

# Construire et déployer son application avec Argo dans Kubernetes

Date : 21 avril 2022

Speakers : Paul-Henry Perrissel (LinkValue) et Nicolas Mpacko Tongo (Postivie Thinking Company)

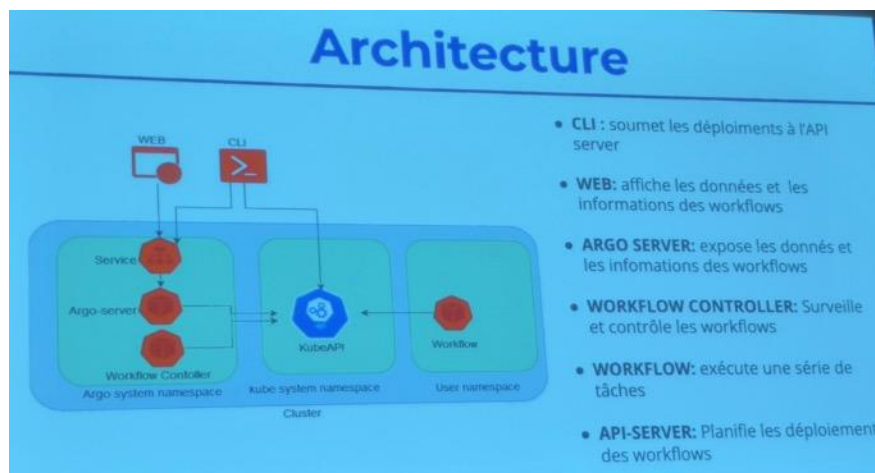
Format : conference (45mn)

## Présentation des différents composants d'Argo

- Boite à outils très complète
- Argo workflow permet de gérer des pipelines
- Argo Rollout : adresse problématique de blue-green, canary release ....
- ArgoCD
- Argo Event

## Zoom sur Argo Workflow

Plateforme d'exécution de workflows, container native et OSS. Permet d'orchestrer des jobs // dans k8s, exécuter des pipelines CI/CD ...



## Concepts :

- Workflow : travail à la chaîne, gestion de l'état => ressemble à un déploiement
- Workflow Specification => deployment specification
- Template => container
- Workflow Instance : Live Object => pod

Un workflow est une ressource k8s.

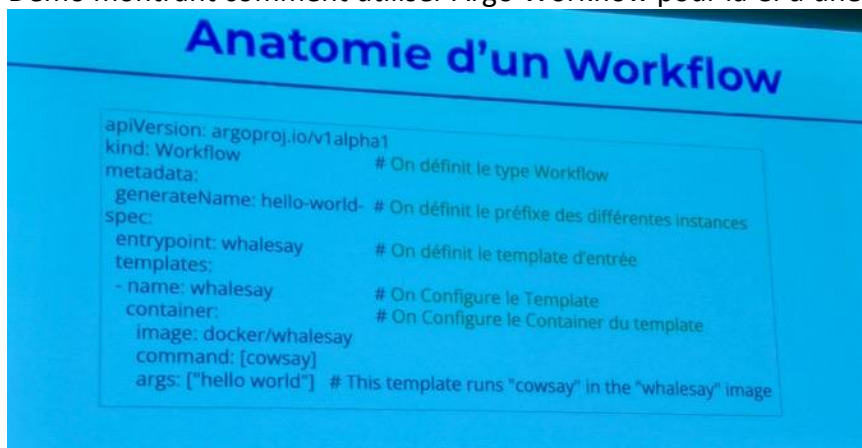
## Template Patterns Language :

- **Container** : portable (plusieurs langages), décorable avec un script
- **Steps invocator** : exécute en // les tâches, orchestration déléguée par Argo. Limite du séquençement
- **DAG Invocator** : arbre de tâches avec des branchements conditionnels
- **Artifacts** : permet d'enregistrer des fichiers sous forme d'artefacts visible à Argo
- **Cache** : cache entre le template et le container permettant d'accélérer les workflows (ex : dépendances Maven et NPM) => cache supporté : configMap
- **Parameters** : permet de variabiliser des templates (ex : port et host). La tâche qui appelle le template passe la valeur des paramètres attendus.
- **Workflow Template** : lorsque 2 workflows partagent les mêmes tâches, on peut déclarer un WorkflowTemplate
- **Cluster Workflow Template** : permet de mutualiser des tâches entre différents namespaces

#### Workflow Pattern :

- **Composite Workflow** : exécution hiérarchique et coordonnées de workflow
- **Cron Workflow** : exécution périodique d'un workflow

Démo montrant comment utiliser Argo Workflow pour la CI d'une webapp.



#### Zoom sur Argo CD

Plateforme de déploiement sur Kubernetes.

Permet de suivre les indicateurs de santé des ressources déployées.

Met à jour, corrige et nettoie les environnements.

Système de Custom Resource Definition (CRD) pour le suivi des applications. Dispose d'une GUI et d'une CLI pour suivre visuellement les applis.

#### Options possibles :

- Détection automatique des outils de déploiement : Helm, Kustomize ....
- Configuration des options de synchro
- Possibilité d'ignorer certaines ressources
- Gestion des lifecycles hooks
- Usage de templates pour la génération d'application
- Génération de status badge (import dans du markdown)

Options de synchronisation permettant un ensemble d'automatisation :

- Self Heal et Auto Prune
- Ajustement de paramètres

5 phases de lifecycle hooks : PreSync, Sync, Skip, PostSync et SyncFail

Exemple : un PreSync permet de migrer le schéma d'une appli

On retrouve le système de hook de Helm

ArgoCD se pilote à travers des manifestes.

ArgoCD peut aller chercher les sources (les Charts) dans un repo et une branche particulière.

Les ApplicationSets permettent l'usage de plusieurs générateurs :

- Git
- Cluster et ClusterDecisionResource
- Pull Request (ne fonctionne qu'avec GitHub)
- List

Combinaison de générateurs : Merge et Matrix.

**Exemple d'ApplicationSet**

```
kind: ApplicationSet
metadata: [...]
spec:
  generators:
  - git:
      repoURL: https://gitlab.com/lopment/travel.git
      revision: release/0.0.1
      directories:
      - path: charts/*
  template:
    metadata:
      name: "travel-{{path.basename}}"
    spec:
      project: default
      source:
        repoURL: https://gitlab.com/lopment/travel.git
        targetRevision: release/0.0.1
        path: "{{path}}"
      destination:
        namespace: travel
        server: https://kubernetes.default.svc
      syncPolicy: [...]
```

Annotations explaining the fields:

- Utilisation du CRD
- La liste de générateurs utilisés
- Utilisation du générateur Git ...
- ... à partir de ce repo ...
- ... et de cette révision
- Les deux répertoires à exploiter
- Le format du nom de l'application
- Source Git comme vu pour les applications ...
- ... avec surcharge du path depuis les répertoires
- Cluster et namespace de destination pour les ressources générées
- Les policies de sync qui peuvent être appliquées sur l'application

Démo d'un ApplicationSets sélectionnant la PR taggée avec envtest => le namespace avec le numéro de la PR détectée par Argo est créé.