example
- name: example
container:
image: alpine
command:
- sh
- -c
- |
cd /workdir
dd if=/dev/zero of=LARGE_FILE bs=1 count=0 seek=10G
resources:
requests:
memory: 1Gi
cpu: 1
volumeMounts:
- name: workdir
mountPath: /workdir

Comparing Cost



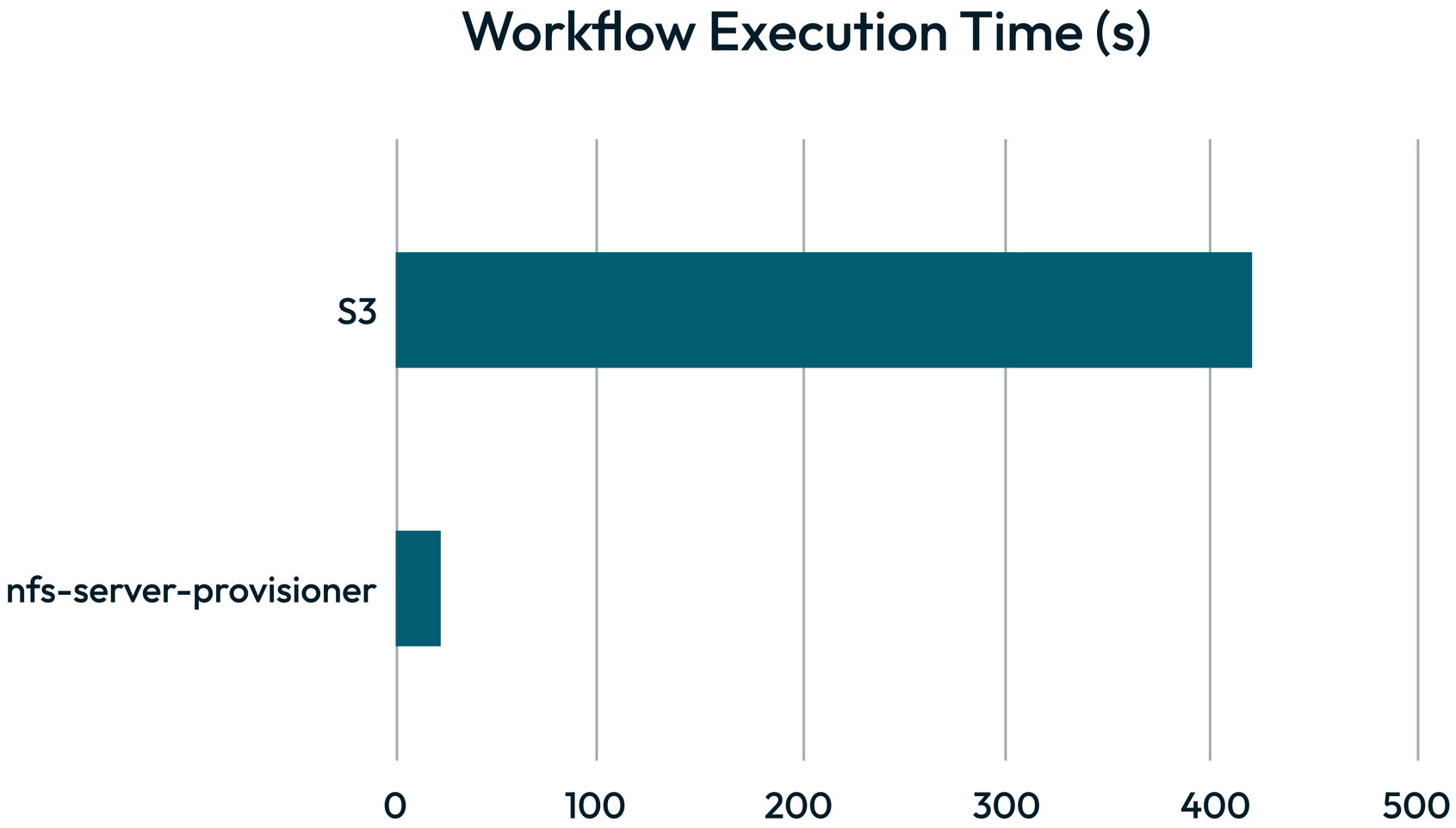
Testing Workflow Speed





These tests can be found at https://github.com/pipekit/talk-demos

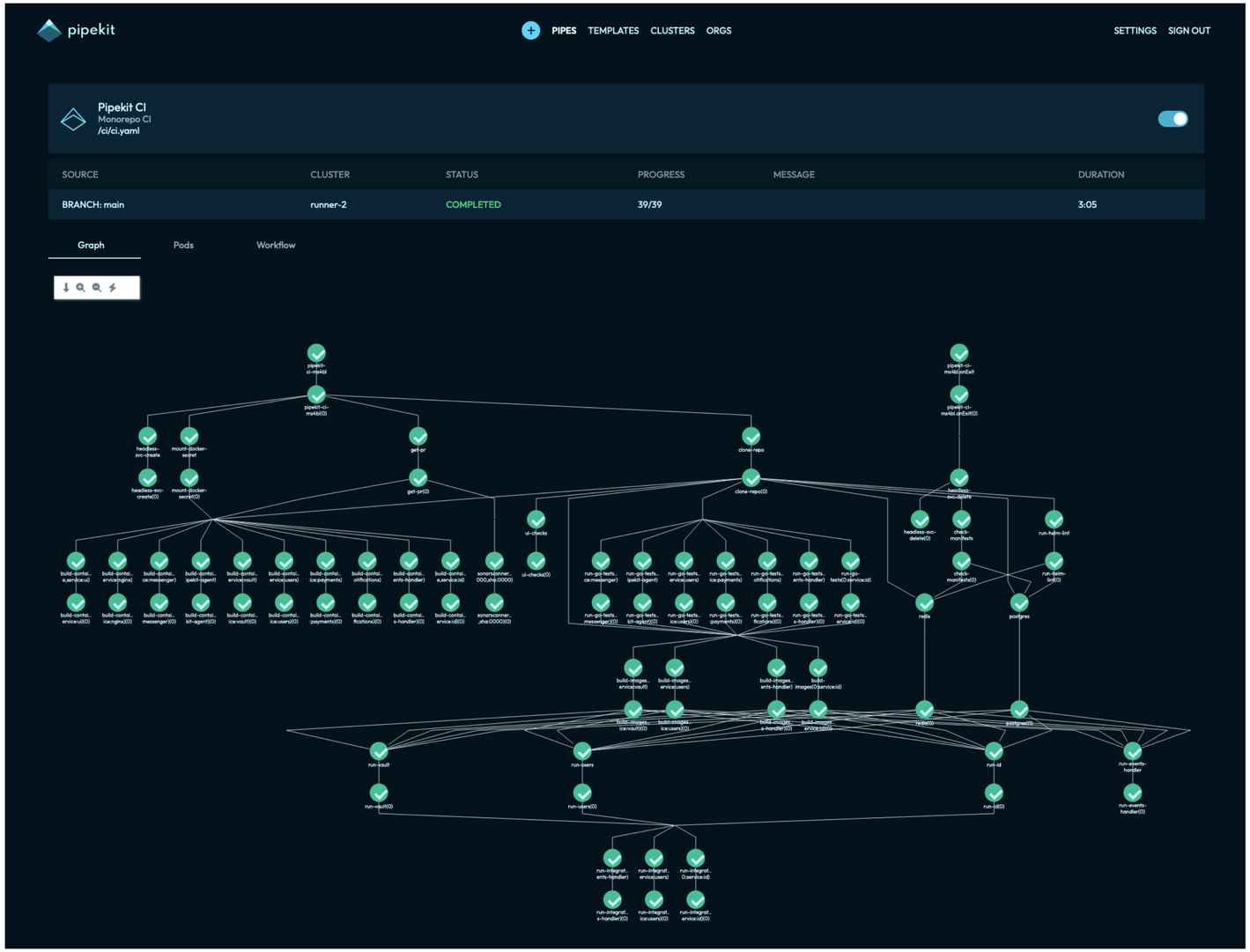
Timing


Workflow Execution Time (s)
S3
nfs-server-provisioner
0
100
200
300
400
500

NFS-Server Provisioner - Tips

- Fatten up your node disk
- Copy from NFS disk to ephemeral at each step if expecting a lot of parallel reads and writes.
- Install NFS-Server-Provisioner twice
  - One backed by an EBS disk for semi-persistent data
  - One backed by ephemeral node disks, for transient data

Conclusion


pipekit
PIPES
TEMPLATES
CLUSTERS
ORGS
SETTINGS
SIGN OUT
Pipekit CI
Monorepo CI
/ci/ci.yaml
SOURCE
CLUSTER
STATUS
PROGRESS
MESSAGE
DURATION
BRANCH: main
runner-2
COMPLETED
39/39
3:05
Graph
Pods
Workflow

Thank you!

ArgoCon

#ArgoCon2023