Felix Hochleitner
Sr. DevOps Engineer
Gepardec

GitOps powered CICD

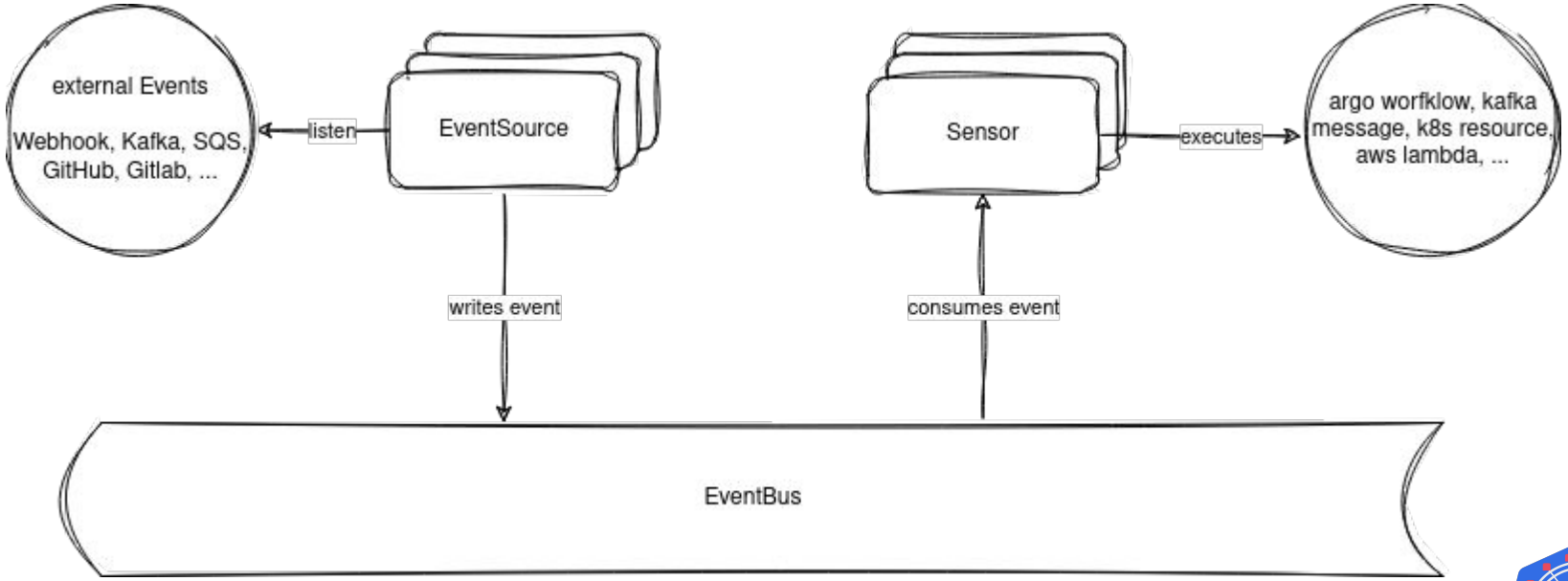
Gluing it together with Argo Events, Workflows & CD

KCD AUSTRIA September 26th-27th 2023

Argo Events

- Event-based dependency manager for kubernetes
- Core components:
  - EventSource
  - Sensor
  - Trigger
  - EventBus

Argo Events - Architecture Overview



Argo Events

```yaml
apiVersion: argoproj.io/v1alpha1
kind: EventSource
metadata:
  name: demo-microservice-configurator
  namespace: argo-events-eventbus
spec:
  service:
    ports:
      - port: 12000
        targetPort: 12000
  github:
    demo-microservice-event:
      owner: "gepaplexx-demos"
      repository: "demo-microservice"
      webhook:
        endpoint: "/event"
        port: "12000"
        method: "POST"
      events:
        - "push"
        - "create"
        - "delete"
      webhookSecret:
        name: workflow-demo-microservice
        key: webhook-secret
      insecure: false
      active: true
```

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Sensor
metadata:
  name: demo-microservice
  namespace: argo-events-eventbus
spec:
  template:
    serviceAccountName: operate-workflow-sa
  dependencies:
    - name: demo-microservice-event
      eventSourceName: demo-microservice-configurator
      eventName: demo-microservice-event
      filters:
        data:
          - path: "body.X-GitHub-Event"
            type: string
            value:
              - "push"
              - "create"
              - "delete"
```

```yaml
triggers:
  - template:
      name: demo-microservice-event-trigger
      conditions: "demo-microservice-event"
      argoWorkflow:
        parameters:
          - src:
              dependencyName: demo-microservice-event
              dataTemplate: '{{ toJson .Input.body }}'
            dest: spec.arguments.parameters.0.value
        operation: submit
        source:
          resource:
            apiVersion: argoproj.io/v1alpha1
            kind: Workflow
            metadata:
              generateName: demo-microservice-configurator-
              namespace: demo-microservice-cicd
            spec:
              arguments:
                parameters:
                  - name: event
                    value: "will-be-set-from-payload"
              entrypoint: configure
              workflowTemplateRef:
                name: workflow-configurator-github
                clusterScope: true
```

Argo Events - Event Transformation

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Sensor
metadata:
  name: demo-microservice-gitlab
  namespace: gp-cicd-eventbus
spec:
  template:
    serviceAccountName: operate-workflow-sa
  dependencies:
    - name: demo-microservice-gitlab-delete
      eventSourceName: demo-microservice-gitlab-configurator
      eventName: demo-microservice-gitlab-event
      filters:
        data:
          - path: body.after
            type: "string"
            operator: "="
            value:
              - "0000000000000000000000000000000000000000"
      transform:
        jq: .body.event_name = "delete"
```

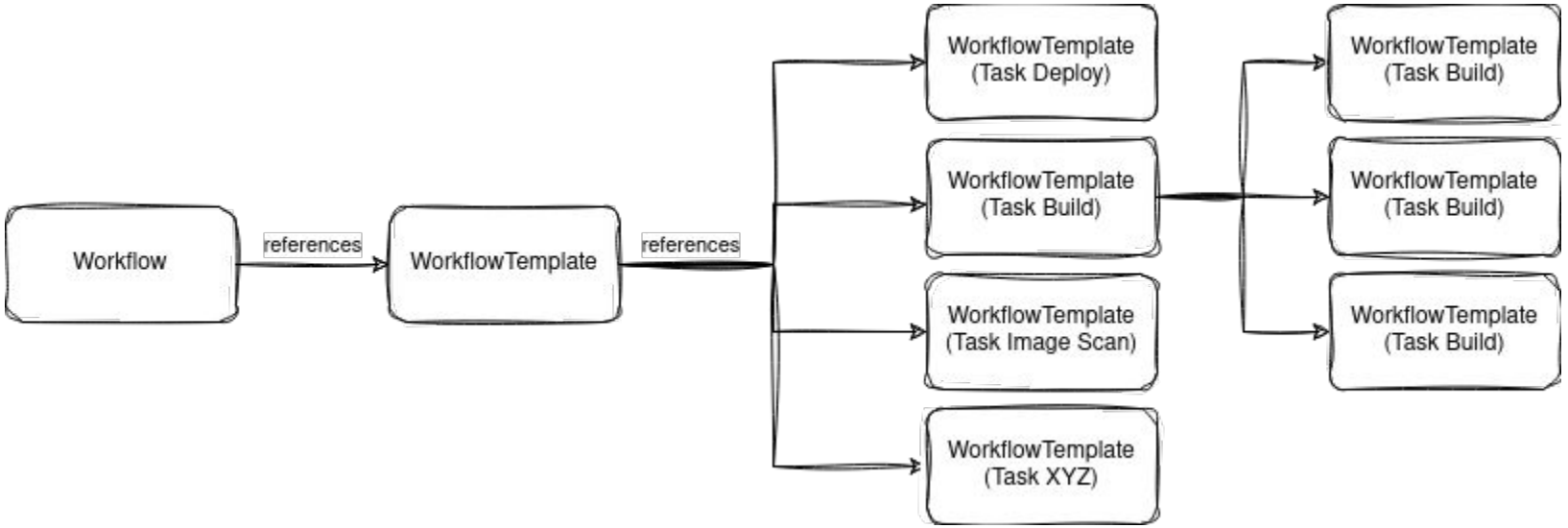
Argo Workflows

- Container based workflow engine
- CICD processes, infrastructure automation, machine learning, data processing, ...
- Core components
  - Workflow
  - (Cluster-)WorkflowTemplate
  - CronWorkflow

Argo Workflows

- Step and DAG Workflows
- Extensive templating support
- Supports various types of steps/tasks
- Prometheus Metrics
- Workflow of Workflows

Workflow of Workflows



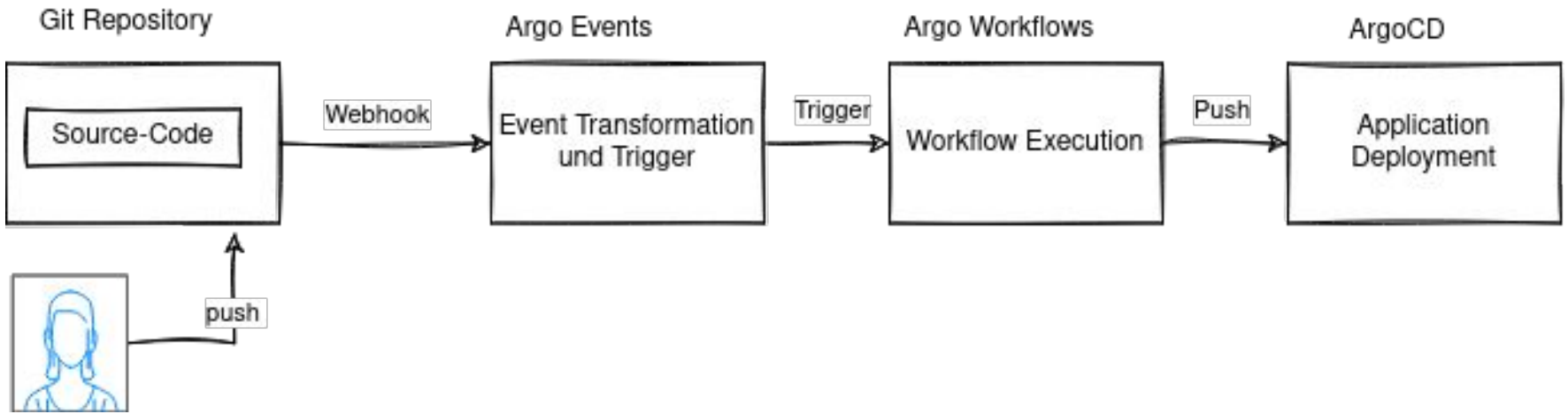
Argo CD

- GitOps engine
- Used for deploying:
  - Argo Events & Workflows
  - WorkflowTemplates
  - EventSources & Sensors
  - Applications
  - Feature-Branch environments

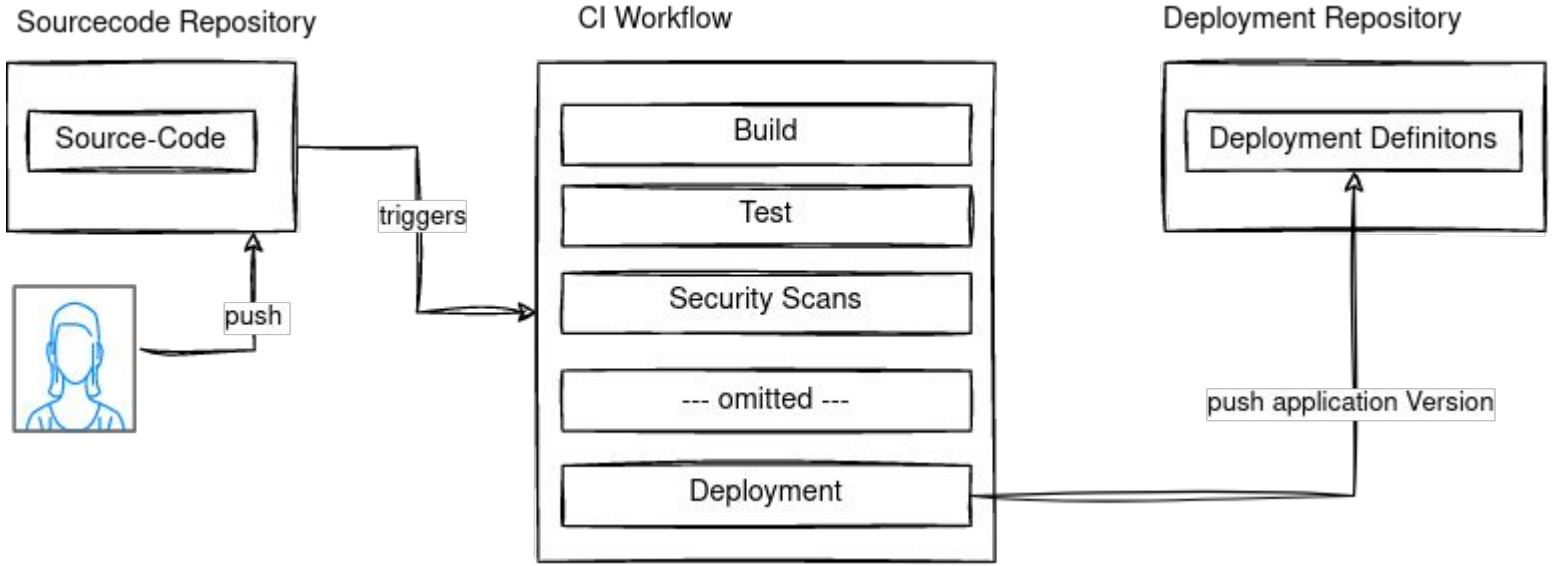
Argo CD - ApplicationSet

- Generate Applications
- Used for dynamic feature-branch environments
- Updated by workflow instance

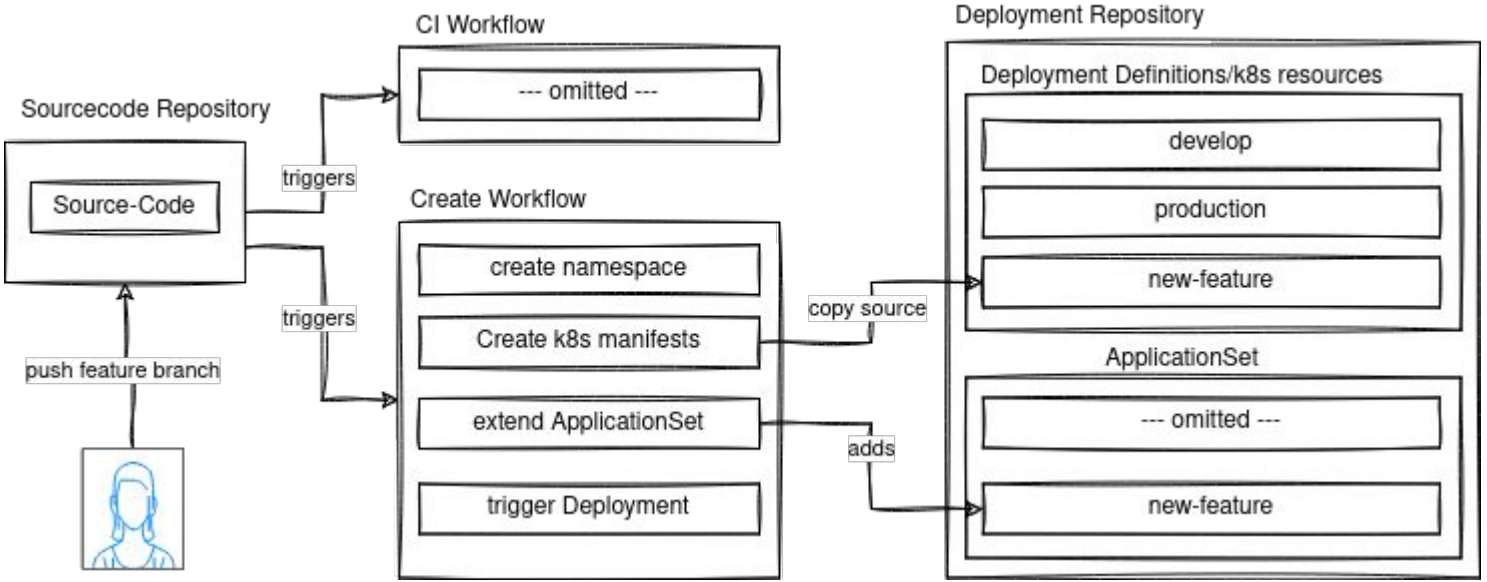
Putting everything together



Build Workflow

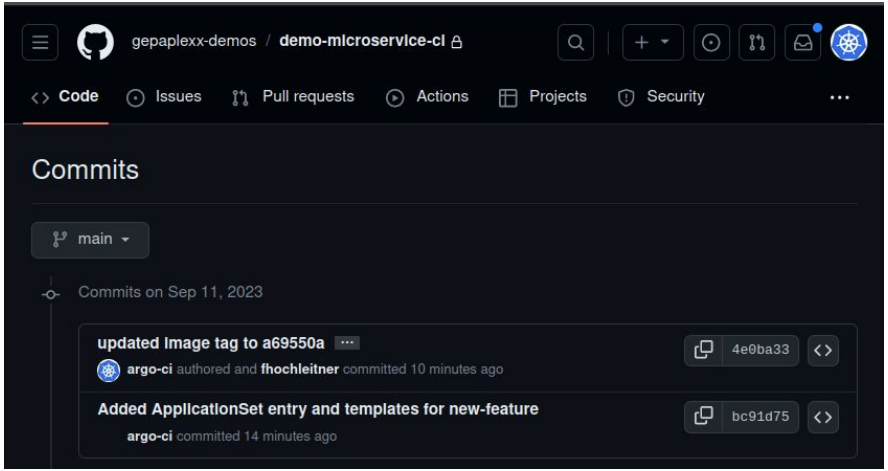


Dynamic Environments



Dynamic Environments

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: demo-microservice
  namespace: gp-cicd-tools
spec:
  syncPolicy:
    preserveResourcesOnDeletion: false
  generators:
    - list:
        elements:
          - cluster: dev
            url: https://kubernetes.default.svc
            branch: dev
          - cluster: prod
            url: https://api.production-cluster.mycompany.com
            branch: prod
          - cluster: new-feature
            url: https://kubernetes.default.svc
            branch: main
  template:
    metadata:
      name: "demo-microservice-{{cluster}}"
    spec:
      project: demo-microservice
      -- ommited for brevity --
      source:
        repoURL: git@github.com:gepaplexx-demos/demo-microservice-ci.git
        targetRevision: "{{ branch }}"
        path: apps/env/{{ cluster }}
      destination:
        server: "{{url}}"
        namespace: "demo-microservice-{{cluster}}"
```



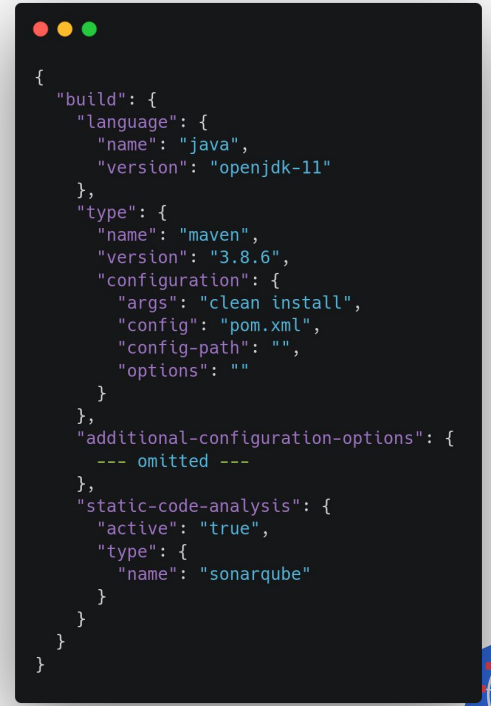
KCD Austria 2023

That’s nice and all but …

… it’s missing flexibility

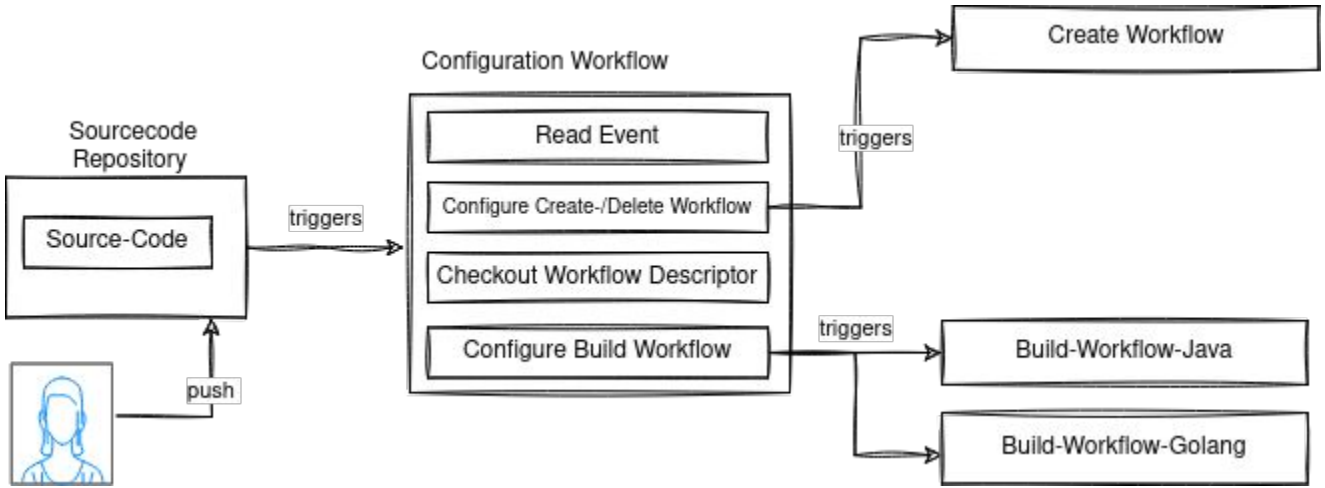
Workflow Descriptor

- Definition of workflow parameters
- Branch specific
- Leverages the power of templating



```
{
  "build": {
    "language": {
      "name": "java",
      "version": "openjdk-11"
    },
    "type": {
      "name": "maven",
      "version": "3.8.6",
      "configuration": {
        "args": "clean install",
        "config": "pom.xml",
        "config-path": "",
        "options": ""
      }
    },
    "additional-configuration-options": {
      --- omitted ---
    },
    "static-code-analysis": {
      "active": "true",
      "type": {
        "name": "sonarqube"
      }
    }
  }
}
```

Extended Workflow Architecture



Workflow Templating at Runtime

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ClusterWorkflowTemplate
metadata:
  name: workflow-build-java
spec:
  entrypoint: pipeline
  arguments:
    parameters:
      --- omitted for brevity ---
  templates:
    - name: pipeline
      metadata:
        labels:
          template: pipeline
      dag:
        tasks:
          - name: checkout
            templateRef:
              name: git-operations
              template: checkout
              clusterScope: true
          - name: build
            depends: "checkout"
            templateRef:
              name: "{{ workflow.parameters.build-type }}-operations"
              template: build
              clusterScope: true
            arguments:
              parameters:
                - name: config
                  value: "{{=jsonpath(workflow.parameters['build-config'], '$.config')}}"
                - name: config-path
                  value: "{{=jsonpath(workflow.parameters['build-config'], '$.config-path')}}"
                - name: args
                  value: "{{=jsonpath(workflow.parameters['build-config'], '$.args')}}"
                - name: options
                  value: "{{=jsonpath(workflow.parameters['build-config'], '$.options')}}"
```

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ClusterWorkflowTemplate
metadata:
  name: maven-operations
spec:
  templates:
    - name: build
      inputs:
        parameters:
          --- omitted for brevity ---
      container:
        name: maven
        image: ghcr.io/gepaplexx/maven:{{ workflow.parameters.build-type-version }}-{{
workflow.parameters.language-version }}
        command:
          - "/usr/bin/mvn-wrapper.sh"
        args:
          - "{{ inputs.parameters.args }}"
          - "{{ inputs.parameters.options }}"
          - "-f"
          - "/mnt/out/{{ workflow.parameters.reponame }}/{{ inputs.parameters.config-path }}/{{
inputs.parameters.config }}"
```

The Good, the Bad and the Ugly

- Everything is a kubernetes resource and can be managed by GitOps Engine
- Very powerful templating mechanisms for supplying generic and reusable workflows
- Can use alertmanager for build notifications via emission of prometheus metrics
- Grafana Dashboards for builds

- Learning curve is quite steep
- High cost of getting started

- Refactor early & chose the right tools for your tasks
- CICD processes impact development & deployment workflows

Code Snippets & Slides

Thank you