



ArgoCon

Mastering Argo Workflows at Scale: A Practical Guide to Scalability Excellence

Tim Collins - Pipekit | Alec Stansell - Fetch Analytics

INTROS



Tim Collins

Senior Infrastructure Engineer @ [Pipekit.io](https://pipekit.io)

- Argo Maintainer. Lives in the CNCF Argo Slack.
- Member of the Pipekit Services team.
- Growing a moustache to annoy his wife.



Alec Stansell

Data Scientist @

[FetchAnalytics.ai](https://fetchanalytics.ai)

- Optimising Argo Workflows past 6 months.
- Water industry, HPC, Data Engineering
- Painting, Free Diving, Gypsy Jazz



ArgoCon

PIPEKIT



Argo experts & maintainers



Save engineering time and up to 60% on compute costs



Add a team of 3 Argo maintainers & 5 Argo contributors to your Slack



Serving startups & Fortune 500 enterprises since 2021:

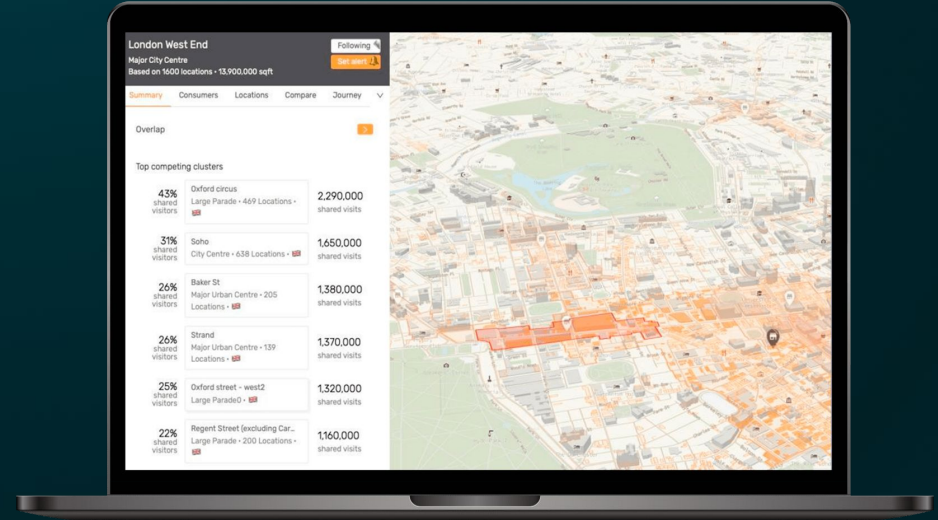
- Enterprise Support for Argo -> Ideal for Platform Eng teams scaling with Argo
- Control Plane for Argo Workflows -> Ideal for Data teams who don't speak K8s OR multi-cluster setups



ArgoCon

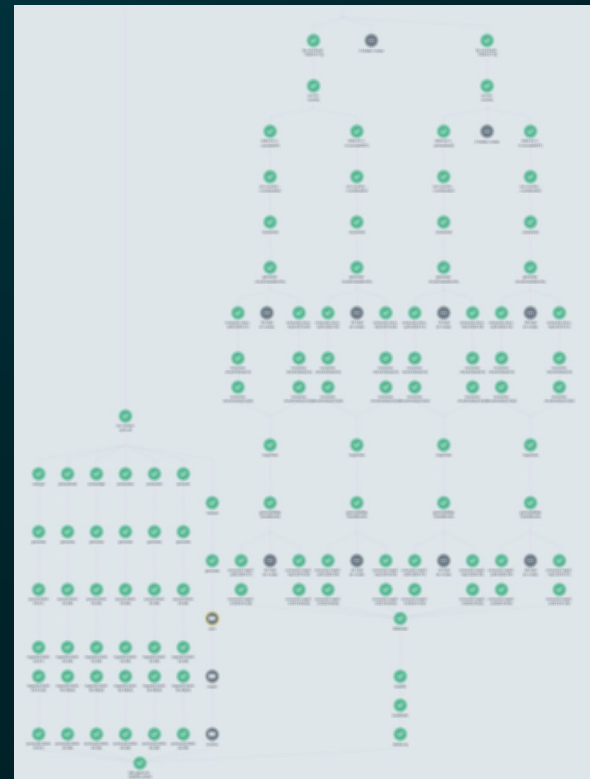
FETCH ANALYTICS

- How people interact with the physical world.
- Metrics: journeys, dwell time, popular locations.
- Coffee shop.
- UK, Sweden, Spain, Netherlands, South Africa.
- Scaling



ARGO WORKFLOWS AT FETCH ANALYTICS

- Core of our business
 - > 500 billion records twice a week.
 - Algorithms deriving mobility patterns.
 - Long running 16 hours.
- Problem:
 - Unreliable.
 - Feeling that it could be faster.
 - Limited transparency of what's going on.
- Share practical insights when scaling your pipeline.



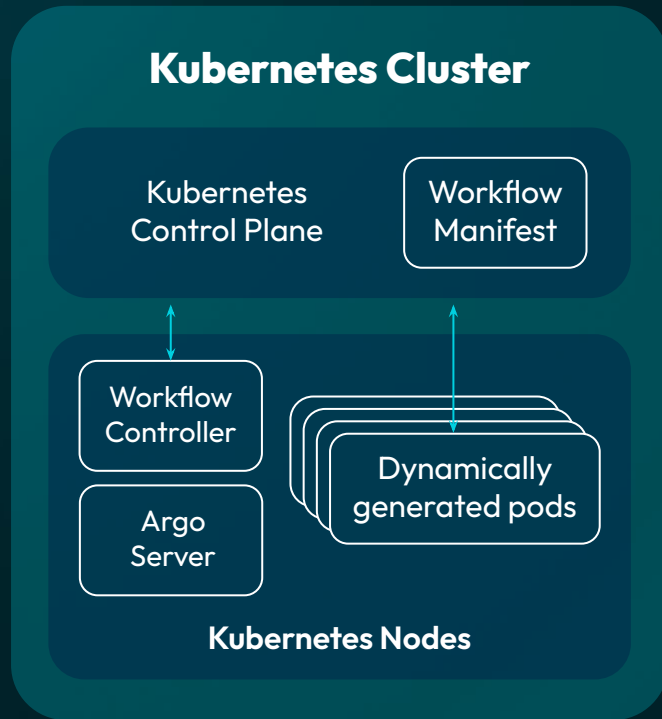
GUIDED ARGO PARAMETER GRID SEARCH

- Search for optimal setup.
- 66 experiments
- 15 Regressors
- Including:
 - Thread usage.
 - Resources per pod, step, node:
 - Network File Store vs Buckets (GCS).
 - Data chunk size and number.
 - Machine specs and type.
- Outputs
 - **Pipeline run time**
 - Reliability
 - Cost
- Results
 - Reliable, consistent, deterministic.
 - Reduced run time from 4 to 1 hour for our biggest market run.
 - “Sleep easy on Sundays” - CTO



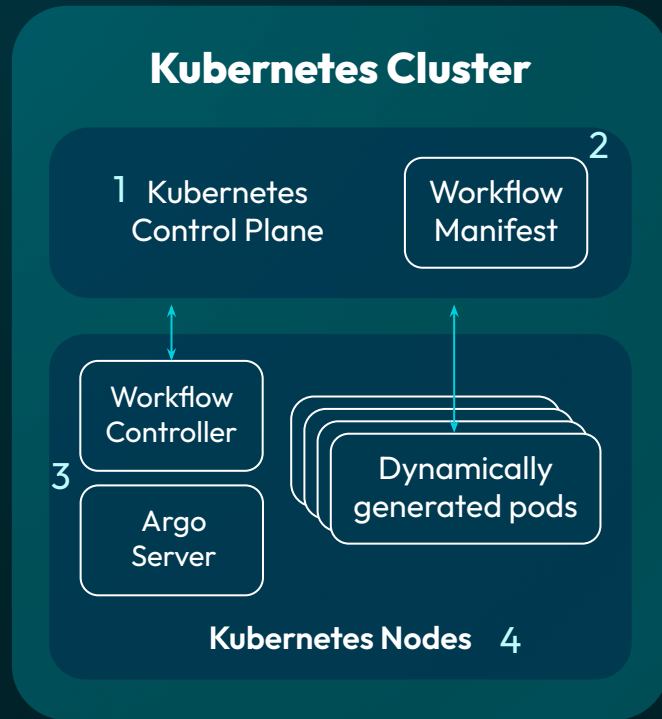
HOW ARGO WORKFLOWS WORKS

- User creates a Workflow manifest. This is stored in etcd.
- Workflow Controller starts working when it finds a workflow manifest.
- Controller asks the K8s API to create pod(s).
- Controller continually asks the k8s api for the status of the pod(s).
- Controller writes the status and parameter values of each workflow step back to the Workflow manifest (via k8s API).
- Users viewing the UI or using the API will cause the Argo Server to query the Kubernetes API for Workflows information.



WHAT THINGS COULD BREAK AT SCALE?

- 1 - Kubernetes Control Plane could get overwhelmed.
- 2 - Workflow manifest could get too large for etcd.
- 3 - Workflow Controller/Server could run out of resources.
- 4 - Cluster could run out of Kubernetes nodes (or other resources) to run the work.



1 - THE KUBERNETES API

Monitor the Kubernetes API Health

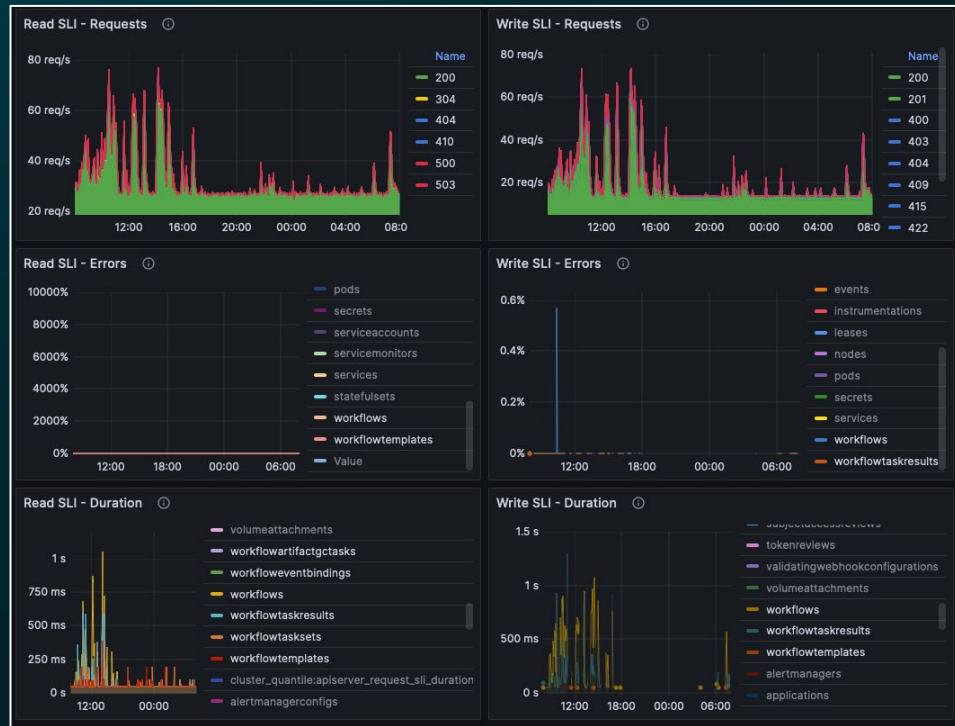
- Kube-prometheus-stack comes with a Grafana dashboard for this.
- Alternatively can use the Workflow Controller logs to get some insight.

Looking after the Kubernetes API

- `DEFAULT_QUEUE_TIME`.
- Workflow Parallelism.
- Limit how often the controller creates k8s

resources: `resourceRateLimit`

- Use Kubernetes 1.27+.
- Spread the work across more Clusters.
- Consider what else the cluster is used for.



2 - LARGE WORKFLOW MANIFEST

Parameters, steps and statuses - everything written back to the Workflow manifest.

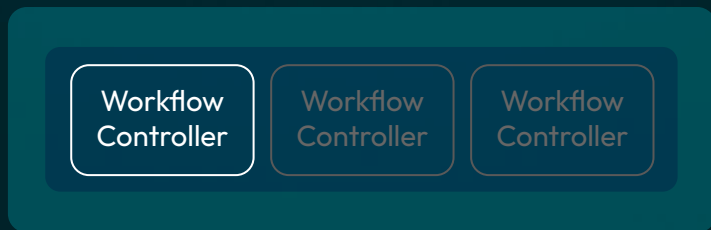
- Are you putting too much data into parameters?
 - [Consider using a shared \(RWX\) disk or artifacts instead](#)
- Are you running too many pods in one workflow?
 - [Consider splitting into smaller workflows](#)
- Still > 3mb
 - [Consider workflow offloading to a Database](#)

```
Status:      Error
Message:     Request entity too large: limit is 3145728
Conditions:
PodRunning   True
Completed    True
Created:     Mon Oct 23 23:48:52 +0200 (8 hours ago)
Started:     Mon Oct 23 23:48:52 +0200 (8 hours ago)
Finished:    Tue Oct 24 01:12:20 +0200 (6 hours ago)
Duration:    1 hour 23 minutes
Progress:    4483/5205
```



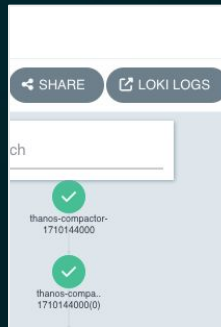
3 - WORKFLOW CONTROLLER RESOURCES

- Resource requests/limits on the Controller are not set by either the official release manifest or the Helm chart.
- The types of workflow you execute will determine the resource usage of the controller.
- You can only have one running Workflow Controller replica at a time.
 - Make use of `PriorityClasses`.
 - `LEADER_ELECTION_DISABLE` is `FALSE` by default, even if you have one workflow controller.



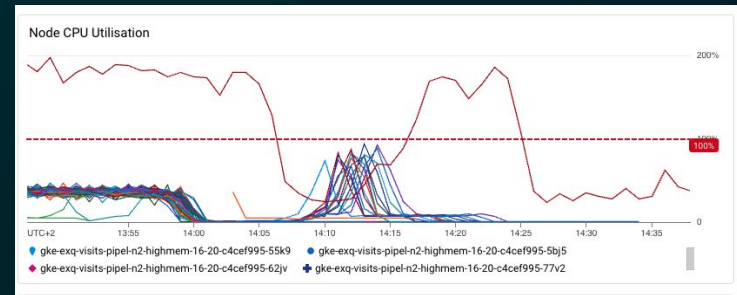
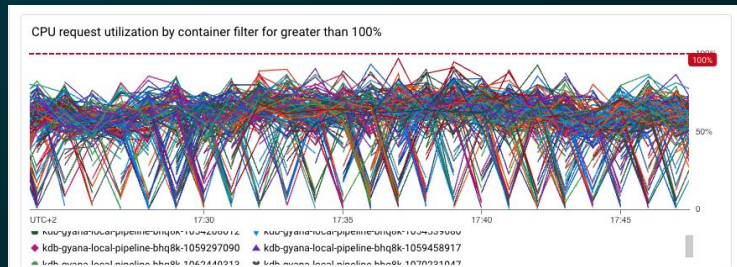
3 - WORKFLOW SERVER RESOURCES

- Server - CLI and UI so not vital for running workflows.
- View approximately 1000 workflows before things get too slow.
 - Depending on size
- Delete old workflows (use `ttlStrategy`) from your cluster.
 - `argo delete -older 10d`
 - If you need workflows, consider archiving to a database.
- Use a dedicated log aggregation tool to view Workflow logs
 - Link to UI via the controller configmap.
- Use direct `kubectl/CLI` commands.
 - `argo get @latest -watch`
- Resource requests/limits on the server are not set by either the official release manifest or the Helm chart.



TUNING NODES AND POD RESOURCES

- Monitoring.
- 80% Roughly node and pod resource utilisation.
- Split out steps with different requirements.
- Requests to ensure other pods don't steal your resources.
- Mindful of other processes leading to performance degradation sidecars, metrics server.
- Tune your workflow controller resources.



4 - RUNNING OUT OF KUBERNETES NODES

Look for pending pods

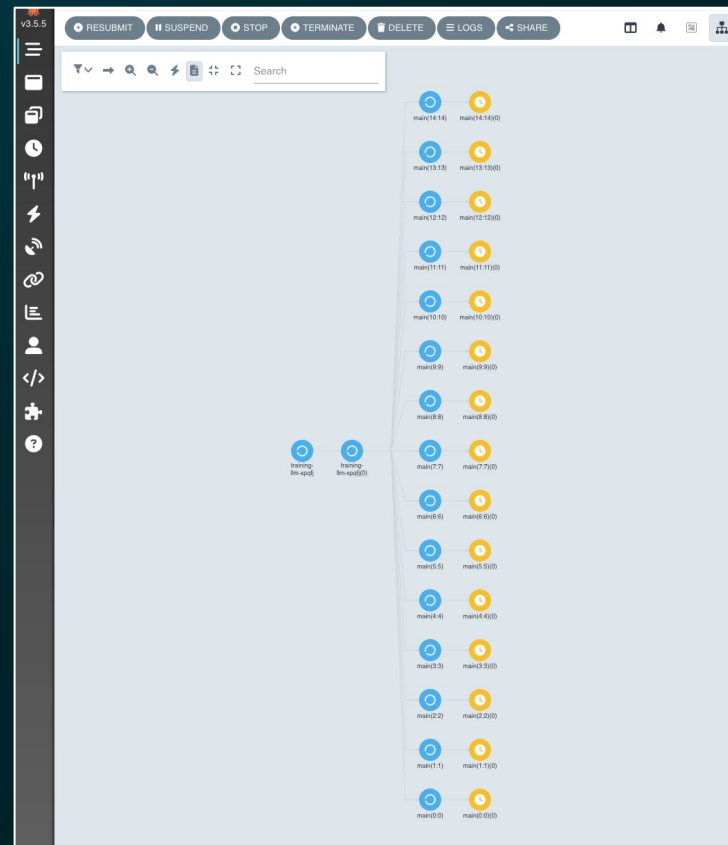
- Pods stuck in pending may indicate that you don't have the Kubernetes nodes at hand.

Autoscaling

- If a node is connected to your cluster, you're already paying for it.
- Scale down when you're not doing things. Consider scaling to 0 at night/weekends (if applicable).

Consider ARM and Spot

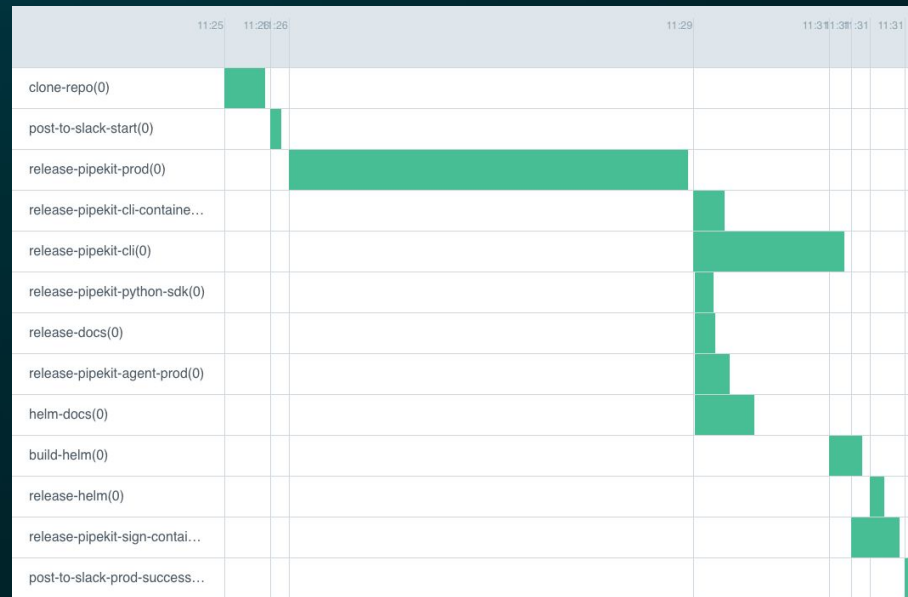
- Cost-savings.
- For Spot, you need to be able to handle sudden node-eviction in your Workflow steps.



WHY IS MY WORKFLOW SLOW?

If you don't have observability, look at the gaps between steps in the Timeline view:

- Pod creation/init time
 - Image size
 - Optimise your image size
 - Image caching
 - <https://github.com/XenitAB/spiegel>
 - Image prefetching
 - Custom Kubernetes node images
 - Many different images (lots of fresh pulls)
 - Can you combine similar images into one?
 - Use a good image tagging methodology (or better yet use the image digest)
 - Container registry rate limiting
 - Do you have enough Kubernetes nodes?
 - Waiting for volumes to mount



WHY IS MY WORKFLOW SLOW?

Workflow Design

As a rough rule of thumb, assume the startup time of a pod is at least 30s:

- If a step is not likely to be parallelised, consider combining it with other steps that won't be run in parallel.
- If a step requires the output of a previous step and both steps are not going to be parallelised, run them in the same pod.
- If a job is reasonably short (2 minutes or less?) and the speed of the execution is critical, consider not using Workflows for the job.
- Consider putting many short-lived steps into one containerset. This results in a single pod.

Large output artifacts

- Takes time to (un)compress and upload/download the files
- You can measure the time it takes to fetch/push artifacts by looking at the logs of the init container.

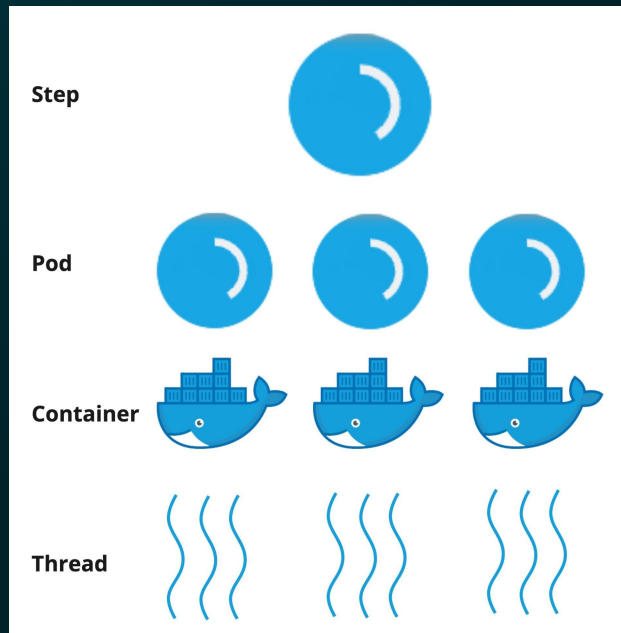
Resource requests for workflow steps. Remember the init/wait containers too.

GC pods after completion



WHERE SHOULD PARALLELISM OCCUR?

- How do you break up your algorithms to scale horizontally?
- Thread, container, pod, node parallelism:
 - Background processes + metrics + side cars
- High Pods Greater load on controller
 - Right number of pods a node / run time of a pod couple of minutes.
- Small vs large workflows - cluster utilisation.



OBSERVABILITY

Open Source Grafana dashboard

- grafana.com/grafana/dashboards/20348-argo-workflows-metrics/
- Import ID: 20348

IDEAL IF:

- You have Prometheus/Grafana set up already.
- You tried the existing one and found it no longer works.
- You want to have a base dashboard to work your own metrics into.



OBSERVABILITY

Observability Enhancements to Argo Workflows Controller (expected in v3.6)

- Large number of new metrics and fixes to old metrics.
- Helping you answer these questions:
 - How long does workflowTemplate foo take to run?
 - When did cronWorkflow bar last run?
 - How many workflows Failed in the last week?
 - What's the reason my pods are pending?

Top 5 Pending Pods by reason in the last 1d

prometheus/runner-prometheus : argo : PodInitializing	1391
prometheus/runner-prometheus : argo : Unschedulable	1242
prometheus/runner-prometheus : argo : ErrImagePull	82
prometheus/runner-prometheus : argo : ImagePullBackOff	50
prometheus/runner-prometheus : argo : ErrImagePull: pull QPS exceeded	40

- Read and upvote the proposal: github.com/argoproj/argo-workflows/issues/12589

Workflow Template Runtime (95th Percentile)



FREE WORKFLOW METRICS BY PIPEKIT

“79% of organizations with observability have saved time or money”

Uncover deep insights into your Argo Workflows and Kubernetes environments.

Gain real-time insights into your systems' health.

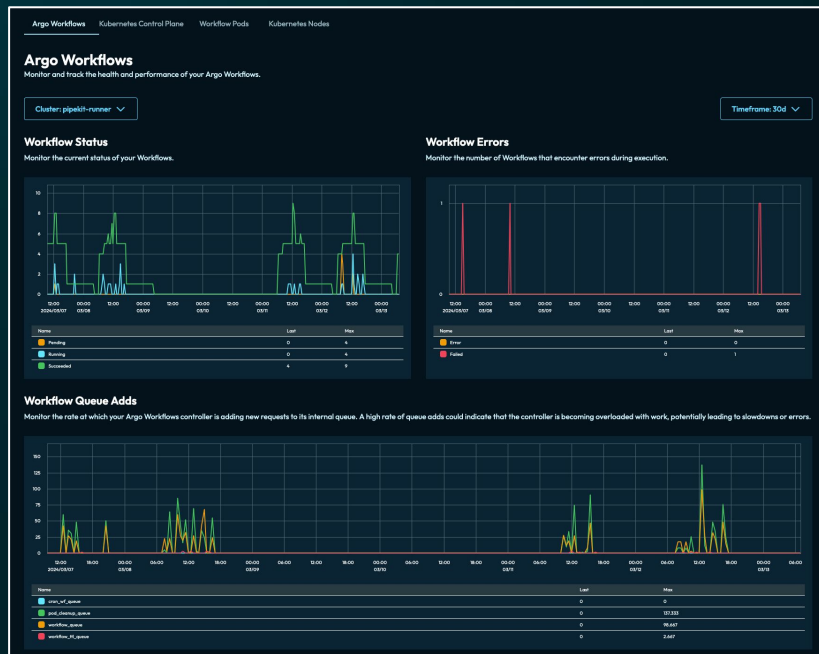
- Open source metrics collector (OTEL).
- We show you all the data we collect.

GET STARTED:

- Register your interest at pipekit.io/metrics-signup.

IDEAL IF:

- You don't have an existing observability stack.
- You don't have time to learn how to compose dashboards.
- You want to take your Argo Workflows scaling to the next level with minimal effort.



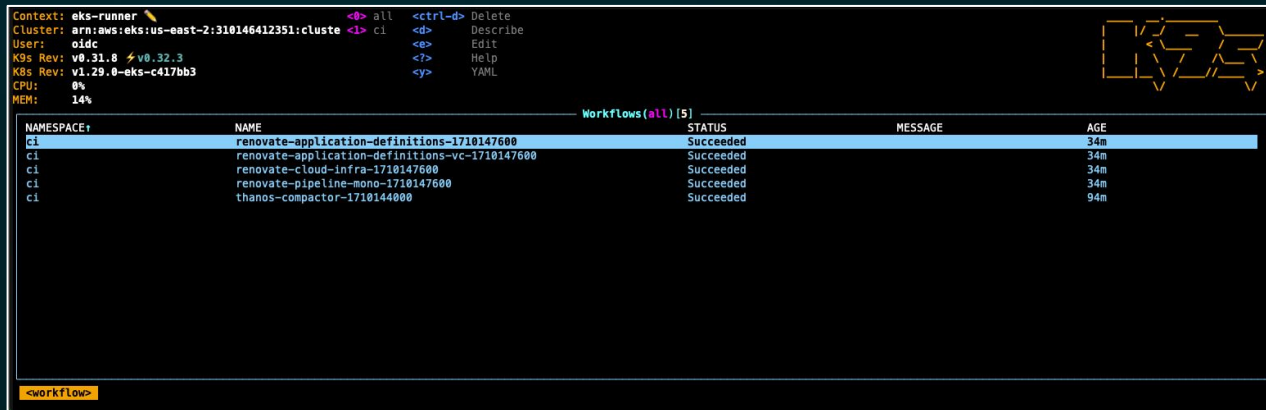
ArgoCon

FETCH'S FINDINGS

- Monitoring Prometheus + Kube State Metrics + Argo Metrics.
- [NFS in production see Tims talk.](#)
- [No container lifecycle hooks.](#)
- Using new tools:
 - [Kustomize](#) for more DRY YAML and cost savings.
 - [K9S](#) CLI for cluster management.
 - [Argo v3.5](#) better observability and performance.
- Performant at pod, container and node level.

- Optimisations:

- ¼ the time.
- Deterministic.
- Reliable



The screenshot shows the K9s CLI interface. At the top, it displays cluster information: Context: eks-runner, Cluster: arn:aws:eks:us-east-2:310146412351:cluster, User: oidc, K9s Rev: v0.31.8, K8s Rev: v1.29.0-eks-c417bb3, CPU: 0%, MEM: 14%. A menu on the right includes options like all, c1, Delete, Describe, Edit, Help, and YAML. Below this, a table titled 'Workflows(all) [5]' lists workflows with columns for Namespace, Name, Status, Message, and Age. All workflows are in a 'Succeeded' state.

Namespace	Name	Status	Message	Age
c1	renovate-application-definitions-1710147600	Succeeded		34m
c1	renovate-application-definitions-vc-1710147600	Succeeded		34m
c1	renovate-cloud-infra-1710147600	Succeeded		34m
c1	renovate-pipeline-mono-1710147600	Succeeded		34m
c1	thanos-compactor-1710144000	Succeeded		94m



FREE STUFF

GITHUB REPO

- github.com/pipekit/talk-demos

GRAFANA DASHBOARD

- grafana.com/grafana/dashboards/20348-argo-workflows-metrics/
- Import ID: 20348

FREE WORKFLOW METRICS BY PIPEKIT

- Register your interest at pipekit.io/metrics-signup



FREE ARGO/INFRASTRUCTURE HELP & ADVICE

- Booth E34
- Weekly Office Hours (pipekit.io/office-hours)



FETCH ANALYTICS

Have reached out on trial for Fetch - awaiting feedback on option.

Keen to add a demo to your repo or equivalent.



ArgoCon