
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

How CERN developers benefit from Kubernetes and CNCF landscape

Antonio Nappi

Who

Antonio Nappi

Computer Engineer at CERN, Kubernetes, Java

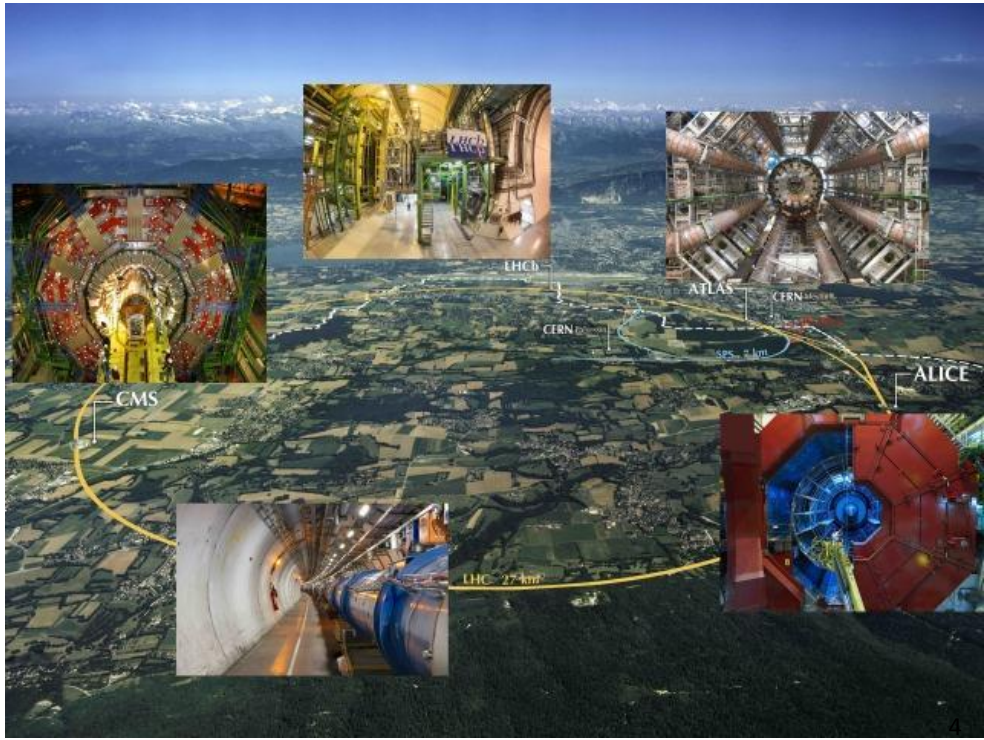
CERN

European Organization for Nuclear Research (CERN)

- study fundamental particles
  - how they interact
  - understand the fundamental laws of nature
- largest particle accelerator in the world
  - 27 km ring
  - 100 meters of depth
- World Wide Web
- science for peace
  - melting pot

What we **DON'T** do:

- black holes!



Agenda

What we do
Our challenges in VM model
Teams' structure
Timeline
Architecture overview
- ❑ Architecture
- ❑ Provisioning of clusters
- ❑ Cluster as cattle
- ❑ Applications deployment (old way)
- ❑ Applications deployment (new way)
- ❑ Monitoring system
- ❑ Logging system

GitOps
- ❑ Adoption
- ❑ Deployment
- ❑ Choices
- ❑ Challenges

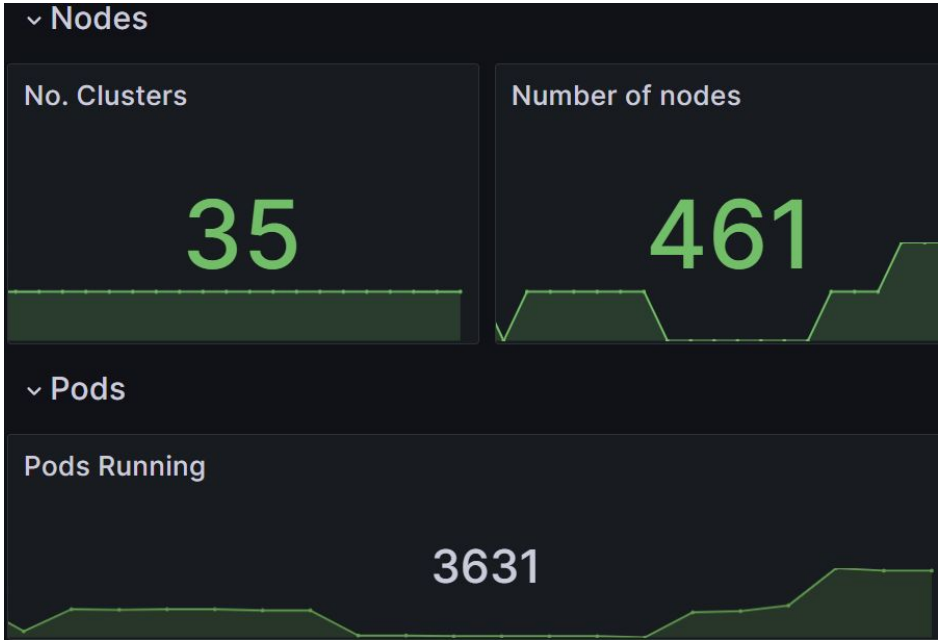
Conclusions
- ❑ Feelings after few years of production
- ❑ Take away

What

What we do:
- hosting critical Java applications for CERN daily life
- hosting Single Sign-On infrastructure based on Keycloak

Our users:
- developers from different areas
  - Finance and Administration
  - Engineering
- IAM engineers

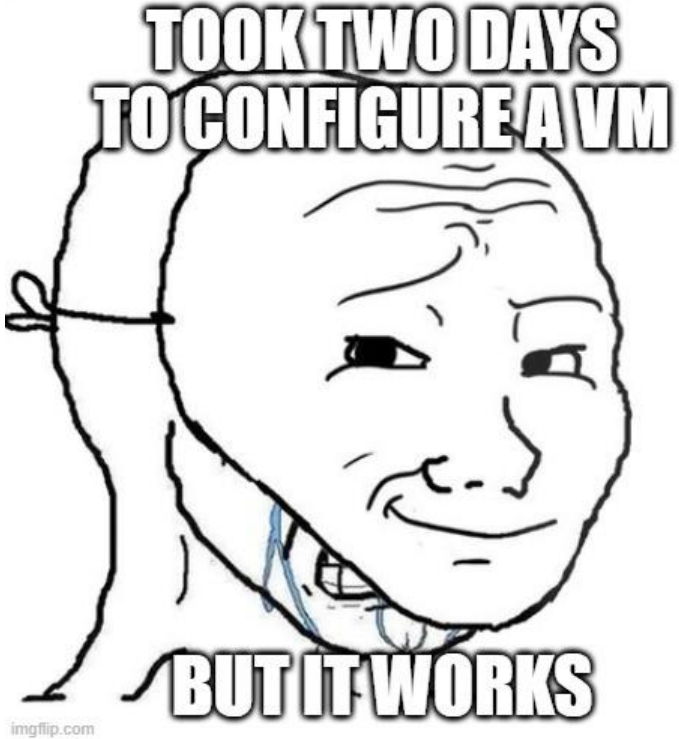
Numbers:
- 80+ applications
- More than 3k pods
- More than 400 nodes
- 35 clusters



Our challenges in VM model

Time wasted in:

- repetitive and easy tasks
- upgrading and provisioning new infrastructure
- maintaining custom scripts and puppet code to automate and speed up operations

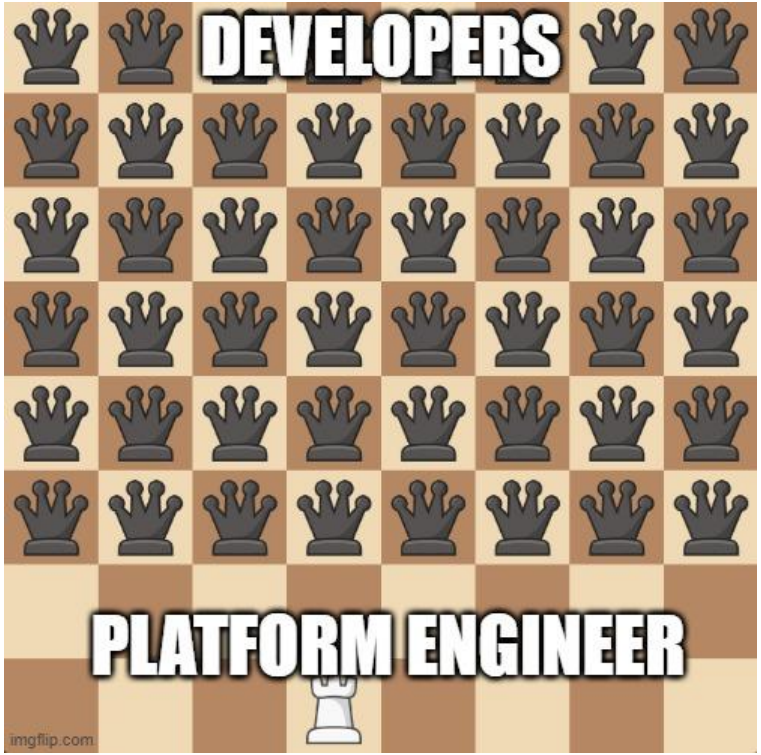


Teams’ structure

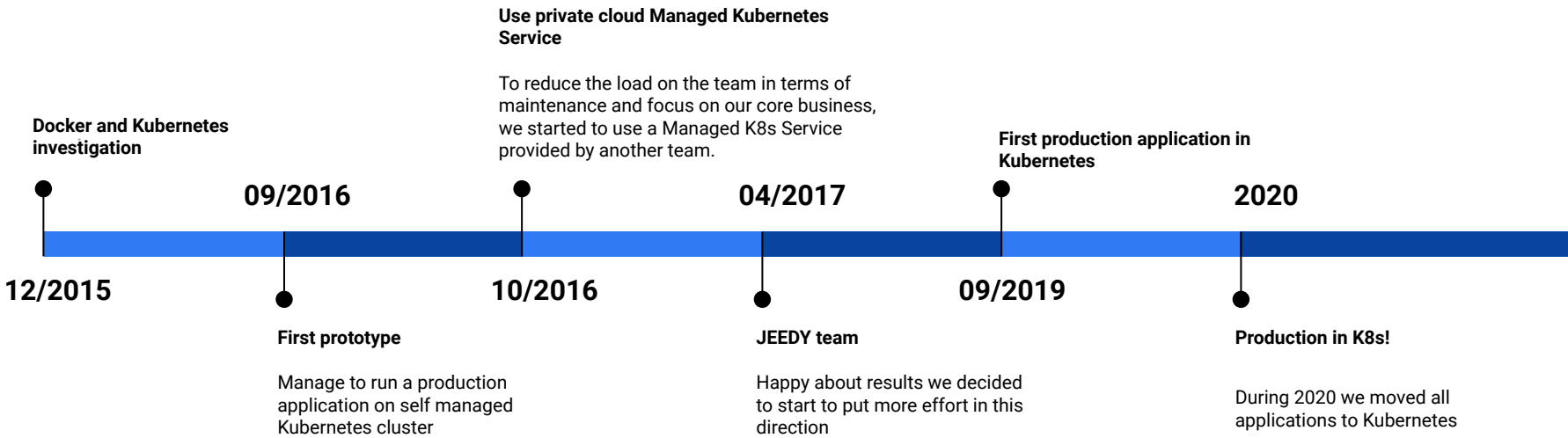
Few platform engineers VS many developers
- same company but different department (independent entities)
- developers' communities are heterogeneous

Only way to survive is automation

Infrastructure is hidden to developers
- no access to machines/pods
- no access to customization

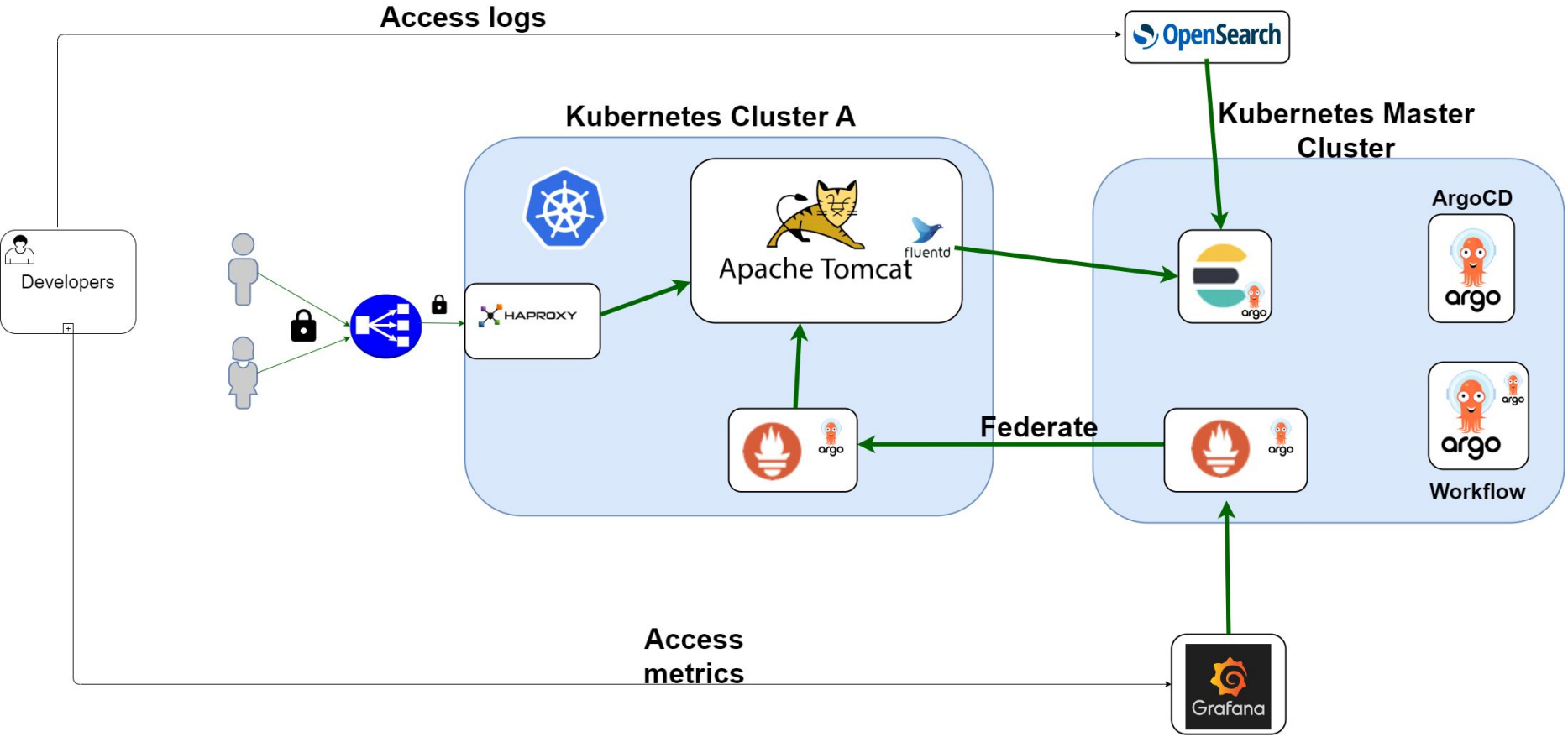


Timeline


Docker and Kubernetes investigation
12/2015
09/2016
First prototype
Manage to run a production application on self managed Kubernetes cluster
Use private cloud Managed Kubernetes Service
To reduce the load on the team in terms of maintenance and focus on our core business, we started to use a Managed K8s Service provided by another team.
10/2016
04/2017
JEEDY team
Happy about results we decided to start to put more effort in this direction
First production application in Kubernetes
09/2019
2020
Production in K8s!
During 2020 we moved all applications to Kubernetes

Architecture overview

Architecture

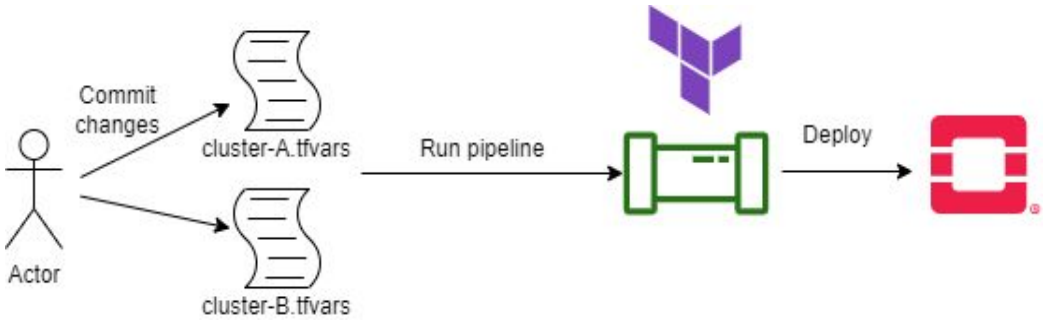


Provisioning of clusters

CERN private cloud based on OpenStack
- Magnum project provides Managed Kubernetes Service

Infrastructure as Code based on Terraform
- clusters' definition
- easy and fast spawn new clusters
- easy to track changes

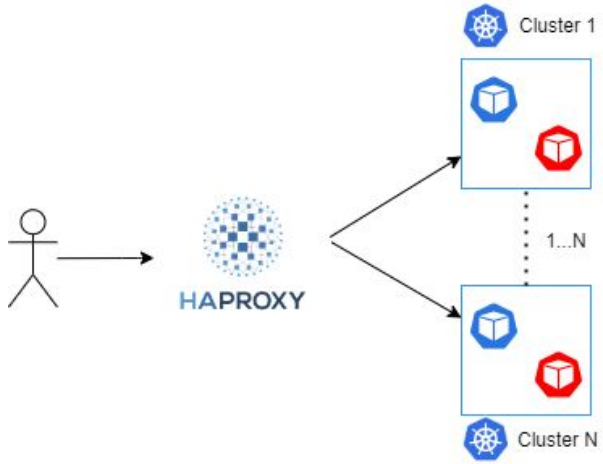
Evaluating alternatives:
- Crossplane but not available for OpenStack yet (even though something is moving)
- Cluster API



Cluster as cattle

Hosted Applications:

- deployed across different availability zones in multiple clusters
- cluster of load balancers to forward traffic across clusters
- each user community has its own set of clusters
- apps share data via DB



Cluster as cattle paradigm

\+ Easy to replace cluster

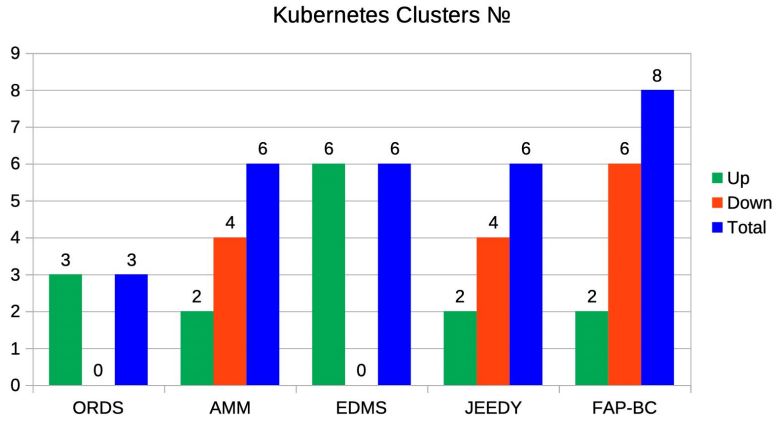
\+ User Isolation

\+ More resilient

- Improve DR/BC (accidentally tested!)

\- Maintenance overhead

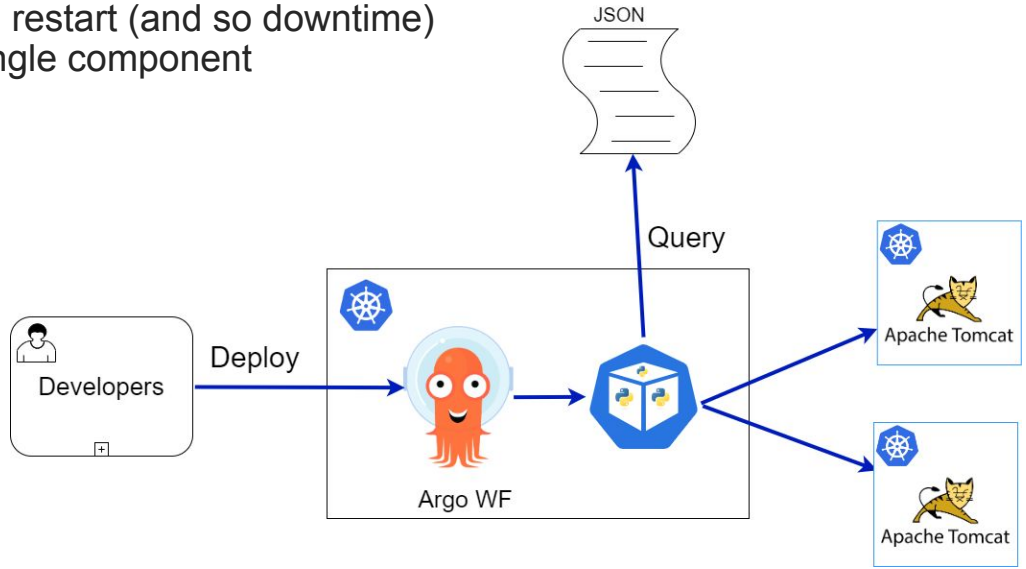
- Investigating virtual clusters



Applications deployment (old way)

Current configuration and deploy model describes Infrastructure as JSON

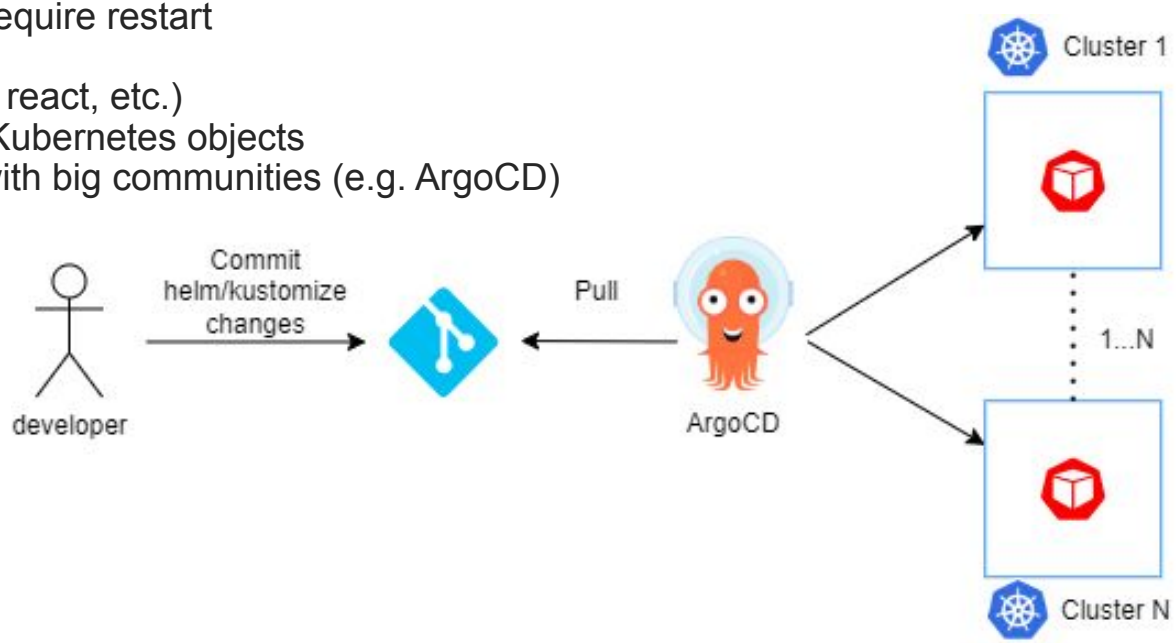
- clusters, applications, proxies as JSON objects
- one single source of truth
- custom python script to extract configuration and inject in Kubernete resource
- on-demand
- any configuration updates require a restart (and so downtime)
- platform engineers control every single component



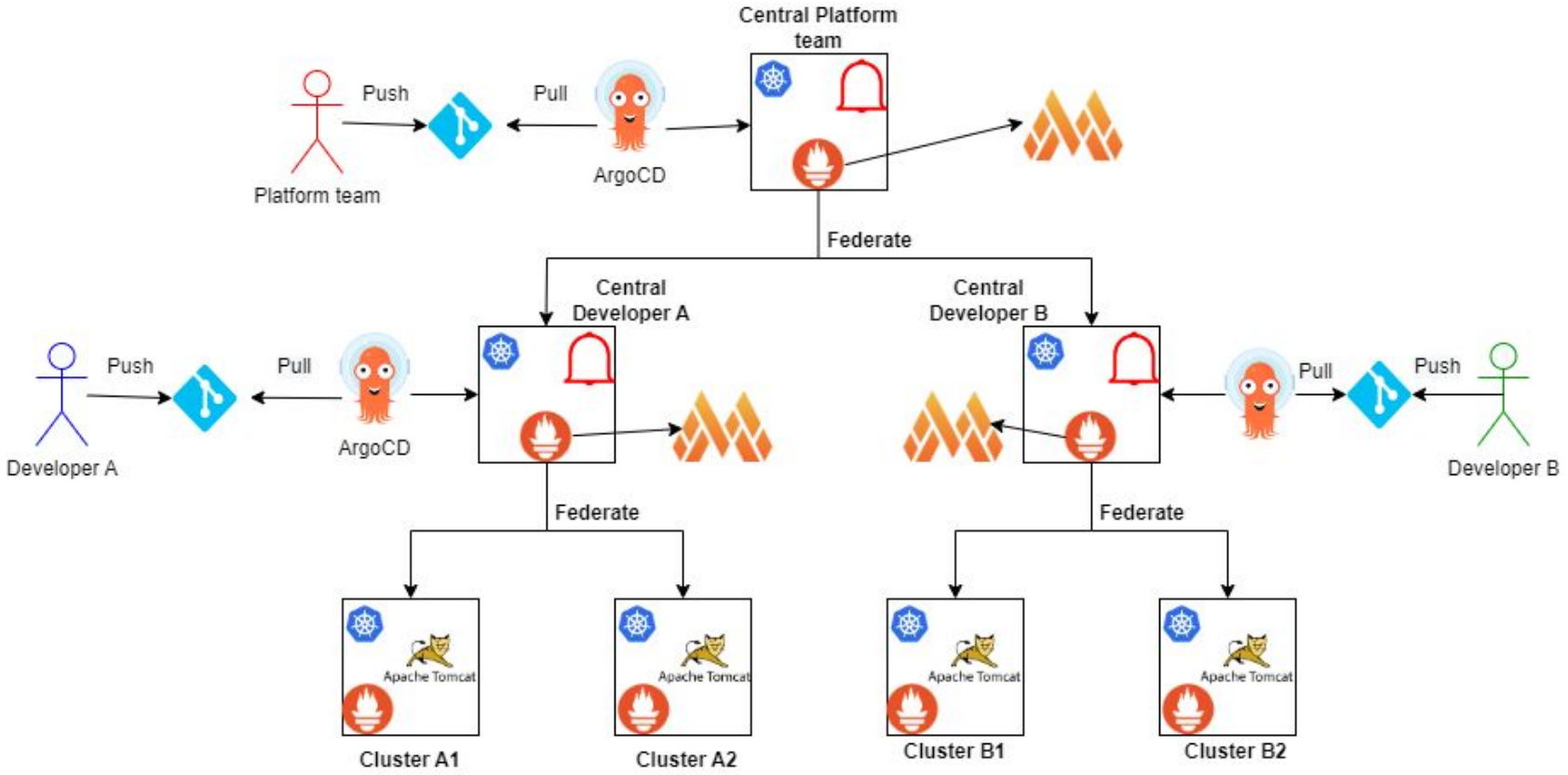
Applications deployment (new way)

Next generation deployment

- continuous reconciliation
  - minimal changes won't require restart
- git source of truth
- new technology stack (nodejs, react, etc.)
- developers have control over Kubernetes objects
- widely adopted technologies with big communities (e.g. ArgoCD)



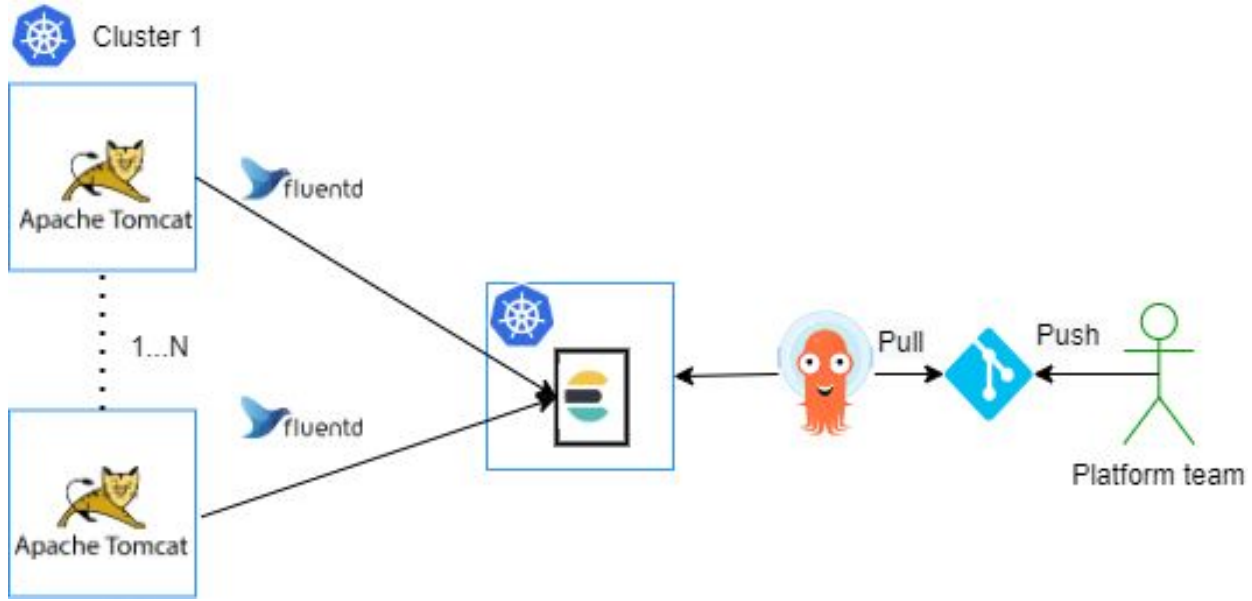
Monitoring system



Logging system

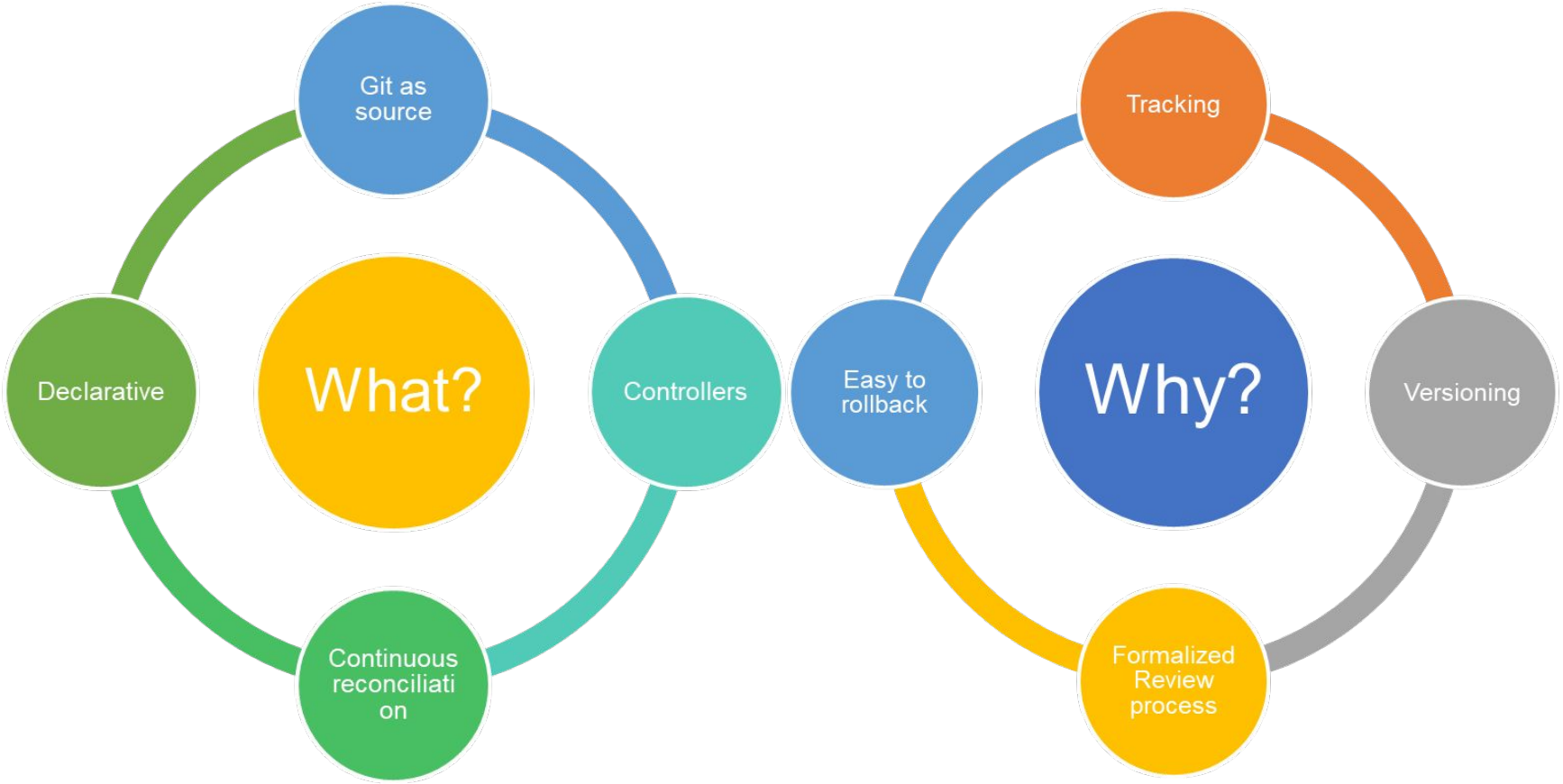
Looking at fluent-bit as replacement of Fluentd

- Still lack of grok support



GitOps

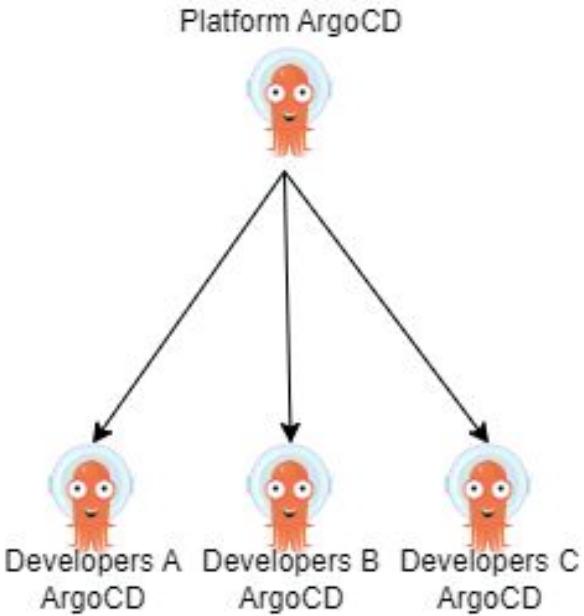
GitOps adoption



GitOps adoption

Current infrastructure

- central ArgoCD instance that manages:
  - monitoring infrastructure
  - logging infrastructure
  - jobs/cronjobs infrastructure (based on Argo WF)
  - developers' ArgoCD instances
  - itself



- developers' ArgoCD instances
  - manage user customizations and applications
    - recording and alerting rules
    - helm chart for applications

GitOps choices

No golden rules to structure repositories

Multiples repositories

+ easy to isolate use cases and users
- not easy to follow which repository defines what

Single branch

+ changes in only one place
- wrong merge could kill whole service

No best tools

- K8s standard tools
- Custom scripts
- Combination of both

```
gitops/
├── ais-users
├── argocd-jeedy-applications
├── argocd-jeedy-sources
├── argo-wf-jeedy-sources
├── authzsvc-users
├── backend-sources
├── eam-users
├── frontend-sources
├── gatekeeper-applications
├── ingress-controller-sources
├── jeedy-users
├── keycloak-operator-sources
├── logging-components-sources
├── maintenance-tools
├── prometheus-applications
└── prometheus-sources
```

```
gitops/prometheus-sources
├── jsonnet
│   ├── alertmanager
│   ├── grafana
│   ├── libs
│   ├── prometheus-operator
│   ├── prometheus-operator-remove-default
│   ├── prometheus-operator-secrets
│   ├── prometheus-operator-storage
│   └── pushgateway
├── kustomize
│   ├── alertmanager
│   ├── grafana
│   ├── prometheus
│   └── prometheus-operator
├── README.md
└── yaml
    └── grafana
```

GitOps challenges 1

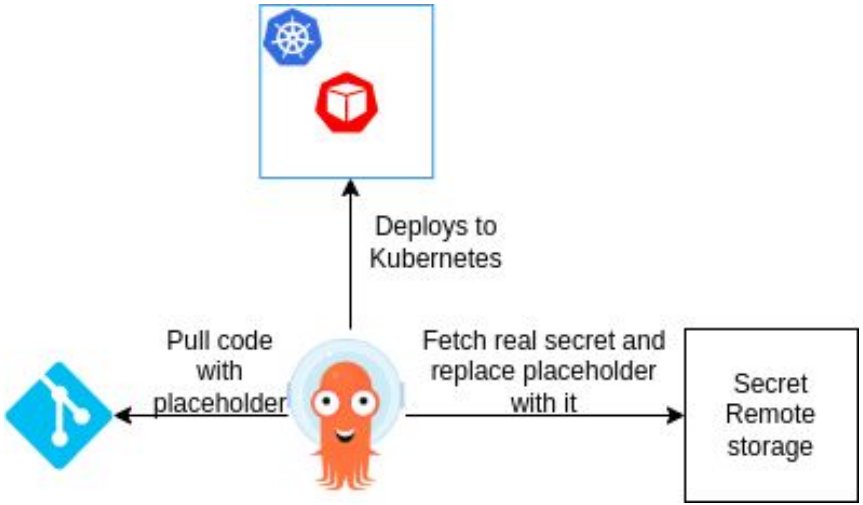
Secrets Management:

- in git (encrypted)
  - \+ close to other data
  - \- maintenance burden
    - key rotations
    - may require additional infrastructure (e.g. Sealed Secrets)
    - a commit for same secret in each repo
- in external store (git contains only place holders)
  - \+ Delegate operations to another expert team
  - \+ Change password in only one place
  - \- External dependency

GitOps challenges 2

Our solution

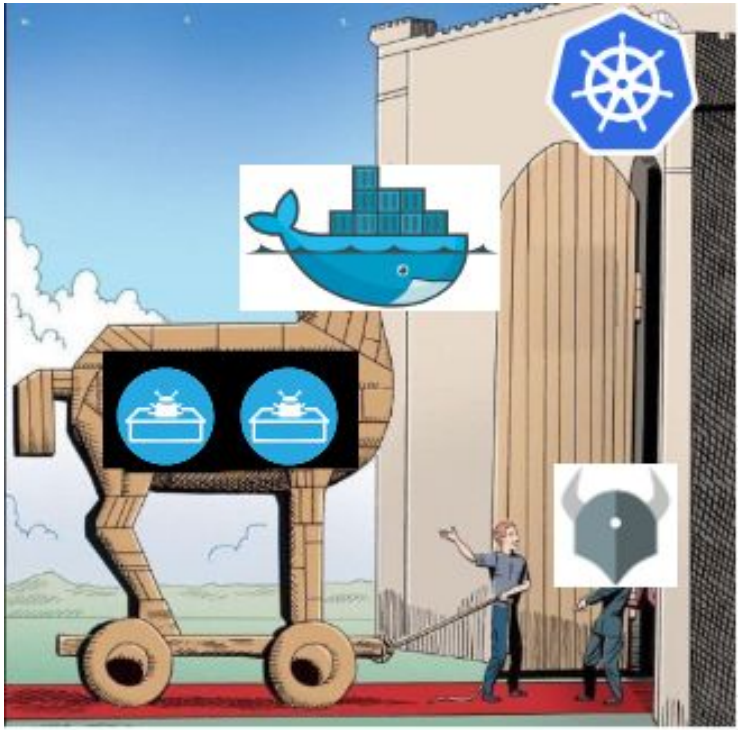
- Based on secrets stored in external store (CERN teigi)
- Extended ArgoCD Vault plugin with support for CERN teigi
- Evaluating alternative:
  - to move secret to Vault
  - Missing in CNCF landscape a secret management system



GitOps challenges 3

Security

- empower developers without loosing control
  - developers have more access to the infrastructure
    - provide their own images
- balance between operations and user experience
- way to mitigate
  - define policies with developers
  - force those policies with Policy Agent
    - Open Policy Agent (i.e. Gatekeeper)
      - evaluating to replace with Kyverno
  - cluster as cattle paradygm



GitOps challenges 4

Full picture

- not easy to follow which repository defines what
  - onboard new people can become a nightmare
  - resources may be defined in multiple repositories
- trying to mitigate
  - naming convention
    - like Kubernetes label
  - extract info from ArgoCD

Conclusions

Feelings after few years of production

	Before Kubernetes	After Kubernetes
Operations	Burden by simple and repetitve tasks	Increased efficiency and ability to focus on other projects
New app deployment	Days	Hours
Stuck application	May require human intervention	Liveness/Readiness probes
Configuration/Tracking	May diverge due to human interaction	Always aware of what we are running
Automation	Plenty of custom scripts	Adoption of multiple CNCF tools
User flexibility	Minimal	Delegate simple actions to developers
Disaster Recover/Business Continuity	Days/weeks	Few hours

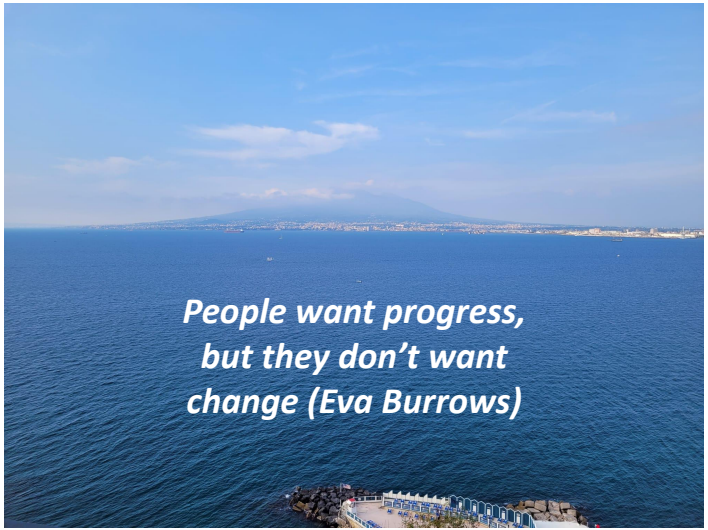
Take away

Extremely happy of our (not finished) journey
More reliable service
Kubernetes helped to

- increase platform team productivity
  - shift attention from simple tasks to new projects and ideas
- replace ad-hoc solutions with standard approaches
  - easier attract new people
  - not reinventing the wheel

Not easy

- documentation is not always at same level of the code
- breaking changes between different versions
- lot of effort in:
  - convince developers/colleagues to move
    - why change something that is working ?
  - making changes painless
    - implementing them in phases



Thank you

Thank you for your attention

Please scan the QR Code above
to leave feedback on this session