Designing a Complete CI/CD Pipeline Using Argo Events, Workflows, and CD

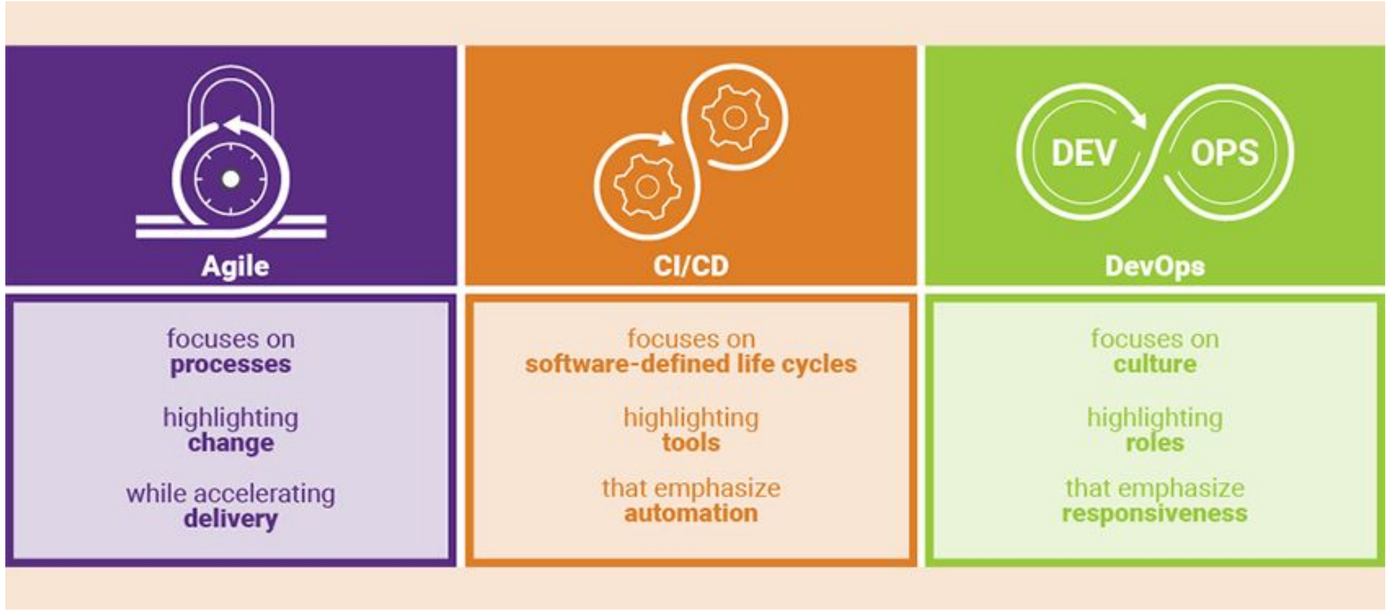
Julian Mazzitelli, CTO BioBox Analytics Inc.

GDG

$ whoami

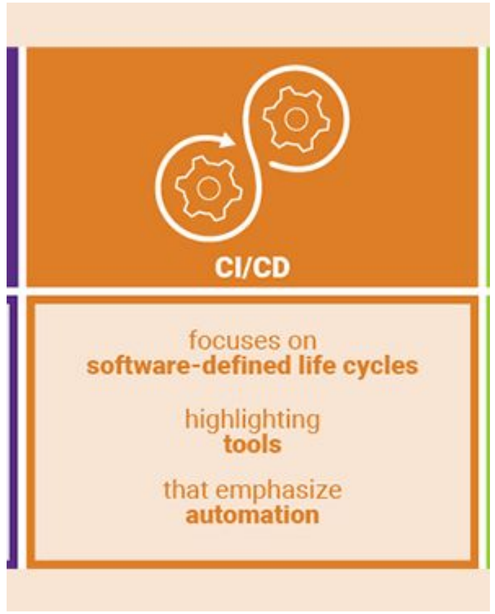
- BioBox Analytics Inc.
- Early stage startup, incorporated April 2019
- 3 full time
  - 2x developers
  - 1x developer + operations “full stack”
- 2 part time
  - Quality Assurance / Product Officer
- Cloud native stack - API talks to K8s
- WE NEED TO MOVE FAST!
  - Want a robust and flexible CI/CD process
  - Want Kubernetes native

Agility + CI/CD + DevOps = success



https://www.synopsys.com/blogs/software-security/agile-cicd-devops-difference/

CI/CD Stakeholders

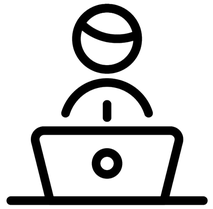


Developers

Operations
+
Security

QA
+
Product Officer
+
Users

CI/CD Stakeholder Concerns



Devs

- Don’t write CI pipelines
- Don’t like application manifests
- Don’t like yaml!
- Want visibility into CI/CD process

Ops

- Don’t like difficult to understand CI configuration
- Don’t like difficult to reuse CI pipelines
- Don’t like inflexible CI/CD
- Don’t like yaml!
- Want K8s native CI pipelines
- Want robust application lifecycle management

QA/PO

- Don’t like not knowing what version of which app they just tested out
- Don’t like not having a list of all deployed applications
- Don’t like incorrectly informing users which features are available on prod

Issues BioBox had with existing tools

- Drone
  - Can achieve modular pipelines via jsonnet plugin...but jsonnet is unfamiliar to developers
  - There was alpha support for Kubernetes runtime, however not configurable from CI config...Drone internally was creating Jobs/Pods, *was later deprecated* - drone/drone-runtime/issues/69
- GitLab
  - K8s GitLab runner a huge blackbox, don't want to maintain a fork, also different scope
  - Reusability via YAML DSL (".partial: &partial", "<<: *partial" !?) is annoying for Ops, difficult for Devs
- Tektoncd/pipeline
  - Was seen as alternative to Argo Workflows, which we already had operational experience with

Objectives - *Things We Knew We Wanted*

- Developers should feel comfortable reading and writing CI configuration
- CI pipelines should be kept DRY and modularized
- Flexible CI/CD configuration
- Audit log from Git event to deployed resources
- CI/CD observability tooling consistent with primary application
- Manual, schedule, or event-based triggering of CI pipelines
- Support many 3rd party dependencies (many Dockerfiles)
- Preview application for all PRs, easily accessible to QA/PO

Architecture overview: CI workflows

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-parameters-
spec:
  # invoke the whalesay template with
  # "hello world" as the argument
  # to the message parameter
  entrypoint: whalesay
  arguments:
    parameters:
    - name: message
      value: hello world

  templates:
  - name: whalesay
    inputs:
      parameters:
      - name: message       # parameter declaration
    container:
      # run cowsay with that message input parameter as args
      image: docker/whalesay
      command: [cowsay]
      args: ["{{inputs.parameters.message}}"]
```

Argo Workflows

- Like Job on steroids
- Parameters, Artifacts (Git, S3, +)
- Linear sequence of steps
- DAG of steps
- Retry-able

GDG

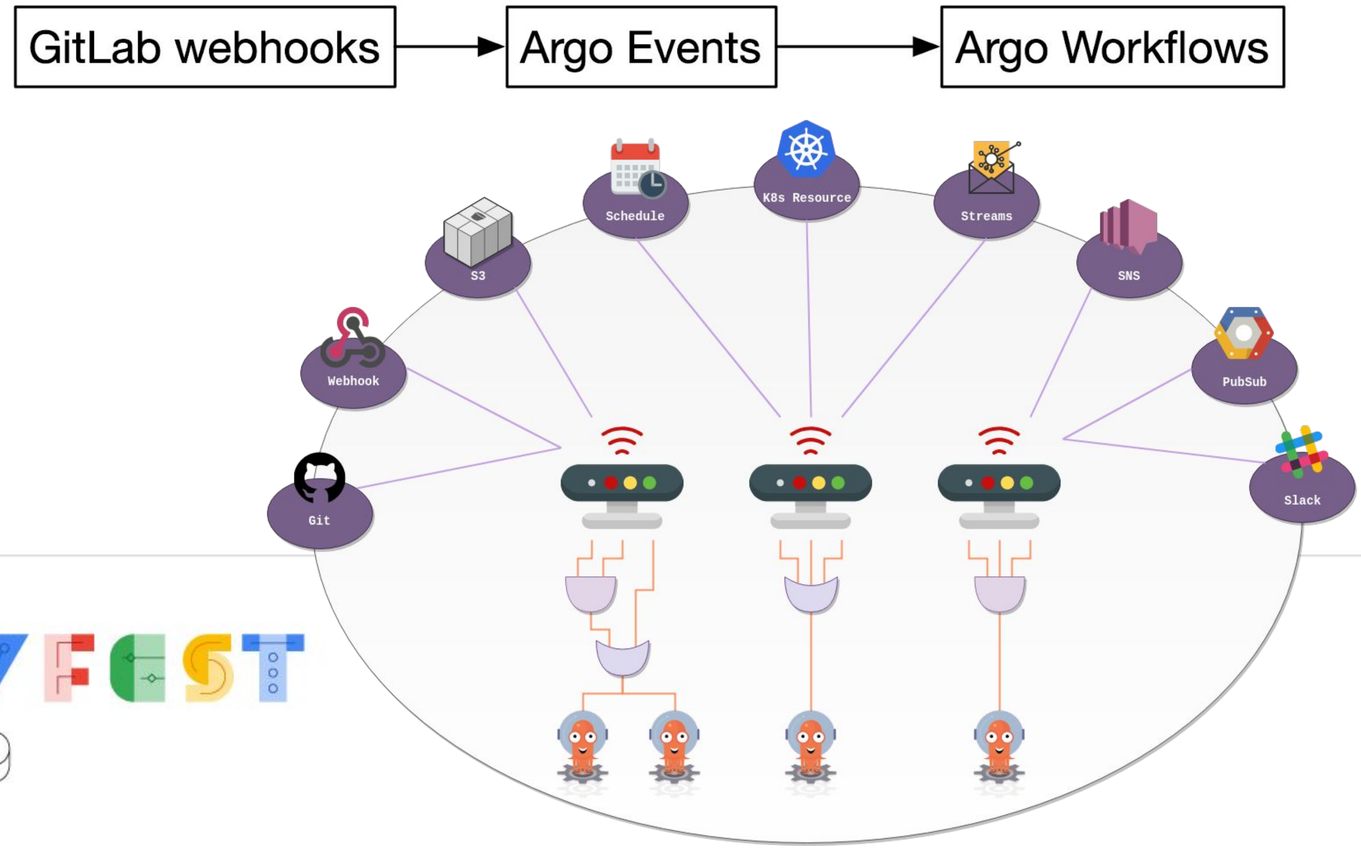
Architecture overview: Git webhooks

GitLab webhooks

Argo Workflows

- Push
- Tag
- New branch
- MR open/close/update
- MR comments

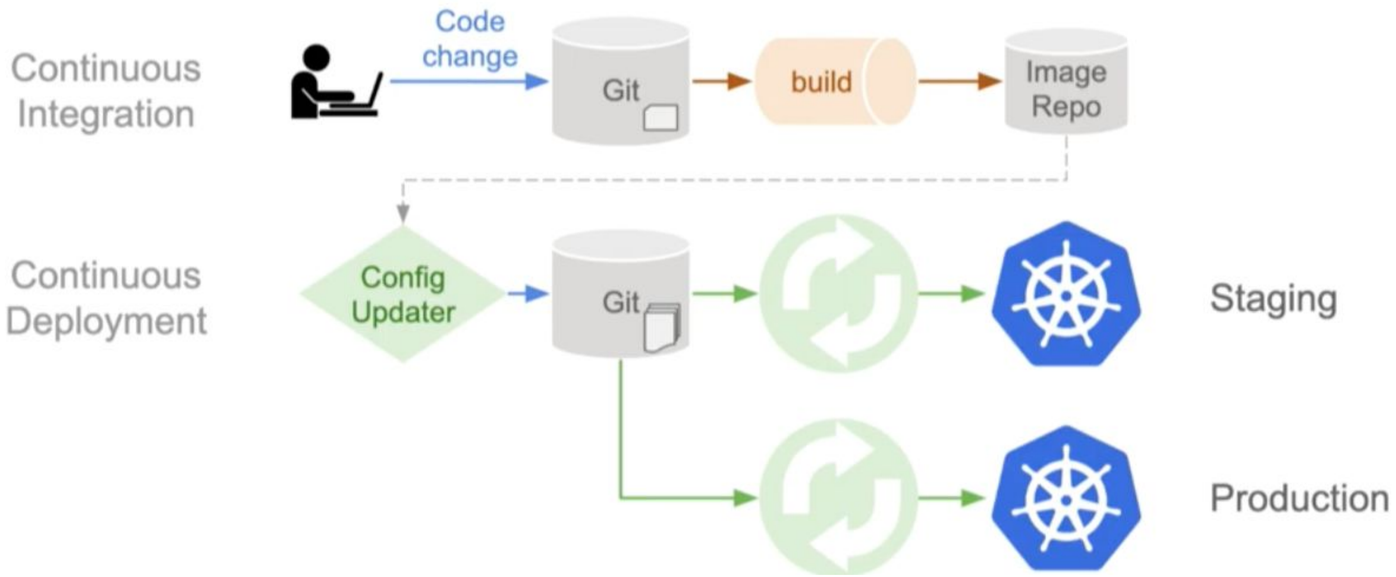
Architecture overview: Webhook to Workflow



"Argo Events is an event-based dependency manager for Kubernetes which helps you define multiple dependencies from a variety of event sources like webhook, s3, schedules, streams etc. and trigger Kubernetes objects after successful event dependencies resolution."

Architecture overview: GitOps “Application”

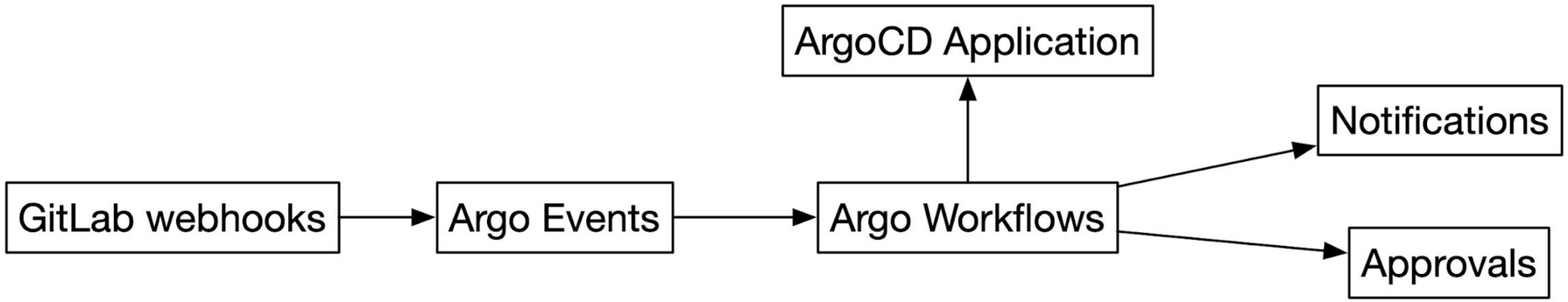
- GitOps “Application” CRD
- Defines git source and tracking
- Defines destination cluster
- Optionally defines tool settings - helm values, jsonnet top-level params, etc



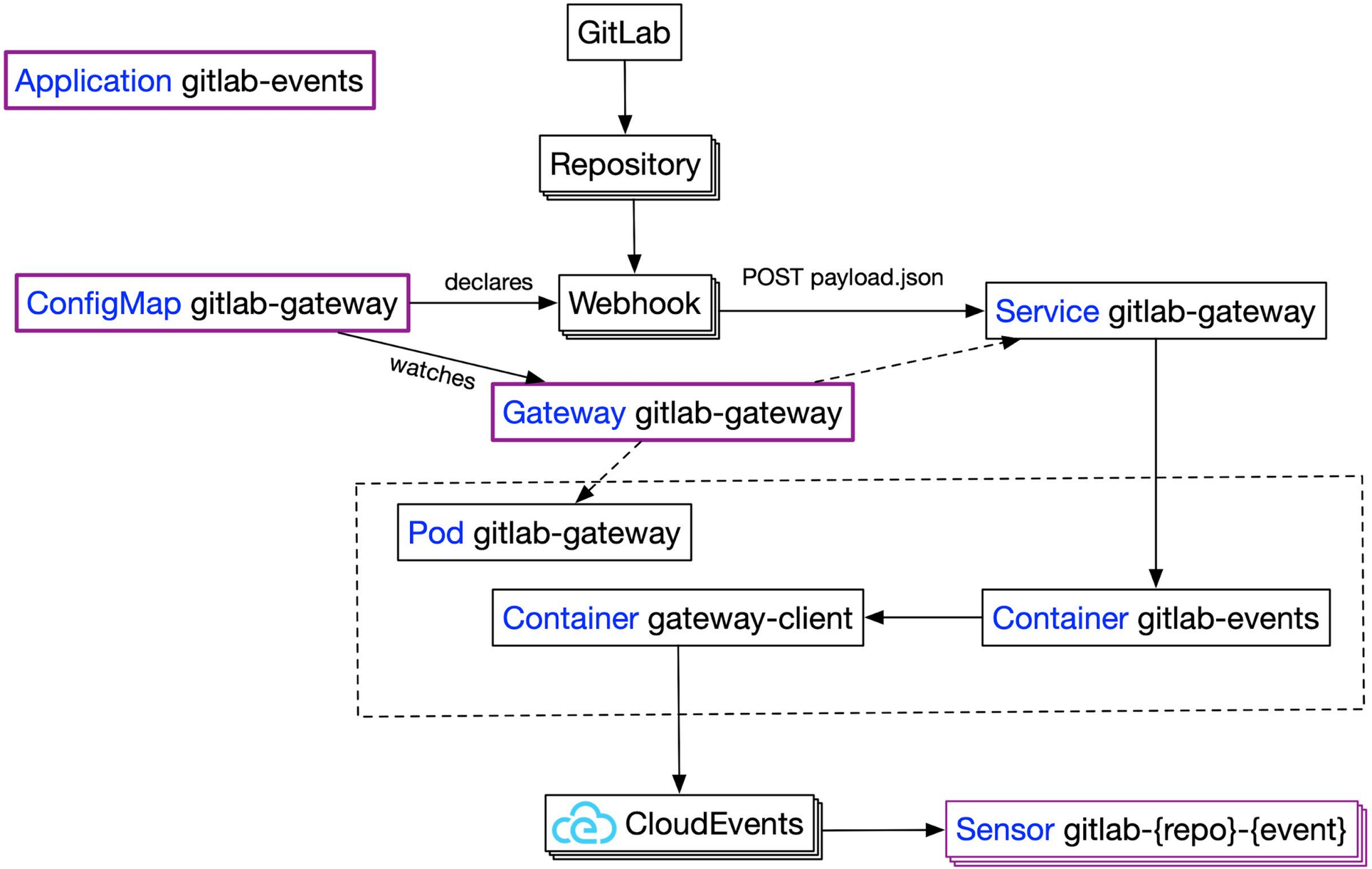
GDG

https://www.weave.works/blog/automate-kubernetes-with-gitops

Architecture overview: What **is** a CI/CD software?

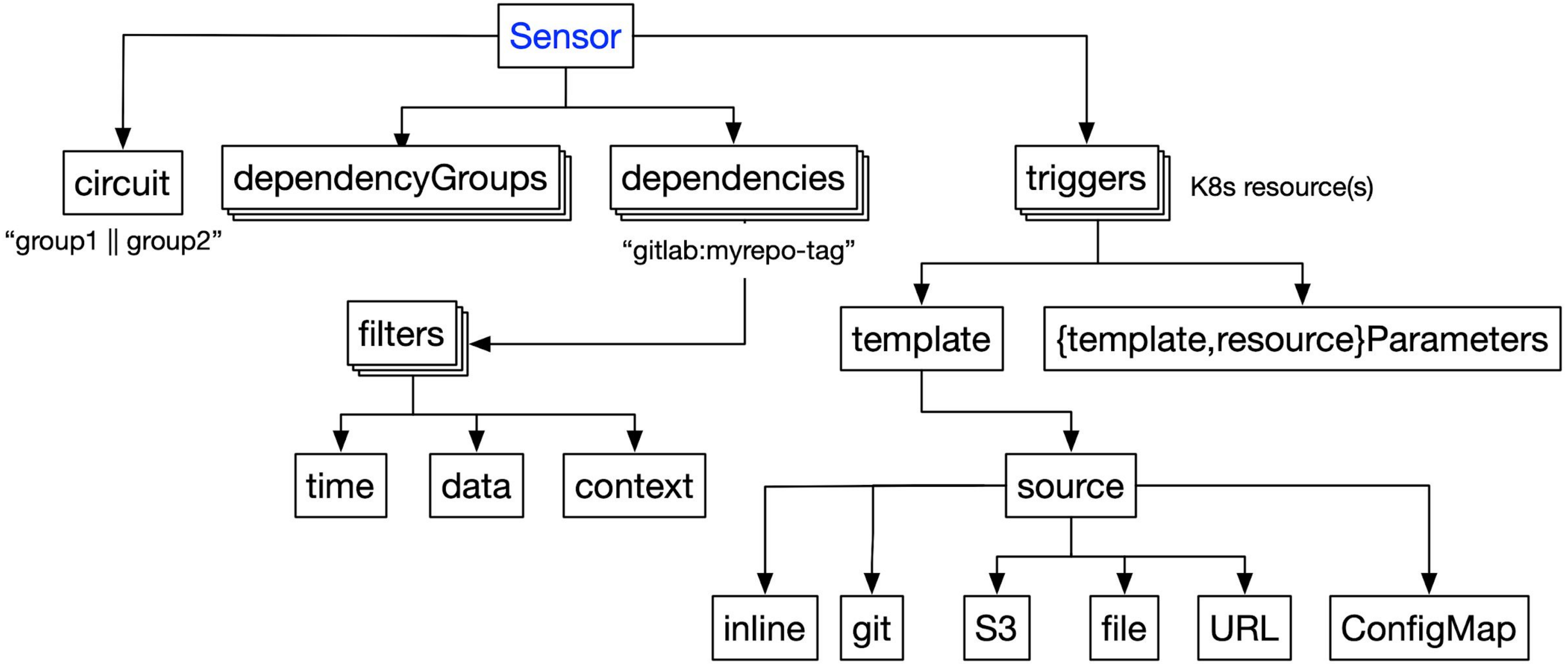


Technical Architecture: Gateway and Sensors

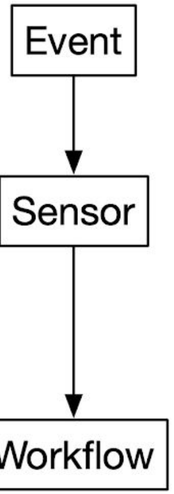


GDG

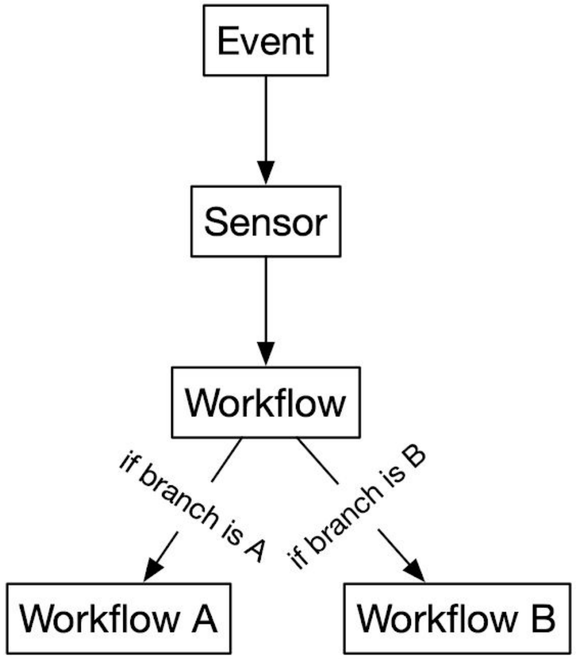
Sensor Spec


Sensor
circuit
"group1 || group2"
dependencyGroups
dependencies
"gitlab:myrepo-tag"
triggers
K8s resource(s)
filters
time
data
context
template
{template,resource}Parameters
source
inline
git
S3
file
URL
ConfigMap

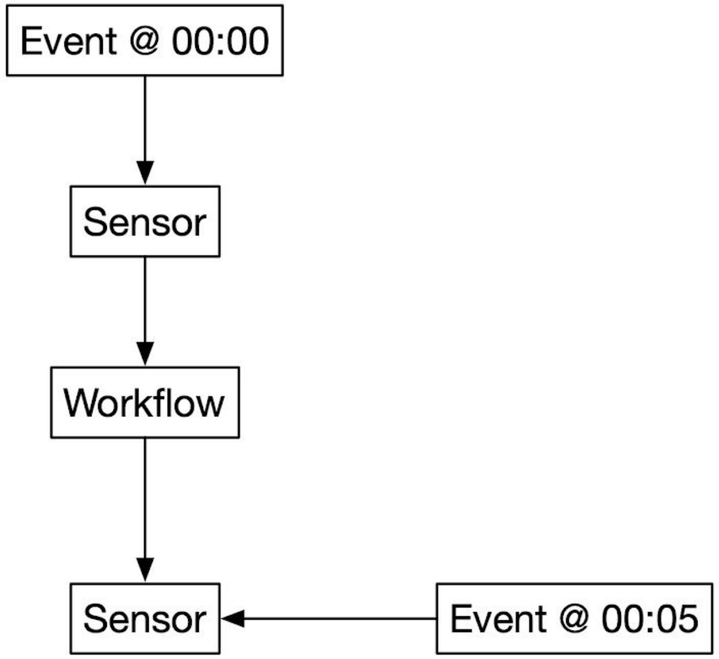
Sensor Designs



logic contained within single Workflow,
CI pipeline runs in single Workflow



logic contained within single Workflow,
parent Workflow programmatically generates children Workflows

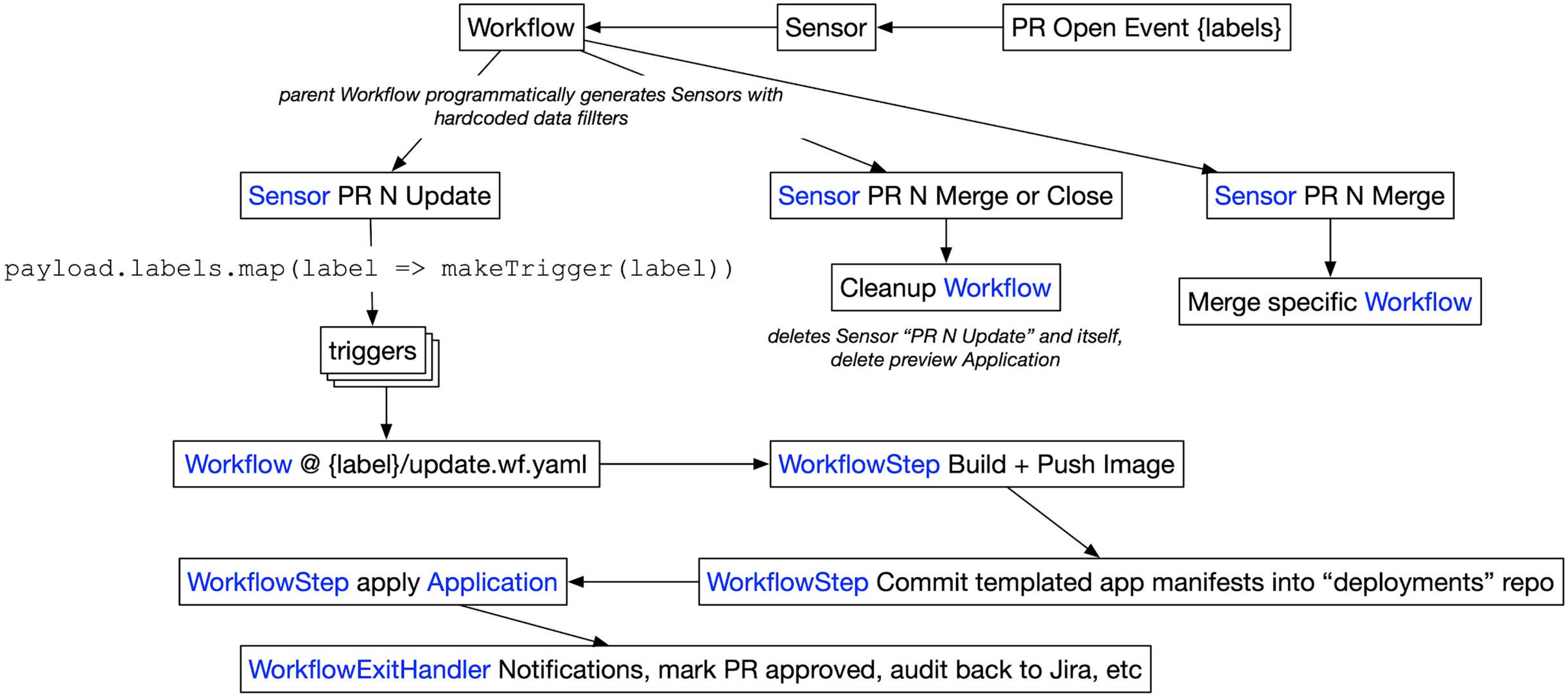


logic contained within filters of Sensor

```
spec:
  dependencies:
  - filters:
      data:
      - path: object_attributes.iid
        type: number
        value:
        - "31"
      - path: object_attributes.action
        type: string
        value:
        - update
      name: merge-request-update
    name: gitlab-gateway:bx-merge-request
```

BioBox Monorepo CI/CD

- Each merge request is annotated with labels specifying which services to deploy in a monorepo
- Developer can test one or more altered services in the context of the entire stack (the rest deployed from whichever was latest release)



Results

- CI logic can be written in any language that developers are comfortable with, breaking down divisional roles between Dev and Ops
- Kubernetes CI Workflows can be labeled/annotated with repo/branch/tag etc (templateParameters)
- GitOps for CD via ArgoCD enables visibility for QA/PO as well as robustness for Ops
- Flexibility - receive a webhook (or event!), code process payload, pick Sensor design that fits task
- Reuse of observability stack for metrics and logging on CI workflows
- CI workflows autoscaling via K8s resource requests, scheduling via tolerations and node taints
- Consistent tooling - developers can get familiar with K8s through CI/CD, same K8s for primary app
- Arbitrary notifications (slack, PR comments, email) written as Argo Workflow steps
- CI workflows can be manually triggered via kubectl/argo CLI, or by Argo Events (e.g. GCR PubSub)

Future Objectives

- Improve multi-event multi-sensor Workflow visibility
- Argo Workflows still in YAML… working on K8s TypeScript client implementation
- Argo Workflows 2.4 release will bring “Template CRD” - reuse workflow steps across Workflows!
- Argo Events has support for NATS instead of HTTP streaming for Gateway
  - Kafka is a supported event source, but have to use NATS for Gateway-Sensor streaming?
  - Event replayability, long term storage, audit logging
- Special pipelines for PRs which are WIP - bring up web IDE, run apps in debug mode

Questions & demo!

Thank you!

- Intuit + Blackrock
- Argo slack
- Devfest organizers!

DEVFEST
2019