

# Cloud Native Development Without the Toil: *Key Practices and Tooling*

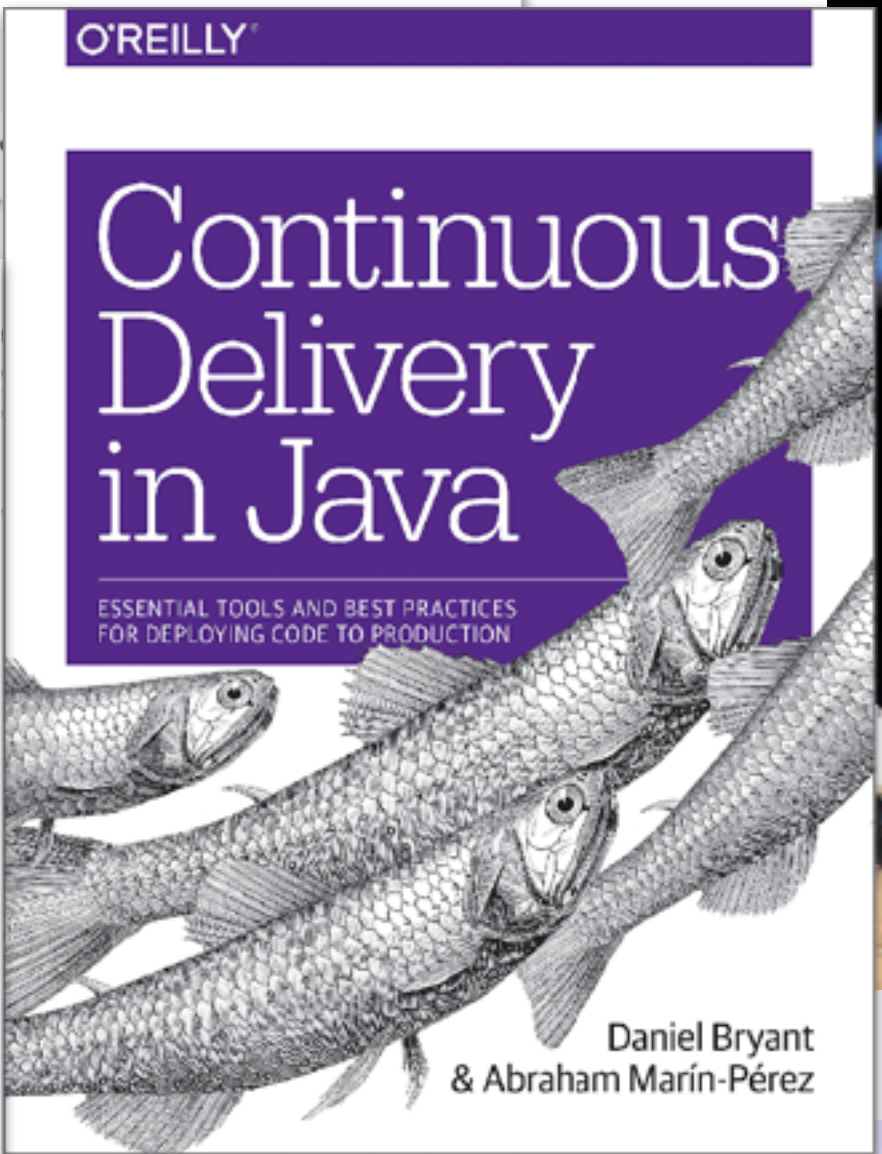
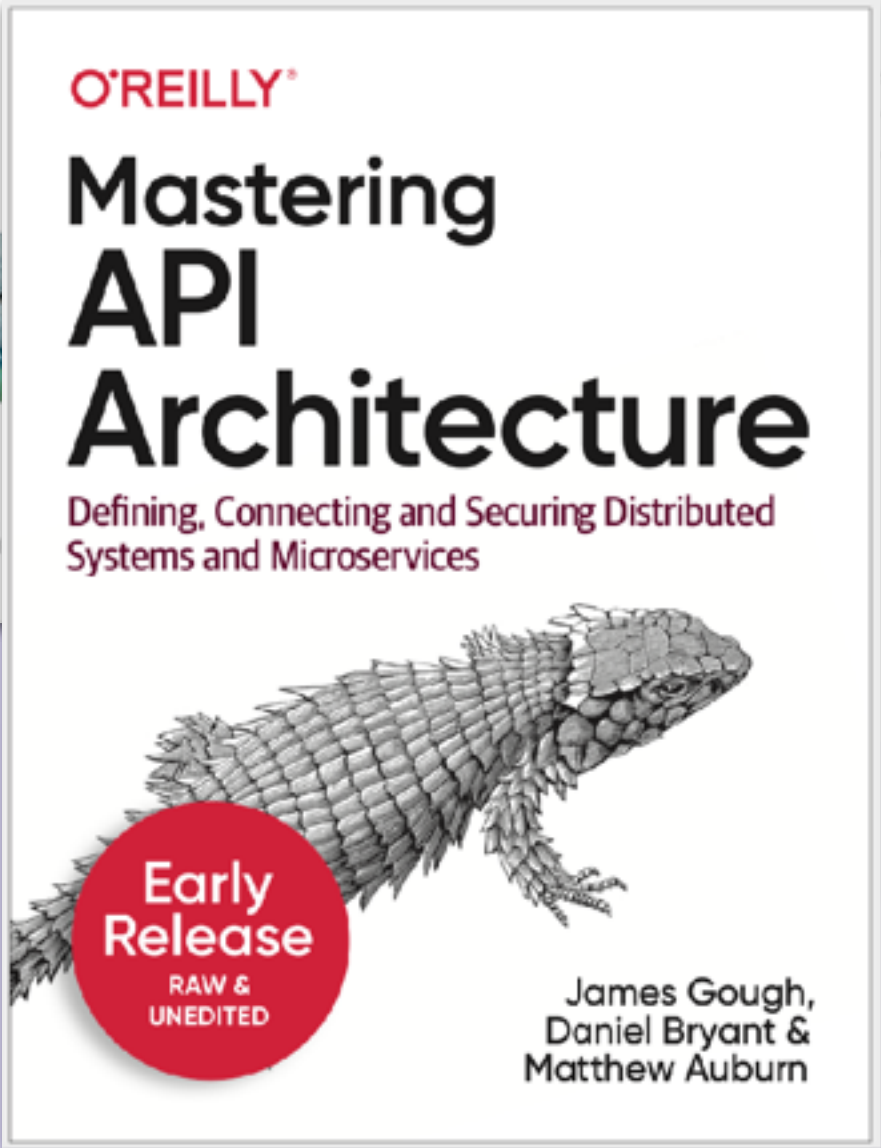
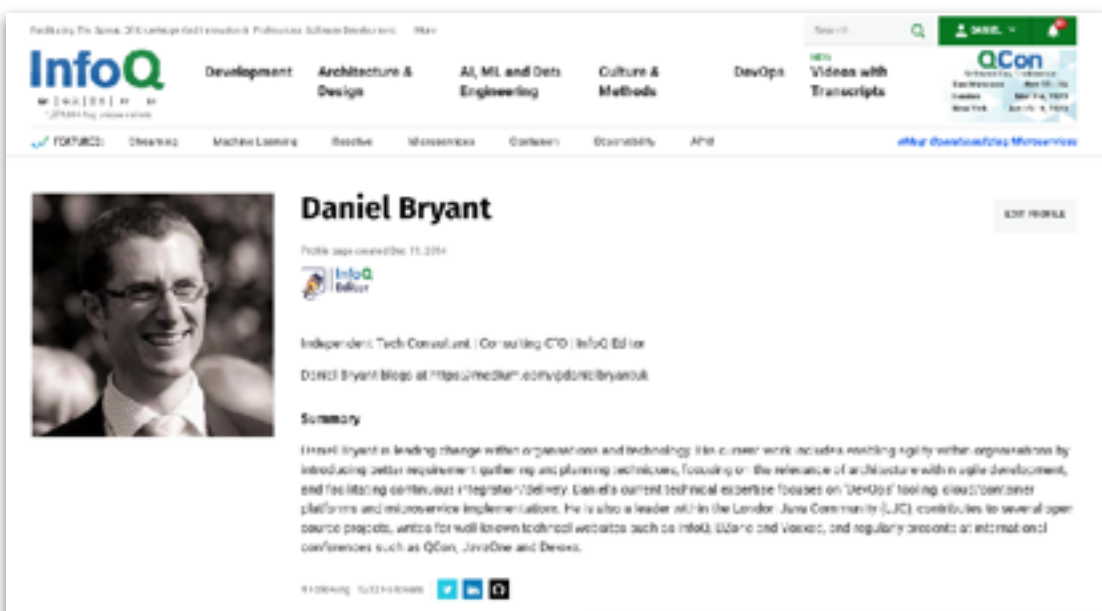
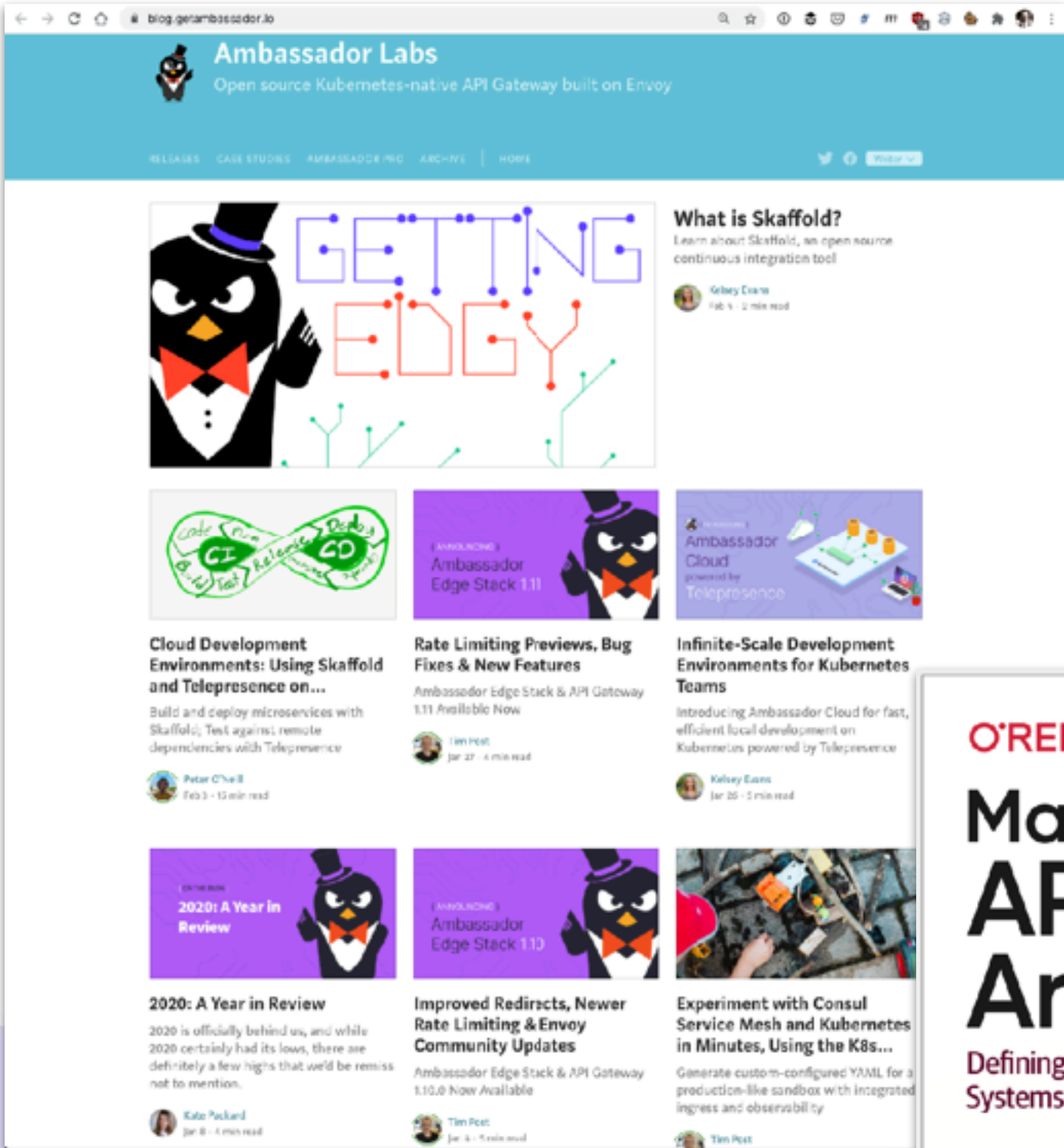
**A▶BASSADOR**

Daniel Bryant

# tl;dr

- Continuous delivery requires safety and speed throughout the engineering workflow
- Bringing old mental models and tools can increase toil when building, testing, and running cloud native services
- Artifact syncing, dev environment bridging, and GitOps increase both safety and speed

@danielbryantuk



# Why we're all here: Happy users



<https://dilbert.com/strips/2014-02-25>

# Continuous delivery of value

“Continuous delivery is achieved when stability and speed can satisfy business demand.

Discontinuous delivery occurs when stability and speed are insufficient.”

- Steve Smith ([@SteveSmith\\_Tech](https://twitter.com/SteveSmith_Tech))

# Continuous delivery of value

“Continuous delivery is achieved when **stability** and **speed** can satisfy business demand.

Discontinuous delivery occurs when stability and speed are insufficient.”

- Steve Smith ([@SteveSmith\\_Tech](https://twitter.com/SteveSmith_Tech))

# Continuous delivery of value

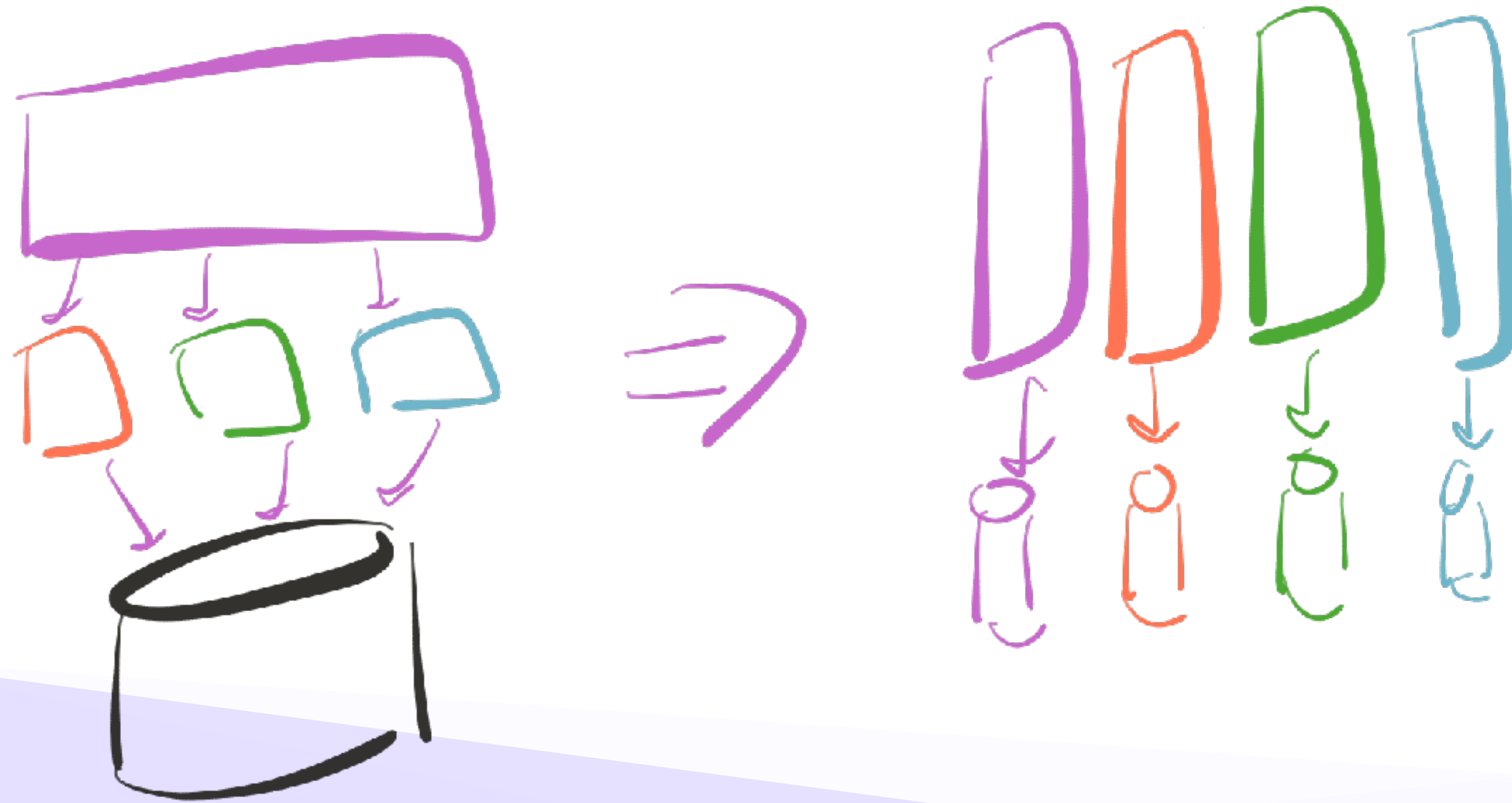
“Continuous delivery is achieved when **stability** and **speed** can **satisfy [your] business demand**.

Discontinuous delivery occurs when stability and speed are insufficient.”

- Steve Smith ([@SteveSmith\\_Tech](https://twitter.com/SteveSmith_Tech))

# Two modern software trends for speed and safety

# Trend #1: Microservices



- Modularisation
  - Aka “microservices”
- Independently:
  - Build
  - Release
  - Scale
  - Operate
- Strive for
  - High cohesion
  - Loose coupling

# Trend #2: Kubernetes

- Automated orchestration of container-based services
- Declarative config (with reconciliation loops)
- Common API/platform to support and extend
- But... a non-trivial set of abstractions/components for developers
- And it's not a PaaS



A long-exposure photograph of a city street at night. The image is dominated by horizontal light trails from moving vehicles, creating a sense of motion and speed. In the background, there are buildings with lit windows and storefronts. One storefront on the right has the text "GIOCATOLI" visible. The overall color palette is warm, with orange and yellow tones from the streetlights and building lights.

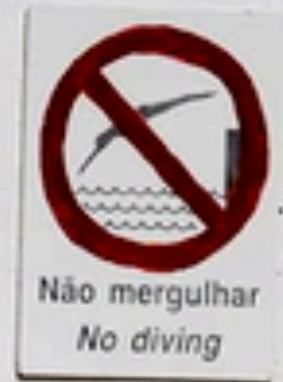
**With cloud native, the focus is often on speed**

# The Kubernetes Migration Journey(s)

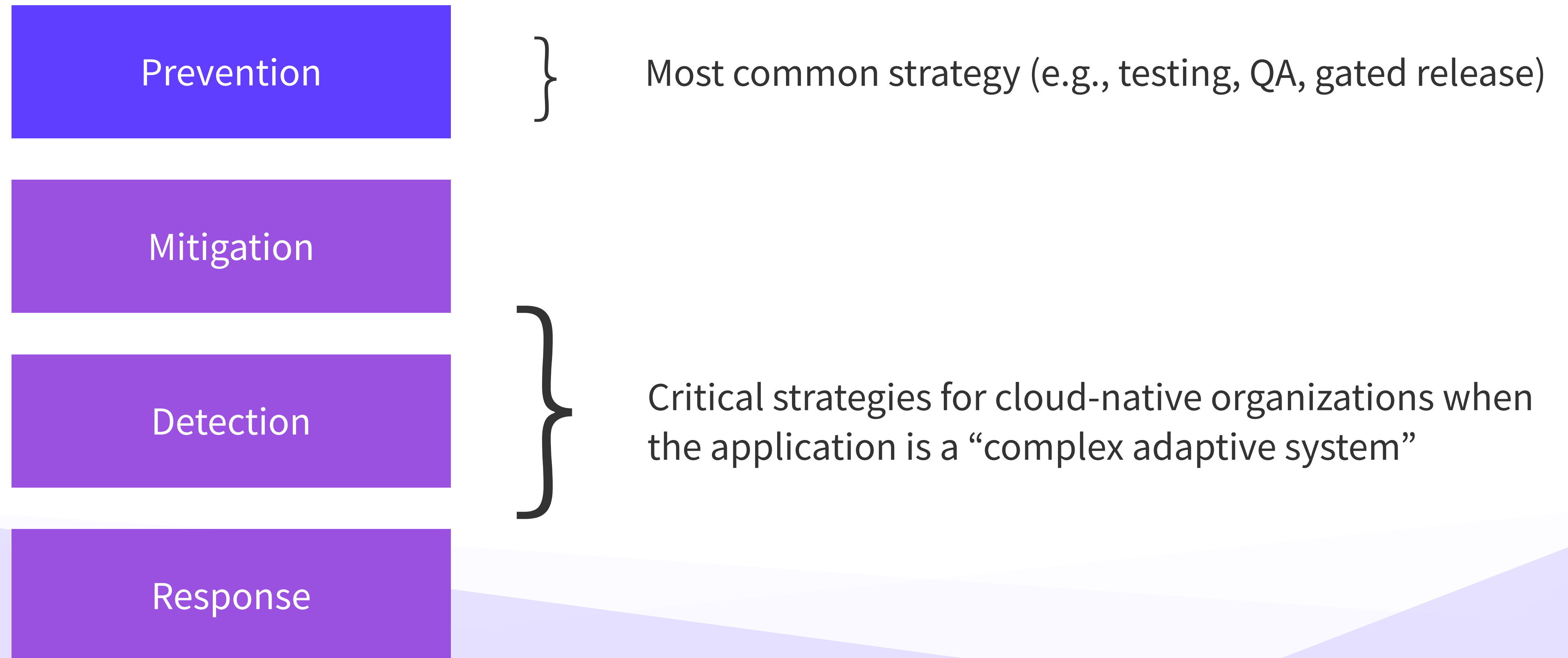


# Safety

The ability to quickly ship  
changes to production,  
***without fear***



# Safety requires multiple strategies



# Safety Strategies in the Cloud

## Prevention

- Testing: integration tests, user acceptance tests, ...
- High-fidelity replica environments for dev & test

## Mitigation

- Engineering for resilience
- Progressive delivery: canary release, blue/green rollouts, ...

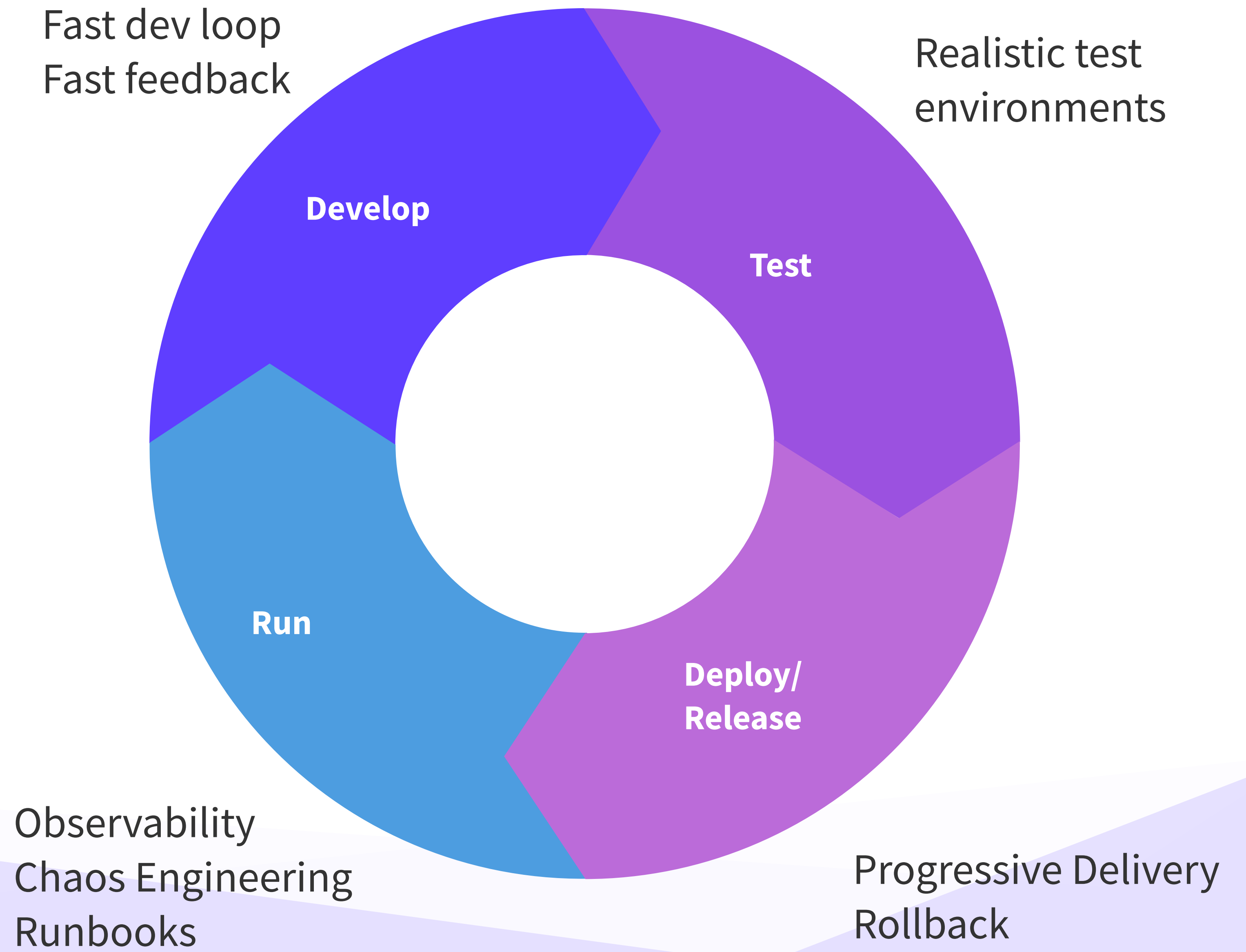
## Detection

- Observability
- Chaos engineering

## Response

- Instant rollback & traceability
- Runbooks, fast dev / test loops
- Blameless postmortems
- Game days

# Safety needs to be part of your workflow.



 Gitpod


Fast dev loop  
Fast feedback



  
S K A F F O L D

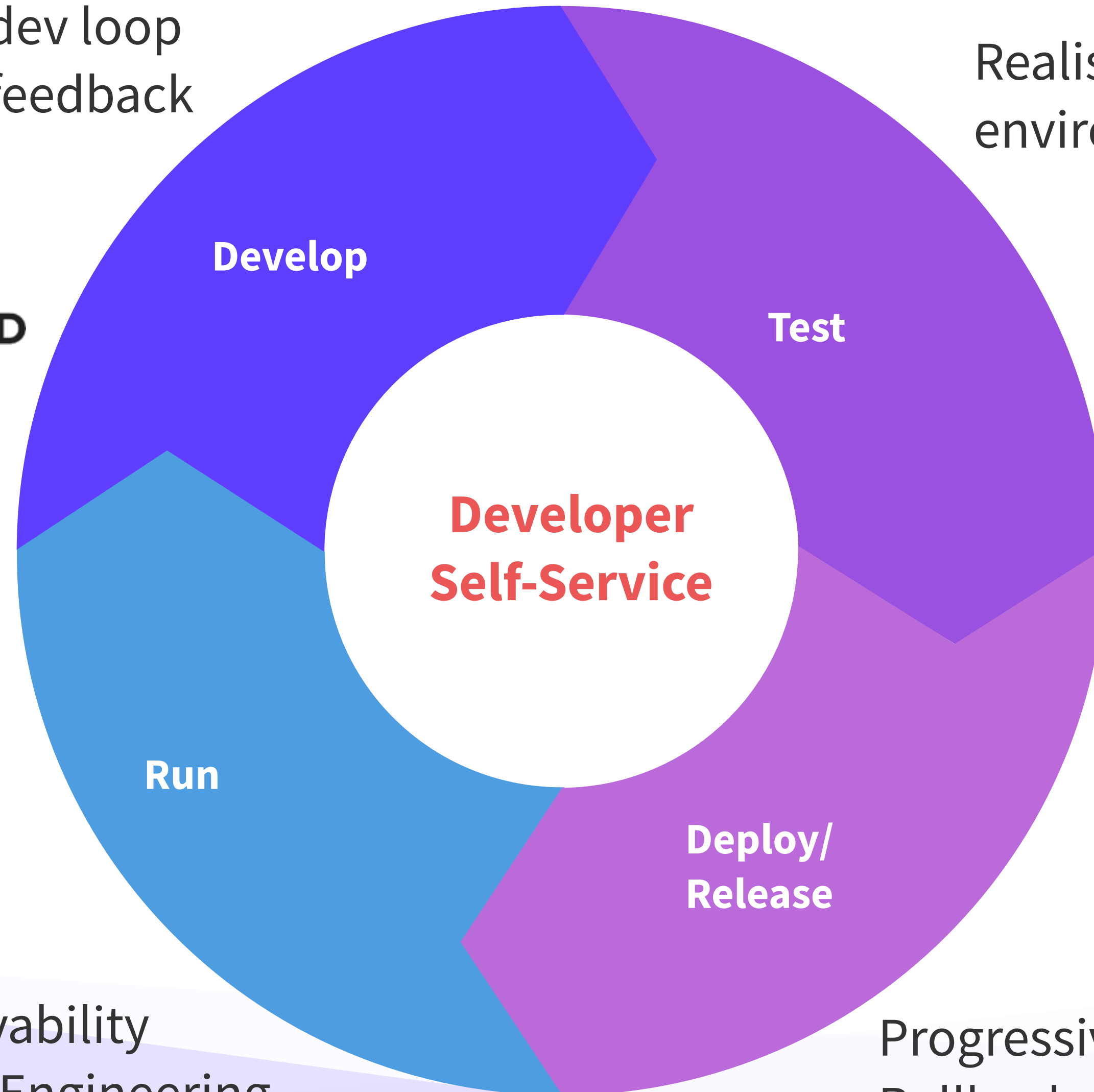
 Buildpacks.io

 OpenTelemetry

 ChaosToolkit

 keptn

Observability  
Chaos Engineering  
Runbooks



Realistic test  
environments



 VELERO

 HELM

 flux



 argo

Progressive Delivery  
Rollback

 Consul

 LINKERD

 Istio

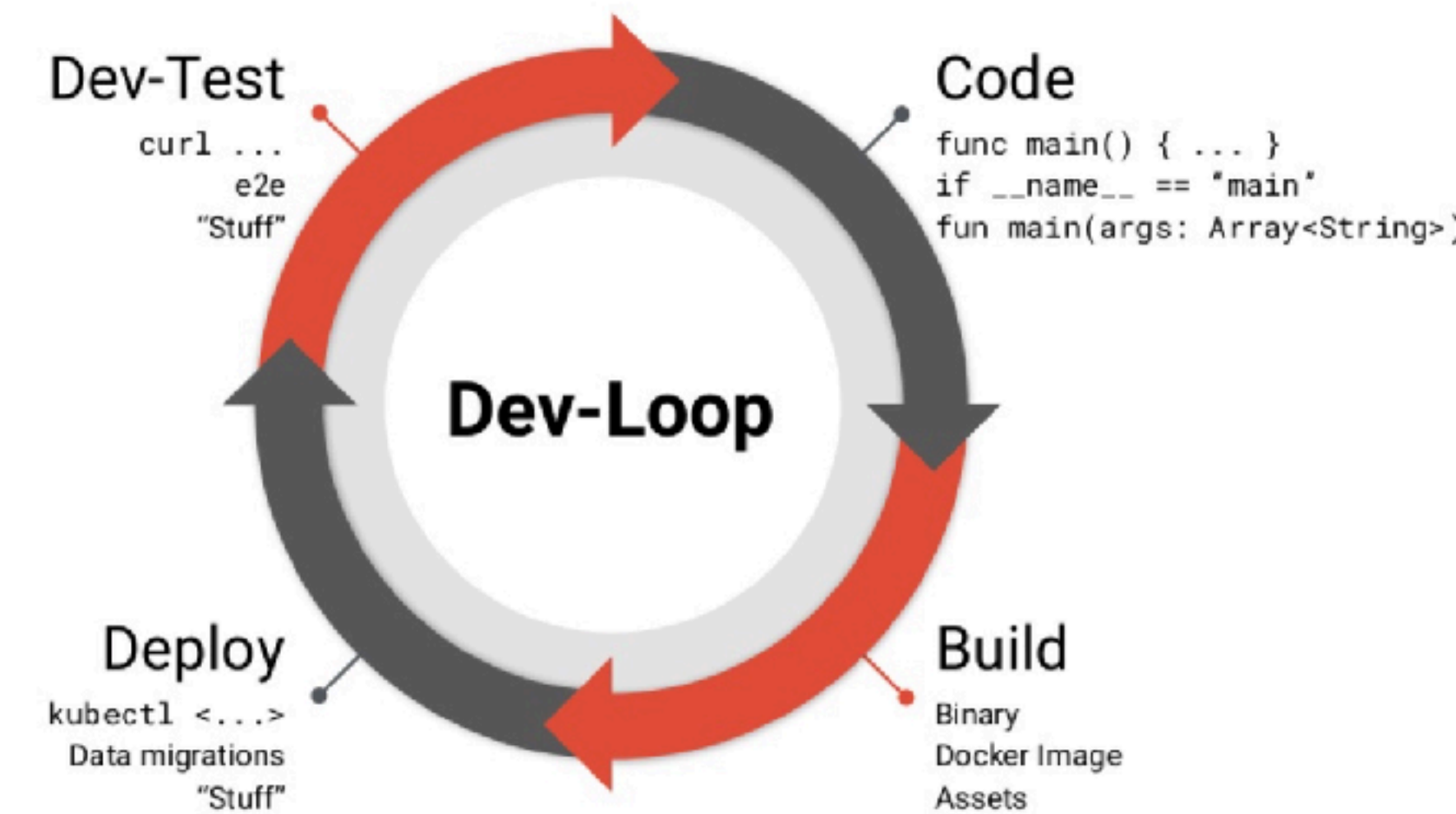
 AxBASSADOR



**Patterns, practices, and tooling**

# Pattern: Artifact Syncing

- Pain points:
  - Can't run all required dependent services locally
  - Sick of code, build image, push to remote registry slow dev loop
  - Integration tests rely on mocks (with implicit assumptions)
- Solution
  - Deploy all services to remote environment and sync local changes (combine with buildpacks)
  - Often combined with development environment bridging
- Example tool
  - Skaffold (ksync, Tilt, Garden)

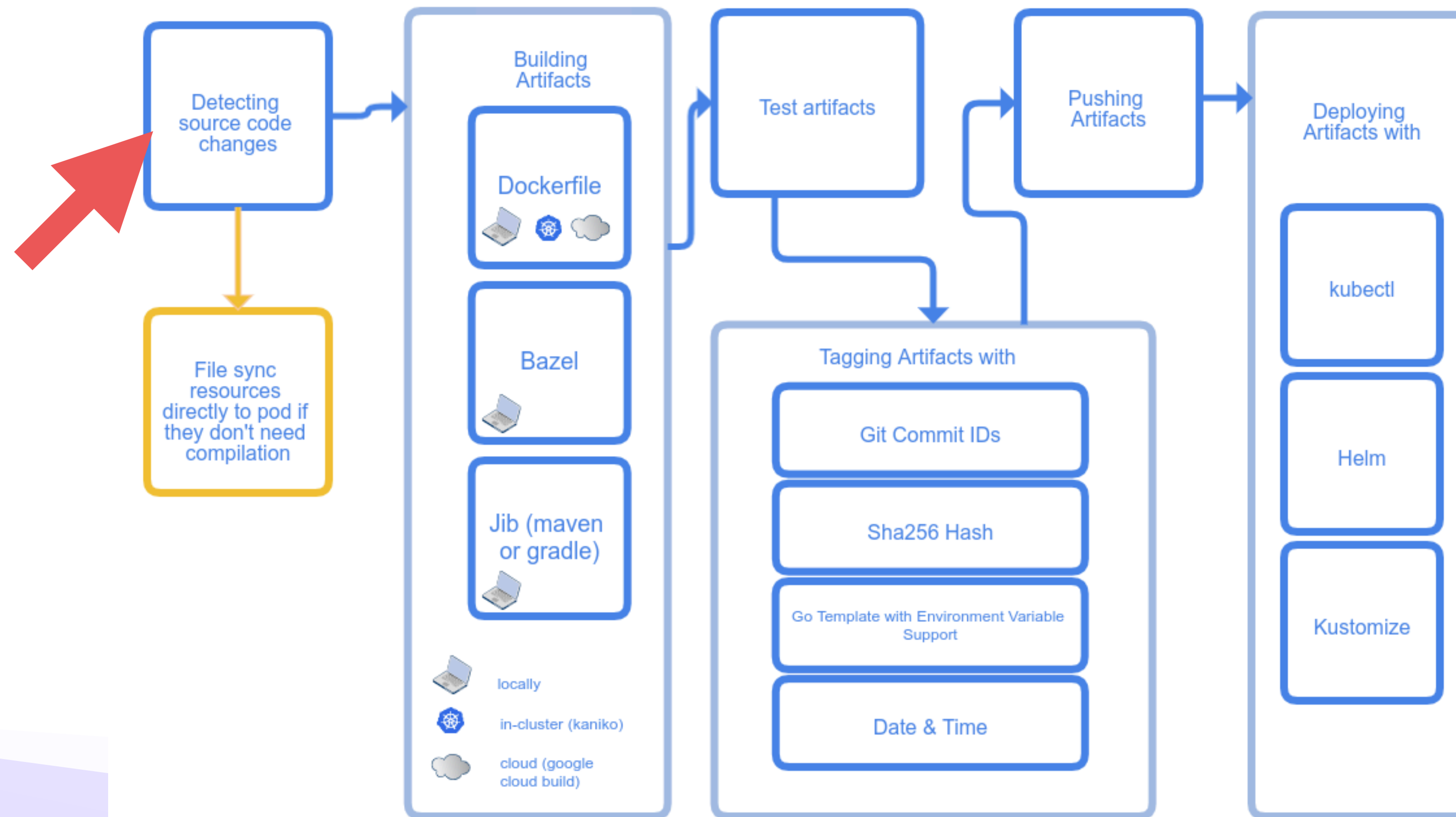


# Skaffold: “Source to K8s” CLI Tool

\$ skaffold dev

```
Listing files to watch...
- skaffold-example
Generating tags...
- skaffold-example -> skaffold-example:v1.1.0-113-g4649f2c16
Checking cache...
- skaffold-example: Not found. Building
Found [docker-desktop] context, using local docker daemon.
Building [skaffold-example]...
Sending build context to Docker daemon 3.072kB
Step 1/6 : FROM golang:1.12.9-alpine3.10 as builder
----> e0d646523991
Step 2/6 : COPY main.go .
----> Using cache
----> e4788ffa88e7
Step 3/6 : RUN go build -o /app main.go
----> Using cache
----> 686396d9e9cc
Step 4/6 : FROM alpine:3.10
----> 965ea09ff2eb
Step 5/6 : CMD ["/app"]
----> Using cache
----> be0603b9d79e
Step 6/6 : COPY --from=builder /app .
----> Using cache
----> c827aa5a4b12
Successfully built c827aa5a4b12
Successfully tagged skaffold-example:v1.1.0-113-g4649f2c16
Tags used in deployment:
- skaffold-example -> skaffold-example:c827aa5a4b12e707163842b803d666eda11b8ec20c7a480198960cfdcb251042
  local images can't be referenced by digest. They are tagged and referenced by a unique ID instead
Starting deploy...
- pod/getting-started created
Watching for changes...
[getting-started] Hello world!
[getting-started] Hello world!
[getting-started] Hello world!
```

# Skaffold: “Source to K8s” CLI Tool



# Skaffold: “Source to K8s” CLI tool

`skaffold dev` & `skaffold debug`

- Detects changes in your source code and handles the pipeline to build, push

`skaffold build` and `skaffold deploy`

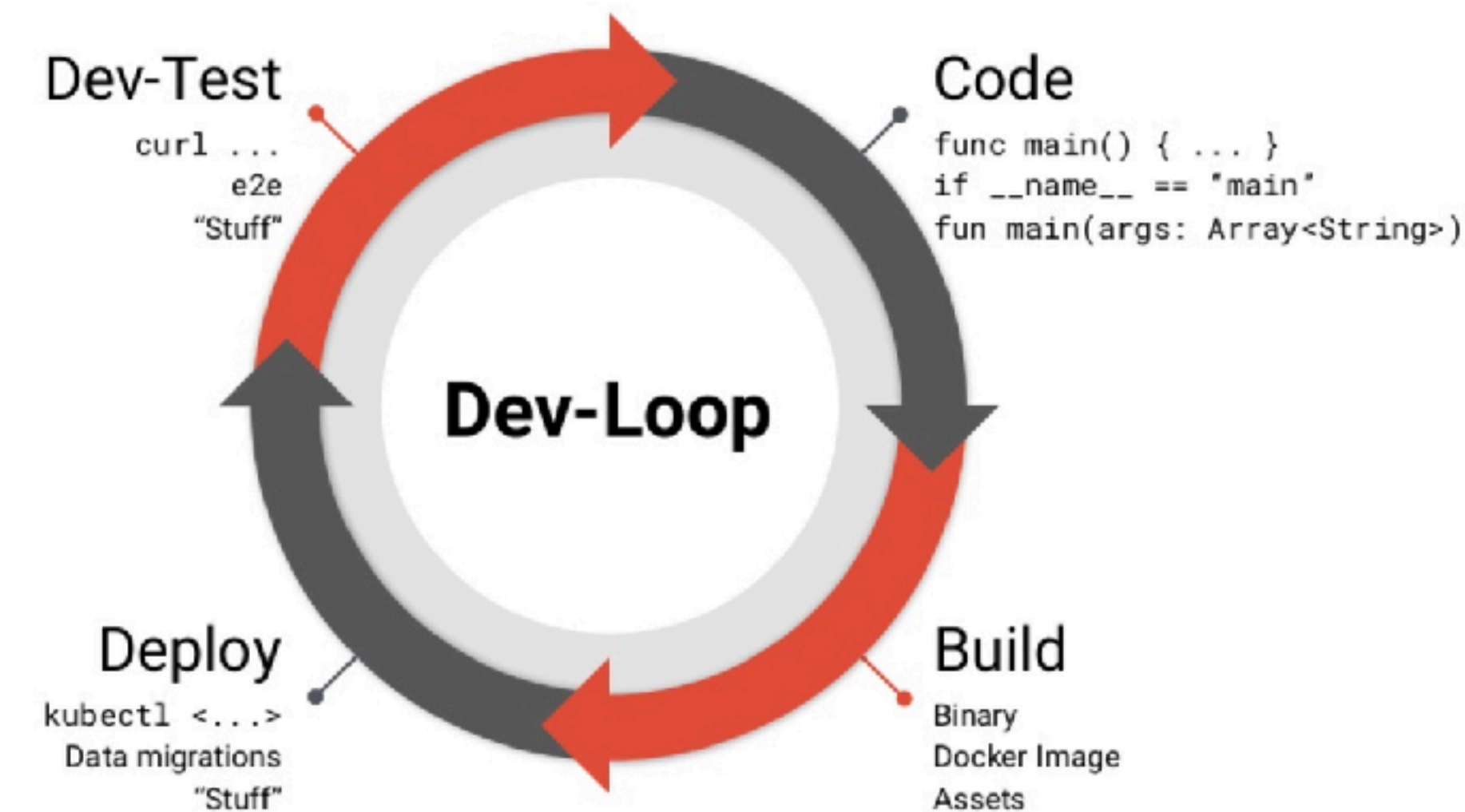
- Execute as part of your CI/CD pipeline, or `skaffold run` end-to-end

`skaffold render`

- Build your images and render templated Kubernetes manifests for use in GitOps workflows

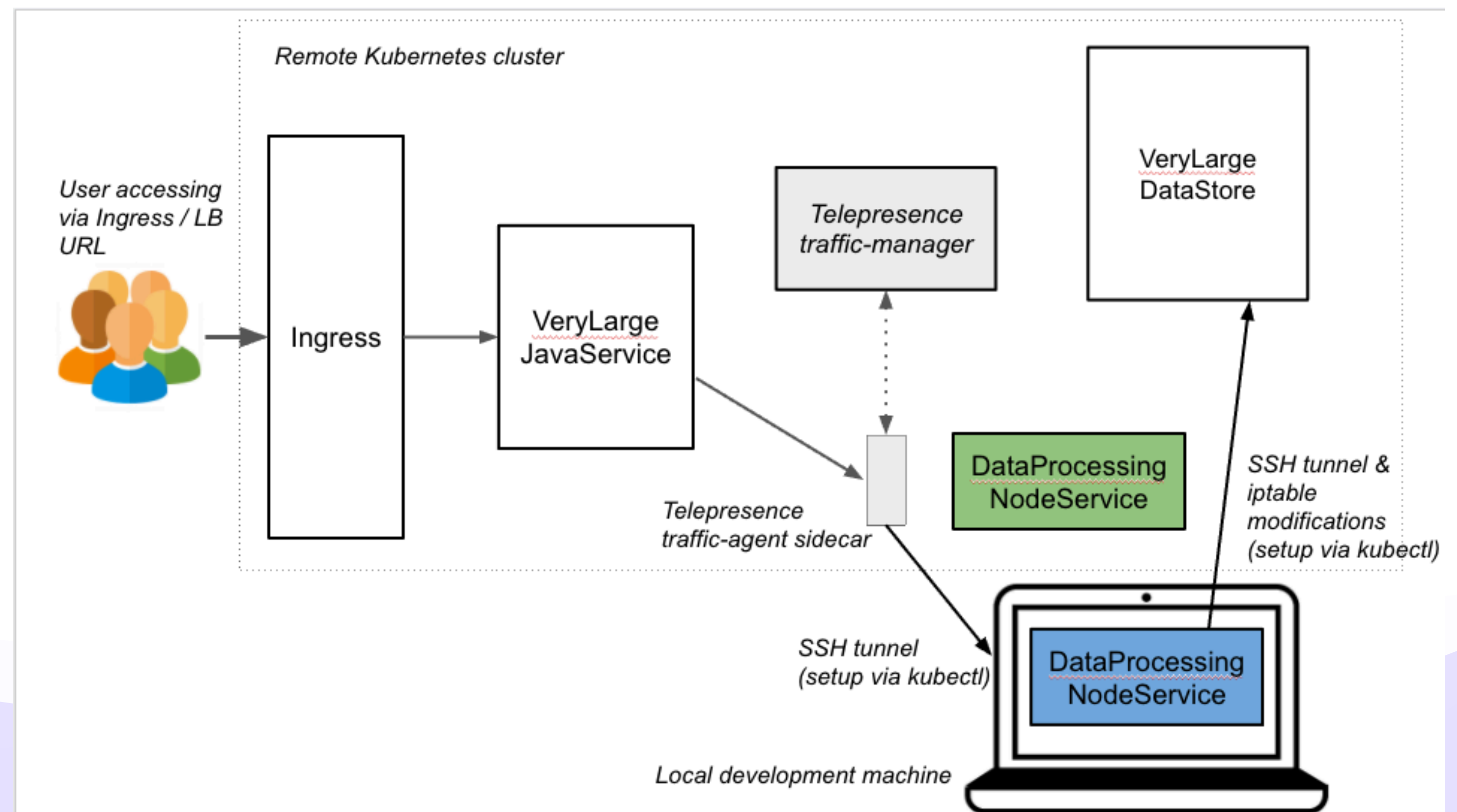
# Pattern: Dev Environment Bridging/Extension

- Pain points:
  - Can't run all required dependent services locally
  - Integration tests rely on mocks (with implicit assumptions)
  - Need “hot reload” coupled to remote services/resources
- Solution
  - Deploy all services to remote environment and proxy/re-route traffic to/from a local running copy of a service (or subset of services)
- Example tool
  - Telepresence (kubefwd, Okteto)

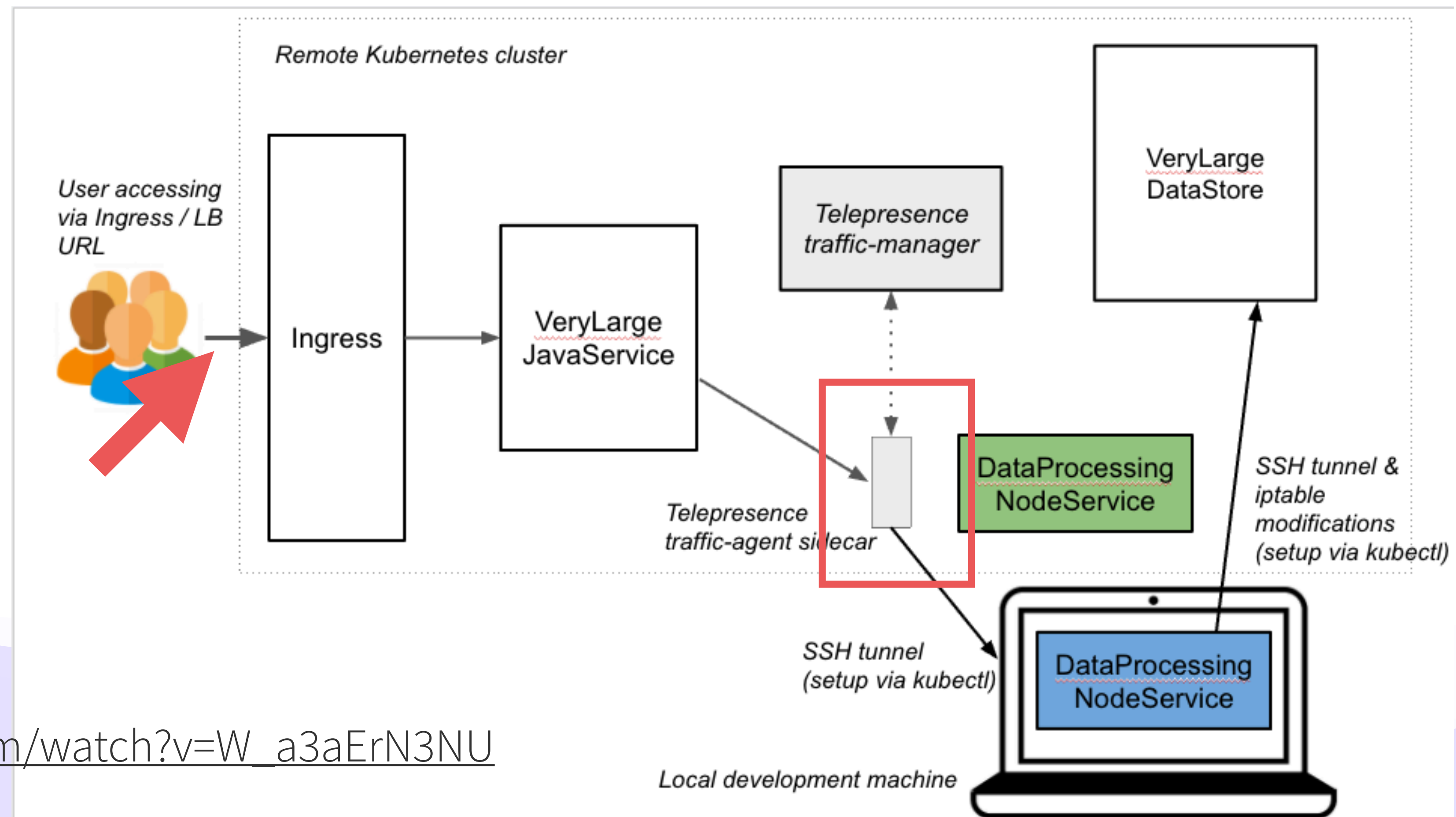


# Telepresence: Local-to-Remote bridge

```
$ telepresence intercept dataprocessingnodeservice --port 3000
```



# Telepresence: Local-to-Remote bridge



[www.youtube.com/watch?v=W\\_a3aErN3NU](https://www.youtube.com/watch?v=W_a3aErN3NU)

# Telepresence: Local-to-Remote bridge

`telepresence connect`

- Open a tunnel to the remote cluster; exposes “in-cluster” services/network/DNS

`telepresence intercept my-service --port 3000`

- Re-routes (intercepts) traffic to my-service in the remote cluster to my local machine

`telepresence login & telepresence intercept`

- Create preview URL to isolate and share results of the intercept

# Telepresence: Local-to-Remote bridge

```
$ telepresence intercept dataprocessingnodeservice --port 3000
```

Using deployment dataprocessingnodeservice  
intercepted

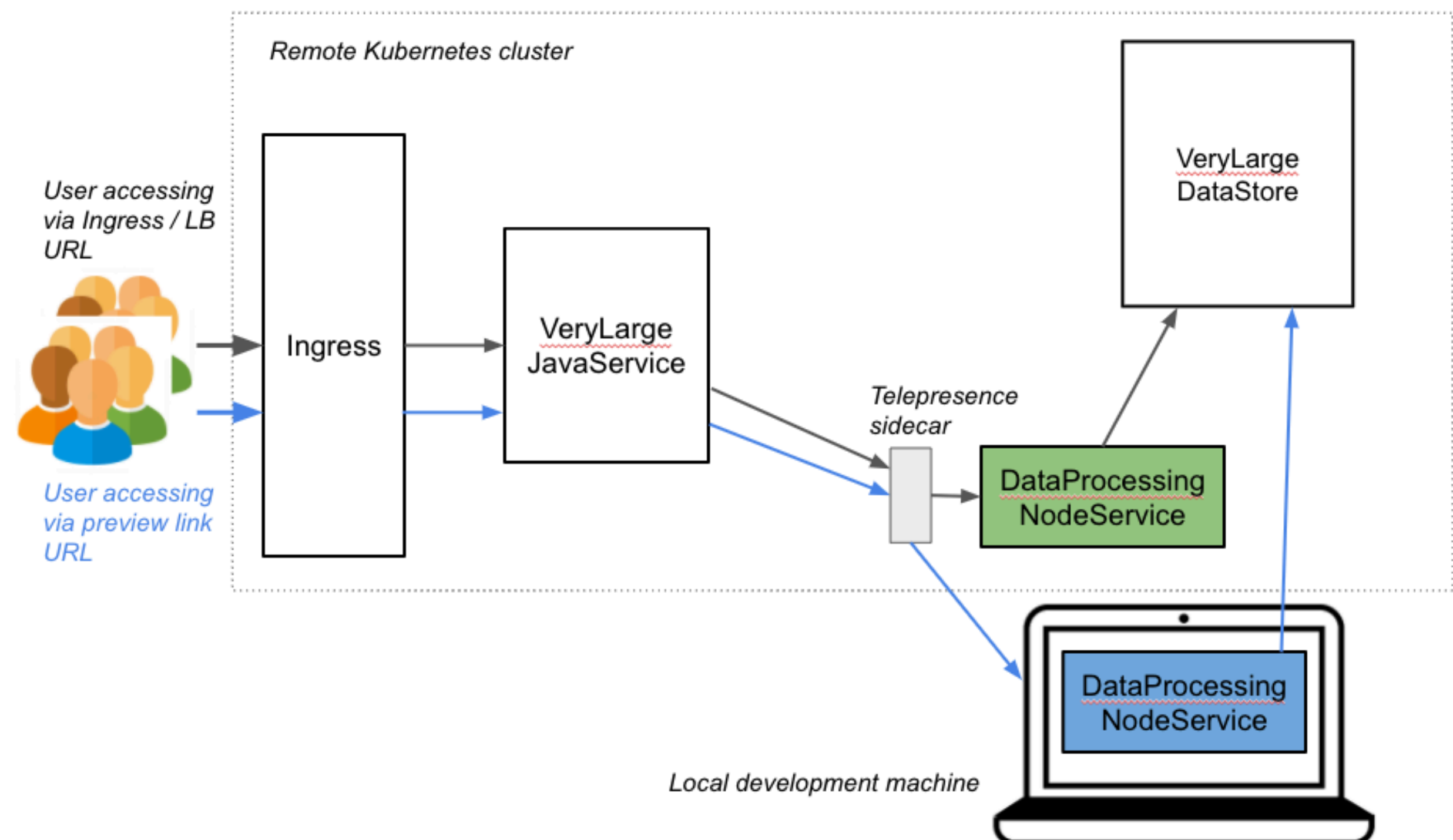
State : ACTIVE

Destination : 127.0.0.1:3000

Intercepting: HTTP requests that match all of:

header("x-telepresence-intercept-id") ~= regexp ("76a1e848-1829-74x-1138-e3294c1e9119:dataprocessingnodeservice")

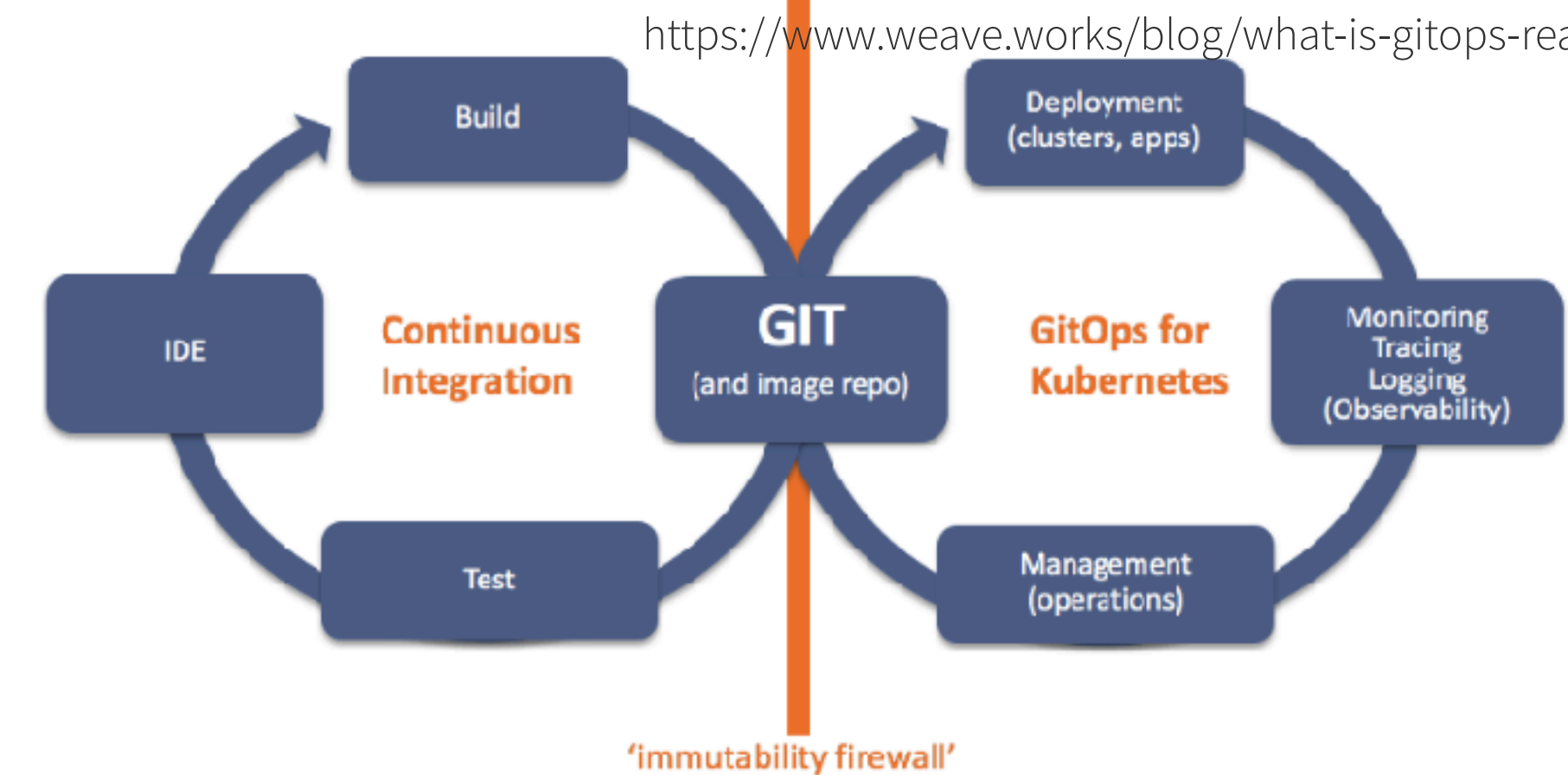
Preview URL : [https://\[random-subdomain\].preview.edgestack.me](https://[random-subdomain].preview.edgestack.me)



[www.youtube.com/watch?v=W\\_a3aErN3NU](https://www.youtube.com/watch?v=W_a3aErN3NU)

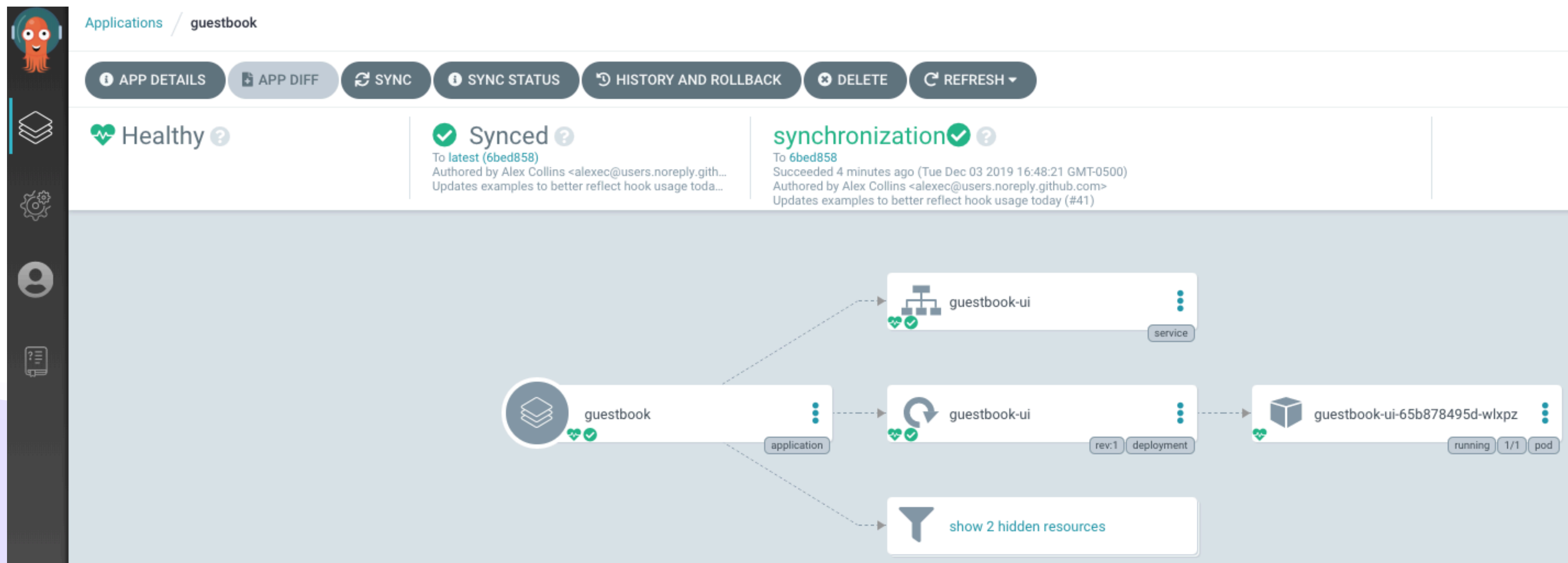
# Pattern: GitOps

- Pain points:
  - No repeatable, auditable, & understandable deployment of infra and apps
  - Slow hand-offs and manual deploys
- Solution
  - Use version control (git) as single source of truth
  - Declarative config; reconciled against target cluster
- Example tool
  - Argo CD (Flux, JenkinsX, werf)

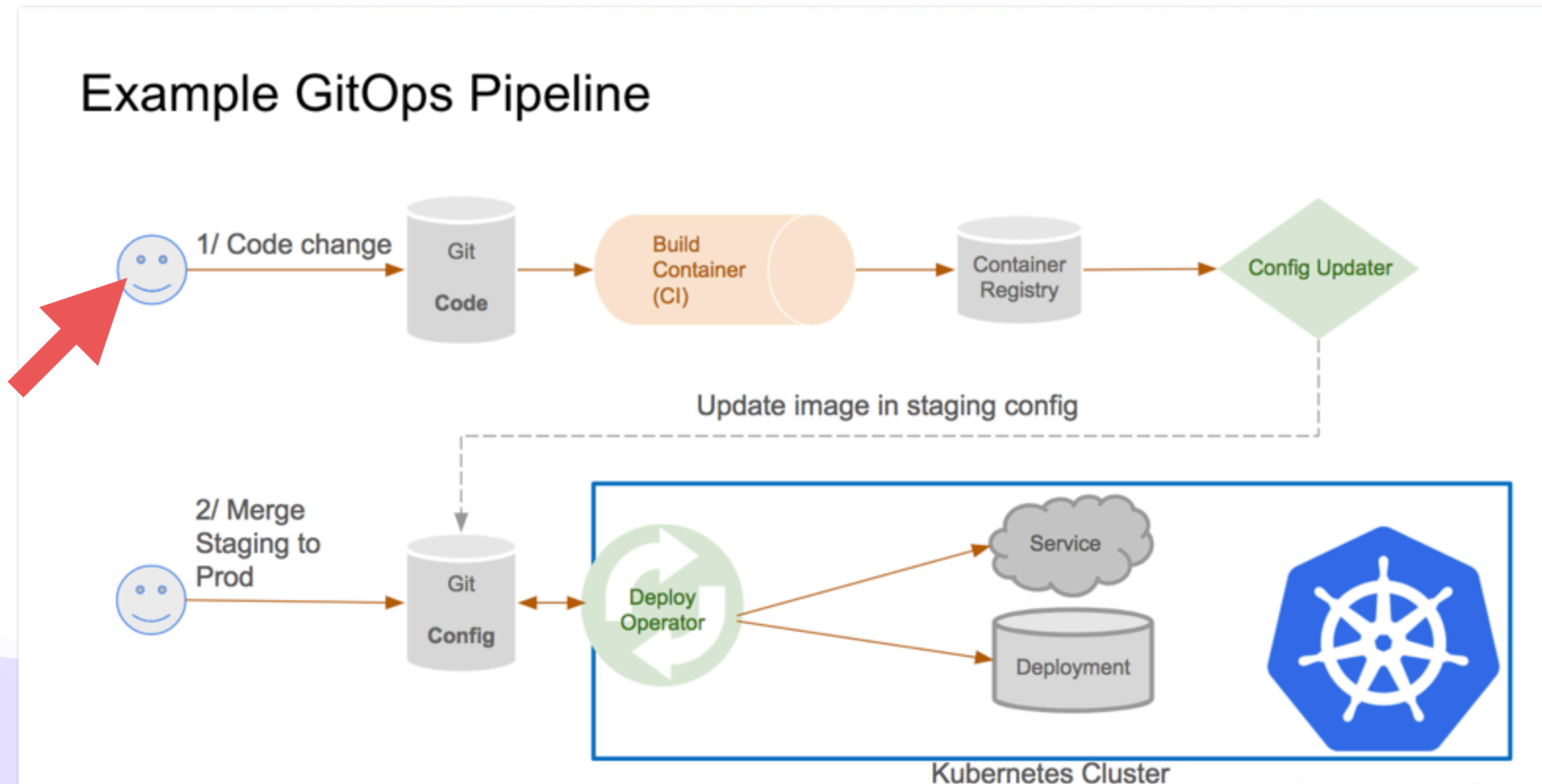


# Argo CD: GitOps for K8s

```
$ argocd app create guestbook --repo https://github.com/argoproj/argocd-example-apps.git /  
--path guestbook --dest-server https://kubernetes.default.svc --dest-namespace default
```




# GitOps in a Nutshell



<https://www.weave.works/blog/the-gitops-pipeline>

# Argo Rollouts: K8s Progressive Delivery Controller



```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: guestbook
spec:
  ...
  strategy:
    canary:
      analysis:
        templates:
          - templateName: success-rate
        startingStep: 2 # delay starting analysis run
        args:
          - name: service-name
            value: guestbook-svc.default.svc.cluster.local
      steps:
        - setWeight: 20
        - pause: {duration: 10m}
        - setWeight: 40
        - pause: {duration: 10m}
        - setWeight: 60
        - pause: {duration: 10m}
        - setWeight: 80
        - pause: {duration: 10m}
```

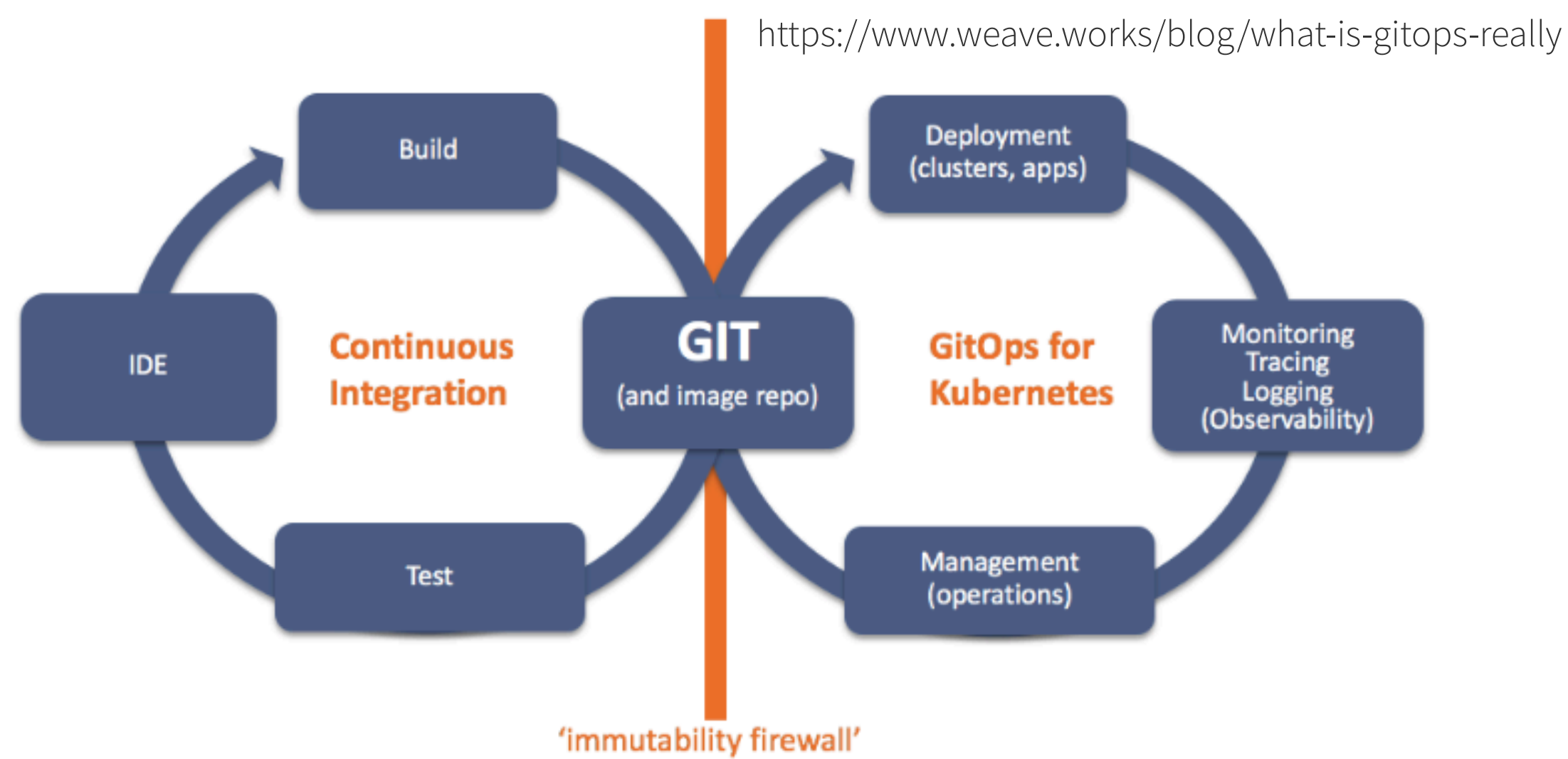
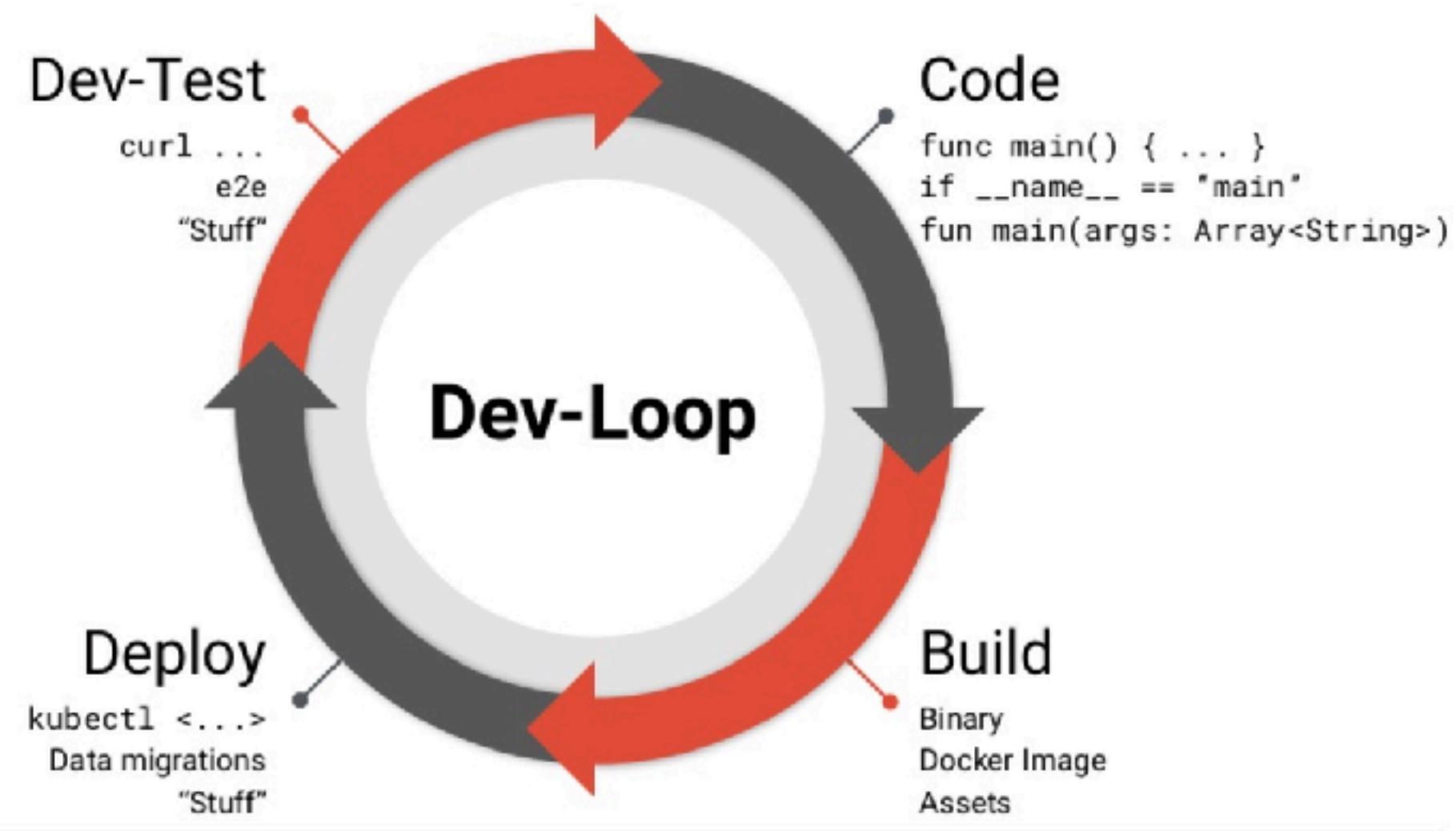
```
apiVersion: argoproj.io/v1alpha1
kind: AnalysisTemplate
metadata:
  name: log-error-rate
spec:
  args:
    - name: service-name
  metrics:
    - name: error-rate
      interval: 5m
      successCondition: result <= 0.01
      failureLimit: 3
      provider:
        datadog:
          interval: 5m
          query: |
            sum:requests.error.count{service:{{args.service-name}}} /
            sum:requests.request.count{service:{{args.service-name}}}
```

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: guestbook
spec:
  ...
  strategy:
    canary:
      steps:
        - setWeight: 20
        - pause: {duration: 5m}
        - analysis:
            templates:
              - templateName: log-error-rate
            args:
              - name: service-name
                value: guestbook-svc.default.svc.cluster.local
```

# Putting it All Together: Safety and Speed

Development

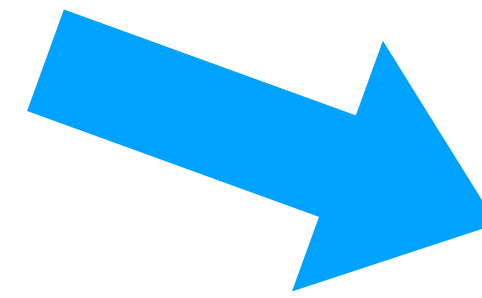
Production



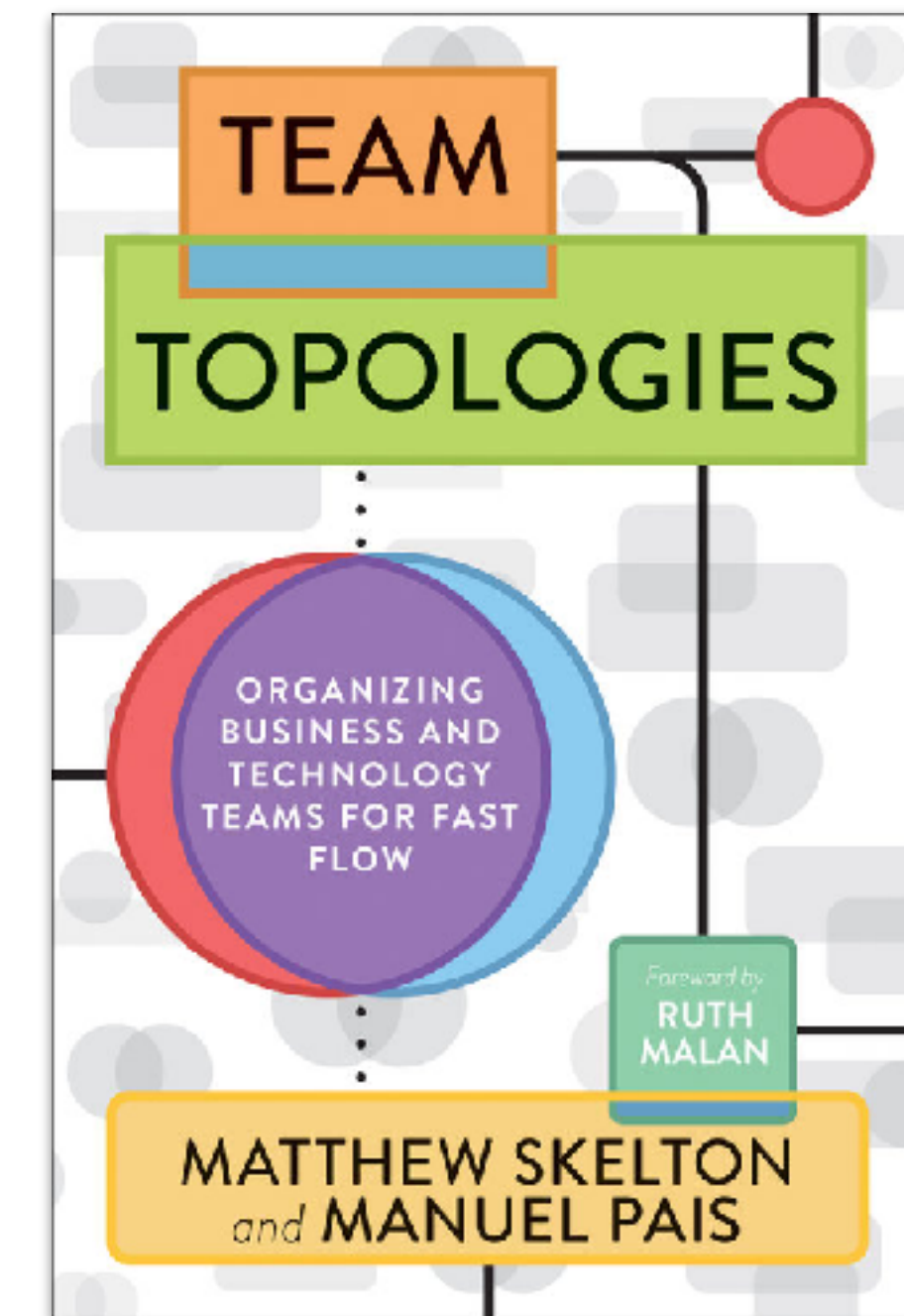
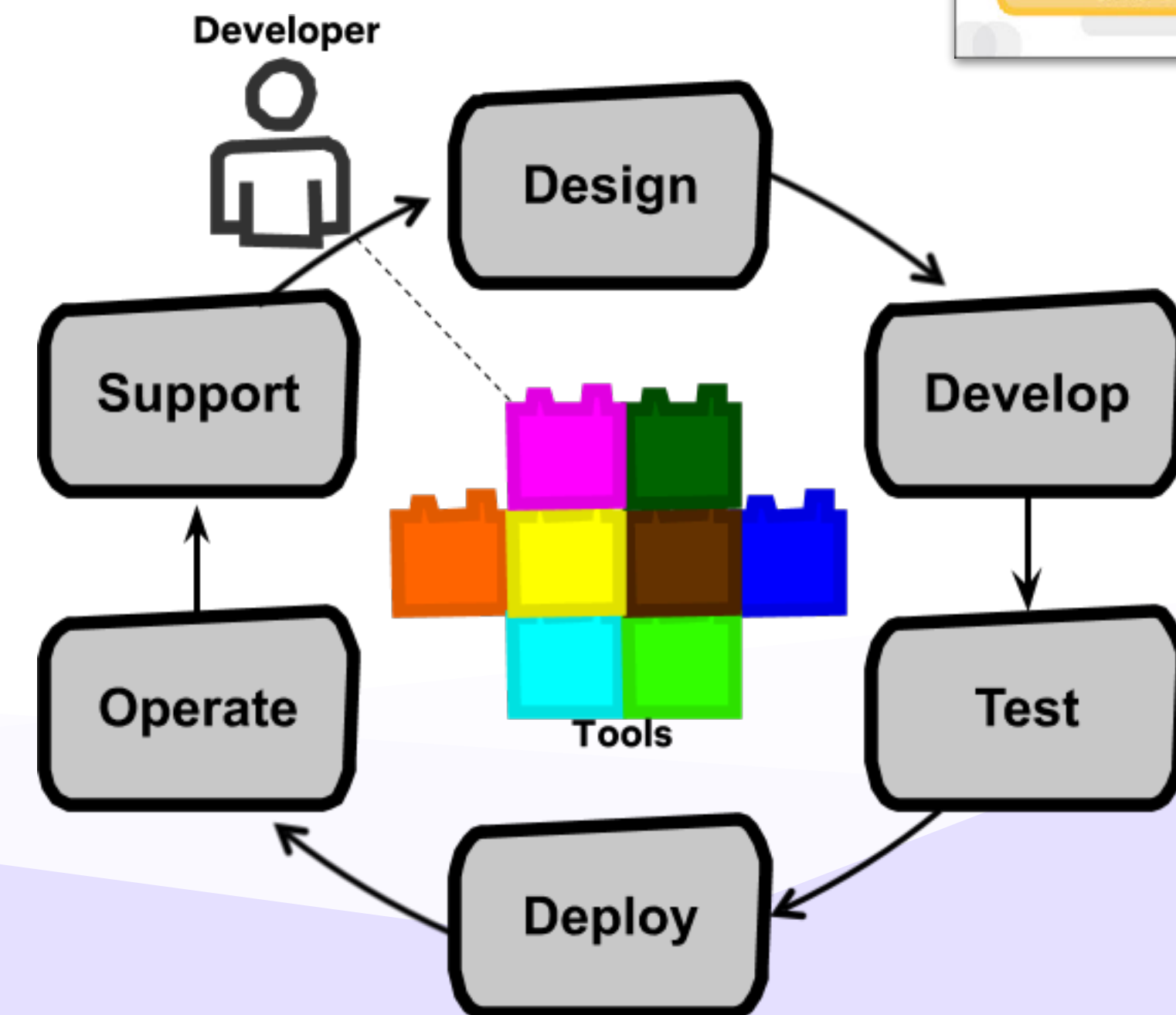
# What about the platform and the people?



# Workflow: Full Cycle Development



- App teams have full responsibility (and authority) for **delivering a service**, and ultimately, **value to users**
- Increase agility by accelerating the feedback loop
- <https://netflixtechblog.com/full-cycle-developers-at-netflix-a08c31f83249>





# Conclusion

- Continuous delivery of value to users requires a focus on safety and speed throughout the engineering workflow
- Bringing old mental models and tools can add toil to the process of building, deploying, and operating cloud native services
- Adopt best practices first, such as artifact syncing, dev environment bridging, and GitOps to increase both safety and speed

# Many thanks!

- Learn more:
  - [www.getambassador.io/resources](https://www.getambassador.io/resources)
  - [www.getambassador.io/use-case/local-kubernetes-development](https://www.getambassador.io/use-case/local-kubernetes-development)
  - [www.infoq.com/profile/Daniel-Bryant](https://www.infoq.com/profile/Daniel-Bryant)
- Find me in:
  - Datawire OSS Slack: [d6e.co/slack](https://d6e.co/slack)
  - Twitter [@danielbryantuk](https://twitter.com/danielbryantuk)

