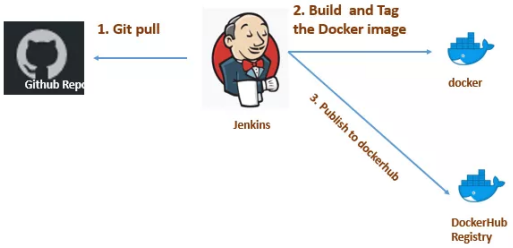
Argo Workflow

Pipeline “”



CI

JenkinsJenkins Jenkins Jenkins X Kubernetes —— Argo Tekton Argo

Argo

`Argo Workflows` Kubernetes Argo Workflows Kubernetes CRDArgo Kubernetes `kubectl` CRD controller server

```
kubectl create ns argo
kubectl apply -n argo -f
https://raw.githubusercontent.com/argoproj/argo/stable/manifests/quick-start-postgres.yaml
```

> *namespace Workflowcluster install https://argoproj.github.io/argo-workflows/*

Argo CRD WorkflowTemplateWorkflowTemplatehttps://argoproj.github.io/argo-workflows/workflow-concepts/

1Template

template template `containerscriptdagstepsresource` `suspend` template Pod ——container /script/resource template Pod dag/steps template template container/script/resource

- **container** Kubernetes container spec
- **script** Container template Source
- **resource** template kubernetes action create, apply, delete template
- **suspend**Suspend template CLI argo resumeAPI UI
- **steps**Steps Template Steps [--] [-]
- **dag**DAG template DAG dependencies DAG https://github.com/argoproj/a...

2Workflow

Workflow spec templates template

hello world

Content Menu

-
- Argo
-
  - 1Template
  - 2Workflow
  - 3 WorkflowTemplate
  - 4Workflow Overview
- Sidecar
  - 1Init
  - 2Wait
- Inputs and Outputs
  - 1Artifact
  - 2Script
  - 3Parameter
  - 4Volume
-
  - 1
  - 2
  - 3
  - 4
  - 5
-

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-
  labels:
    workflows.argoproj.io/archive-strategy: "false"
spec:
  entrypoint: whalesay
  templates:
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [cowsay]
      args: ["hello world"]
```

Workflow templates container template whalesay

workflow

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: steps-
spec:
  entrypoint: hello-hello-hello

  # 在 templates 中有两个 template，一个为 hello-hello-hello，一个为 whalesay
  templates:
  - name: hello-hello-hello
    # Instead of just running a container
    # This template has a sequence of steps
    steps:                    # 该 template 的类型是 steps
    - - name: hello1          # 在 steps 类型中，[--] 代表顺序执行，[-] 代表并行执行
        template: whalesay    # 这里引用了下面的 template
        arguments:
          parameters:
          - name: message
            value: "hello1"
    - - name: hello2a         # 两个短杠 [--] => 顺序执行
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "hello2a"
      - name: hello2b         # 一个短杠 [-] => 并行执行
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "hello2b"
# 第二个 template
  - name: whalesay
    inputs:
      parameters:
      - name: message
    container:
      image: docker/whalesay
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

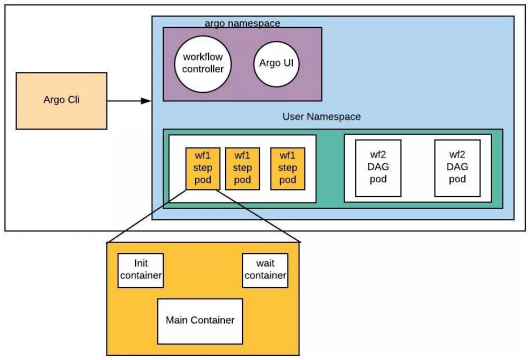
3WorkflowTemplate

WorkflowTemplate Workflow Workflow template WorkflowTemplate Workflow

```yaml
apiVersion: argoproj.io/v1alpha1
kind: WorkflowTemplate
metadata:
  name: workflow-template-submittablespec:
  entrypoint: whalesay-template
  arguments:
    parameters:
      - name: message
        value: hello world
  templates:
    - name: whalesay-template
      inputs:
        parameters:
          - name: message
      container:
        image: docker/whalesay
        command: [cowsay]
        args: ["{{inputs.parameters.message}}"]
```

4Workflow Overview

ARGO Workflow Overview



Argo Argo Workflow**Workflow Argo**

-
-

Workflow Active “”

Workflow Template Workflow Workflow WorkflowTemplate Workflow WorkflowTemplate Submit Workflow

Workflow entrypoint template entrypoint workflow template Pod Pod Main Container **Sidecar**

Sidecar

Argo Sidecar **`argoexec`**Argo executor

1Init

template inputs `artifact` script script Argo pod **InitContainer** —— argoexec argoexec init `Init Container artifact`

```
func loadArtifacts() error {
        wfExecutor := initExecutor()
        defer wfExecutor.HandleError()
        defer stats.LogStats()

        // Download input artifacts
        err := wfExecutor.StageFiles()
        if err != nil {
                wfExecutor.AddError(err)
                return err
        }
        err = wfExecutor.LoadArtifacts()
        if err != nil {
                wfExecutor.AddError(err)
                return err
        }
        return nil
}
```

2Wait

Resource templateArgo Wait Container Main Container Sidecar Wait Container `argoexec` `argoexec wait`Resource Resource template `argoexec` Main Container

```
func waitContainer() error {
        wfExecutor := initExecutor()
        defer wfExecutor.HandleError()
        defer stats.LogStats()
        stats.StartStatsTicker(5 * time.Minute)

        defer func() {
                // Killing sidecar containers
                err := wfExecutor.KillSidecars()
                if err ≠ nil {
                        log.Errorf("Failed to kill sidecars: %s", err.Error())
                }
        }()

        // Wait for main container to complete
        waitErr := wfExecutor.Wait()
        if waitErr ≠ nil {
                wfExecutor.AddError(waitErr)
                // do not return here so we can still try to kill sidecars & save outputs
        }

        // Capture output script result
        err := wfExecutor.CaptureScriptResult()
        if err ≠ nil {
                wfExecutor.AddError(err)
                return err
        }
        // Capture output script exit code
        err = wfExecutor.CaptureScriptExitCode()
        if err ≠ nil {
                wfExecutor.AddError(err)
                return err
        }
        // Saving logs
        logArt, err := wfExecutor.SaveLogs()
        if err ≠ nil {
                wfExecutor.AddError(err)
                return err
        }
        // Saving output parameters
        err = wfExecutor.SaveParameters()
        if err ≠ nil {
                wfExecutor.AddError(err)
                return err
        }
        // Saving output artifacts
        err = wfExecutor.SaveArtifacts()
        if err ≠ nil {
                wfExecutor.AddError(err)
                return err
        }
        err = wfExecutor.AnnotateOutputs(logArt)
        if err ≠ nil {
                wfExecutor.AddError(err)
                return err
        }

        // To prevent the workflow step from completing successfully, return the error
occurred during wait.
        if waitErr ≠ nil {
                return waitErr
        }

        return nil
}
```

Inputs and Outputs

Workflow Step **Argo Artifact Parameter**

1Artifact

Argo Artifact Artifact Artifact Repository Config Map Workflow Argo

Name	Inputs	Outputs	Usage (Feb 2020)
Artifactory	Yes	Yes	0.11
GCS	Yes	Yes	-
Git	Yes	No	-
HDFS	Yes	Yes	0.03
HTTP	Yes	No	0.02
OSS	Yes	Yes	-
Raw	Yes	No	0.05
S3	Yes	Yes	0.86

Artifact

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: artifact-passing-spec:
  entrypoint: artifact-example
  templates:
  - name: artifact-example
    steps:
    - - name: generate-artifact
        template: whalesay
    - - name: consume-artifact
        template: print-message
        arguments:
          artifacts:
          # bind message to the hello-art artifact
          # generated by the generate-artifact step
          - name: message
            from: "{{steps.generate-artifact.outputs.artifacts.hello-art}}"

  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["cowsay hello world | tee /tmp/hello_world.txt"]
    outputs:
      artifacts:
      # generate hello-art artifact from /tmp/hello_world.txt
      # artifacts can be directories as well as files
      - name: hello-art
        path: /tmp/hello_world.txt

  - name: print-message
    inputs:
      artifacts:
      # unpack the message input artifact
      # and put it at /tmp/message
      - name: message
        path: /tmp/message
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["cat /tmp/message"]
```

Artifact tgz(tar +gzip) archive

whalesay template cowsay /tmp/hello-world.txt hello-art Artifact print-message template message Artifact /tmp/message cat /tmp/message

Sidecar **Init Container Artifact** Sidecar Argo

2Script

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: scripts-bash-spec:
  entrypoint: bash-script-example
  templates:
  - name: bash-script-example
    steps:
    - - name: generate
        template: gen-random-int-bash
    - - name: print
        template: print-message
        arguments:
          parameters:
          - name: message
            value: "{{steps.generate.outputs.result}}"  # The result of the here-script

  - name: gen-random-int-bash
    script:
      image: debian:9.4
      command: [bash]
      source: |                                          # Contents of the here-script
        cat /dev/urandom | od -N2 -An -i | awk -v f=1 -v r=100 '{printf "%i\n", f + r * $1
/ 65536}'

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

  - name: print-message
    inputs:
      parameters:
      - name: message
    container:
      image: alpine:latest
      command: [sh, -c]
      args: ["echo result was: {{inputs.parameters.message}}"]
```

script templatescript source commandbashpythonjs etc

Script template result template **{{steps.generate.outputs.result}}** generate template

> *{{xxx}} Argo*
>
> - *https://github.com/argoproj/a...*
> - *https://github.com/argoproj/a...*

Sidecar Wait Container

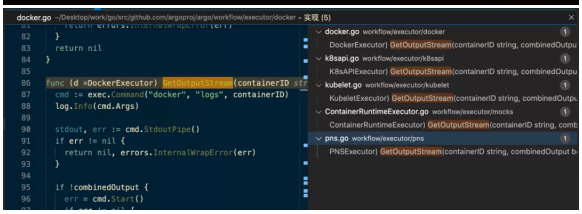
```
// CaptureScriptResult will add the stdout of a script template as output result
func (we *WorkflowExecutor) CaptureScriptResult() error {

        ...

        log.Infof("Capturing script output")
        mainContainerID, err := we.GetMainContainerID()
        if err != nil {
                return err
        }
        reader, err := we.RuntimeExecutor.GetOutputStream(mainContainerID, false)
        if err != nil {
                return err
        }
        defer func() { _ = reader.Close() }()
        bytes, err := ioutil.ReadAll(reader)
        if err != nil {
                return errors.InternalWrapError(err)
        }
        out := string(bytes)
        // Trims off a single newline for user convenience
        outputLen := len(out)
        if outputLen > 0 && out[outputLen-1] == '\n' {
                out = out[0 : outputLen-1]
        }

        const maxAnnotationSize int = 256 * (1 << 10) // 256 kB
        // A character in a string is a byte
        if len(out) > maxAnnotationSize {
                log.Warnf("Output is larger than the maximum allowed size of 256 kB, only
the last 256 kB were saved")
                out = out[len(out)-maxAnnotationSize:]
        }

        we.Template.Outputs.Result = &out
        return nil
}
```



Wait Container Volume Mount

```
volumeMounts:
    - mountPath: /argo/podmetadata
      name: podmetadata
    - mountPath: /var/run/docker.sock
      name: docker-sock
      readOnly: true
    - mountPath: /argo/secret/my-minio-cred
      name: my-minio-cred
      readOnly: true
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-b5grl
      readOnly: true
```

Wait Container docker.sock service account Main Container Workflow Workflow Workflow Step Workflow

3Parameter

Parameter Parameter stdout

```
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["echo -n hello world > /tmp/hello_world.txt"]  # generate the content of
hello_world.txt
    outputs:
      parameters:
      - name: hello-param       # name of output parameter
        valueFrom:
          path: /tmp/hello_world.txt    # set the value of hello-param to the contents of
this hello-world.txt
```

4Volume

Argo Volume Inputs Outputs Workflow Spec Volume

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: volumes-pvc-spec:
  entrypoint: volumes-pvc-example
  volumeClaimTemplates:                 # define volume, same syntax as k8s Pod spec
  - metadata:
      name: workdir                     # name of volume claim
    spec:
      accessModes: [ "ReadWriteOnce" ]
      resources:
        requests:
          storage: 1Gi                  # Gi => 1024 * 1024 * 1024
```

template mount volume

```
  - name: whalesay
    container:
      image: docker/whalesay:latest
      command: [sh, -c]
      args: ["echo generating message in volume; cowsay hello world | tee
/mnt/vol/hello_world.txt"]
      # Mount workdir volume at /mnt/vol before invoking docker/whalesay
      volumeMounts:                 # same syntax as k8s Pod spec
      - name: workdir
        mountPath: /mnt/vol
```

1

Workflow :

```
  templates:
  - name: loop-example
    steps:
    - - name: print-message
        template: whalesay
        arguments:
          parameters:
          - name: message
            value: "{{item}}"
        withItems:              # invoke whalesay once for each item in parallel
        - hello world           # item 1
        - goodbye world         # item 2
```

withItems step

```
// expandStepGroup looks at each step in a collection of parallel steps, and expands all
steps using withItems/withParam
func (woc *wfOperationCtx) expandStepGroup(sgNodeName string, stepGroup
[]wfv1.WorkflowStep, stepsCtx *stepsContext) ([]wfv1.WorkflowStep, error) {
        newStepGroup := make([]wfv1.WorkflowStep, 0)
        for _, step := range stepGroup {
                if !step.ShouldExpand() {
                        newStepGroup = append(newStepGroup, step)
                        continue
                }
                expandedStep, err := woc.expandStep(step)
                if err != nil {
                        return nil, err
                }
                if len(expandedStep) == 0 {
                        // Empty list
                        childNodeName := fmt.Sprintf("%s.%s", sgNodeName, step.Name)
                        if woc.wf.GetNodeByName(childNodeName) == nil {
                                stepTemplateScope := stepsCtx.tmplCtx.GetTemplateScope()
                                skipReason := "Skipped, empty params"
                                woc.log.Infof("Skipping %s: %s", childNodeName, skipReason)
                                woc.initializeNode(childNodeName, wfv1.NodeTypeSkipped,
stepTemplateScope, &step, stepsCtx.boundaryID, wfv1.NodeSkipped, skipReason)
                                woc.addChildNode(sgNodeName, childNodeName)
                        }
                }
                newStepGroup = append(newStepGroup, expandedStep...)
        }
        return newStepGroup, nil
}
```

2

when

```
  templates:
  - name: coinflip
    steps:
    # flip a coin
    - - name: flip-coin
        template: flip-coin
    # evaluate the result in parallel
    - - name: heads
        template: heads                 # call heads template if "heads"
        when: "{{steps.flip-coin.outputs.result}} == heads"
      - name: tails
        template: tails                 # call tails template if "tails"
        when: "{{steps.flip-coin.outputs.result}} == tails"
```

3

```
  templates:
  - name: retry-backoff
    retryStrategy:
      limit: 10
      retryPolicy: "Always"
      backoff:
        duration: "1"      # Must be a string. Default unit is seconds. Could also be a
Duration, e.g.: "2m", "6h", "1d"
        factor: 2
        maxDuration: "1m"  # Must be a string. Default unit is seconds. Could also be a
Duration, e.g.: "2m", "6h", "1d"
```

4

Template

```yaml
apiVersion: argoproj.io/v1alpha1kind: Workflowmetadata:
  generateName: coinflip-recursive-spec:
  entrypoint: coinflip
  templates:
  - name: coinflip
    steps:
    # flip a coin
    - - name: flip-coin
        template: flip-coin
    # evaluate the result in parallel
    - - name: heads
        template: heads                 # call heads template if "heads"
        when: "{{steps.flip-coin.outputs.result}} == heads"
      - name: tails                     # keep flipping coins if "tails"
        template: coinflip
        when: "{{steps.flip-coin.outputs.result}} == tails"

  - name: flip-coin
    script:
      image: python:alpine3.6
      command: [python]
      source: |
        import random
        result = "heads" if random.randint(0,1) == 0 else "tails"
        print(result)

  - name: heads
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was heads\""]
```

```
argo get coinflip-recursive-tzcb5

STEP                        PODNAME                               MESSAGE
 ✓ coinflip-recursive-vhph5
 ├──✓ flip-coin             coinflip-recursive-vhph5-2123890397
 └──✓ heads                 coinflip-recursive-vhph5-128690560
    └─o tails

STEP                         PODNAME                                MESSAGE
 ✓ coinflip-recursive-tzcb5
 ├──✓ flip-coin              coinflip-recursive-tzcb5-322836820
 └──o heads
    └─✓ tails
      ├──✓ flip-coin         coinflip-recursive-tzcb5-1863890320
      └──o heads
         └─✓ tails
           ├──✓ flip-coin    coinflip-recursive-tzcb5-1768147140
           └──o heads
              └─✓ tails
                ├──✓ flip-coin  coinflip-recursive-tzcb5-4080411136
                └──✓ heads      coinflip-recursive-tzcb5-4080323273
                   └─o tails
```

5

workflow template

```yaml
spec:
  entrypoint: intentional-fail
  onExit: exit-handler                # invoke exit-handler template at end of the
workflow
  templates:
  ...
```

- https://segmentfault.com/a/1190000038979821
- https://github.com/argoproj/argo-workflows