

GOTO **AARHUS 2023**

#GOTOaar

Software rollout at scale:

Using gitops to scale Kubernetes rollouts





Thor Anker Kvisgård Lange
Platform Development Specialist

Bridging Dev and Ops.

- 20+ years of experience as developer and architect
- Working with Kubernetes 5+ years
- Working with Netic Kubernetes offerings



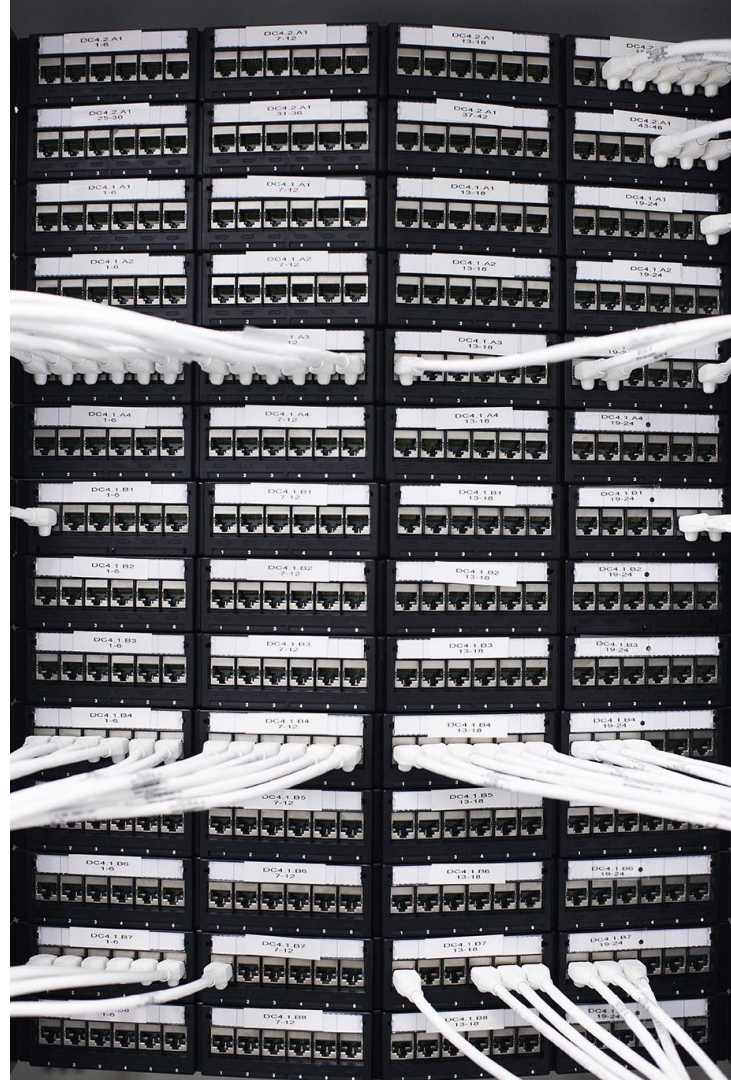
in/thor-lange-26b388



@langecode

Agenda

- What is gitops anyway?
- Managing a Kubernetes platform
- Risk mitigation by versioning
- Opportunity versus risk
- Scaling rollouts



What is gitops?

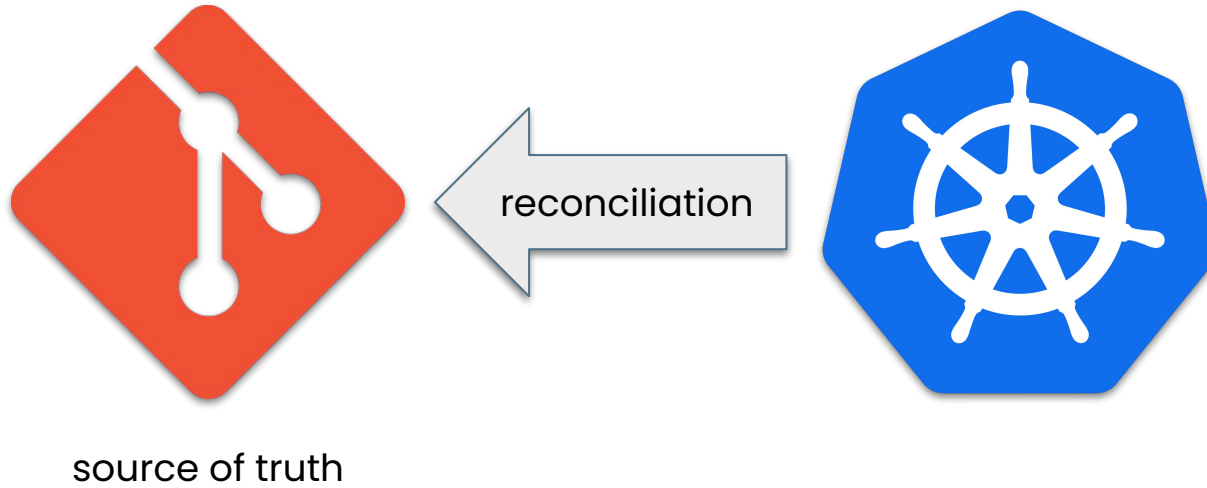
”

GitOps leverages Git as the single source of truth to define every part of a cloud-native system. Once declared in Git, a GitOps agent (Flux) automatically applies all code, configuration, and policies across dev, test, staging, and production environments.

What is GitOps?

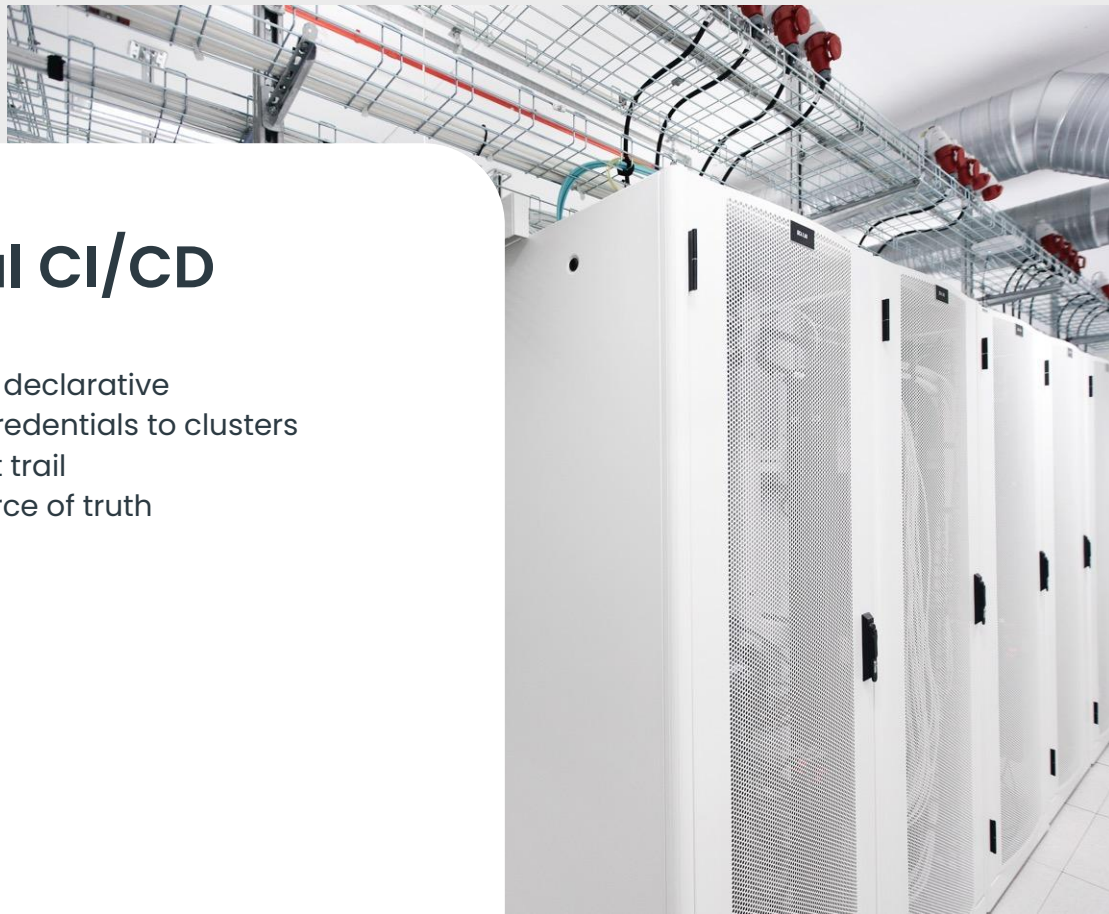
<https://www.weave.works/technologies/gitops/>

What is gitops?



Traditional CI/CD

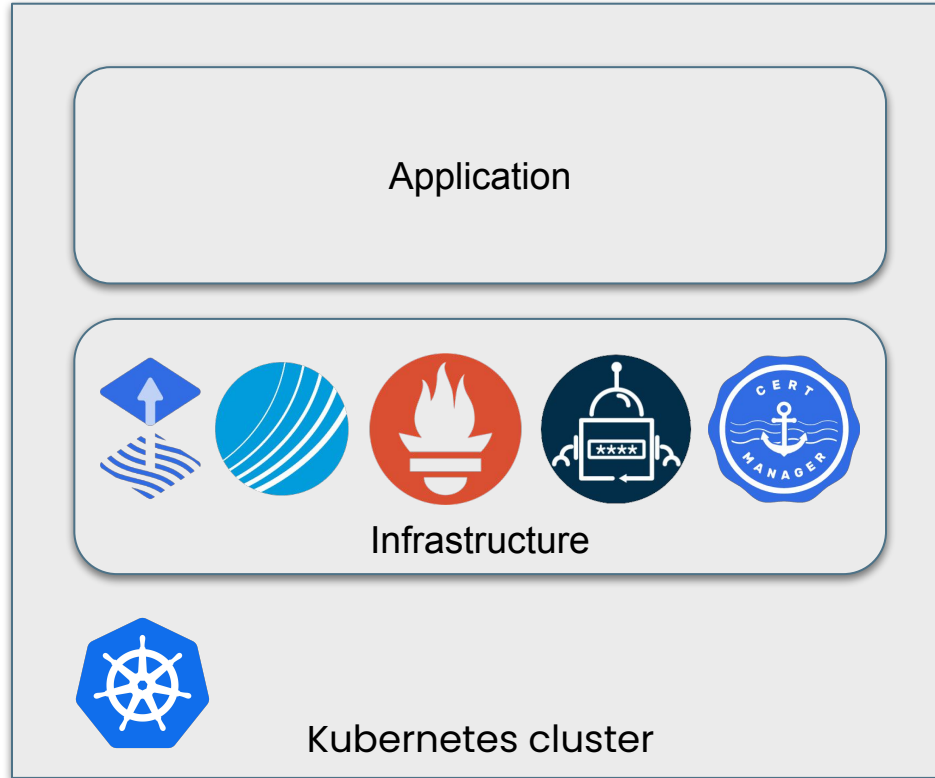
- Imperative vs declarative
- CI/CD need credentials to clusters
- No clear audit trail
- No single source of truth

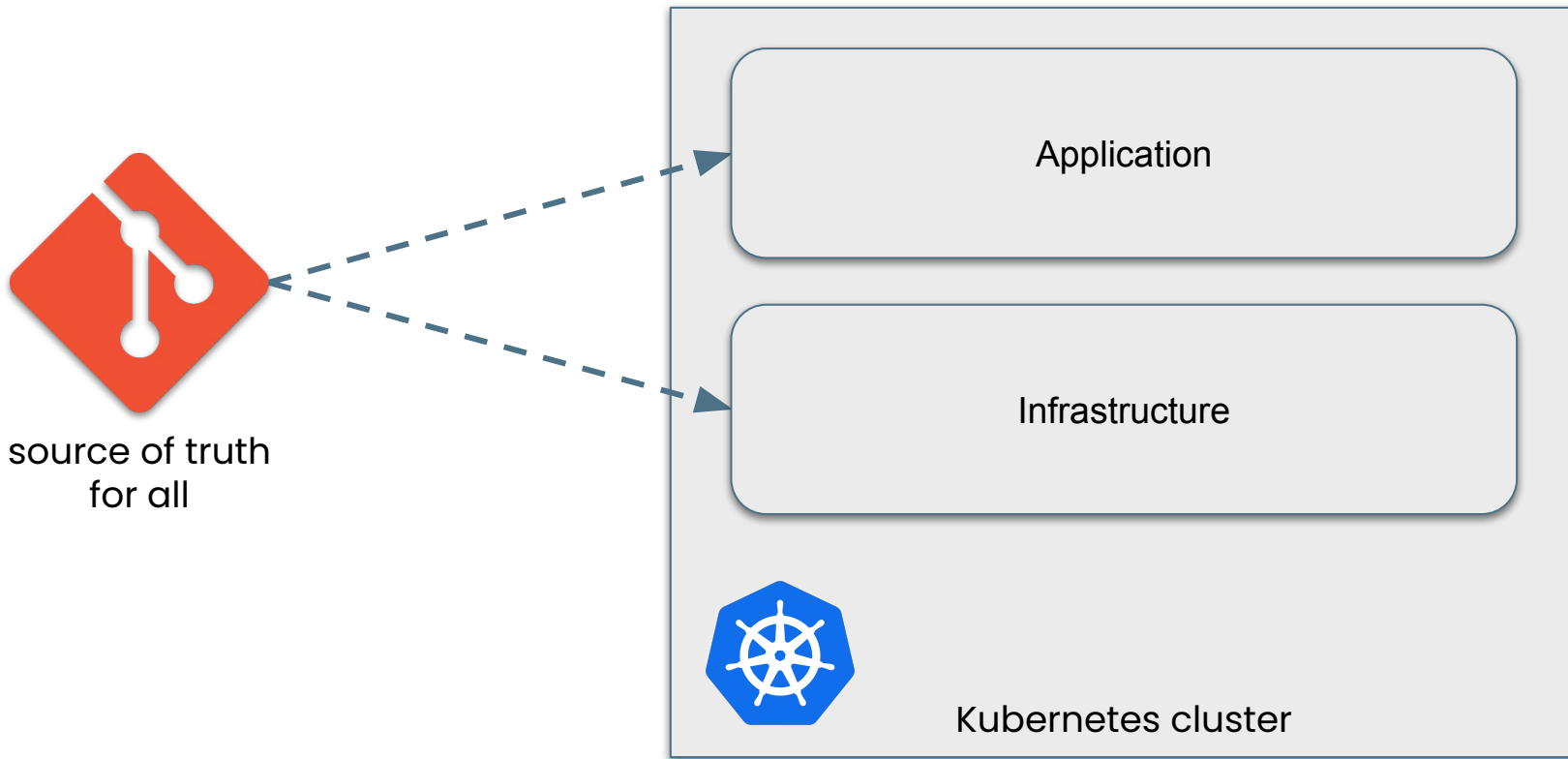




What is a Kubernetes platform?

Wait – there is more to Kubernetes?





Apply gitops to a single cluster

Initial Approach

- Single cluster
- Single repository
- Every commit to `main` changes cluster state

├ applications
└ infrastructure

main



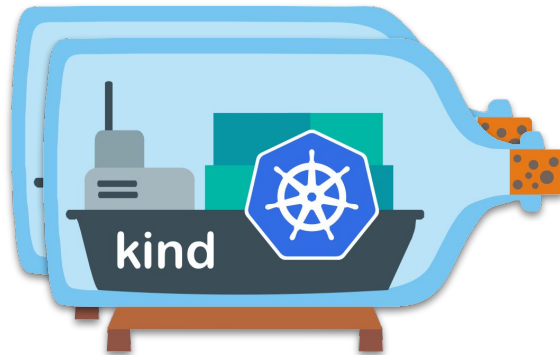
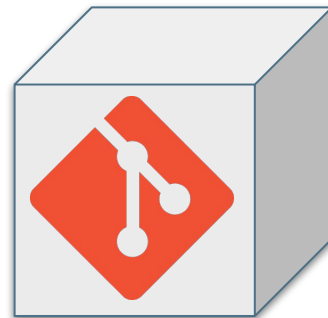
Lets see how it works...

Demo environment

- All examples running on laptop
- Simple ssh git server running as container image
- Kubernetes clusters based on kind (Kubernetes in Docker)

Application: Podinfo

Infrastructure: nginx



Demo – how does it work?

Gitops: Initial approach

Install flux and setup initial reconciliation (for demo purposes):

```
flux bootstrap git --url=...
```

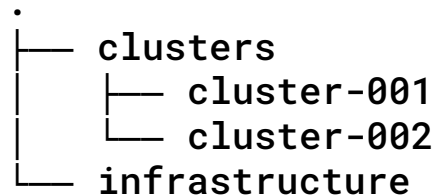
```
apiVersion: source.toolkit.fluxcd.io/v1
kind: GitRepository
metadata:
  name: flux-system
spec:
  url: ssh://git@git/...
  ref:
    branch: main
```

```
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: flux-system
spec:
  sourceRef:
    kind: GitRepository
    name: flux-system
  path: ./
```

Scaling to multiple clusters

Extend to multiple clusters

- Multi cluster
- Single repository
- Allow reuse across clusters
- Changes are rolled out to all clusters



main



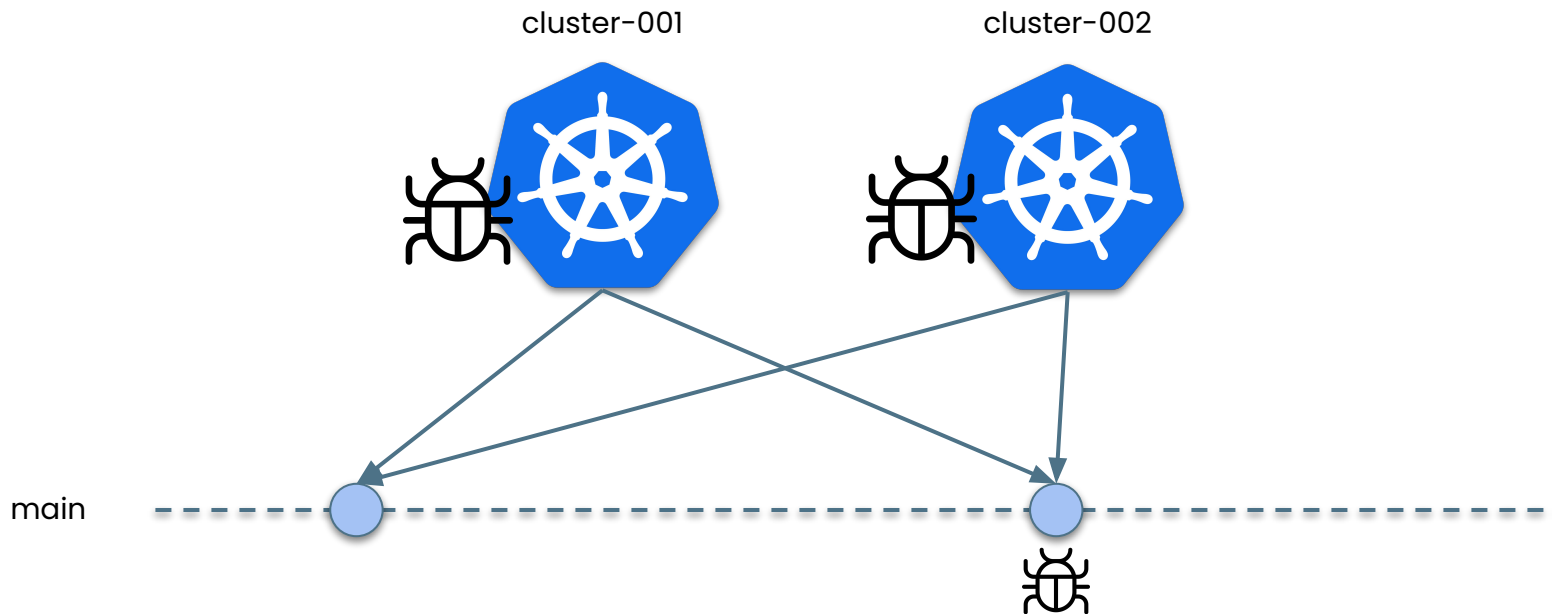
Gitops: Multiple clusters

- Create structure to support multiple clusters
- Reconcile a cluster specific path
- Include common infrastructure

```
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: flux-system
spec:
  sourceRef:
    kind: GitRepository
    name: flux-system
    path: ./clusters/cluster-001
```

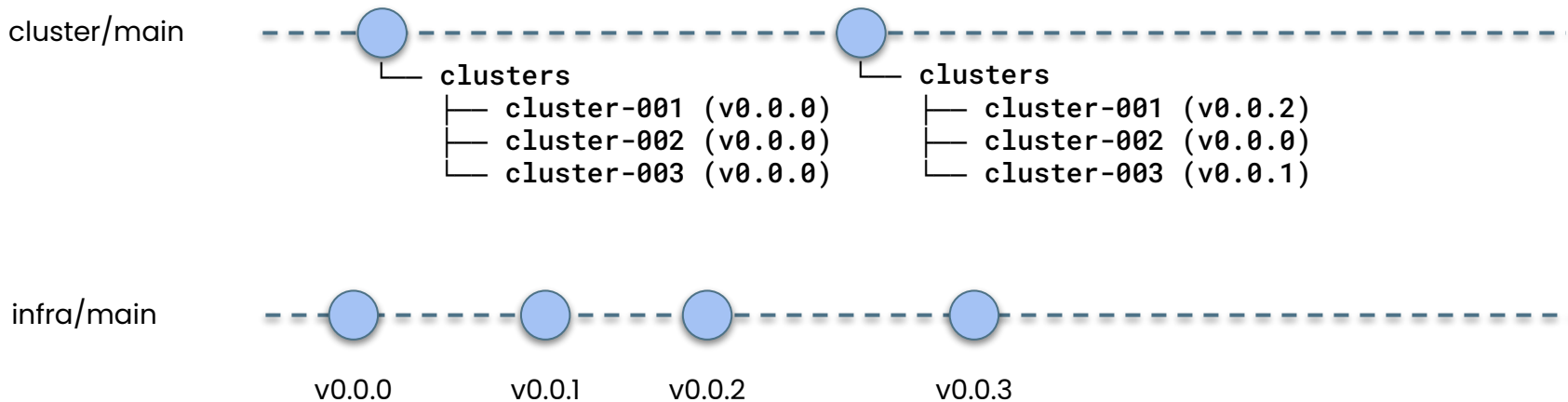
```
apiVersion: kustomize.toolkit.fluxcd.io/v1
kind: Kustomization
metadata:
  name: flux-system
spec:
  sourceRef:
    kind: GitRepository
    name: flux-system
    path: ./clusters/cluster-002
```

What just happened?



Mitigation: Version the cross cluster config

- Multi cluster
- Multi repository
- Risk mitigated through versioning (tagging)





Problem solved?

- Roll-out is fully versioned
- Risk is mitigated
- Scaling is linear with number of clusters
- But each new cluster add maintenance and thus complexity
- What else could we do?

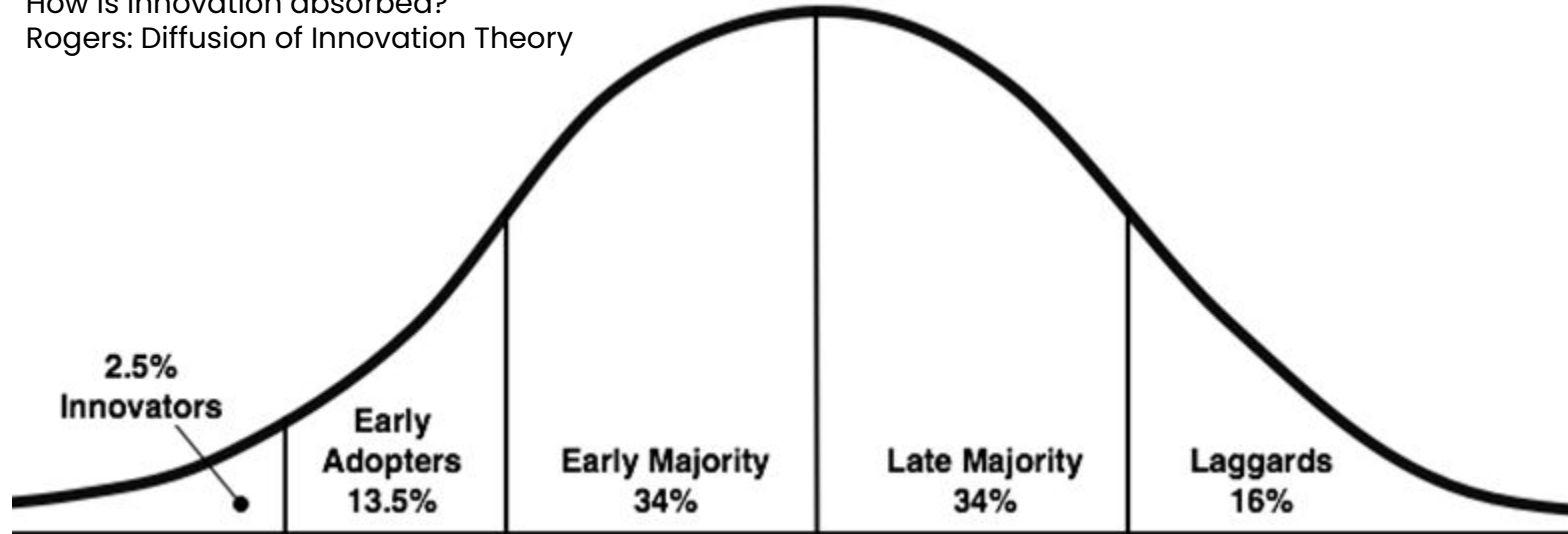
Sustainable scaling

- Need to roll-out to multiple clusters
- Need to be able to define groups
- “Traditional” approach dev, test, ... – won’t work
- As a service provider most clusters are production



Opportunity versus risk

- How is innovation absorbed?
- Rogers: Diffusion of Innovation Theory



Source: Everett Rogers, Diffusion of innovations model

<https://sphweb.bumc.bu.edu/otlt/mph-modules/sb/behavioralchangetheories/behavioralchangetheories4.html>

Archetype definitions

Innovators – These are people who want to be the first to try the innovation. They are venturesome and interested in new ideas. These people are very willing to take risks, and are often the first to develop new ideas.

Early Adopters – These are people who represent opinion leaders. They enjoy leadership roles, and embrace change opportunities. They are already aware of the need to change and so are very comfortable adopting new ideas.

Early Majority – These people are rarely leaders, but they do adopt new ideas before the average person. That said, they typically need to see evidence that the innovation works before they are willing to adopt it.

Late Majority – These people are skeptical of change, and will only adopt an innovation after it has been tried by the majority.

Laggards – These people are bound by tradition and very conservative. They are very skeptical of change and are the hardest group to bring on board.

Applying the theory: Resilience zones

- Using the Rogers categories
- When to receive updates?

innovators



cluster-001

E.g. development – quickly testing out new features

early-adopters



cluster-002

E.g. production for innovative applications

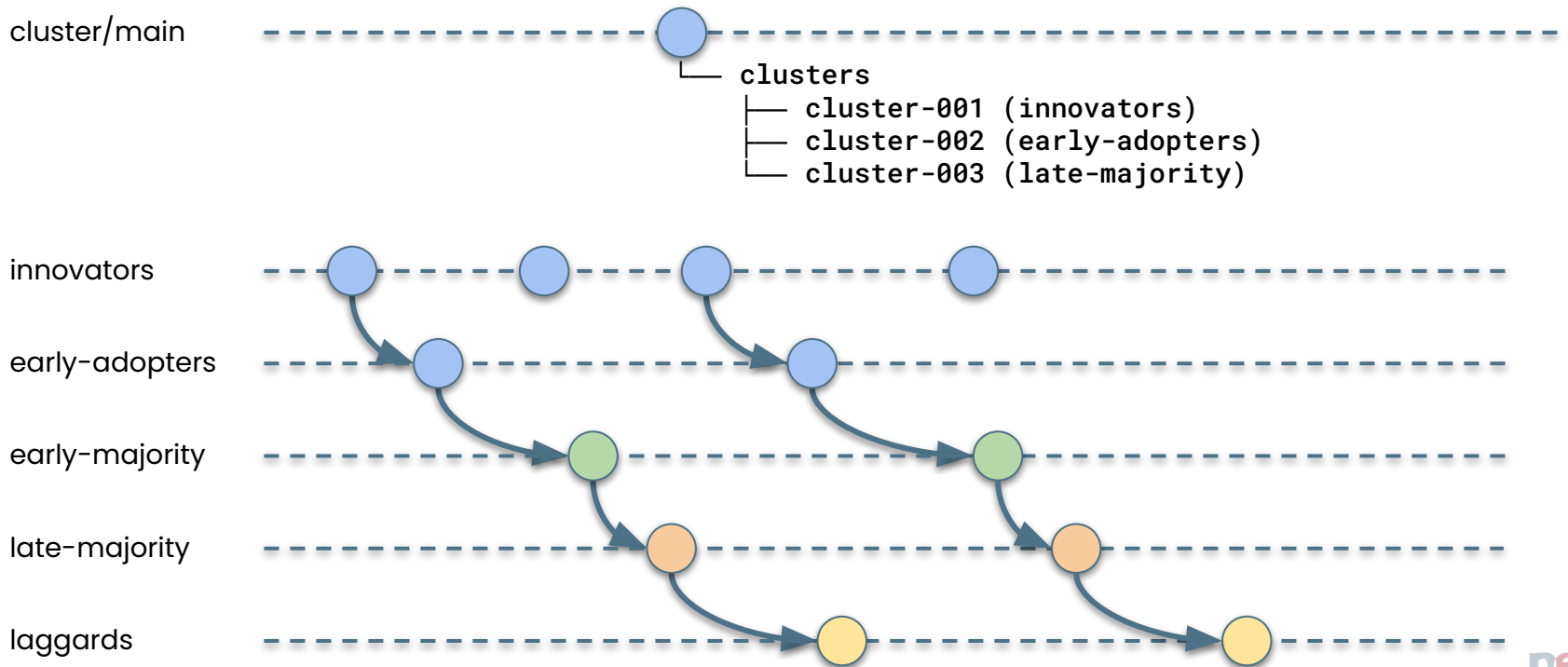
late-majority



cluster-003

E.g. production for critical applications

Branching out: Implementing with gitops



Demo: Same roll-out as before...



Takeaways

- Gitops is very powerful (also) at scale
- Don't repeat yourself (DRY principle) but...
- Mitigate the risks when scaling up
- Consider what is right in your context
- Automate and be aware of the complexity

Thank you.

<https://github.com/neticdk/k8s-workshop>



Alfred Nobels vej 25, DK-9220 Aalborg Ø • Tlf. 77 77 08 88



netic.dk



company/netic-as

Don't forget to
vote for this session
in the **GOTO Guide app**