



Beyond stand-alone performance: AI Workflow Session 3

Argo Workflows

"Workflows, the Kubernetes way."

By:  at  **FORTH** & Antonis Tapanlis



Introduction to Workflows & Argo

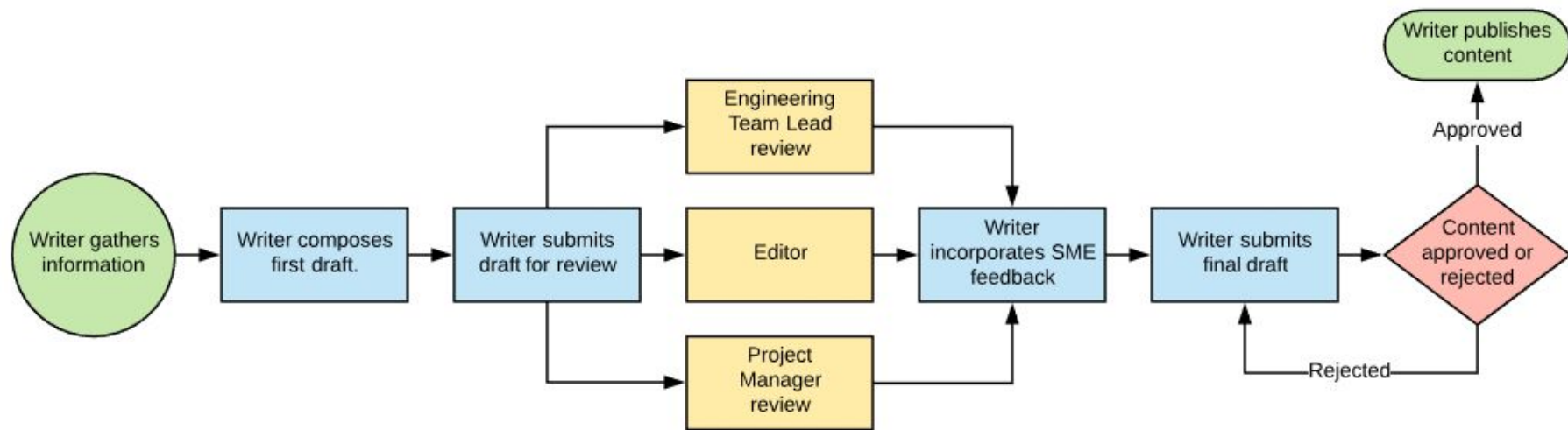
Argo Workflows Components

Argo Language & Examples

What is a Workflow?

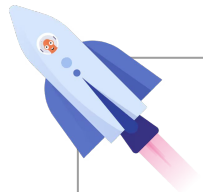


- A workflow is a **series of steps** necessary to complete a task
- Each step in a workflow has a preceding step and a following step (except for the first and last steps)
- Each step may have dependencies, such as files, user actions, or scheduled execution times
- Often modeled as Directed Acyclic Graphs (**DAGs**) or represented as flowcharts





- Open source tools for Kubernetes to run workflows, manage clusters, and do GitOps
- Accepted on CNCF (Cloud Native Computing Foundation) at 2020, Graduated at 2022



Argo CD

Declarative continuous delivery with a fully-loaded UI

20k★

Argo Events

Event based dependency management for Kubernetes - 2.5k★



Argo Rollouts

Advanced Kubernetes deployment strategies

3k★



Argo Workflows

K8s-native workflow engine supporting DAG and step-based workflows - 16k★



- Workflow Engine
 - Is a **Language & a Runtime System**
 - Defines, orchestrates, and monitors a series of tasks as DAGs
 - Ensures the correct running sequence and the right dependencies
- Kubernetes-native Workflow Engine
 - Leverages Kubernetes features
 - Mainly consists of
 - Kubernetes Objects (CRDs)
 - A Controller (Kubernetes Operator)
 - Each Workflow **task runs in a Container**

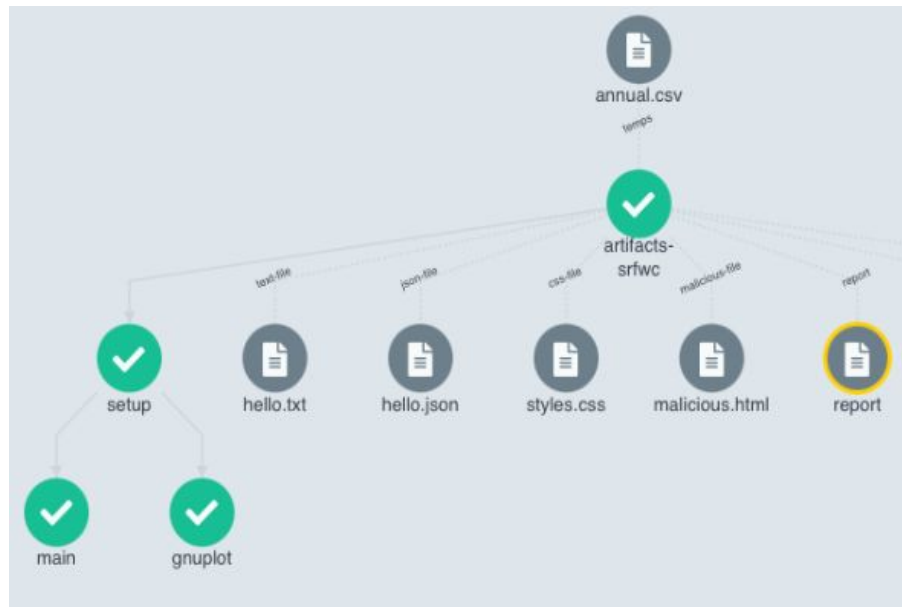


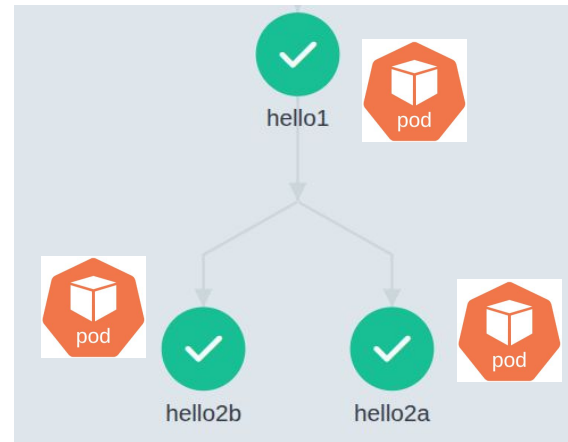
Image source: <https://argo-workflows.readthedocs.io/en/latest/artifact-visualization/>



- Language -> Argo **yaml** (Kubernetes Objects)
- Runtime -> Argo Workflows components running on **Kubernetes**
- Main object -> **Workflow** (similar to Job)
- Each step/task of a Workflow is a **pod**
- Use/write **containers** for each step

Why do we need a workflow-tool like this?

- To manage workflows through CLI or Web UI, getting advantage of the Kubernetes power
- To mainly use it for
 - Compute-intensive jobs (ML, data processing)
 - Automating software infrastructure (CI/CD)





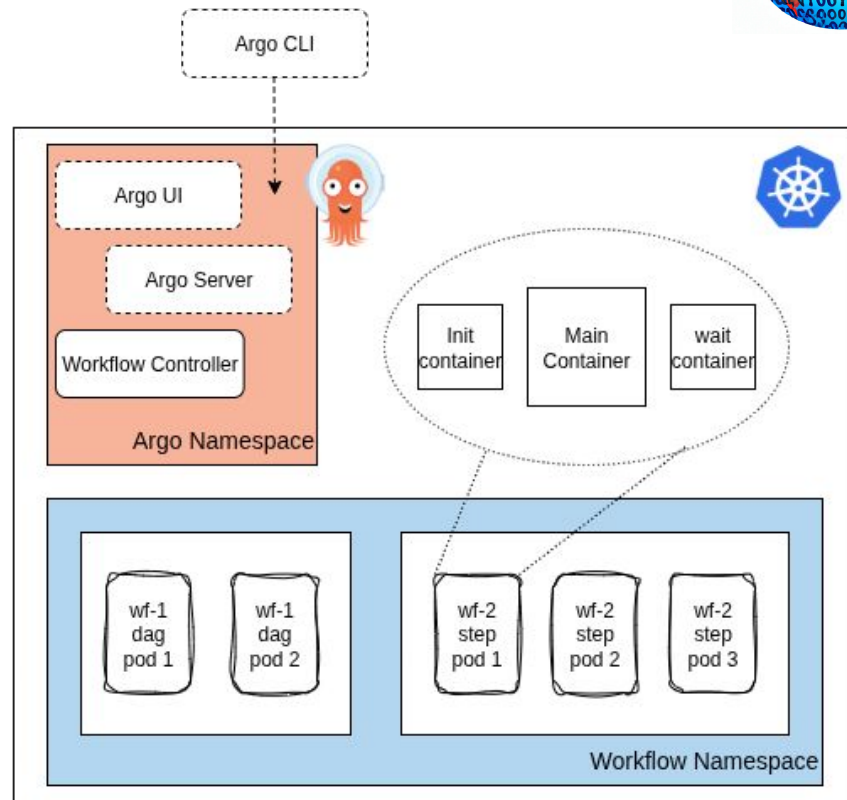
Introduction to Workflows & Argo

Argo Workflows Components

Argo Language & Examples



- **Argo CLI & UI (optional use)**
 - User endpoint for interaction with Argo
 - Extended functionalities for Argo Objects
- **Argo Server (optional use)**
 - Used to take advantage of CLI and UI
 - Alternatively “*kubect*” and kubernetes API can be used
- **Workflow Controller**
 - Deploys, monitors and manages workflow objects
 - Pods are spawned per step, based on YAML-defined logic
 - Three containers per Pod





- **argo submit** hello-world.yaml -> submit a workflow to Kubernetes
 - **--wait** -> wait for the workflow to complete
 - **--watch** -> watch the workflow until it completes
- **argo list** -> list current workflows
 - **--running** or **--completed**
 - **-n <namespace>** or **-A** (for all namespaces)
- **argo get** hello-world-xxx -> get info about a specific workflow
- **argo logs** hello-world-xxx -> print the logs from a workflow
- **argo watch** hello-world-xxx -> watch the flow in terminal
- **argo stop** hello-world-xxx
- **argo resume** hello-world-xxx
- **argo delete** hello-world-xxx



STEP	TEMPLATE	PODNAME	DURATION
✓ steps-z2zdn	hello-hello-hello		
└─ ✓ hello1	print-message	steps-z2zdn-27420706	2s
└─ ✓ hello2a	print-message	steps-z2zdn-2006760091	3s
└─ ✓ hello2b	print-message	steps-z2zdn-2023537710	3s



Home page

Workflows / knot-antap

+ SUBMIT NEW WORKFLOW

WORKFLOWS SUMMARY ⓘ

Running workflows 0

Pending 0 Succeeded 8 Failed 0 Error 0

NAMESPACE

knot-antap

NAME CONTAINS

LABELS

WORKFLOW TEMPLATE

	NAME	NAMESPACE	STARTED	FINISHED	DURATION	PROGRESS
☐	coinflip-sequence-vwb2j	knot-an...	15m39s ago	15m8s ago	30s	6/6
☐	coinflip-sequence-9xq68	knot-an...	15m53s ago	15m23s ago	30s	6/6
☐	coinflip-sequence-tj9tb	knot-an...	16m10s ago	15m40s ago	30s	6/6
☐	coinflip-items-z56mg	knot-an...	31m35s ago	31m5s ago	30s	6/6
☐	coinflip-items-8zvt9	knot-an...	1h17m ago	1h16m ago	20s	6/6
☐	steps-8vz7x	knot-an...	1h43m ago	1h42m ago	40s	3/3
☐	steps-svpss	knot-an...	1h46m ago	1h46m ago	20s	3/3
☐	steps-z42k9	knot-an...	1h50m ago	1h50m ago	21s	3/3

Workflow page

RESUBMIT DELETE LOGS SHARE

Search

```

graph TD
    steps-svpss[steps-svpss] --> hello1[hello1]
    hello1 --> hello2b[hello2b]
    hello1 --> hello2a[hello2a]
    
```

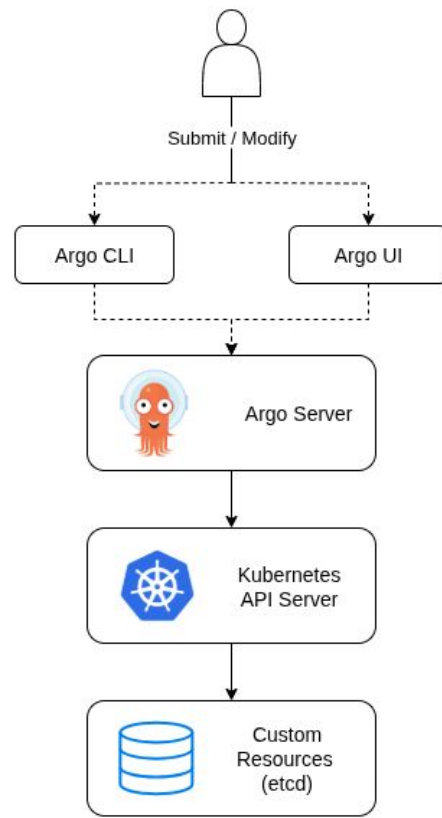
SUMMARY CONTAINERS INPUTS/OUTPUTS

NAME	steps-svpss
ID	steps-svpss
TYPE	Steps
PHASE	Succeeded
START TIME	9/16/2025, 12:22:45 AM (2d19h ago)
END TIME	9/16/2025, 12:23:05 AM (2d19h ago)
DURATION	20s
PROGRESS	3/3
MEMOIZATION	N/A
RESOURCES DURATION	0s*(1 cpu),14s*(100Mi memory)

MANIFEST RETRY NODE

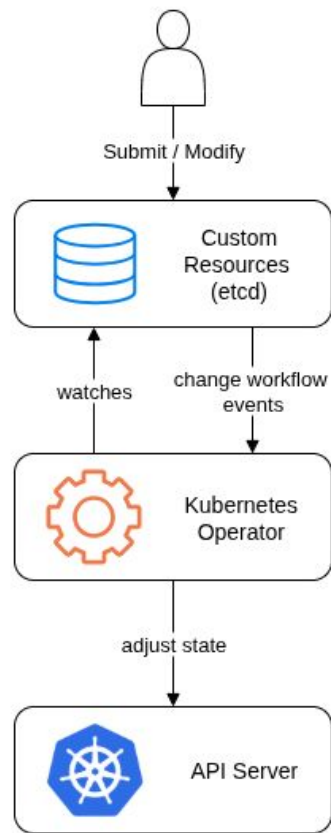


- Mostly used for the communication with the outside world
- Exposes Argo Workflows API
- Provides Argo UI – Considered to make Argo Workflows more accessible to those less familiar with Kubernetes
- Optional use – “kubectl” use for users prefer the Kubernetes way
- Argo CLI talks to the Argo Server

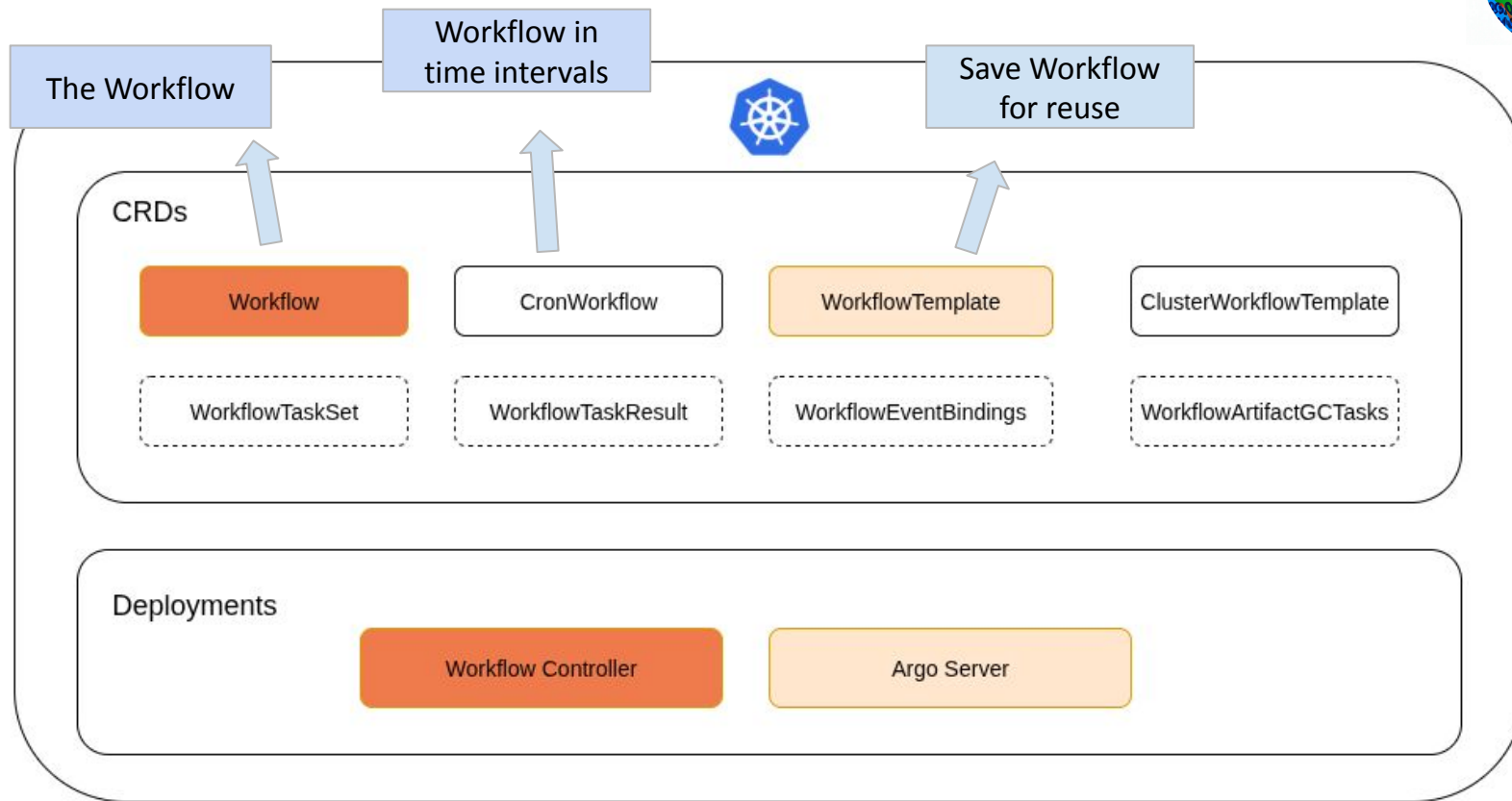




- Main component of Argo Workflows
- Manages Workflows and everything related to Workflows (Argo Workflows CRDs)
- Talks with Kubernetes API Server
- Implements the Kubernetes Operator pattern
- Can be modified and/or have multiple deployments (High availability mode, Namespaced mode)



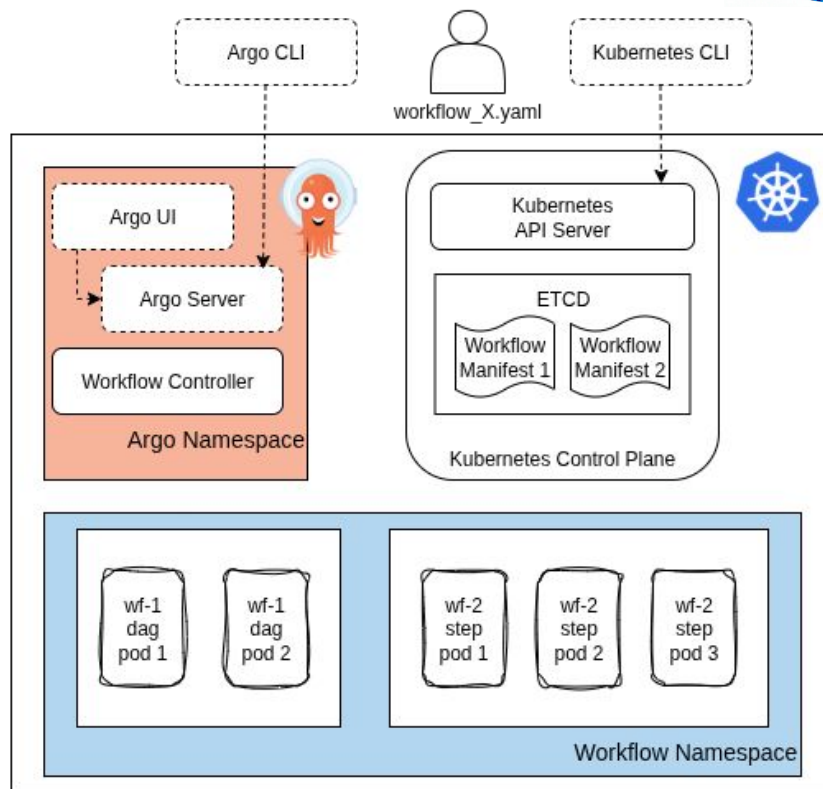
Argo Workflows - Overall



Argo Workflows - Working



1. **User creates** a Workflow manifest using either K8s CLI, Argo CLI, or Argo UI
 2. K8s API Server stores the Workflow in **etcd**
 3. Workflow Controller **detects it** and starts running
 4. Controller asks the Kubernetes API Server to **create the pod(s)** for the read-to-run steps
 5. Controller continually asks the Kubernetes API for the **status of the pod(s)**
 6. Controller **writes the status** and parameter values of each workflow step back to the Workflow manifest (via k8s API)
- Steps 4–6 are repeated until the workflow is Succeeded or Failed





Introduction to Workflows & Argo

Argo Workflows Components

Argo Language & Examples



Language part 1

- Hello World !
- Workflow Structure
 - Workflow Types - (Template Invocators)
 - Step Templates Types - (Template Definitions)
- Hello World !!!!
- Bonus Workflow Structure



Hello World !



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: hello-world-
5    annotations:
6      workflows.argoproj.io/description: |
7        This is a simple hello world example.
8  spec:
9    entrypoint: hello-world
10   templates:
11     - name: hello-world
12       container:
13         image: busybox
14         command: [echo]
15         args: ["hello world"]
```



hello-world-wbcfs

SUMMARY

CONTAINERS

NAME	main
IMAGE	busybox
COMMAND	echo
ARGS	'hello world'





Language part 1

- Hello World !
- Workflow Structure
 - Workflow Types - (Template Invocators)
 - Step Templates Types - (Template Definitions)
- Hello World !!!!
- Bonus Workflow Structure



Workflow Types - (Template Invocators)



Single Task

One-task Workflow.
(Hello World)

Step-based

Sequence of steps.
Implicit dependencies

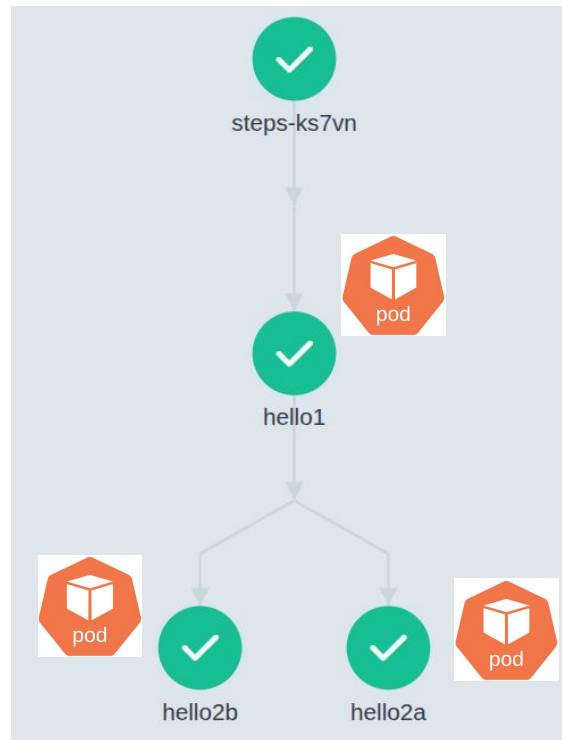
DAG

Graph of tasks.
Explicit dependencies

Workflow Types - Steps



```
3  apiVersion: argoproj.io/v1alpha1
4  kind: Workflow
5  metadata:
6    generateName: steps-
7  spec:
8    entrypoint: hello-hello-hello
9    templates:
10     - name: hello-hello-hello
11       steps:
12         - - name: hello1
13           (config)
14         - - name: hello2a
15           (config)
16         - - name: hello2b
17           (config)
```



Step Template Types - (Template Definitions)



- We need something to implement this steps
- We want reusable code
- We better keep the Workflow “clean”
- Solution: Template Definitions!
 - **Container**
 - Script
 - ...

```
3   apiVersion: argoproj.io/v1alpha1
4   kind: Workflow
5   metadata:
6     generateName: steps-
7   spec:
8     entrypoint: hello-hello-hello
9     templates:
10      - name: hello-hello-hello
11        steps:
12          - - name: hello1
13
14              (config)
15
16          - - name: hello2a
17
18              (config)
19
20          - name: hello2b
21
22              (config)
23
```

Step Template Types - Container



```
3  apiVersion: argoproj.io/v1alpha1
4  kind: Workflow
5  metadata:
6    generateName: steps-
7  spec:
8    entrypoint: hello-hello-hello
9    templates:
10     - name: hello-hello-hello
11       steps:
12         - - name: hello1
13             template: print-message
14             (config)
15         - - name: hello2a
16             template: print-message
17             (config)
18         - name: hello2b
19           template: print-message
20           (config)
```

```
25  - name: print-message
26
27  (config)
28
29  container:
30    image: busybox
31
32  (config)
```





Language part 1

- Hello World !
- Workflow Structure
 - Workflow Types - (Template Invocators)
 - Step Templates Types - (Template Definitions)
- Hello World !!!!
- Bonus Workflow Structure

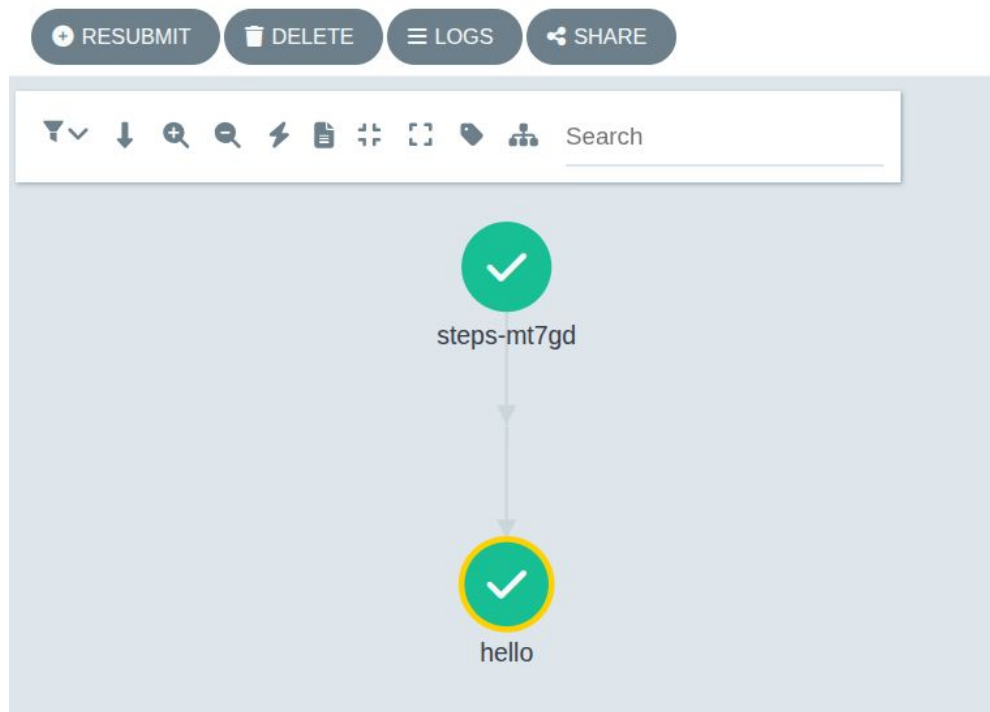


Hello World !! - (single inline step)



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: steps-
5  spec:
6    entrypoint: hello
7    templates:
8      - name: hello
9        steps:
10         - - name: hello
11           inline:
12             container:
13               image: busybox
14               command: [echo]
15               args: [hello world!]
```

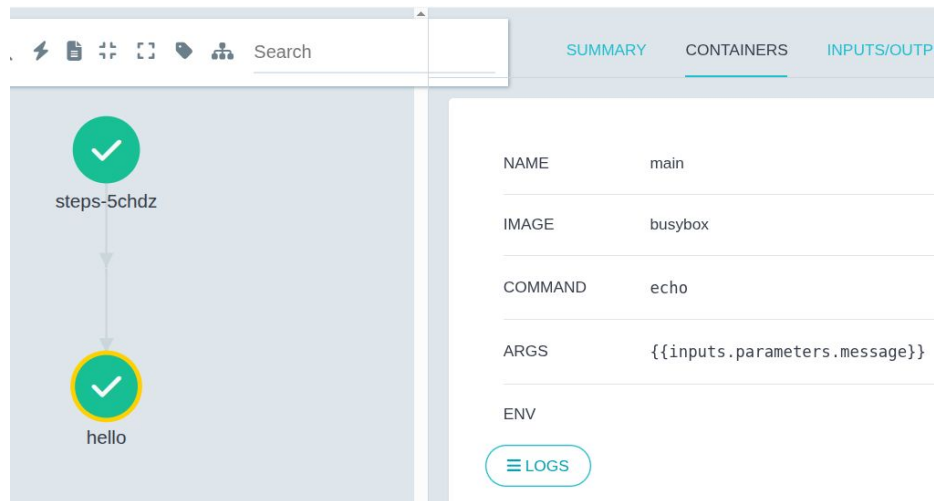
(inline: Not the best practice !)



Hello World !!! - (single template step)



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: steps-
5  spec:
6    entrypoint: hello
7    templates:
8      - name: hello
9        steps:
10          - - name: hello
11              template: print-message
12              arguments:
13                parameters: [{name: message, value: "hello world!"}]
14
15      - name: print-message
16        inputs:
17          parameters:
18            - name: message
19        container:
20          image: busybox
21          command: [echo]
22          args: ["{{inputs.parameters.message}}"]
```

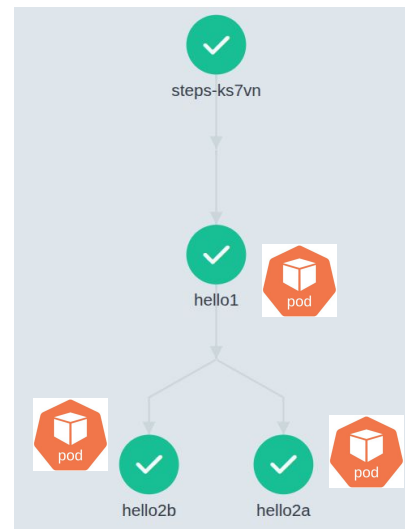


Hello World !!!! - (3-step Workflow using 1 template)



```
3  apiVersion: argoproj.io/v1alpha1
4  kind: Workflow
5  metadata:
6    generateName: steps-
7  spec:
8    entrypoint: hello-hello-hello
9    templates:
10   - name: hello-hello-hello
11     steps:
12     - - name: hello1
13         template: print-message
14         arguments:
15           parameters: [{name: message, value: "hello1"}]
16     - - name: hello2a
17         template: print-message
18         arguments:
19           parameters: [{name: message, value: "hello2a"}]
20     - - name: hello2b
21         template: print-message
22         arguments:
23           parameters: [{name: message, value: "hello2b"}]
```

```
25   - name: print-message
26     inputs:
27       parameters:
28         - name: message
29     container:
30       image: busybox
31       command: [echo]
32       args: ["{{inputs.parameters.message}}"]
```





Language part 1

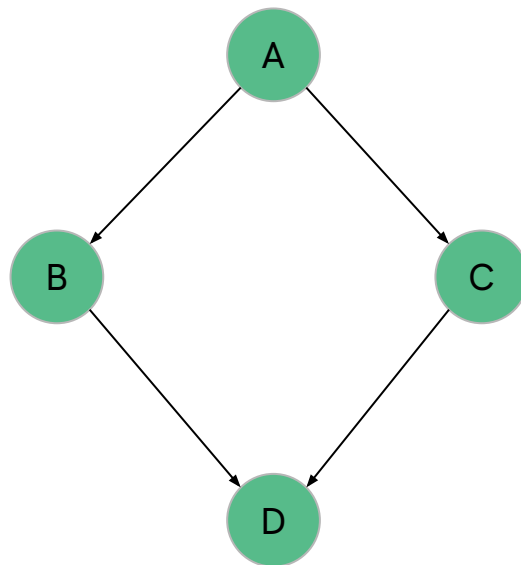
- Hello World !
- Workflow Structure
 - Workflow Types - (Template Invocators)
 - Step Templates Types - (Template Definitions)
- Hello World !!!!
- Bonus Workflow Structure



Workflow Types - DAG



```
13   entrypoint: diamond
14   templates:
15     - name: diamond
16       dag:
17         tasks:
18           - name: A
19             (config)
20           - name: B
21             depends: "A"
22             (config)
23           - name: C
24             depends: "A"
25             (config)
26           - name: D
27             depends: "B && C"
28             (config)
```



Step Template Types - Script (Container wrapper)



```
13   entrypoint: bash-script-example
14   templates:
15   - name: bash-script-example
16     steps:
17     - - name: generate
18         template: gen-random-int
19     - - name: print
20         template: print-message
21       arguments:
22         parameters:
23         - name: message
24           value: "{{steps.generate.outputs.result}}"
25
26   - name: gen-random-int
27     script:
28       image: debian:9.4
29       command: [bash]
30       source: |
31         cat /dev/urandom | od -N2 -An -i | awk -v f=1 -v r=100 '{printf "%i\n", f + r * $1 / 65536}'
32
33   - name: print-message
34     inputs:
35     parameters:
36     - name: message
37   container:
38     image: alpine:latest
39     command: [sh, -c]
40     args: ["echo result was: {{inputs.parameters.message}}"]
```

```
26   - name: gen-random-int
27     script:
28       image: python:alpine3.6
29       command: [python]
30       source: |
31         import random
32         i = random.randint(1, 100)
33         print(i)
```



Step Template Types - Some more



More than 1
container
per pod

```
12   entrypoint: main
13   templates:
14     - name: main
15       containerSet:
16         containers:
17           - name: a
18             image: argoproj/argosay:v2
19           - name: b
20             image: argoproj/argosay:v2
```

Kubernetes
objects

```
13   entrypoint: k8s-set-owner-reference
14   templates:
15     - name: k8s-set-owner-reference
16       resource:
17         action: create
18         setOwnerReference: true
19         manifest: |
20           apiVersion: v1
21           kind: ConfigMap
22           metadata:
23             generateName: owned-eg-
24             data:
25               some: value
```

HTTP request

```
14     - name: http
15       inputs:
16         parameters:
17           - name: url
18             http:
19               timeoutSeconds: 20 # Default 30
20               url: "{{inputs.parameters.url}}"
21               method: "GET" # Default GET
22               headers:
23                 - name: "x-header-name"
24                   value: "test-value"
25               # Template will succeed if evaluated to true, otherwise will fail
26               # Available variables:
27               # request.body: string, the request body
28               # request.headers: map[string][]string, the request headers
29               # response.url: string, the request url
30               # response.method: string, the request method
31               # response.statusCode: int, the response status code
32               # response.body: string, the response body
33               # response.headers: map[string][]string, the response headers
34               successCondition: "response.body contains \"google\"" # available
35               body: "test body" # Change request body
```





Language part 2

- Runtime Parameters
- WorkflowTemplates
- Conditionals + Step Output
- Loops



Runtime Parameters



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: steps-
5  spec:
6    entrypoint: hello-hello-hello
7    arguments:
8      parameters:
9        - name: workflow-param-1
10          value: "default-hello2b" # default value if none is passed
11  templates:
12    - name: hello-hello-hello
13      steps:
```

*argo submit example.yaml
-p 'workflow-param-1="zdravo!"'*

MANIFEST	PARAMETERS	METADATA
Parameters		
workflow-param-1	default-hello2b	✕
Name...	Value...	✕

```
18  - name: hello2a
19    template: print-message
20    arguments:
21      parameters: [{name: message, value: "hello2a"}]
22  - name: hello2b
23    template: print-message
24    arguments:
25      parameters:
26        - name: message
27          value: "{{workflow.parameters.workflow-param-1}}"
28
29  - name: print-message
30    inputs:
31      parameters:
32        - name: message
33    container:
34      image: busybox
35      command: [echo]
36      args: ["{{inputs.parameters.message}}"]
```



WorkflowTemplate



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: steps-
5  spec:
6    entrypoint: hello-hello-hello
7    arguments:
8      parameters:
9        - name: workflow-param-1
10          value: "default-hello2b" # default value if none is passed
```



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: WorkflowTemplate
3  metadata:
4    name: steps-template
5  spec:
6    entrypoint: hello-hello-hello
```

Submit Workflow

knot-antap/steps-template

Entrypoint

<default>

Parameters

workflow-param-1

default-hello2b

Labels

submit-from-ui=true ✕

MANIFEST

PARAMETERS

METADATA

JSON/YAML

▶ METADATA

▶ SPEC

```
1  metadata:
2    generateName: steps-template-
3    namespace: knot-antap
4    labels:
5      workflows.argoproj.io/workflow-template: steps-template
6      submit-from-ui: "true"
7  spec:
8    arguments:
9      parameters:
10        - name: workflow-param-1
11          value: default-hello2b
12      workflowTemplateRef
13        name: steps-template
```



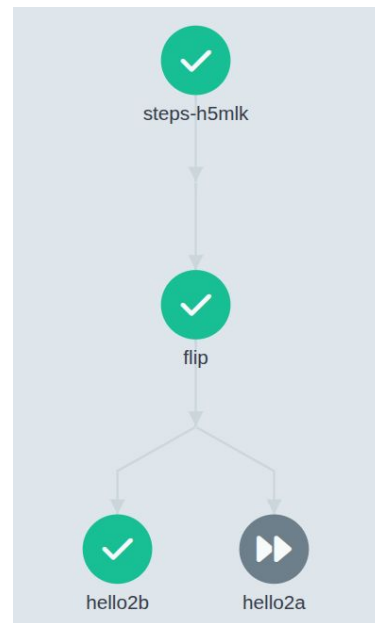
Conditionals + Step Output



```
8   - name: coin-flip-workflow
9     steps:
10    - - name: flip
11        template: flip-coin
12    - - name: hello2a
13        template: print-message
14        arguments:
15          parameters: [{name: message, value: "hello2a"}]
16          when: "{{steps.flip.outputs.result}}" == heads"
17    - name: hello2b
18        template: print-message
19        arguments:
20          parameters: [{name: message, value: "hello2b"}]
21          when: "{{steps.flip.outputs.result}}" == tails"
```

`steps.flip.outputs.result`
=
`steps.<step_name>.outputs.STDOUT`

```
23  # Template for flipping a coin
24  - name: flip-coin
25    script:
26      image: python:alpine3.6
27      command: [python]
28      source: |
29        import random
30        result = "heads" if random.randint(0,1) == 0 else "tails"
31        print(result)
```



Loop (for)



```
9   - name: coinflip-experiment
10  steps:
11    - name: run-coinflip
12      template: coinflip-round
13      withSequence:
14        count: 3 # repeat experiment 3 times
15
16  # One round of coin flip + conditional steps
17  - name: coinflip-round
18    steps:
19      - name: flip
20        template: flip-coin
21      - name: hello2a
22        template: print-message
23        arguments:
24          parameters: [{name: message, value: "heads chosen"}]
25        when: "{{steps.flip.outputs.result}} == heads"
26      - name: hello2b
27        template: print-message
28        arguments:
29          parameters: [{name: message, value: "tails chosen"}]
30        when: "{{steps.flip.outputs.result}} == tails"
```



Loop (with items)



```
23 - name: coinflip-round
24   inputs:
25     parameters:
26       - name: round
27   steps:
28     - - name: flip
29       template: flip-coin
30     - - name: hello2a
31       template: print-message
32       arguments:
33         parameters:
34           - name: message
35             value: "{{inputs.parameters.round}} round: heads chosen"
36       when: "{{steps.flip.outputs.result}} == heads"
37   - name: hello2b
38     template: print-message
39     arguments:
40       parameters:
41         - name: message
42           value: "{{inputs.parameters.round}} round: tails chosen"
43       when: "{{steps.flip.outputs.result}} == tails"
```

```
9   - name: coinflip-experiment
10     steps:
11       - - name: run-coinflip
12         template: coinflip-round
13         arguments:
14           parameters:
15             - name: round
16               value: "{{item}}"
17         withItems:
18           - "1st"
19           - "2nd"
20           - "3rd"
```





Language part 3

- Data between Steps - Input/Output
 - Artifacts - Input/Output files + Artifact Repository
 - Hardwired artifacts
 - Files - Shared Filesystem

Data between Steps - Input/Output



Parameters

Booleans, loop values,
strings/JSON
(e.g. the output of a
python script).
[~256 KB cap]

Easy use.
No setup is needed

Artifacts

Files, Large files
(Argo way)

Moderate use.
- For **Output** files, set up
of Artifact Repository (or
Cloud Storage) is needed.
- No setup is needed
for **Input** files

Volumes

Files, Large files,
XLarge files
(Kubernetes way)

Advanced use.
Filesystem-style
sharing



Shared Filesystem

Files, Large files,
XLarge files
(Filesystem attached to
all containers)

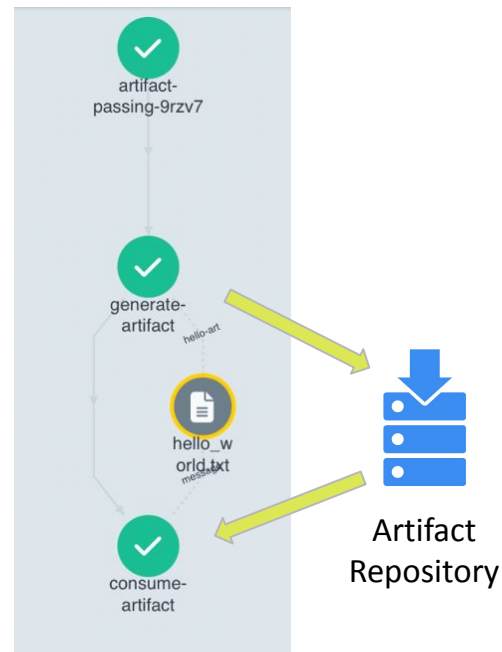
Easy to use once it's
set up. User just writes
the **filepath**
Existing in Knot system

Artifacts - Input/Output files + Artifact Repository



```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: artifact-passing-
5  spec:
6    entrypoint: artifact-example
7    templates:
8      - name: artifact-example
9        steps:
10       - - name: generate-artifact
11           template: hello-world-to-file
12       - - name: consume-artifact
13           template: print-message-from-file
14         arguments:
15           artifacts:
16             # bind message to the hello artifact
17             # generated by the generate-artifact step
18             - name: message
19               from: "{{steps.generate-artifact.outputs.artifacts.hello}}"
```

```
21   - name: hello-world-to-file
22     container:
23       image: busybox
24       command: [sh, -c]
25       args: ["echo hello world | tee /tmp/hello_world.txt"]
26     outputs:
27       artifacts:
28         # generate hello artifact from /tmp/hello_world.txt
29         # artifacts can be directories as well as files
30         - name: hello
31           path: /tmp/hello_world.txt
32
33   - name: print-message-from-file
34     inputs:
35       artifacts:
36         # unpack the message input artifact
37         # and put it at /tmp/message
38         - name: message
39           path: /tmp/message
40     container:
41       image: alpine:latest
42       command: [sh, -c]
43       args: ["cat /tmp/message"]
```



Artifacts - Hardwired [git, http, S3]



```
14 inputs:
15   artifacts:
16     # Check out the main branch of the argo repo and place it at /src
17     # revision can be anything that git checkout accepts: branch, commit, tag, etc.
18     - name: argo-source
19       path: /src
20   → git:
21     repo: https://github.com/argoproj/argo-workflows.git
22     revision: "main"
23   # Download kubectl 1.8.0 and place it at /bin/kubectl
24   - name: kubectl
25     path: /bin/kubectl
26     mode: 0755
27   → http:
28     url: https://storage.googleapis.com/kubernetes-release/release/v1.8.0/bin/linux/amd64/kubectl
29   # Copy an s3 compatible artifact repository bucket (such as AWS, GCS and MinIO) and place it at /s3
30   - name: objects
31     path: /s3
32   → s3:
33     endpoint: storage.googleapis.com
34     bucket: my-bucket-name
35     key: path/in/bucket
36     accessKeySecret:
37       name: my-s3-credentials
38       key: accessKey
39     secretKeySecret:
40       name: my-s3-credentials
41       key: secretKey
```

```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: hardwired-artifact-
5  spec:
6    entrypoint: hardwired-artifact
7    templates:
8      - name: hardwired-artifact
9        container:
10          image: debian
11          command: [sh, -c]
12          args: ["ls -l /src /bin/kubectl /s3"]
```

AWS S3 acts as Artifact Repository and can be used for output Artifacts too! 🌟





```
1  apiVersion: argoproj.io/v1alpha1
2  kind: Workflow
3  metadata:
4    generateName: catalog-demo-
5  spec:
6    entrypoint: run-catalog
7    templates:
8      - name: run-catalog
9        steps:
10          - - name: foo-step
11              template: process-foo
12            - name: bar-step
13              template: process-bar
```

```
15    - name: process-foo
16      inputs:
17        parameters:
18          - name: in-file
19            value: "/files/private/tiffs/foo.tiff"
20      metadata: {}
21      container:
22        image: alpine:3.19
23        command: ["sh", "-c"]
24        args: ["echo Processing {{inputs.parameters.in-file}}"]
25
26    - name: process-bar
27      inputs:
28        parameters:
29          - name: in2-file
30            value: "/files/private/tiffs/bar.tiff"
31      metadata: {}
32      container:
33        image: alpine:3.19
34        command: ["sh", "-c"]
35        args: ["echo Processing {{inputs.parameters.in2-file}}"]
```



“Take-Home Message”

Argo lets you focus on workflows, not infrastructure.



Run complex workflows with Kubernetes power.
No deep Kubernetes knowledge required !



- <https://argoproj.github.io/>
- <https://argo-workflows.readthedocs.io/>
- <https://github.com/argoproj/argo-workflows>
- <https://www.youtube.com/watch?v=FBRMURQYbgw>
- <https://dafab-platform.eu/>
- <https://github.com/AntonisT/Dafab-Summer-School-Argo>
- <https://github.com/argoproj/argo-workflows/discussions/7338> ✨
- <https://argo-workflows.readthedocs.io/en/latest/configure-artifact-repository/#accessing-non-default-artifact-repositories> ✨

