ArgoCon
EUROPE

Demystifying Argo Workflows:
An Architectural Deep Dive

Darko Janjić
Senior Software Engineer,
Pipekit

Becky Pauley
Solutions Engineer,
Venafi Jetstack Consult

Image credit: Phil Botha, Unsplash

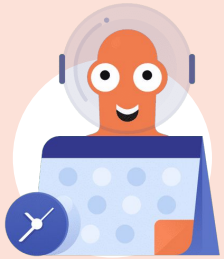
What is Argo Workflows?

Argo Workflows

Kubernetes-native workflow engine supporting DAG and step-based workflows

Argo CD

Declarative continuous delivery with a fully-loaded UI



Argo Events

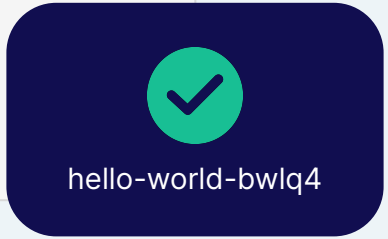
Advanced Kubernetes deployment strategies such as Canary and Blue-Green made easy.

Argo Rollouts

Event based dependency management for Kubernetes.

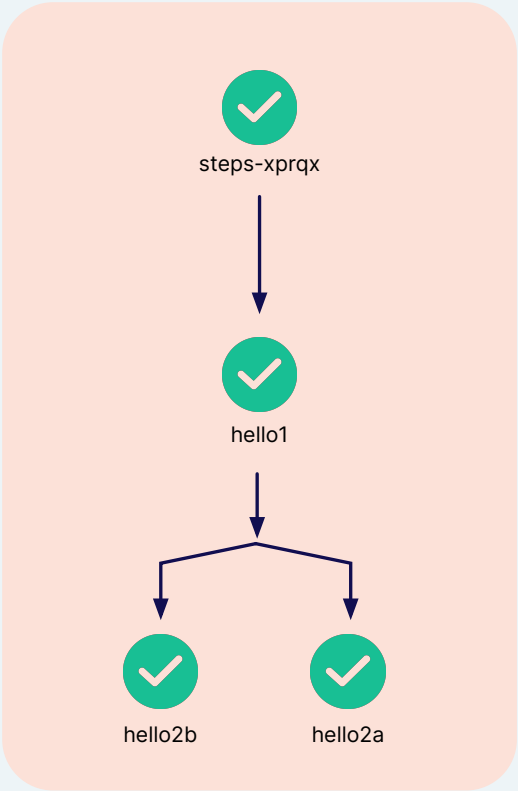
A workflow is...

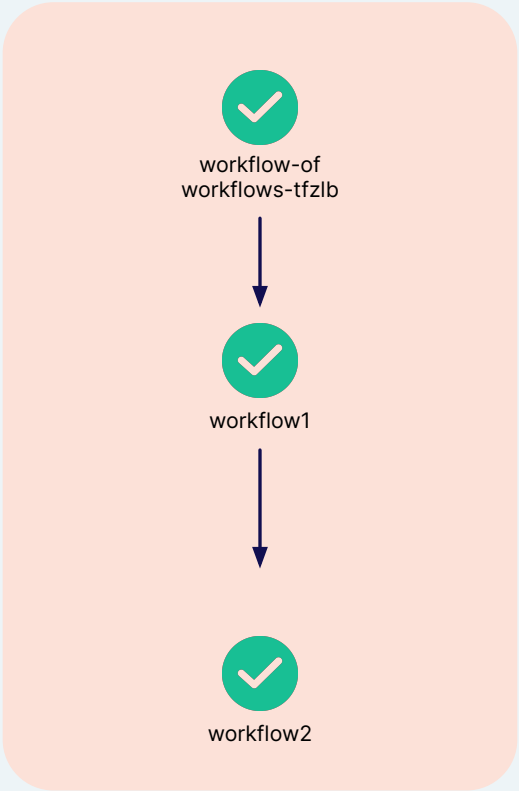
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow                  # new type of k8s spec
metadata:
  generateName: hello-world-    # name of the workflow spec
spec:
  entrypoint: whalesay          # invoke the whalesay template
  templates:
    - name: whalesay              # name of the template
      container:
        image: docker/whalesay
        command: [ cowsay ]
        args: [ "hello world" ]
        resources: # limit the resources
          limits:
            memory: 32Mi
            cpu: 100m
```

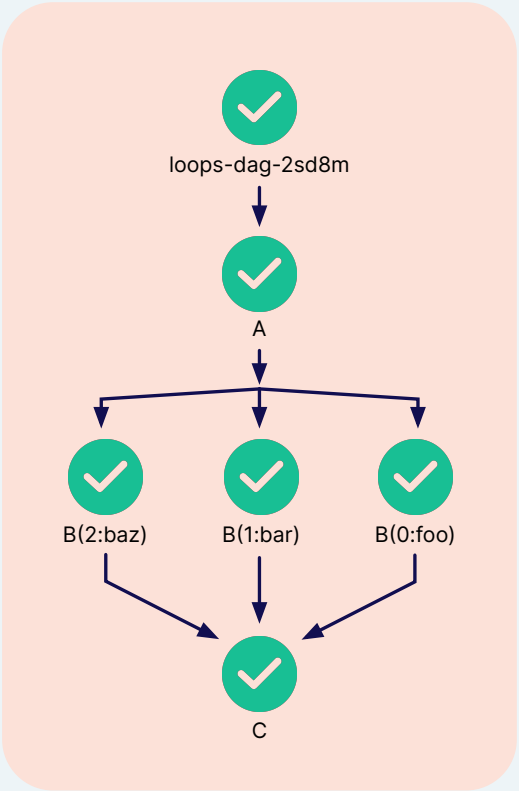


Ref: https://argo-workflows.readthedocs.io/en/latest/walk-through/hello-world/

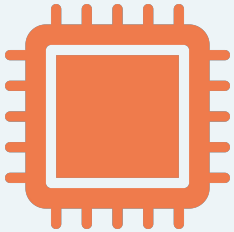
A workflow is...



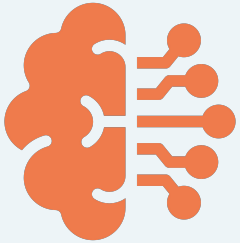




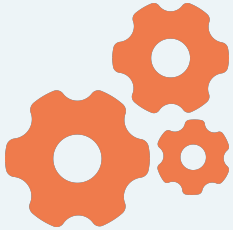
What is Argo Workflows?



Easily run
compute-intensive jobs



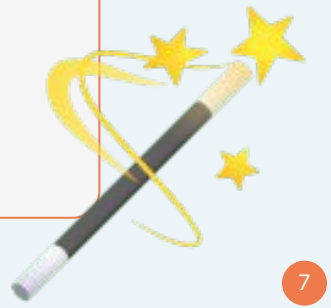
Machine learning,
data processing



Automating infrastructure
and CI/CD pipelines

What is Argo Workflows?

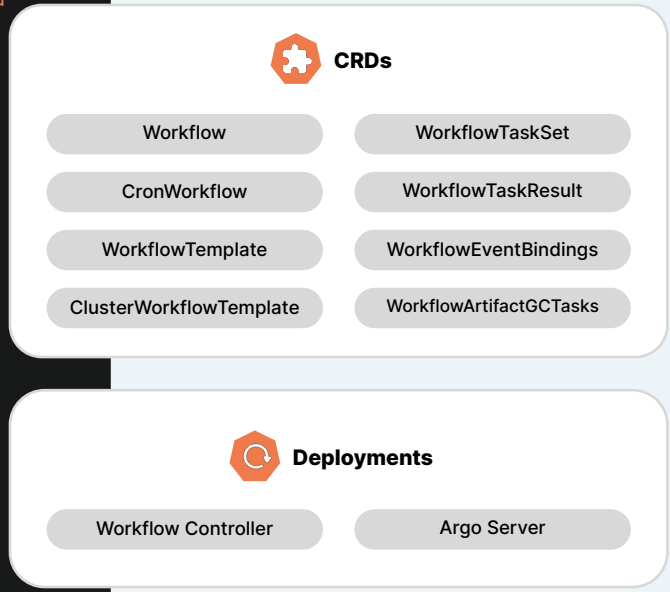
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow                  # new type of k8s spec
metadata:
  generateName: hello-world-    # name of the workflow spec
spec:
  entrypoint: whalesay          # invoke the whalesay template
  templates:
    - name: whalesay              # name of the template
      container:
        image: docker/whalesay
        command: [ cowsay ]
        args: [ "hello world" ]
        resources: # limit the resources
          limits:
            memory: 32Mi
            cpu: 100m
```



Ref: https://argo-workflows.readthedocs.io/en/latest/walk-through/hello-world/

What exists in my cluster?

```
customresourcedefinition.apiextensions.k8s.io/cronworkflows.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/clusterworkflowtemplates.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workflowartifactgctasks.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workfloweventbindings.argoproj.io
customresourcedefinition.apiextensions.k8s.io/workflowtaskresults.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workflowtasksets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workflowtemplates.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/workflows.argoproj.io created
serviceaccount/argo created
serviceaccount/argo-server created
role.rbac.authorization.k8s.io/argo-role created
clusterrole.rbac.authorization.k8s.io/argo-aggregate-to-admin created
clusterrole.rbac.authorization.k8s.io/argo-aggregate-to-edit created
clusterrole.rbac.authorization.k8s.io/argo-aggregate-to-view created
clusterrole.rbac.authorization.k8s.io/argo-cluster-role created
clusterrole.rbac.authorization.k8s.io/argo-server-cluster-role created
rolebinding.rbac.authorization.k8s.io/argo-binding created
clusterrolebinding.rbac.authorization.k8s.io/argo-binding created
clusterrolebinding.rbac.authorization.k8s.io/argo-server-binding created
configmap/workflow-controller-configmap created
service/argo-server created
priorityclass.scheduling.k8s.io/workflow-controller created
deployment.apps/argo-server created
deployment.apps/workflow-controller created
```



Recommend installing from: https://github.com/argoproj/argo-helm

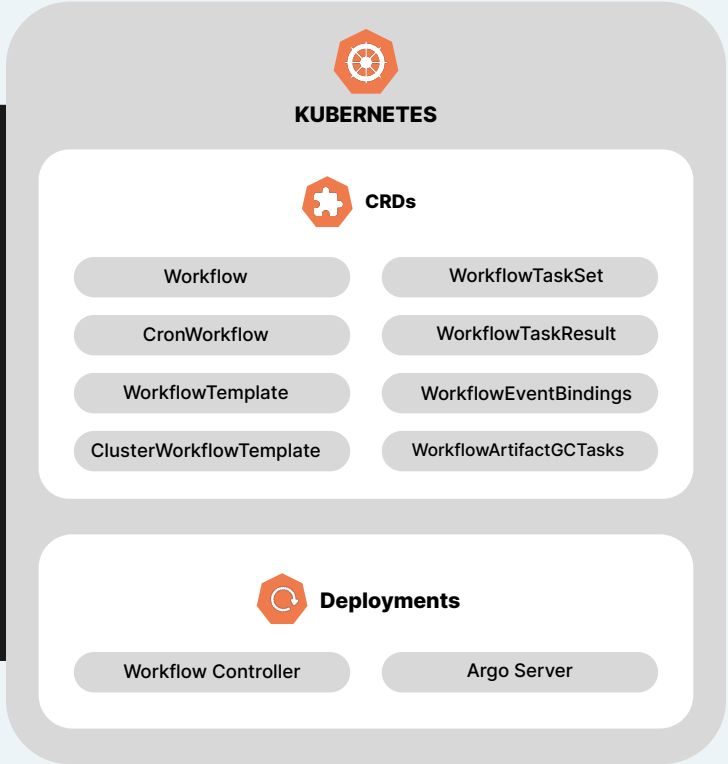
Custom Resource Definitions

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: workflows.argoproj.io
spec:
  group: argoproj.io
  names:
    kind: Workflow
    listKind: WorkflowList
    plural: workflows
    shortNames:
    - wf
    singular: workflow
  scope: Namespaced
  versions:
  - additionalPrinterColumns:
    - description: Status of the workflow
      jsonPath: .status.phase
      name: Status
      type: string
    - description: When the workflow was started
      format: date-time
      jsonPath: .status.startedAt
      name: Age
      type: date
    - description: Human readable message indicating
        is in this condition.
      jsonPath: .status.message
      name: Message
      type: string
```

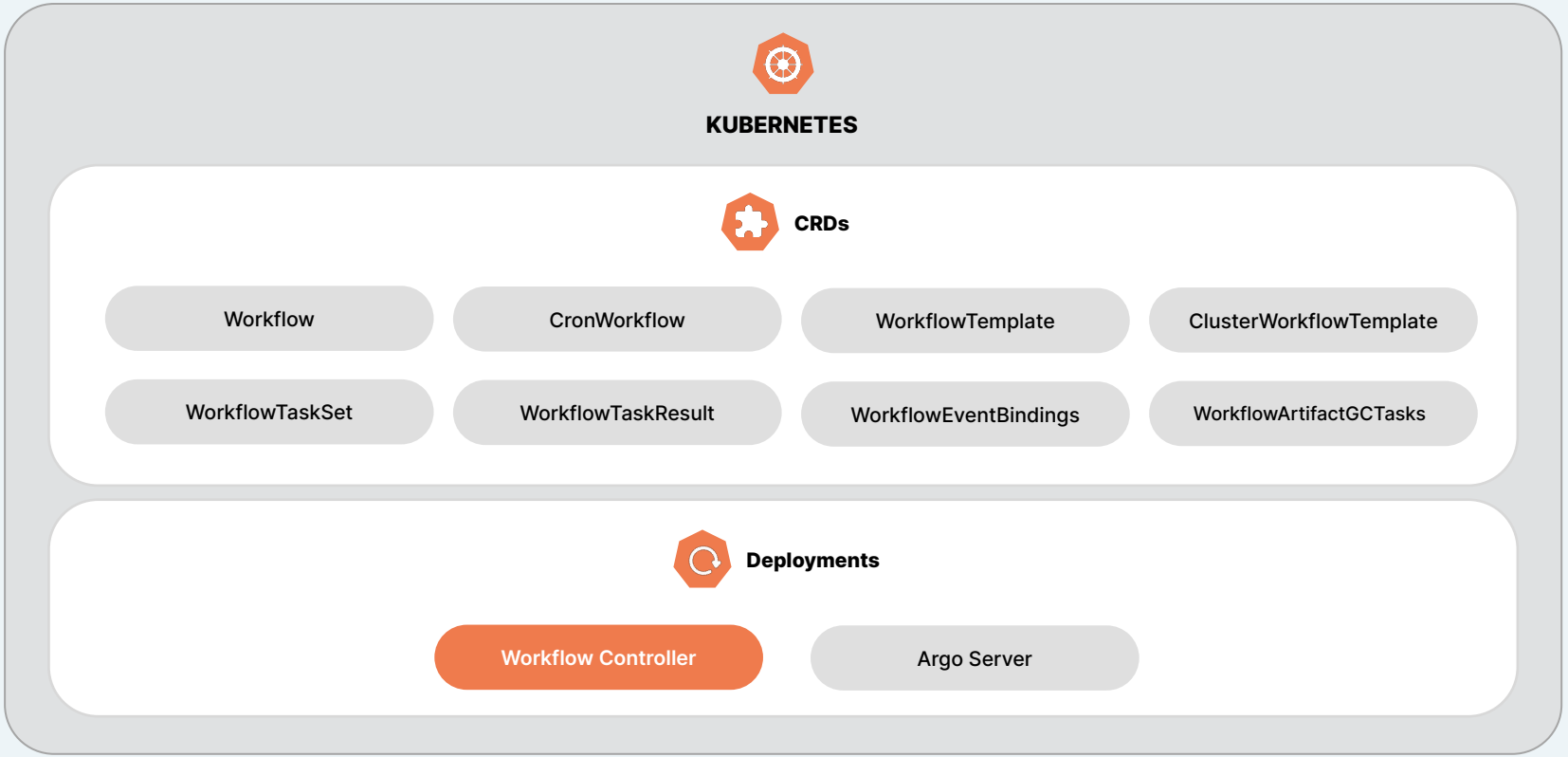
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-
spec:
  entrypoint: whalesay
  templates:
    - name: whalesay
      container:
        image: docker/whalesay
        command: [ cowsay ]
        args: [ "hello world" ]
        resources: # limit the r
          limits:
            memory: 32Mi
            cpu: 100m
```

Custom Resource Definitions

```
NAME
clusterworkflowtemplates.argoproj.io
cronworkflows.argoproj.io
workflowartifactgctasks.argoproj.io
workfloweventbindings.argoproj.io
workflows.argoproj.io
workflowtaskresults.argoproj.io
workflowtasksets.argoproj.io
workflowtemplates.argoproj.io
```



How do we orchestrate Workflows?



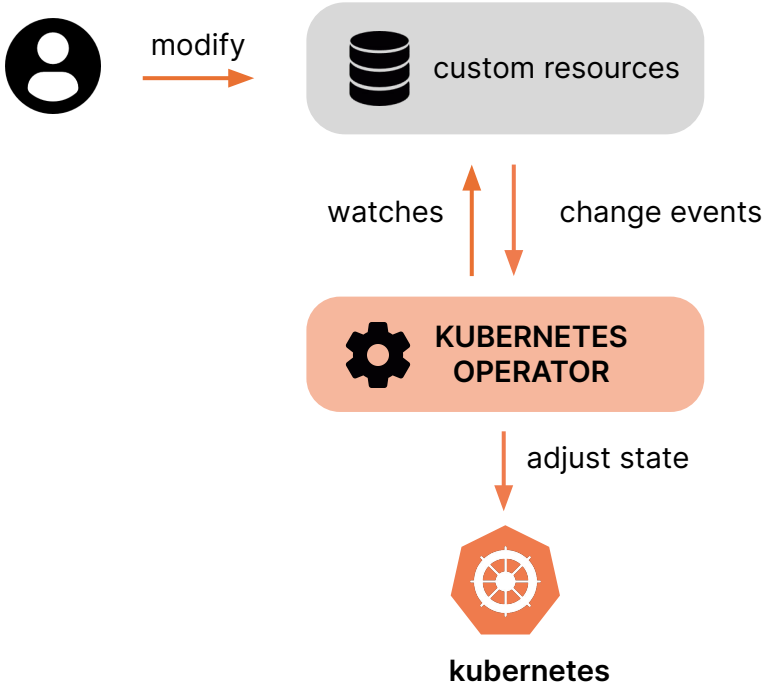
Workflow Controller

Main component of Argo Workflows

Works with custom resources

Manages workflows and everything Argo Workflows related (CronWorkflows, WorkflowTemplates...)

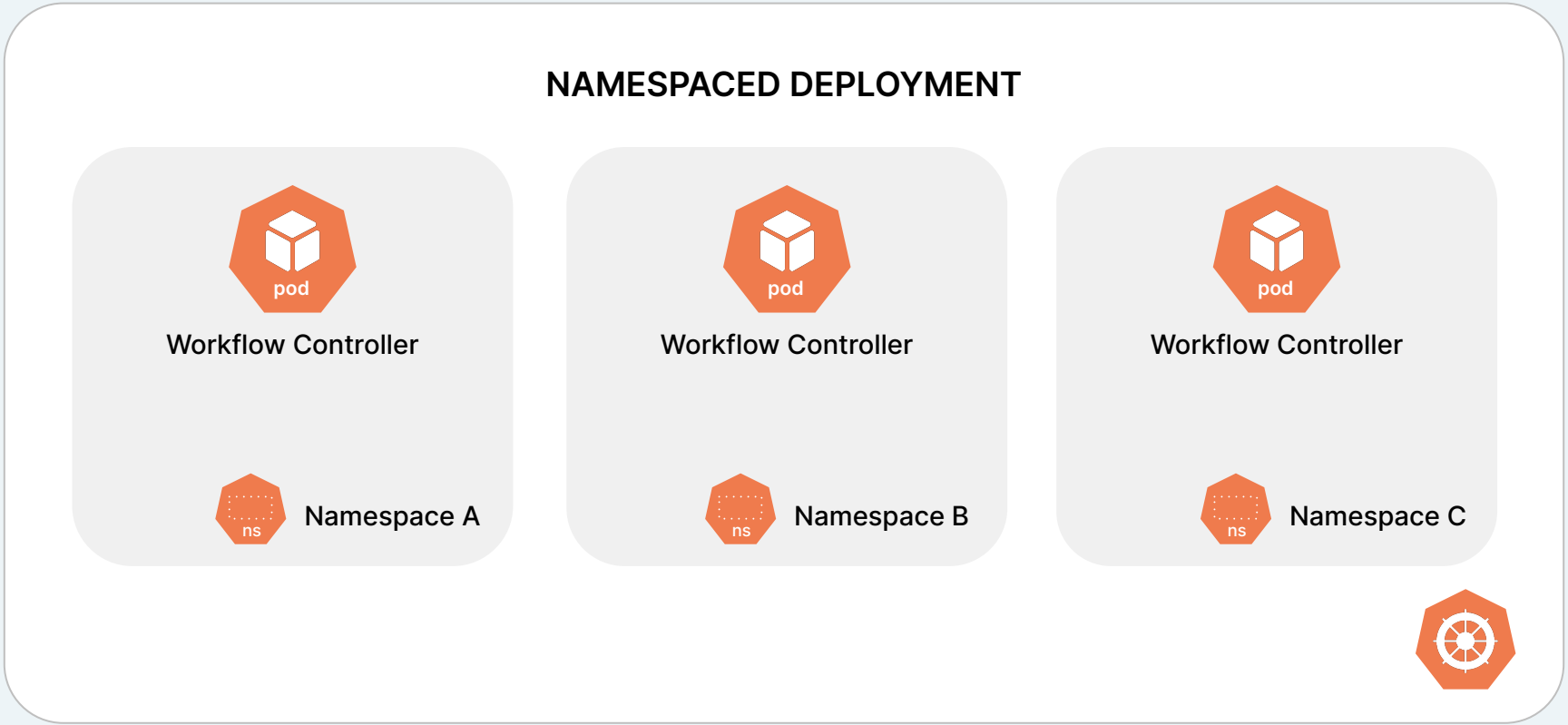
Workflow Controller



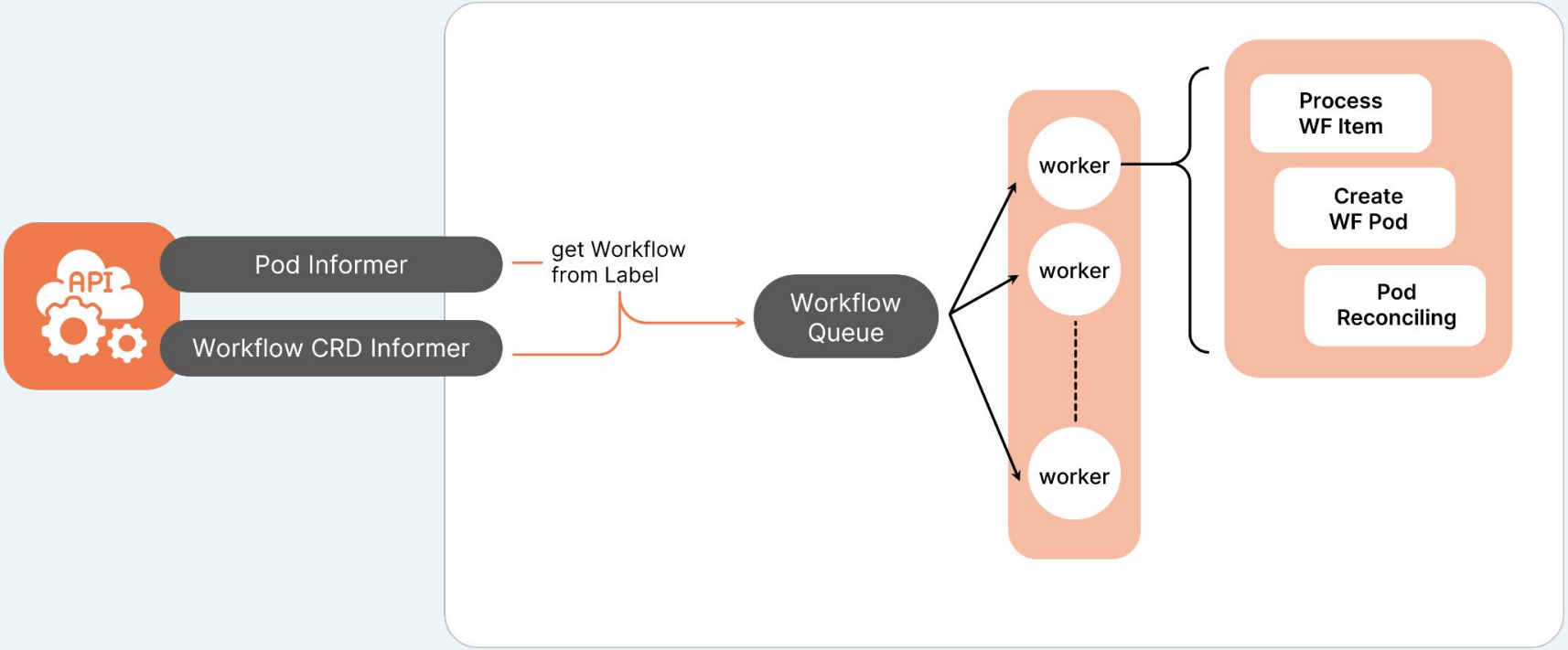
- Implements Kubernetes operator pattern
- Mostly talks with Kubernetes
- Can run in HA or have namespaced deployment

Workflow Controller - modes

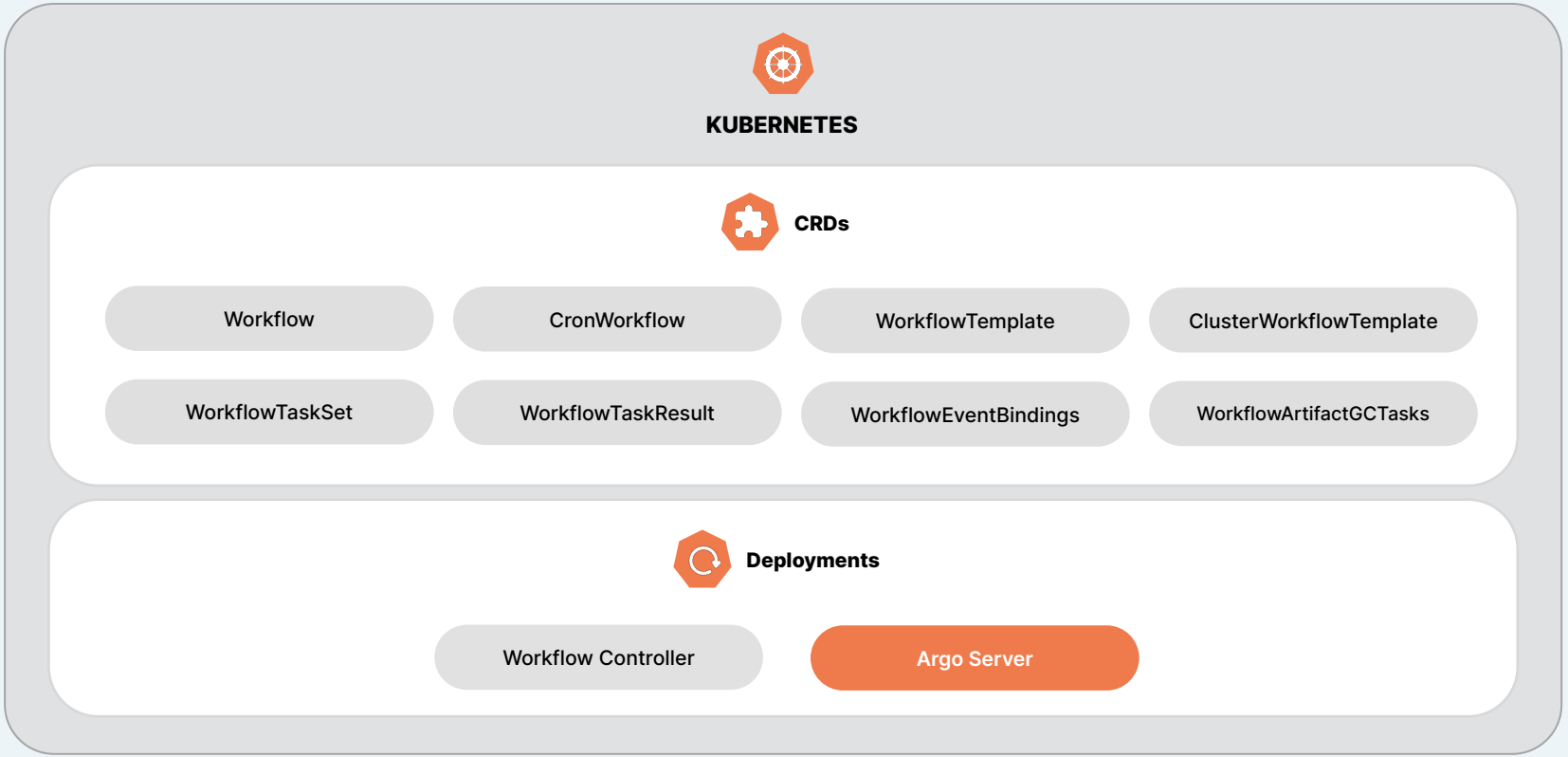
Workflow Controller - modes



Workflow Controller - modes



How do we orchestrate Workflows?



Argo Server

Mostly used for the communication with the outside world

For people that are not Kubernetes experts

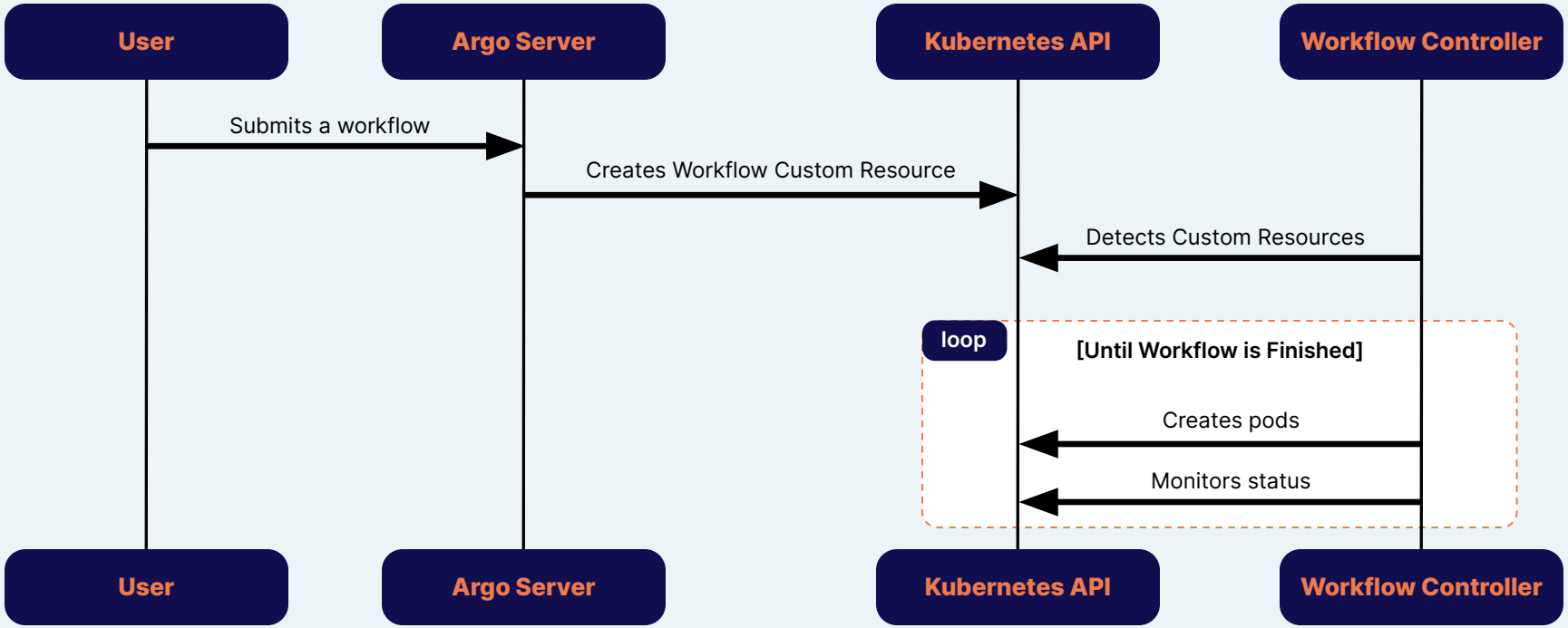
Exposes Argo Workflows API

Provides UI

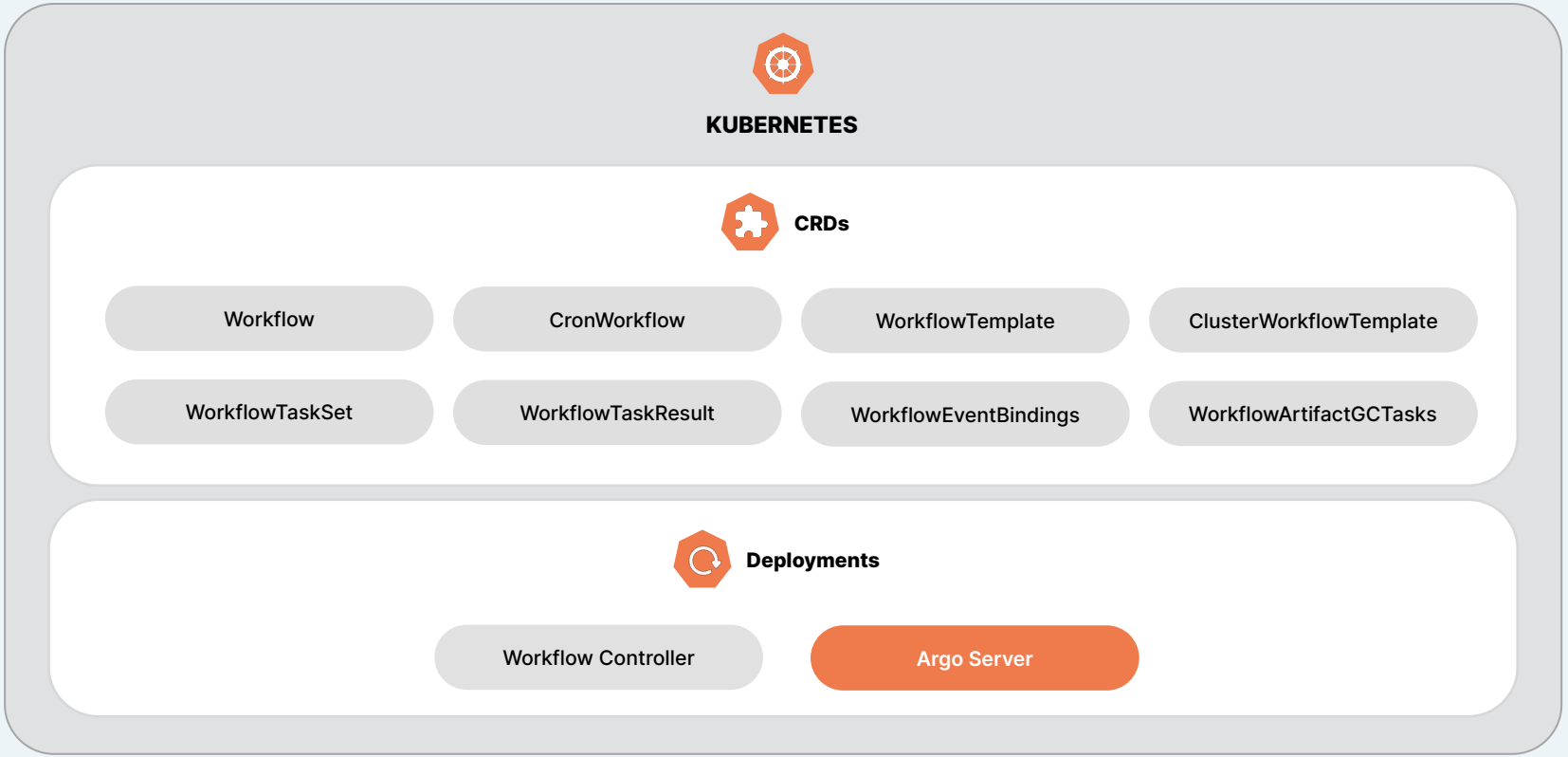
Argo Server

- Argo Workflow CLI is using Argo Server
- Not a mission-critical component
- Does the authentication
- Interacts with Kubernetes API
- Features like Workflow Archive, Offloading large workflows and Events need Argo Server

The Flow


User
Argo Server
Kubernetes API
Workflow Controller
Submits a workflow
Creates Workflow Custom Resource
Detects Custom Resources
loop
[Until Workflow is Finished]
Creates pods
Monitors status
User
Argo Server
Kubernetes API
Workflow Controller

Revisiting the Big Picture



The humble Pod

KUBERNETES

CRDs

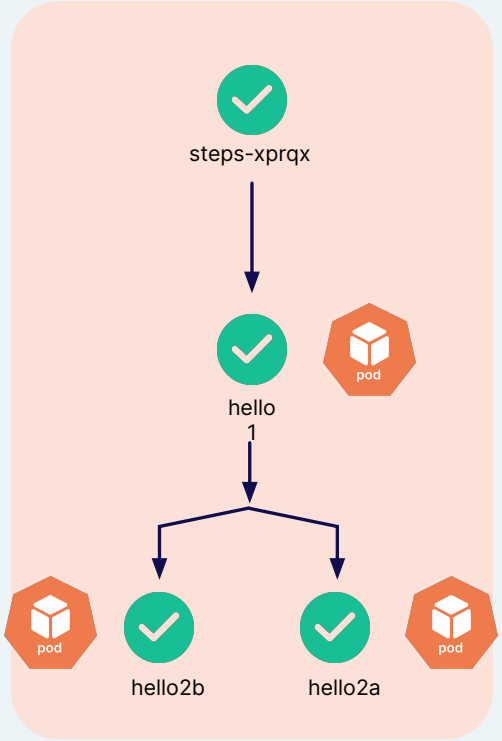
- Workflow
- WorkflowTaskSet
- CronWorkflow
- WorkflowTaskResult
- WorkflowTemplate
- WorkflowEventBindings
- ClusterWorkflowTemplate
- WorkflowArtifactGCTasks

Deployments

- Workflow Controller
- Argo Server

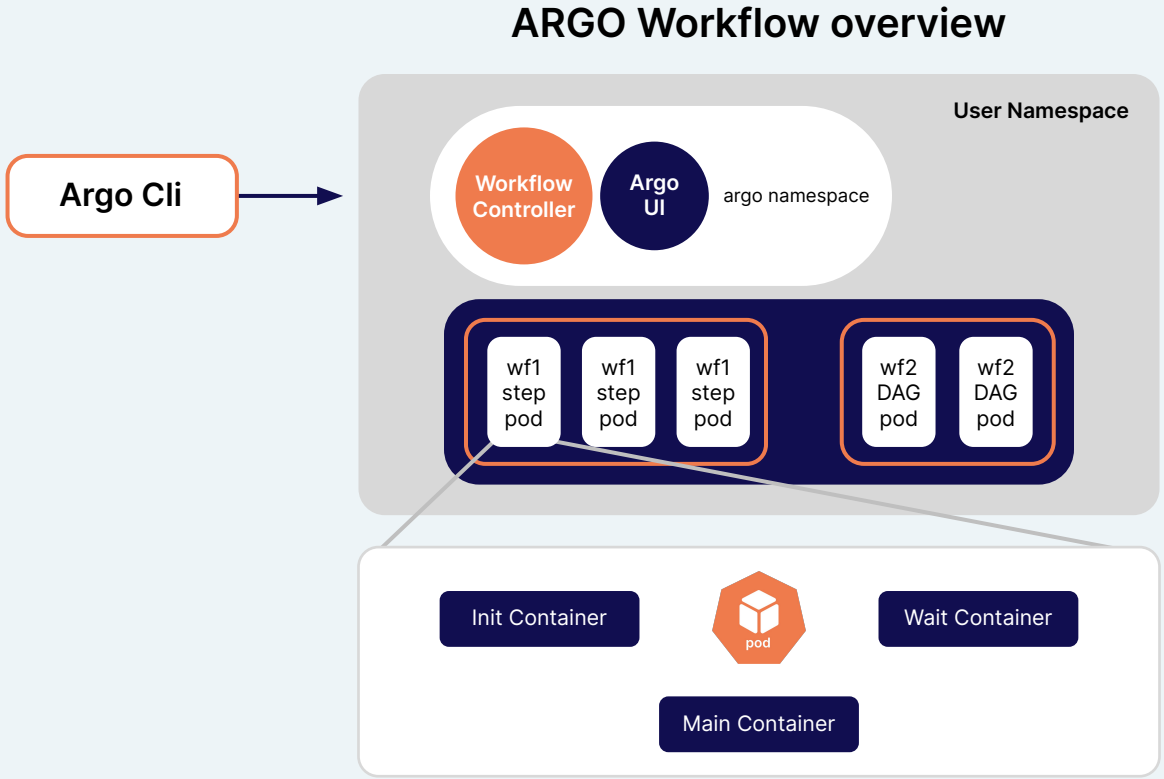
The basis of our Workflow steps

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow                  # new type of k8s spec
metadata:
  generateName: hello-world-    # name of the workflow spec
spec:
  entrypoint: whalesay          # invoke the whalesay template
  templates:
    - name: whalesay              # name of the template
      container:
        image: docker/whalesay
        command: [ cowsay ]
        args: [ "hello world" ]
        resources: # limit the resources
          limits:
            memory: 32Mi
            cpu: 100m
```



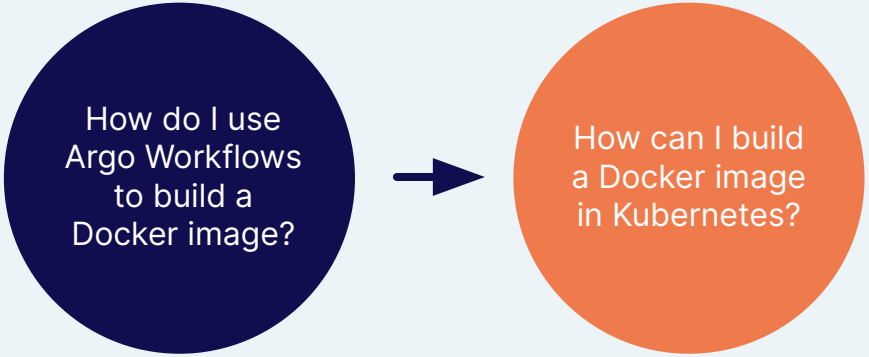
❌ suspend, workflow of workflows

Step == Pod


ARGO Workflow overview
Argo Cli
User Namespace
Workflow Controller
Argo UI
argo namespace
wf1 step pod
wf1 step pod
wf1 step pod
wf2 DAG pod
wf2 DAG pod
Init Container
pod
Wait Container
Main Container

How do I...?

"If you can do a thing in a normal kubernetes pod, you can do it in a workflow"



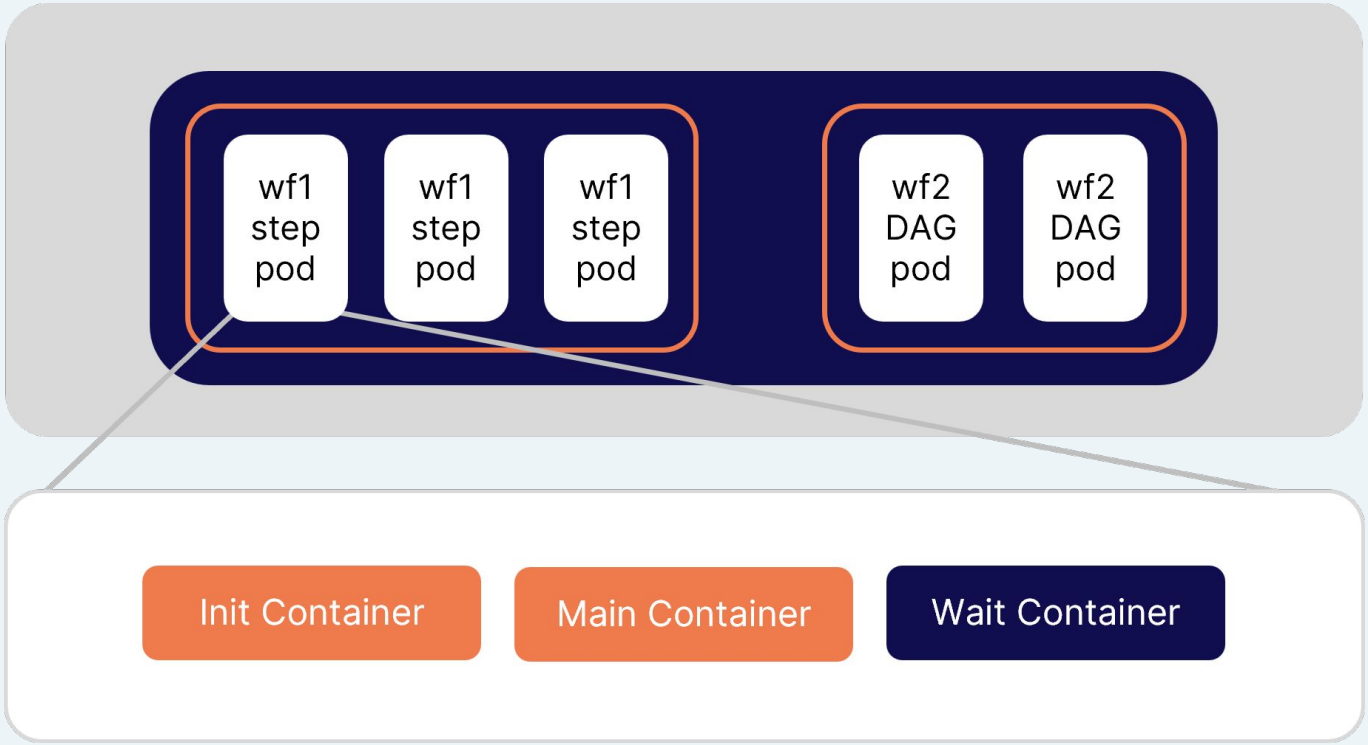
Kaniko

moby/buildkit

‘Podness’ helps us understand…

- Resource utilisation
- RBAC
- Mount volumes/other containers
- Workspace
- Very short-lived steps
- Metrics

The anatomy of a Step (a Pod!)



We can see this when...

```yaml
spec:
  entrypoint: python-script-example
  templates:
  - name: python-script-example
    steps:
    - - name: generate
        template: gen-random-int
    - - name: print
        template: print-message
        arguments:
          parameters:
          - name: message
            value: "{{steps.generate.outputs.result}}"

  - name: gen-random-int
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        i = random.randint(1, 100)
        print(i)
```

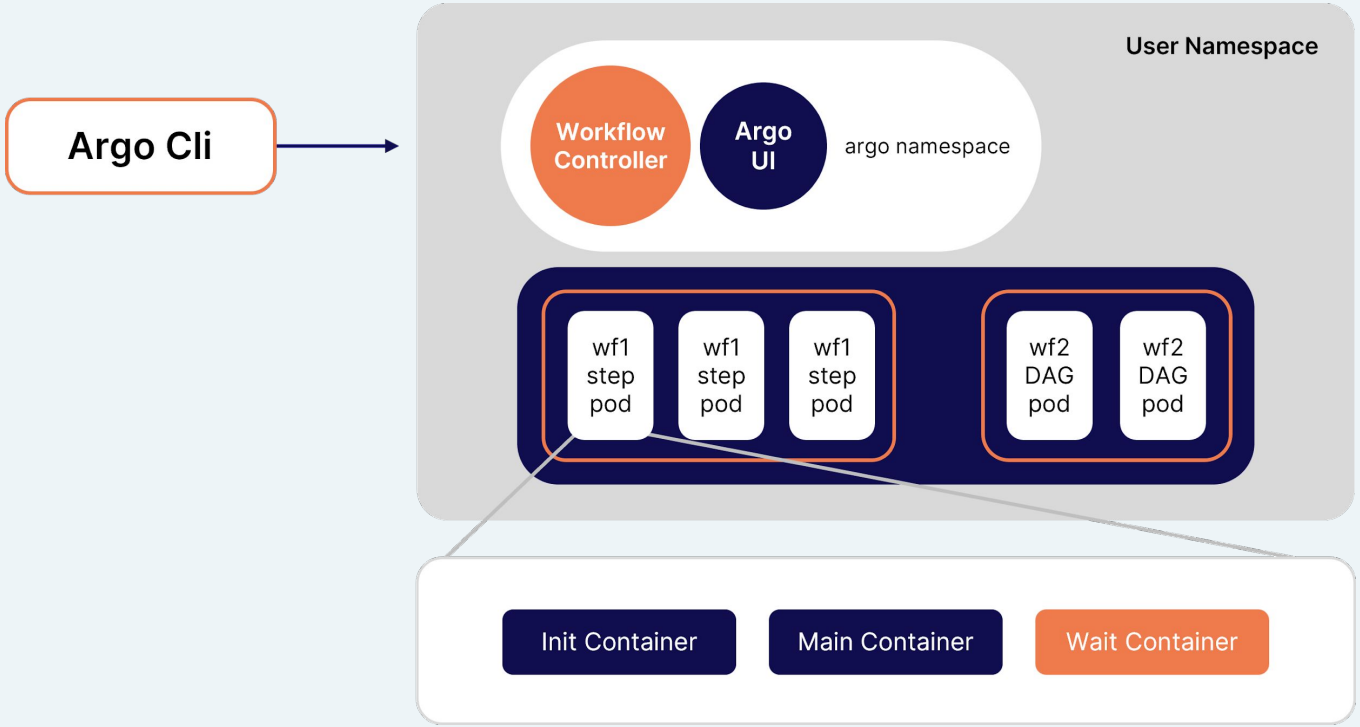
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: container-set-template-
spec:
  entrypoint: main
  templates:
    - name: main
      volumes:
        - name: workspace
          emptyDir: { }
      containerSet:
        volumeMounts:
          - mountPath: /workspace
            name: workspace
        containers:
          - name: a
            image: argoproj/argosay:v2
            command: [sh, -c]
            args: ["echo 'a: hello world' >> /workspace/message"]
          - name: b
            image: argoproj/argosay:v2
            command: [sh, -c]
            args: ["echo 'b: hello world' >> /workspace/message"]
```

But what about...

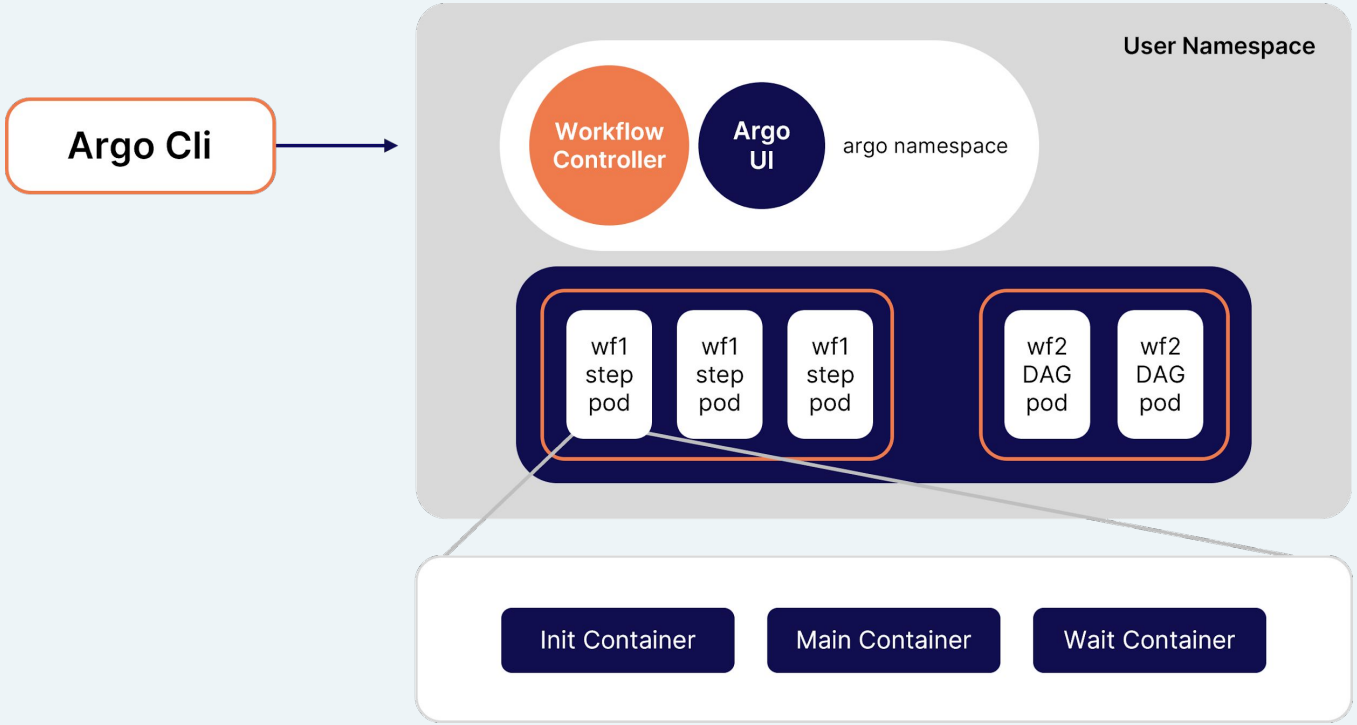
```yaml
entrypoint: main
templates:
  - name: main
    steps:
      - - name: get-google-homepage
          template: http
          arguments:
            parameters: [{name: url, value: "https://www.google.com"}]
  - name: http
    inputs:
      parameters:
        - name: url
    http:
      timeoutSeconds: 20 # Default 30
      url: "{{inputs.parameters.url}}"
      method: "GET" # Default GET
      headers:
        - name: "x-header-name"
          value: "test-value"
      # Template will succeed if evaluated to true, otherwise will fail
      # Available variables:
      #  request.body: string, the request body
      #  request.headers: map[string][]string, the request headers
      #  response.url: string, the request url
      #  response.method: string, the request method
      #  response.statusCode: int, the response status code
      #  response.body: string, the response body
      #  response.headers: map[string][]string, the response headers
      successCondition: "response.body contains \"google\"" # available since v3
      body: "test body" # Change request body
```

```yaml
- name: create-configmap
  resource:
    action: create
    manifest: |
      apiVersion: v1
      kind: ConfigMap
      metadata:
        name: resource-delete-with-flags
        labels:
          cleanup: "true"
      data:
        key: value
```

The anatomy of a Step (a Pod!)



The Big Picture



Want to learn more?

Try the Killer Coda course
https://killercoda.com/argoproj/course/argo-workflows/

Read through the docs
https://argo-workflows.readthedocs.io

Visit GitHub (especially the examples directory)
https://github.com/argoproj/argo-workflows

Stay up to date with our blogs
https://pipekit.io/blog
https://venafi.com/blog/category/cloud-native/

Visit Pipekit Office Hours

About 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

Cloud native ecosystem knowledge and expertise

Venafi Jetstack Consult provides strategic consulting and advisory services for companies that are embracing cloud native

CLOUD NATIVE COMPUTING FOUNDATION

https://venafi.com/jetstack-consult/

Thank you!

Darko Janjić
Senior Software Engineer, Pipekit

darko@pipekit.io

Becky Pauley
Solutions Engineer, Venafi Jetstack Consult

becky.pauley@venafi.com

Feedback

Q&A

Do you have any questions?