Continuous Delivery Meets Custom Kubernetes Controller

A Declarative Configuration Approach to CI/CD

By:

- Simon Cochrane
- Suneeta Mall

CI/CD on Kubernetes

Kubernetes specifically states that it

"Does not deploy source code and does not build your application. Continuous Integration, Delivery, and Deployment (CI/CD) workflows are determined by organization cultures and preferences as well as technical requirements."

nearmap

Configuration files

- Recommended not to use the *:latest* tag

```
---
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-app
      image: nearmap/my-app:latest
      ports:
        - containerPort: 80
```

nearmap

Configuration files

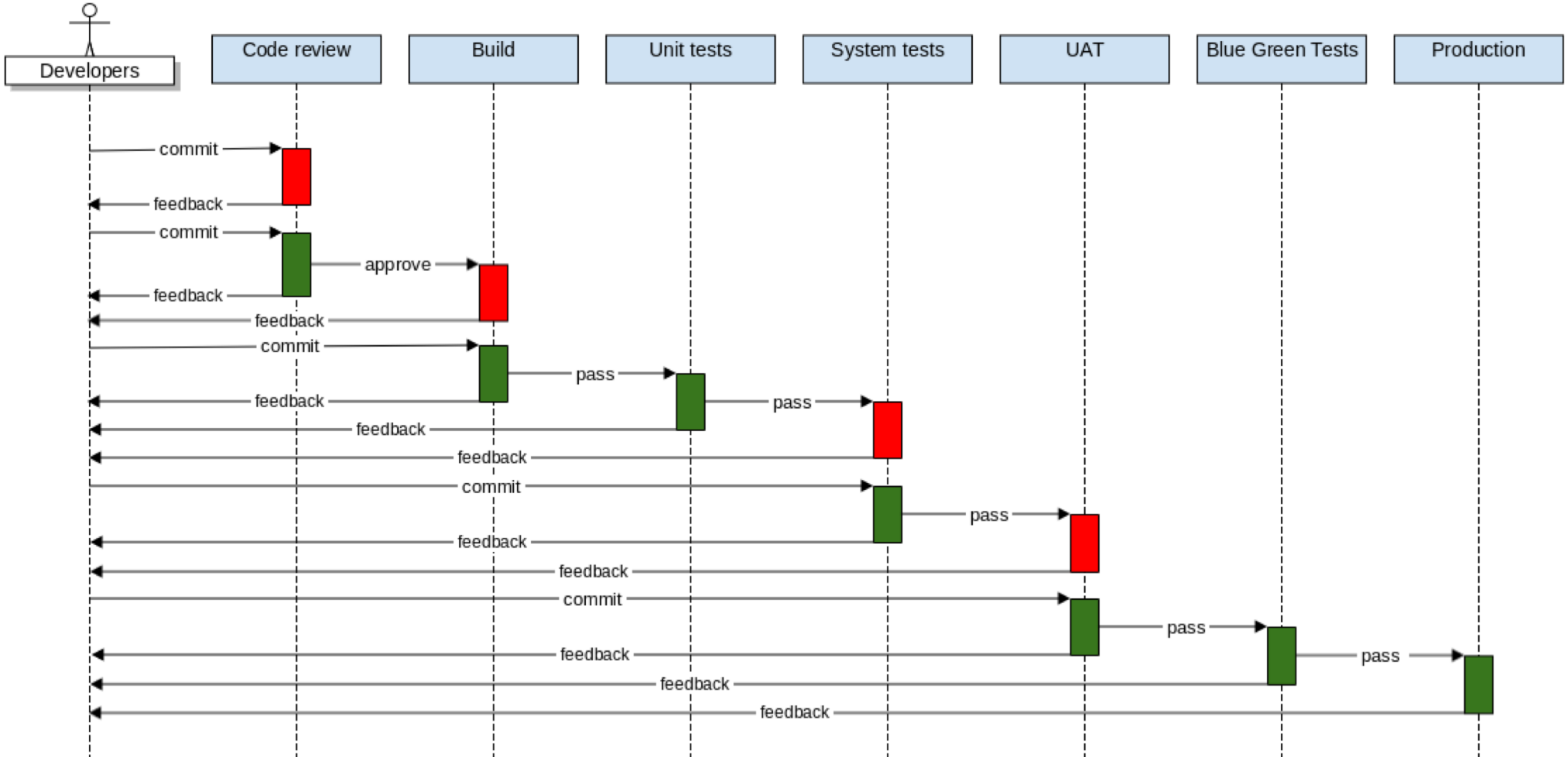
- Specify a version number (digest or git hash)
- Should be in source control
- How to manage multiple environments?

```
---
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-app
      image: nearmap/my-app:1873b440fd288d51c6fc56cc727bc658e9312d50
      ports:
        - containerPort: 80
```

nearmap

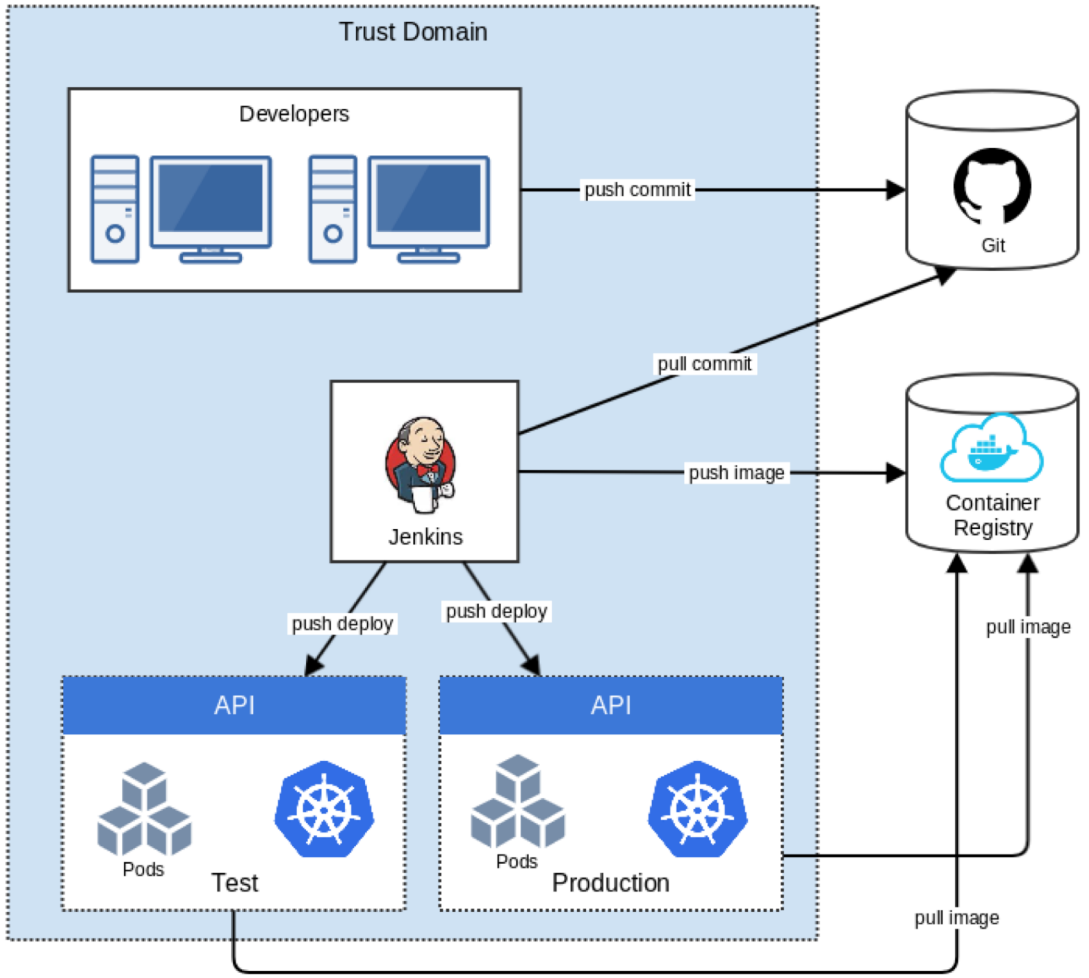
Continuous Delivery

Set of workflows and validations that provide a reliable process for releasing software.



Self-hosted CD

e.g. Jenkins, TeamCity

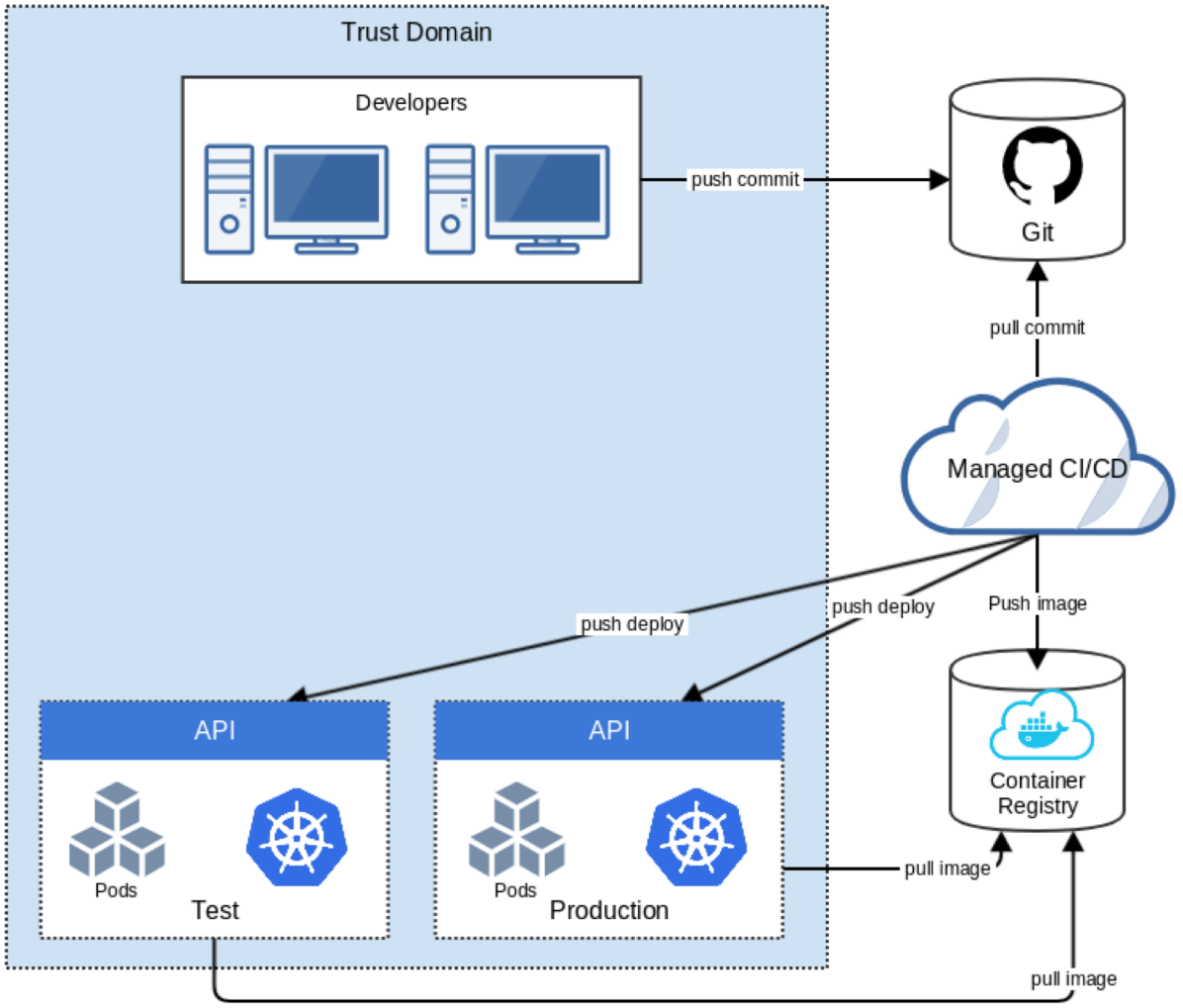


nearmap

Managed CD

E.g. Circle CI, Shippable,

AWS CodePipeline



nearmap

Existing Solutions?

nearmap

Configuration

- Git is the source of truth for application configuration
- Lots of version changes can be noisy
  - Especially when rest of the configuration rarely changes
- Should application releases be treated differently to configuration?
  - Continuous Deployments should be common events
  - Configuration changes are comparatively rare
  - Still need to deal with history and roll back

nearmap

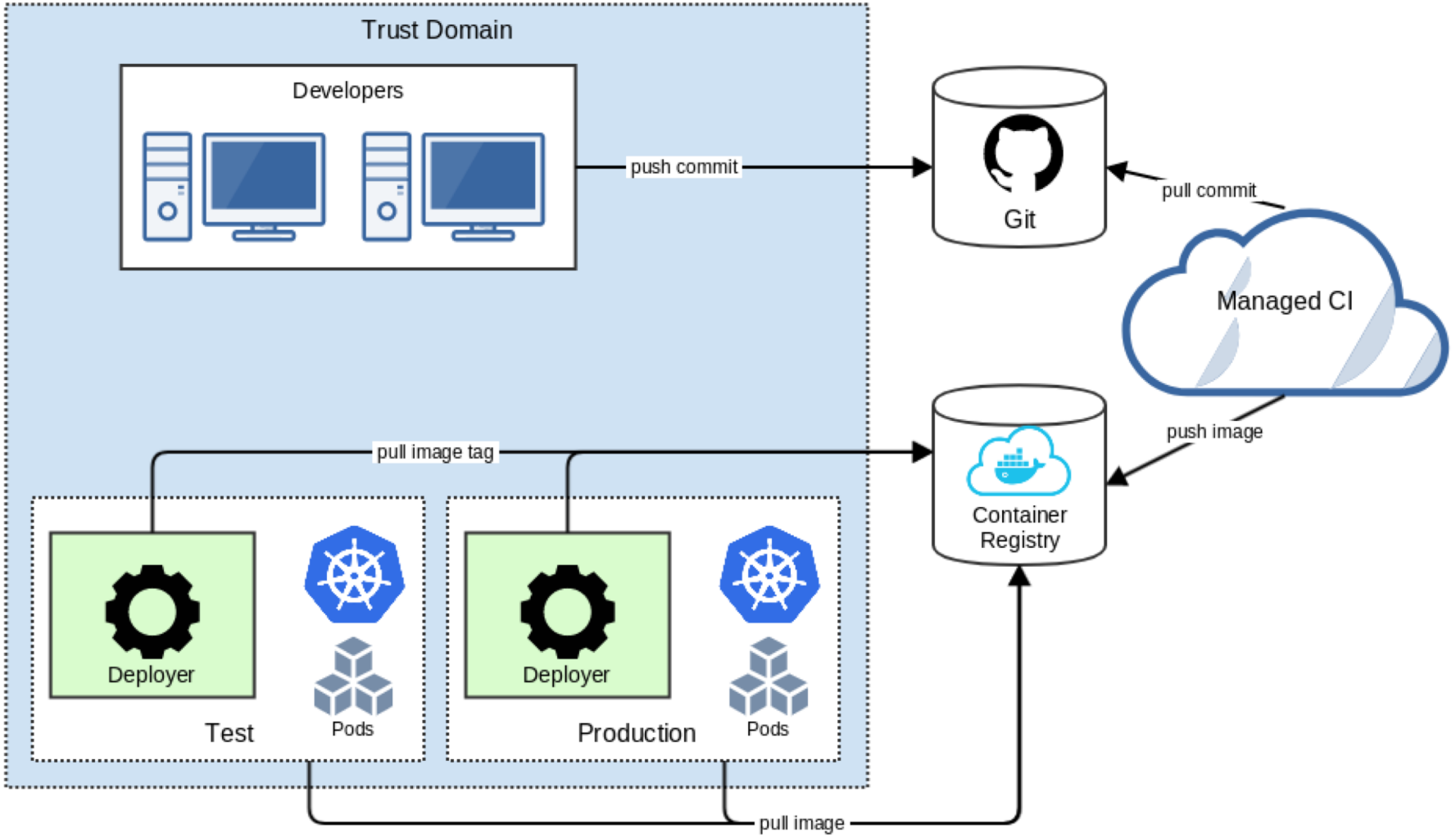
CD-lite

A simplified approach to Continuous Delivery

- BYO continuous integration tool
- Minimise infrastructure setup and management costs
- Build on existing Kubernetes concepts
- Support manual steps and full automation
- Support best practices
  - Blue-green deployments
  - Version history and rollback
  - Secure environments
  - Instrumentation/visibility

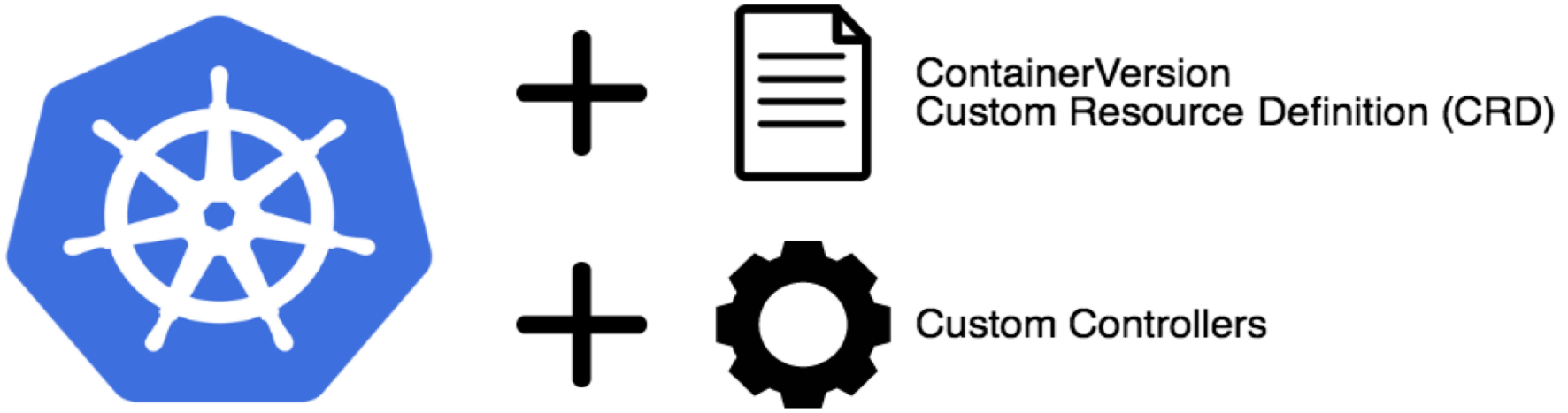
nearmap

Simpler Pipeline



CD-lite: Container Version Manager

Solve these challenges by using intrinsic kubernetes principles and native abstractions



nearmap

Advantages

- Clusters and tools don't need access to additional resources
  - e.g. don't need git read or write access
- Doesn't require a separate config repo and config update process
- Simplified configuration
  - Can exist alongside the application code
  - Avoid configuration per-environment
  - Reduces noise in configuration version history

nearmap

Benefits

- Low management and maintenance requirements
  - Onus of CD pipeline maintenance is on Kubernetes
- Separation of concerns
  - Cluster is responsible for managing its own versions
- Eliminates security risk by centralizing communication to CR
- Native solution
- Better visibility into CI/CD
  - Current release
  - Release history
- Monitoring CI/CD:
  - Dashboard and metrics

nearmap

Demo

Thank you!