End-to-end testing in complex GitOps environments

27-06-2022 - Etienne Tremel

Agenda

1. Who am I
2. Journey to continuous deployment
3. GitOps
4. Promoting images across environments
5. Image promotion the GitOps way
6. Testing
7. Simplicity to complexity
8. End-to-end testing
9. Demo

Who am I

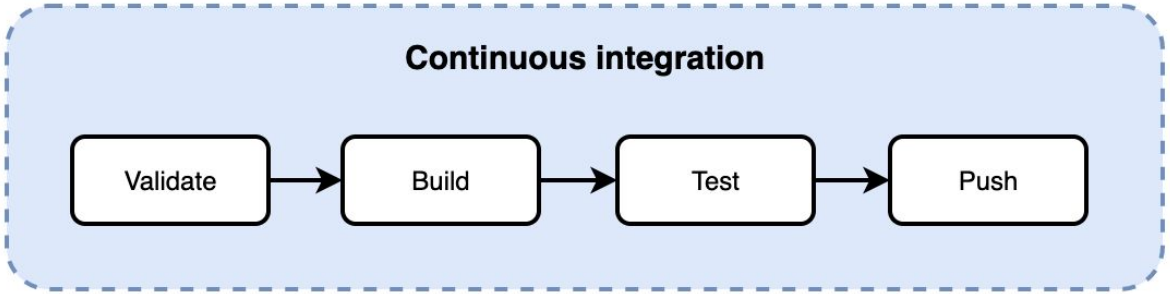
- Cloud engineer?
- Currently contracting for the group Ahold Delhaize (Albert Heijn)
- Part of the collective FikaWorks
- Cloud Pirate
- Love open source

Disclaimer

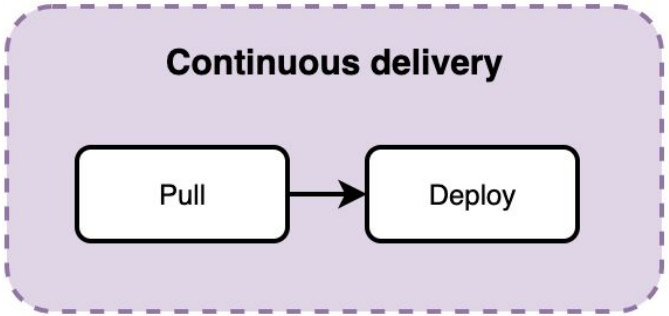
Terms used during this presentation can be misused.

Journey to continuous deployment

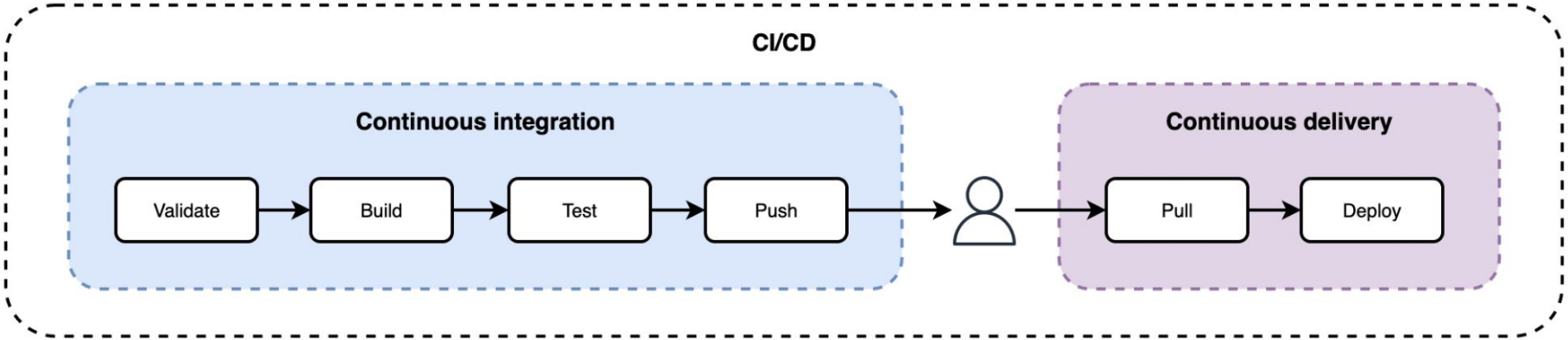
Continuous Integration



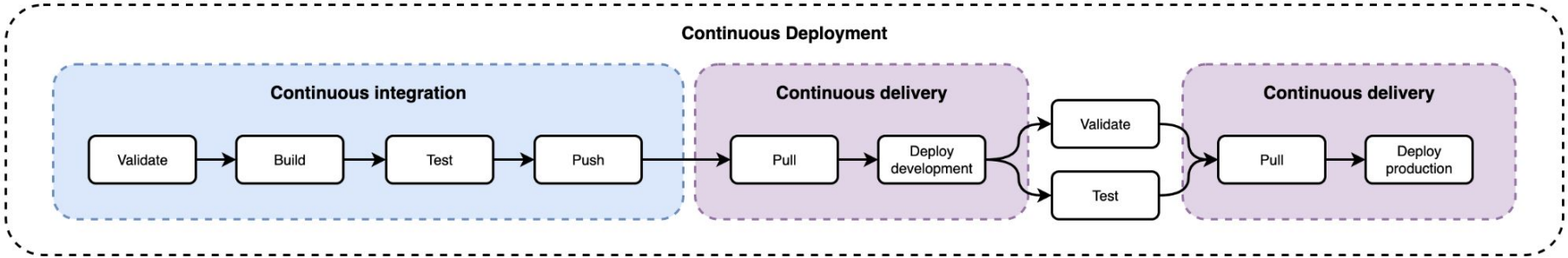
Continuous Delivery



CI/CD


CI/CD
Continuous integration
Validate
Build
Test
Push
Continuous delivery
Pull
Deploy

Continuous deployment



Typical journey of application delivery

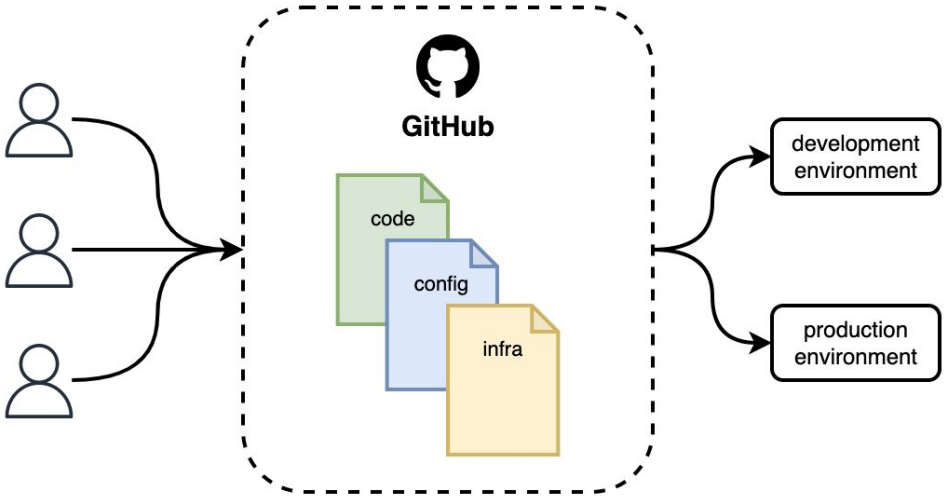
- Peer review
- Lint
- Source code analysis
- Security scanning
- Code quality
- Build
- Publish
- Tests
  - Unit tests
  - Integration tests
  - Performance tests

GitOps

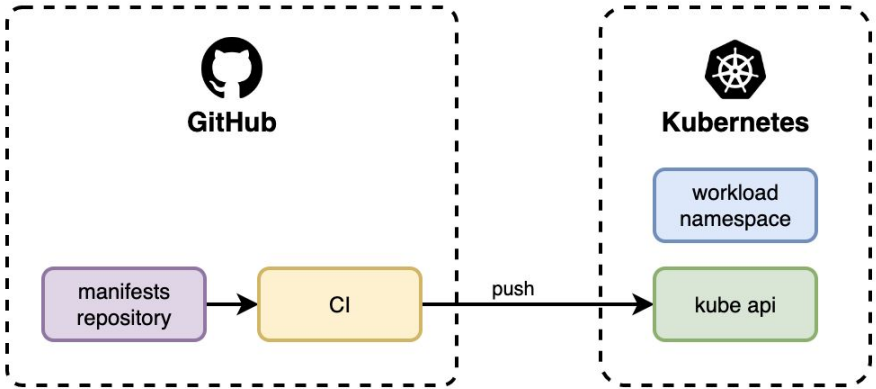
A way of implementing Continuous Deployment for cloud native applications focusing on developer-centric experience.

Source: https://www.gitops.tech

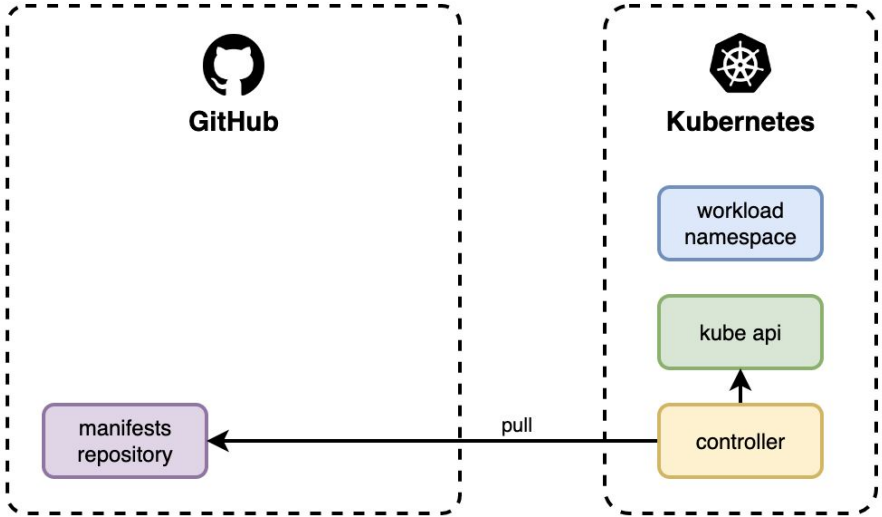
Infrastructure as code



Traditional push model



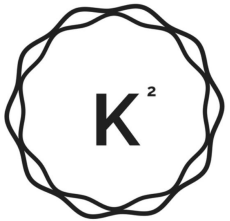
Pull model


GitHub
Kubernetes
workload namespace
kube api
controller
pull
manifests repository

Controllers

Active projects

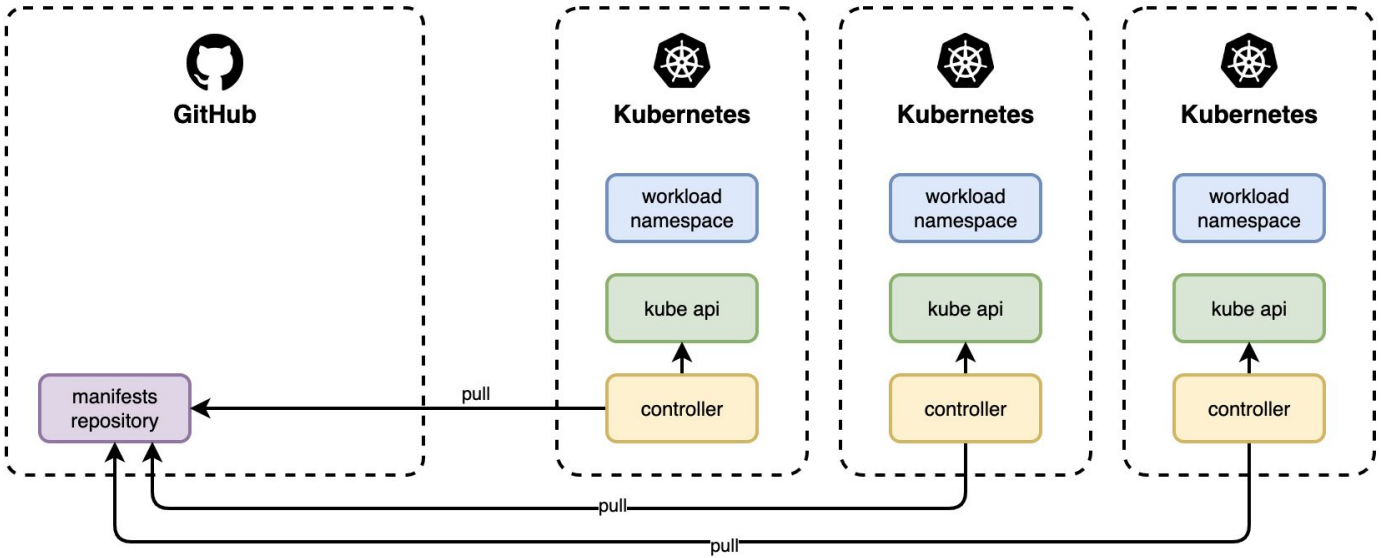
- FluxCD *(CNCF incubation)*
- ArgoCD *(CNCF incubation)*
- Keel
- Werf



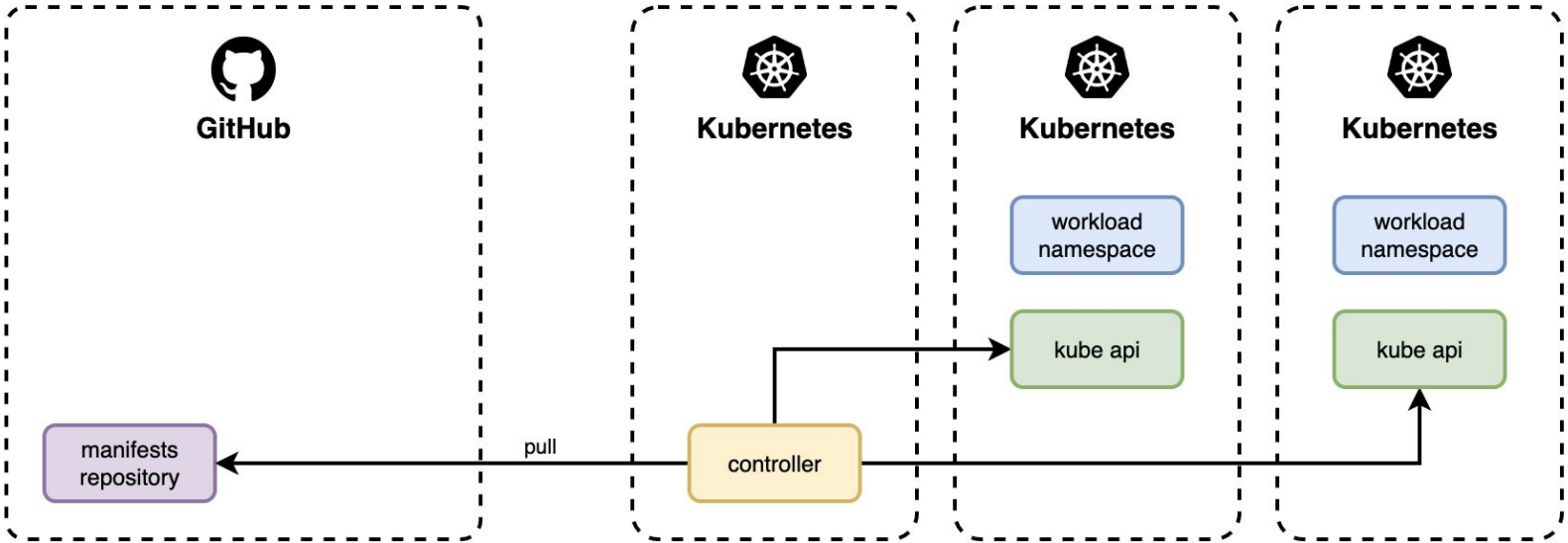
Abandoned projects

- Faros
- GitKube

Pull model per cluster



Pull model from a management cluster



GitOps principles

Declarative

A system managed by GitOps must have its desired state expressed declaratively.

Versioned and Immutable

Desired state is stored in a way that enforces immutability, versioning and retains a complete version history.

Pulled Automatically

Software agents automatically pull the desired state declarations from the source.

Continuously Reconciled

Software agents continuously observe actual system state and attempt to apply the desired state.

Source: https://opengitops.dev/

Promoting images across environments

Promotion by retagging images

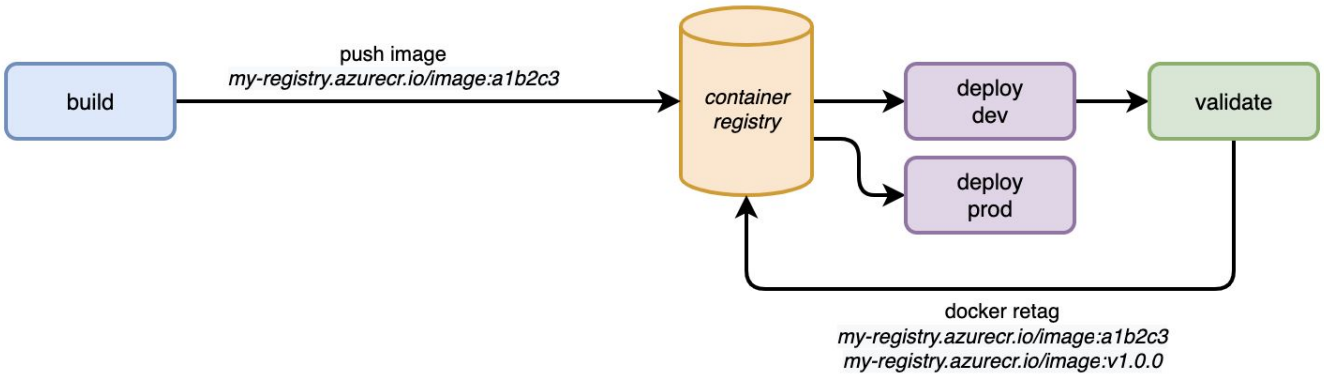


Image tagging conventions

<image>:<commit sha> — initial image

backend:2fd82529590a02f1d06ca619aee64b359f6ea7b5

<image>:<semver> — tag match release tag

backend:v1.0.0

<image>:<environment> — tag match target environment

- backend:dev
- backend:prod

<image>:<custom> — custom convention

- backend:1.0.0-alpha.0 — could match development environment
- backend:1.0.0-beta.0 — could match acceptance environment
- backend:1.0.0 — could match production environment

Image promotion based on multiple container registries

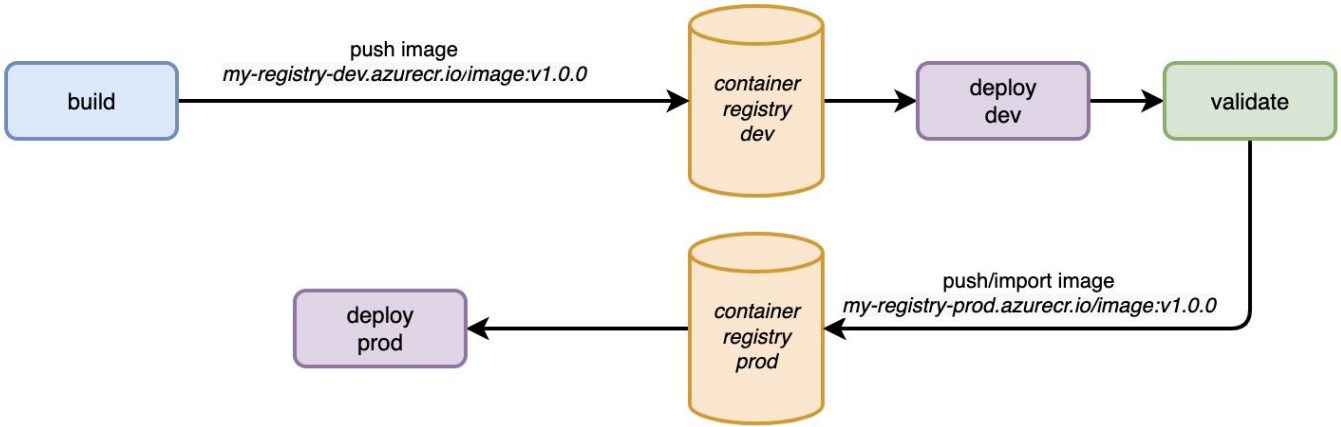
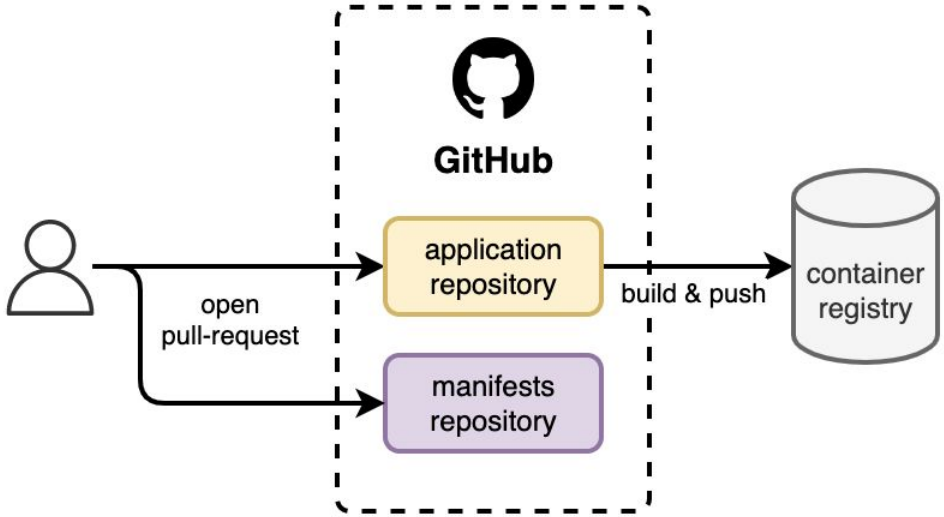


Image promotion the GitOps way

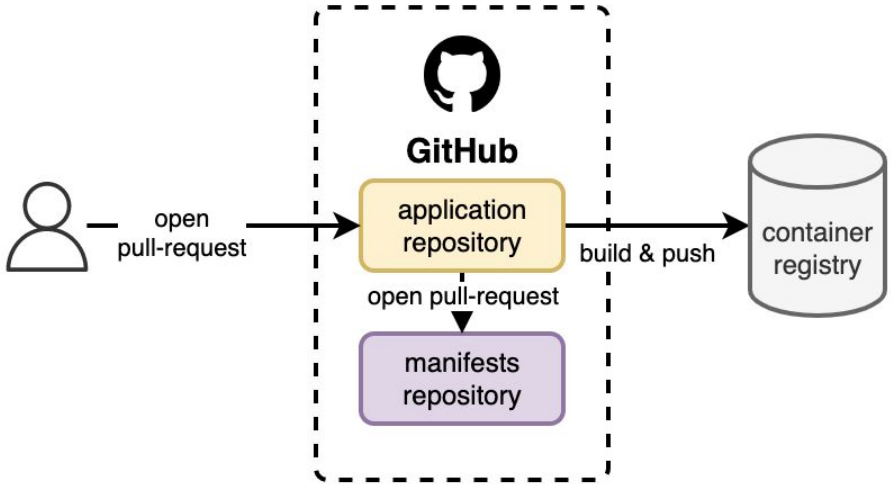
Update image version manually

Someone has to manually update the manifests repository with the correct version of the image.



Update image version automatically from the CI

In the application repository CI, a step update the manifests repository via pull-requests / or directly to main branch.

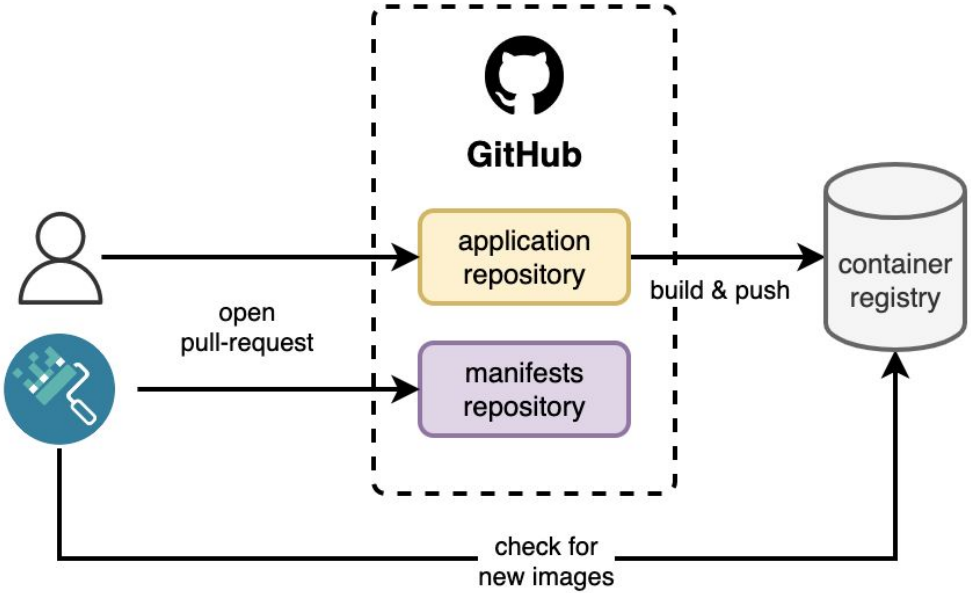


Update image version automatically with controllers

A controller detect new images pushed to the registry and update the manifests repository (via pull-request or directly to main branch).

Example of controllers that can do that:

- RenovateBot
- ArgoCD image updater
- Flux image-reflector-controller and image-automation-controller

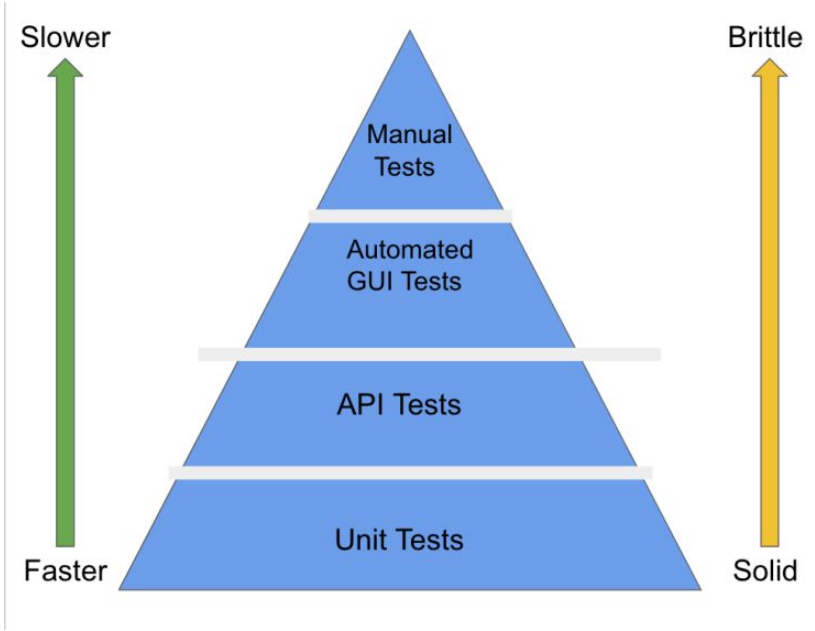


Testing phase for continuous deployment

Test pyramid

Make the process of testing faster, efficient and cost-effective.

- Maximum functionality tested by unit tests
- More functionality tested by less brittle tests like unit and API tests
- Most testing should be automated



Source: https://automationstepbystep.com/2020/05/02/what-is-a-test-pyramid/

Type of tests

- **Unit tests — *small piece of code / at the function level***
- Integration tests —*between services*
- UI tests — *user interface*
- **End-to-end tests — *from beginning to end***
- Performance tests — *ui performance*
- Load tests — *behavior under high traffic*
- Fuzzy — *random input*
- Security tests — *security*
- Smoke tests — *tests determined by business for newly developed feature*
- Regression testing — *make sure that newly introduced features doesn't break previous tests*
- …

From clear to complex

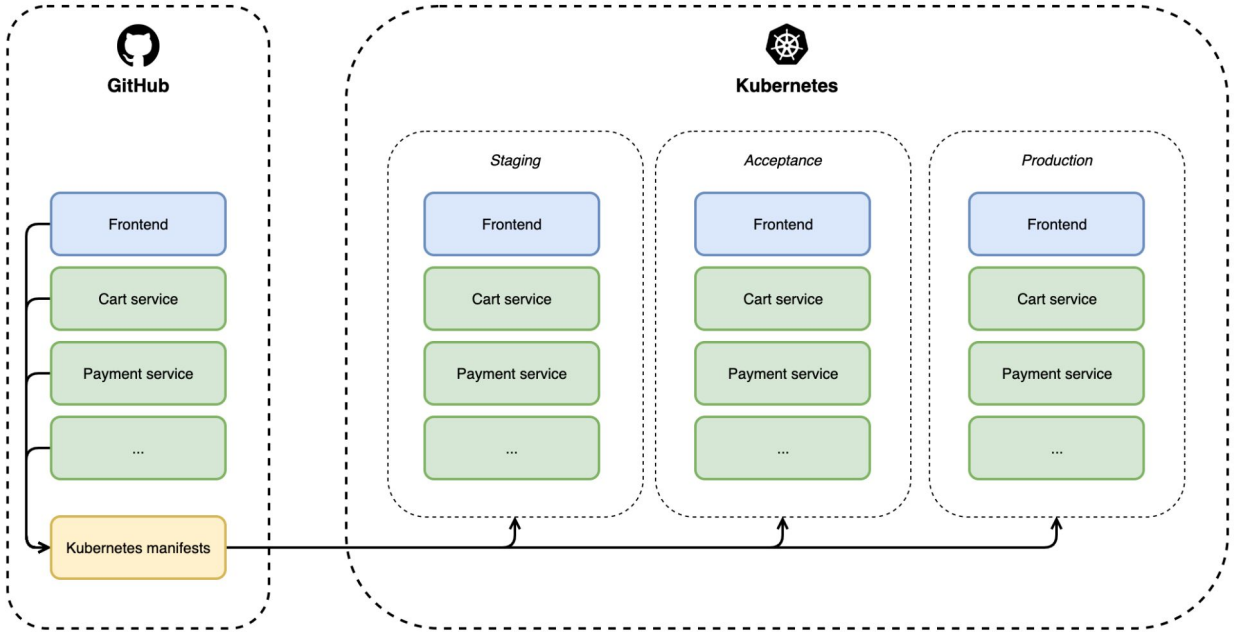
Cynefin model

It helps managers to identify how they perceive situations and make sense of their own and other people's behaviour.

- clear/obvious = safe and battle tested industry standard
- complicated = expert opinionated on the implementation
- complex = no expert, no guideline, few people with knowledges
- chaotic = unknown territory, no expert, no guidelines, no knowledges

Source: https://itrevolution.com/cynefin-four-frameworks-of-portfolio-management/

Single-tenant



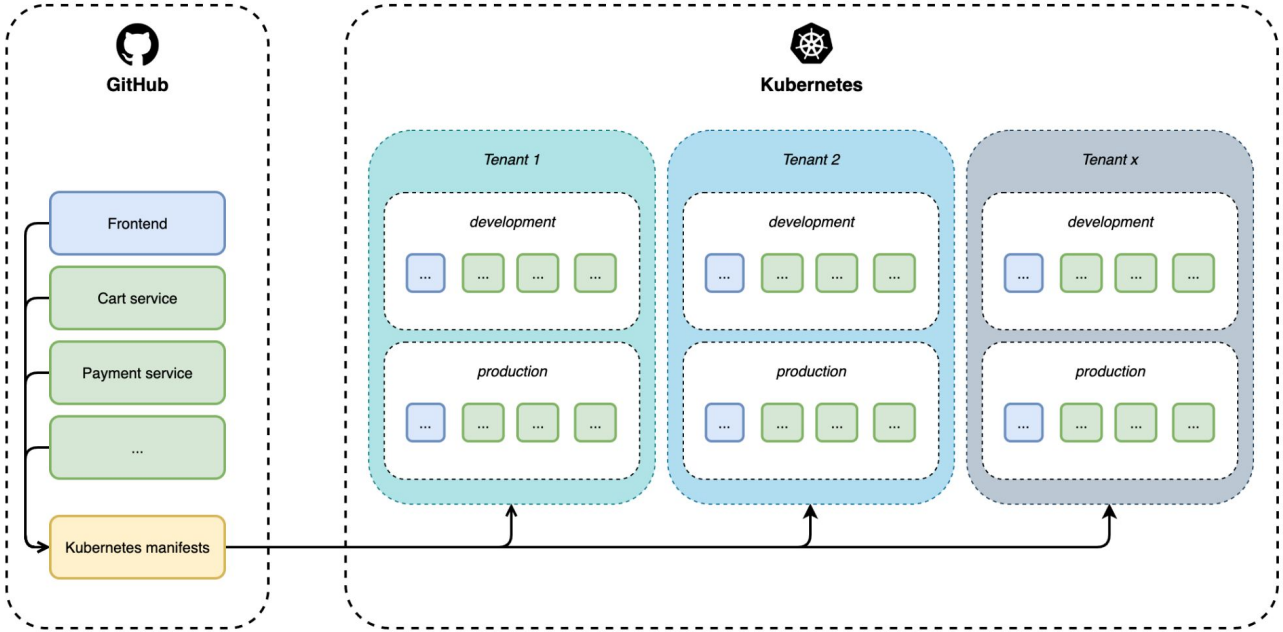
Single-tenant vs Multi-tenant

Single-tenant cloud architecture is one where a single software instance and its supporting infrastructure/database serve only one customer. In a single-tenant environment, all customer data and interactions are separate from every other customer. Customer data is not housed in the same database and there's no sharing of data in any way.

A multi-tenant architecture is one where a single software instance and database serves multiple customers (i.e. tenants).

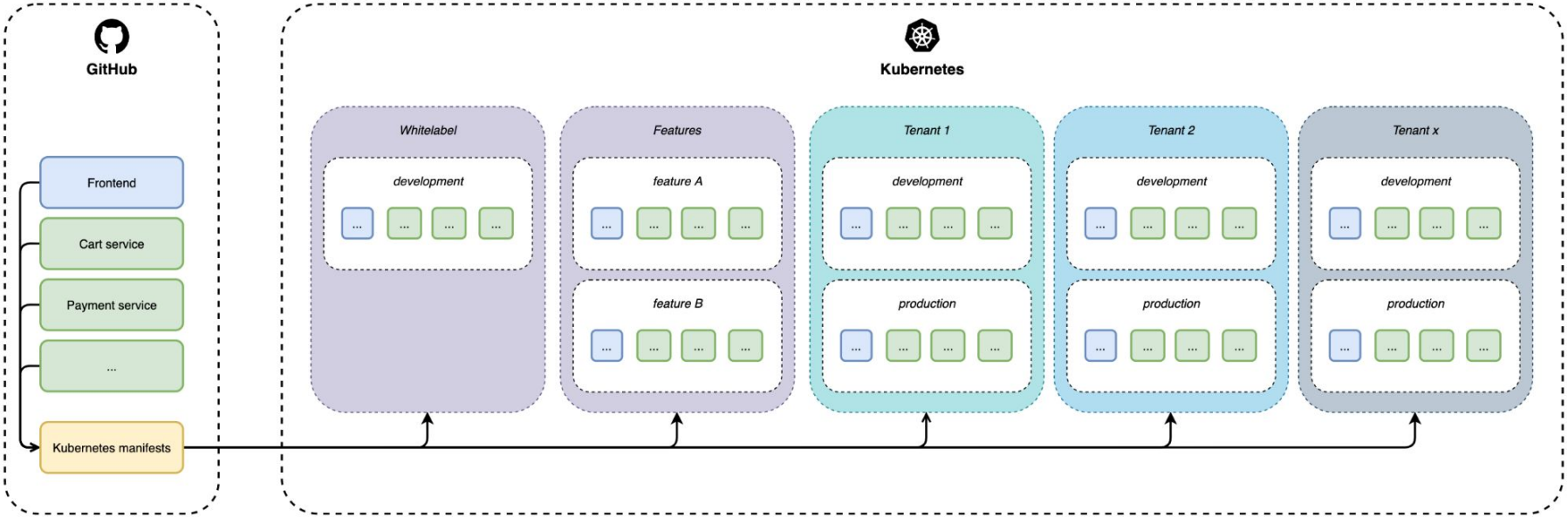
Source: https://www.cloudzero.com/blog/single-tenant-vs-multi-tenant

Multi-tenant

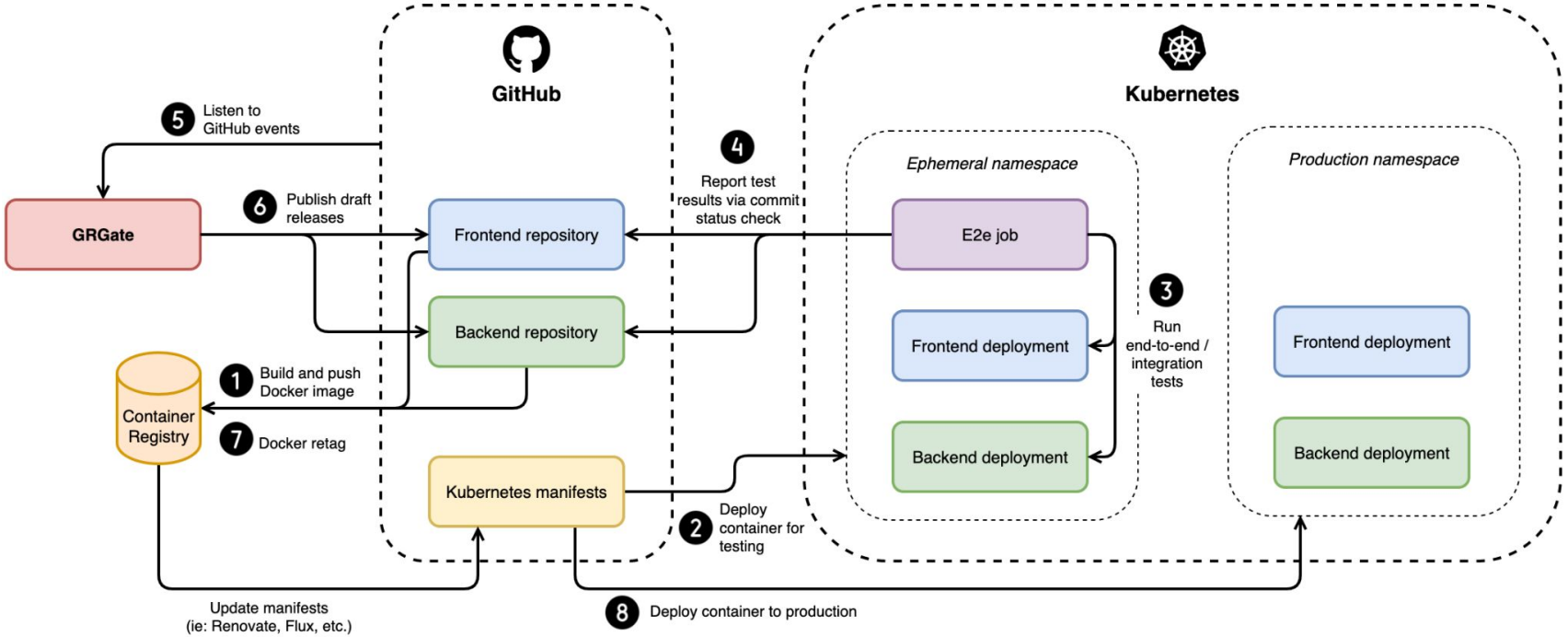


Complex environment

Multiple clusters
Multiple teams
Multiple configurations

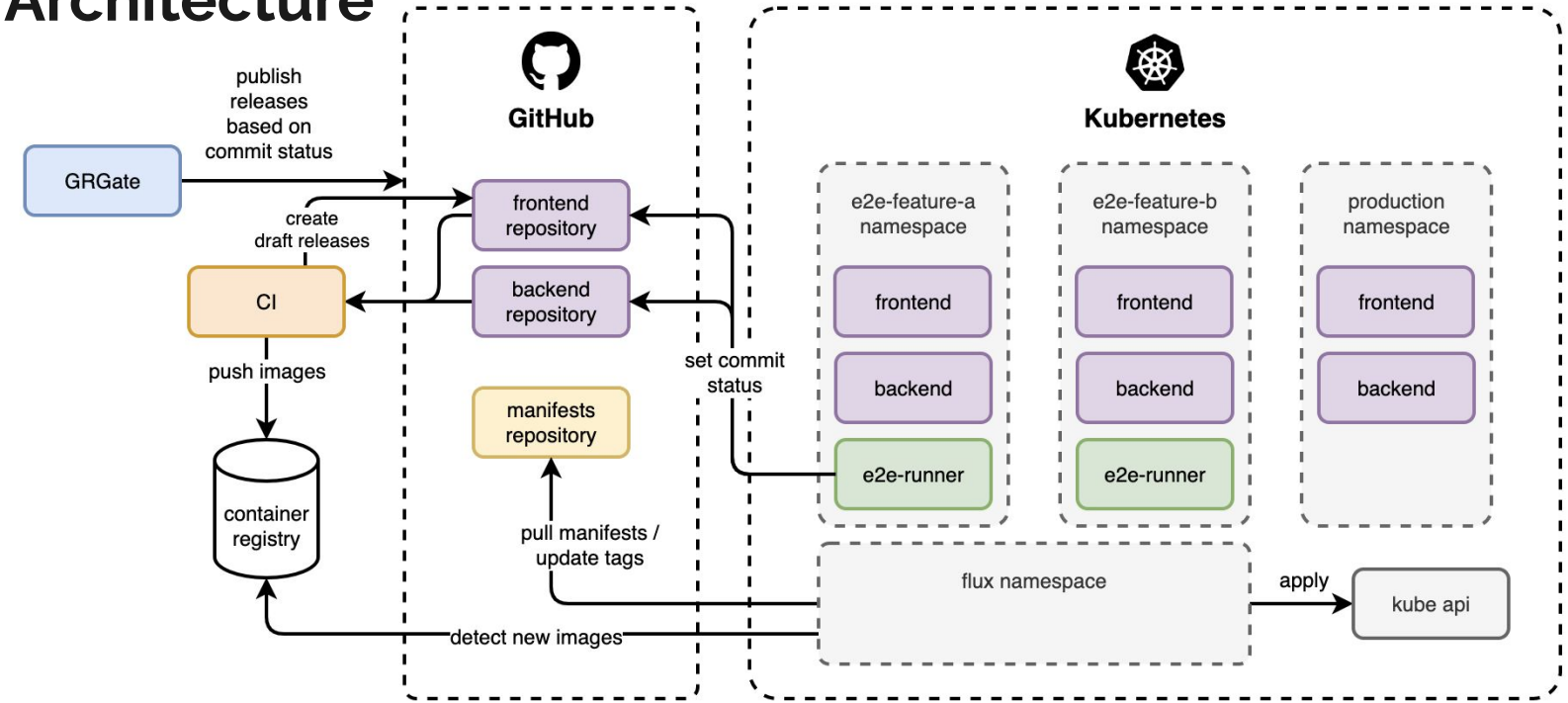


Example of continuous deployment implementation for complex environments



Demo

Architecture



Questions?

Useful links

- FluxCD: https://fluxcd.io
- Git release gate project: https://grgate.dev
- GRGate demo repositories: https://github.com/grgate

Thank you!