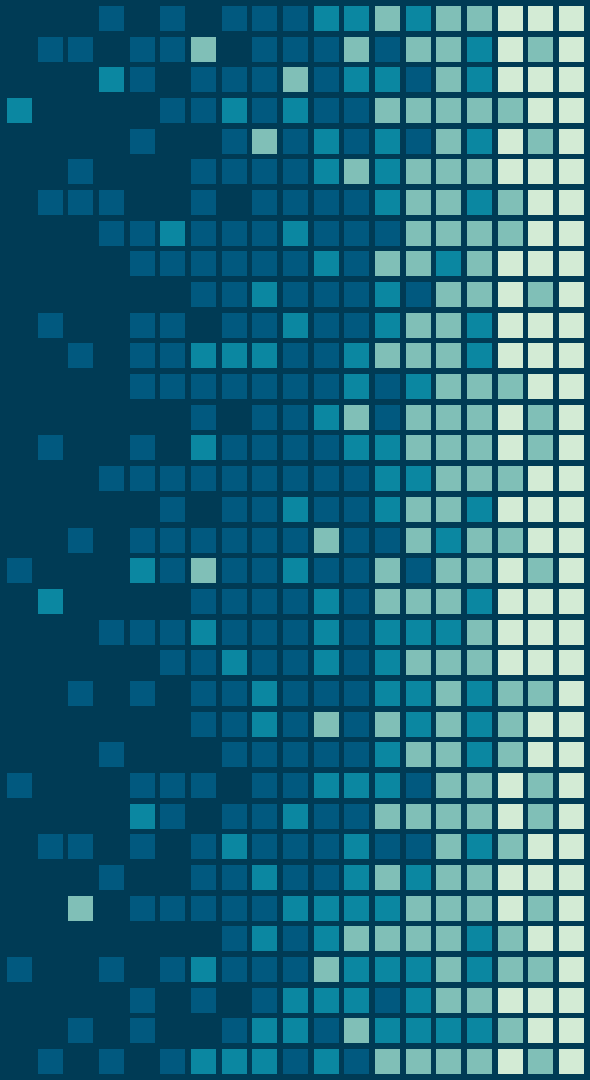
Adopter pleinement GitOps

Lucien Boix



ConFoo.CA
DEVELOPER CONFERENCE

Lucien Boix

- 38 ans, franco-canadien
- Je viens de Lyon (France)
- Je vis depuis 2013 à Montréal (Canada)
- Je suis Spécialiste DevOps (SRE)
  - à 360.Agency

Fondée en 2010

- 51ème employé en 2013 (moi) -> 250 en 2024
- Suite d'outils pour les concessionnaires automobiles
  - CMS, module complet de vente en ligne, CRM, etc.
- 473 clients, 906 websites
- Partenariat avec Kijiji Autos lancé en 2020
- Certifications avec une dizaine de manufacturiers

Le plan

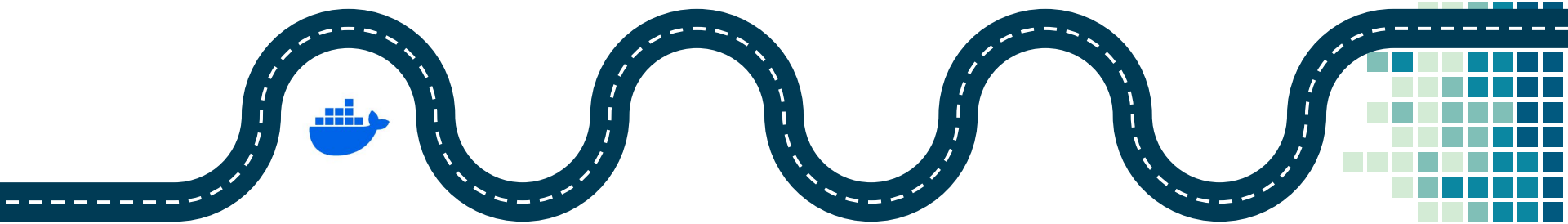
- **Timeline (de Docker à GitOps)**
- **Introduction à Kubernetes (k8s)**
- **Les 4 piliers de GitOps**
- **GitOps en pratique**
- **Retour d’expérience à 360 Agency**
- **Conclusion / Takeaways**
- **Aller plus loin**

Timeline (de Docker à GitOps)

Timeline

Création de **Docker**

2013



La révolution des containers : éphémères, portables, developer friendly

Timeline

Création de **Docker**

2013

La révolution des containers : éphémères, portables, developer friendly

Création de **Kubernetes**

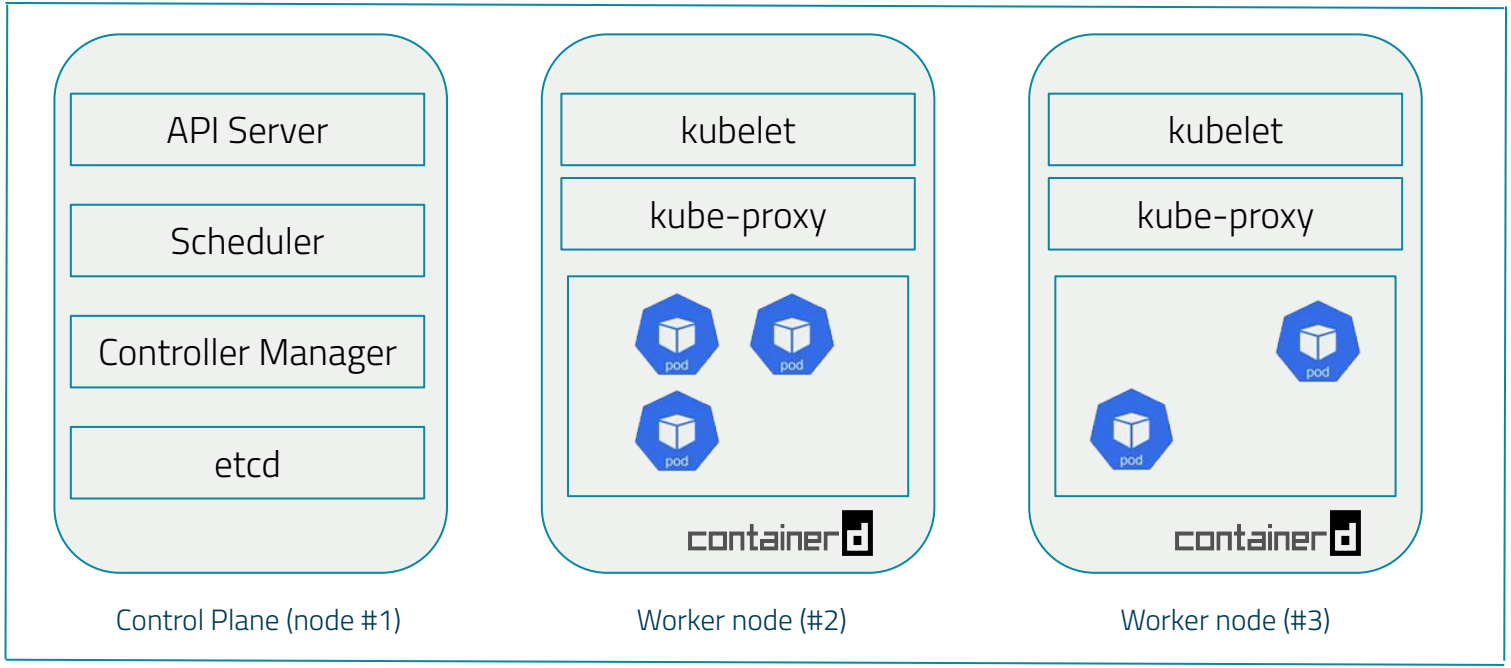
2014

Open source
Orchestrateur de containers
Le plus populaire

Cluster self-managed ou
EKS (Amazon)
GKE (Google)
AKS (Microsoft)

Introduction à Kubernetes (k8s)

Schéma d'un cluster Kubernetes



Kubernetes : gestion des pods

- Commande **kubectl**
  - Pour les créer
  - Ou appliquer un fichier **.yaml**
    - *Yet Another Markup Language*
    - On y décrit simplement ce que l'on veut

Kubernetes : schéma d'un fichier **.yaml**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: &appName microservice-vehicles
  namespace: prod
spec:
  selector:
    matchLabels:
      app: *appName
  replicas: 2
  template:
    metadata:
      labels:
        app: *appName
    spec:
      containers:
        - name: *appName
          image: dockerhub/microservice-vehicles:release-1171-d6884731
          ports: [{containerPort: 80}]
          resources: {requests: {memory: 128Mi, cpu: 100m}, limits: {memo
          env:
            - {name: APPLICATION_TOKEN, valueFrom: {secretKeyRef: {name:
            - {name: LOG_FORMAT, value: "json"}
            - {name: DD_VERSION, value: "release-1171-d6884731"}
```

kubectl create Deployment ...
kubectl apply -f file.yaml

Kubernetes : les challenges à ses débuts

Au niveau des déploiements (changer le nom de l'image)

- **kubectl** manuel
  - error prone, pas de review / approval
  - parfois besoin de ping une autre équipe (bottleneck)
- Ou scripts de déploiement (rollout)
  - à maintenir
  - accès au cluster (pas idéal)
    - URLs et structure interne du cluster exposées

Kubernetes : les challenges à ses débuts

Au niveau des changements

- Aucun audit
  - Exemple : feature toggle, tests dans un coin
    - comment être sûr que rien n'a bougé ?

- Rollback
  - OK mais quoi exactement ?

Timeline

Création de **Docker**

2013

La révolution des containers : éphémères, portables, developer friendly

Création de **Kubernetes**

2014

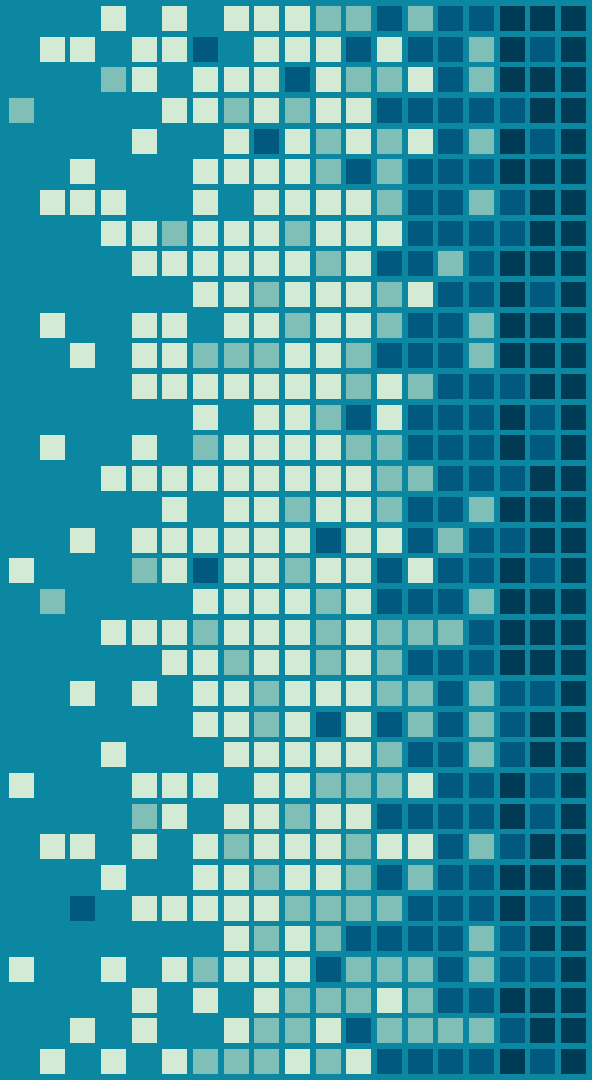
Open source
Orchestrateur de containers
Le plus populaire

Cluster self-managed ou
EKS (Amazon)
GKE (Google)
AKS (Microsoft)

2017

Le terme "**GitOps**" apparaît en 2017 dans un blog post de Alexis Richardson (cofondateur et CEO de Weaveworks, fermée en 2024)

> *It's a methodology for developer tooling to drive operations. This post discussed the use of declarative tools and best practices of configurations being code and therefore should be version controlled.*



Anecdote en 2016 chez Weaveworks

- Changement risqué planifié
- Clusters Kubernetes wiped dans AWS
- Toute la stack a été remontée en 45 minutes
- Toutes les configurations étaient dans Git
  - Ils se forçaient à le faire
    - 1 changement en prod = 1 commit

- Pourquoi ne pas l'automatiser ?

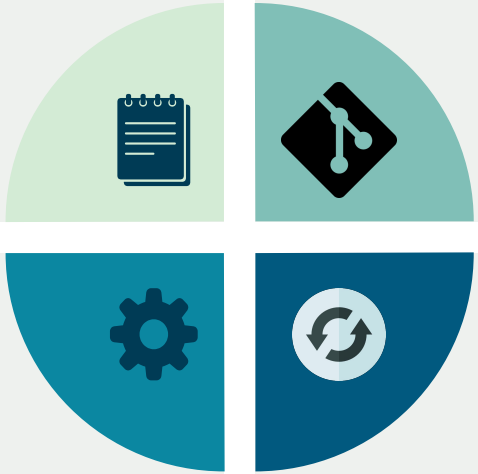
Les 4 piliers de GitOps

GitOps Principles v1.0.0 (lien) en 2021

Les 4 piliers de GitOps

Déclaratif

L'état désiré du système est décrit de manière déclarative



Versionné & immutable

L'état désiré du système est versionné dans Git

Les changements approuvés sont automatiquement appliqués au système

Pull automatique

Un agent s'exécutant à l'intérieur du système détecte et corrige le drift

Réconciliation

1. Déclaratif

L'état désiré est décrit de manière déclarative :

- Différent de impérative
  - instructions précises à suivre dans un ordre
- Exemple : 5 commandes **kubectl** VS 1 fichier **.yaml**

- L'outil GitOps s'assure d'arriver à cet état désiré
  - Peu importe les étapes suivies

2. Versionné et immutable

- Git est le choix naturel : tout le monde connaît
- Source de vérité (audit natif)

- Cohérence
  - Algorithme de hachage SHA
  - Checksum sur chaque fichier indexé
  - 1 commit = 1 point dans le temps

3. Pull automatique

- Différent du push deployment (comme **kubectl**)
- Pull deployment
  - L'agent GitOps pull le repo continuellement
  - Si changement (nouveau commit)
  - Alors il les applique au système (cluster k8s)

- Rollback facile donc (1 revert)

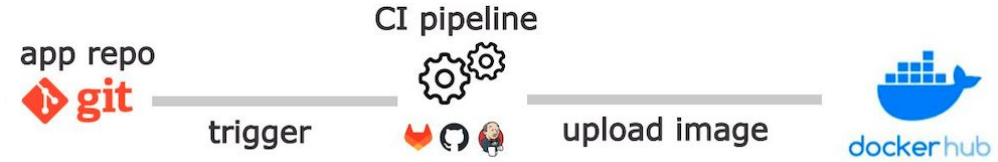
4. Réconciliation

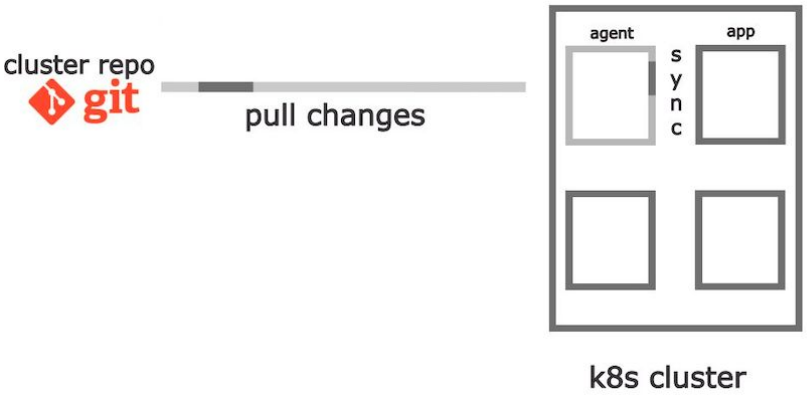
- L'agent GitOps check en continu
  - L'état désiré (Git) VS l'état observé (cluster k8s)
  - Si drift constaté
    - On revient sur l'état désiré

  - **kubectl** manuel ?
    - Il sera écrasé peu de temps après

GitOps en pratique

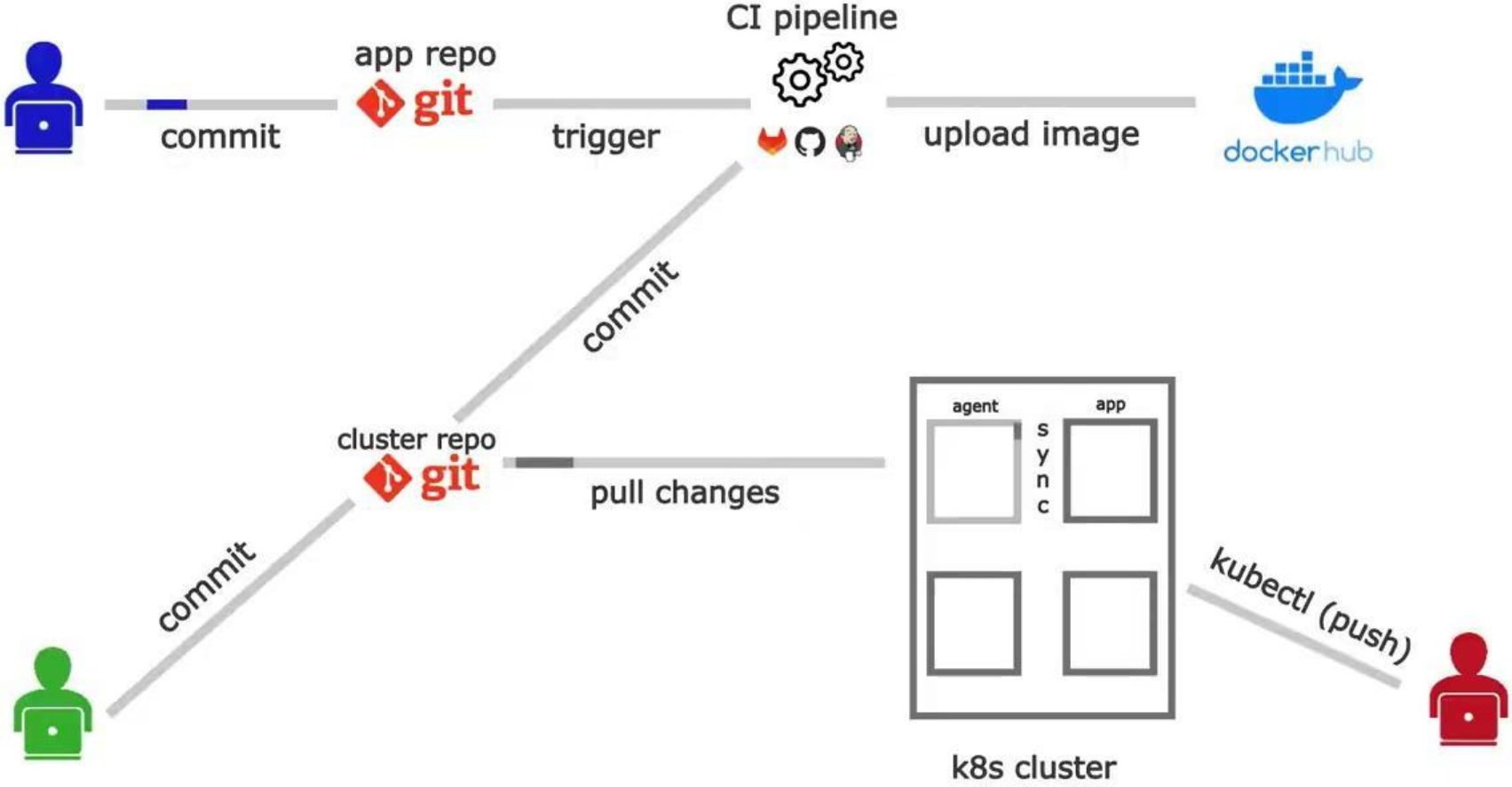
Voici le scope de notre exemple





3 collègues travaillent sur cette app

- veut déployer une nouvelle version de l'app
- veut tuner les ressource allouées
  - un peu plus de memory limit
- veut changer une variable d'environnement
  - désactivation de la cache pour un test


CI pipeline
app repo
git
commit
trigger
upload image
docker hub
commit
cluster repo
git
pull changes
agent
app
sync
commit
kubectl (push)
k8s cluster

If previous embedded video is broken or does not play, use this link:

https://www.youtube.com/watch?v=_m_-TWw6i2A

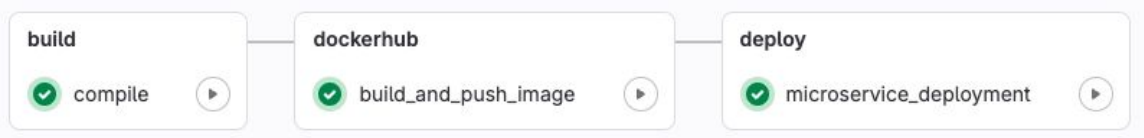
Retour d'expérience à 360 Agency

Migration à Kubernetes en 2018

Déploiements avant GitOps

Push deployments :

- Le script du pipeline appelle un microservice maison
  - Qui **kubectl** le cluster
    - Pour changer le nom de l'image à rouler
- Il attend la fin du rollout des pods (deploy vert = live)



Quelques outils GitOps pour Kubernetes

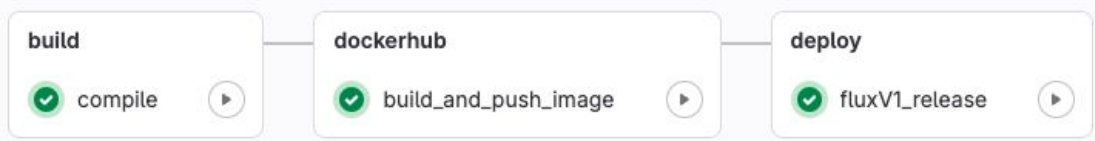
- FluxCD (open source)
  - créé par Weaveworks
  - donné à la CNCF par la suite en 2019
    - *Cloud Native Computing Foundation*
      - containerd, etcd, k8s, CoreDNS, Helm, etc.

- ArgoCD (open source)
- Jenkins X (open source)

2020 : premiers pas dans GitOps (FluxV1)

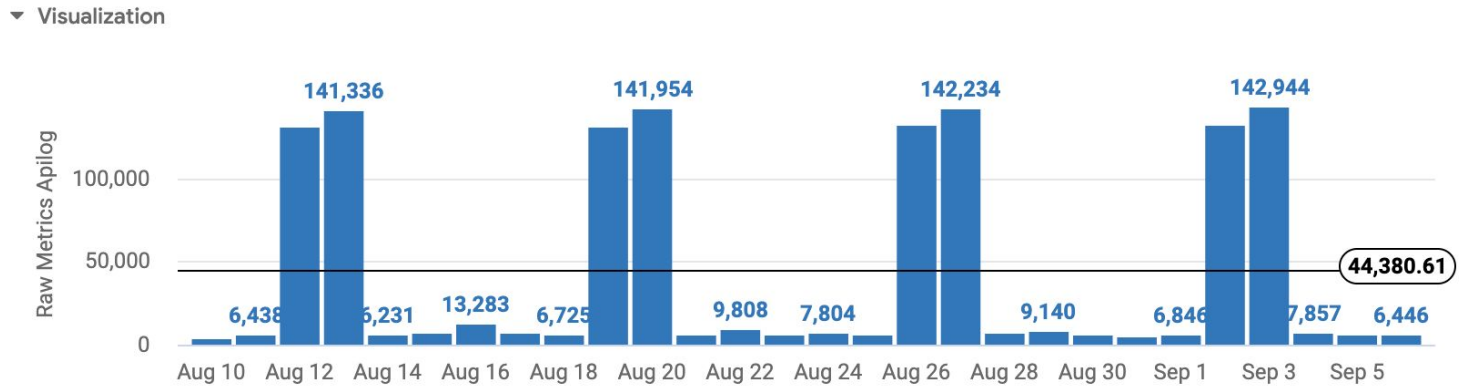
La fin du pipeline appelle un script

- Qui utilise le CLI de FluxV1 : fluxctl release
  - Commit / push de la nouvelle image
  - L'agent détecte et applique le changement
- Il attend toujours la fin du rollout des pods (vert = live)



2022 : courriel reçu de Docker

- “Votre compte fait en moyenne 44 000 pulls / jour”
- Le max autorisé est 5000 sinon il faut un add-on
  - 32 950 USD par an

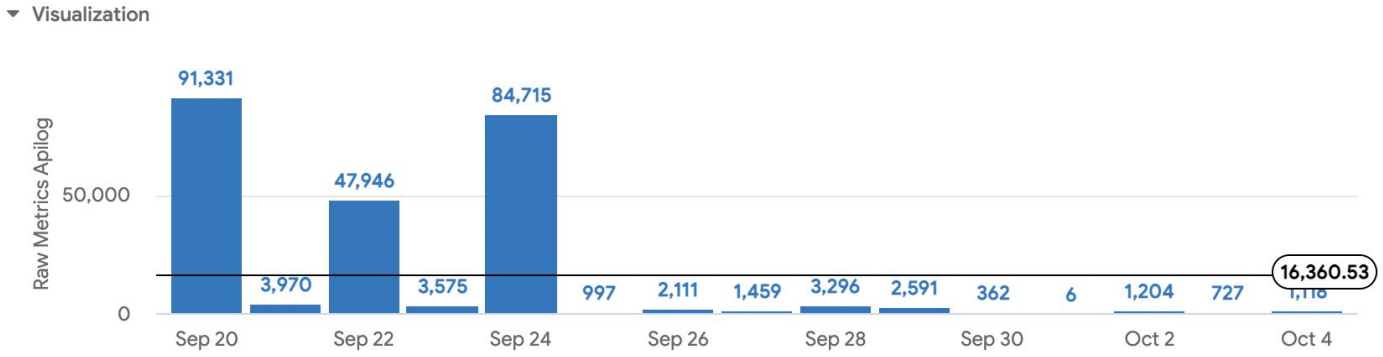


2022 : courriel reçu de Docker

- L'agent (pod flux) scan le Dockerhub en continu
  - Et stocke la liste dans un Memcached (2nd pod)
- Il fallait que l'agent ait vu l'image dans Dockerhub
  - S'il n'a pas vu l'image, le fluxctl release échoue
- C'était annoncé de leur part : version 1 deprecated

2022 : courriel reçu de Docker

- On désactive donc le “registry scanning”
  - L’agent gère toujours les changements
- Mais tous nos pipelines sont brisés
  - Il faut commit / push manuellement



2022 : réflexion sur un script maison

Sachant que :

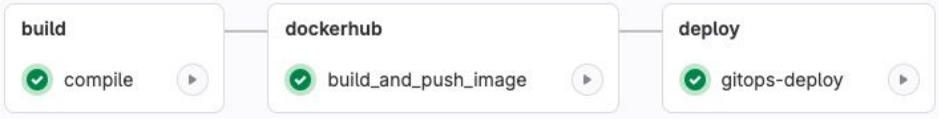
- L'image à déployer existe déjà dans DockerHub
- Notre script actuel
  - A accès au cluster
  - Est fortement lié à un CLI, deprecated en plus
    - À maintenir dans le futur (FluxV2, ArgoCD ?)

- On a besoin d'une approche plus générique

2022 : réflexion sur un script maison

Création de l'image gitops-deploy (open source)

- git clone puis commit / push du nom de l'image
  - dans le bon fichier .yaml
- aucune dépendance avec un CLI
- aucun accès au cluster

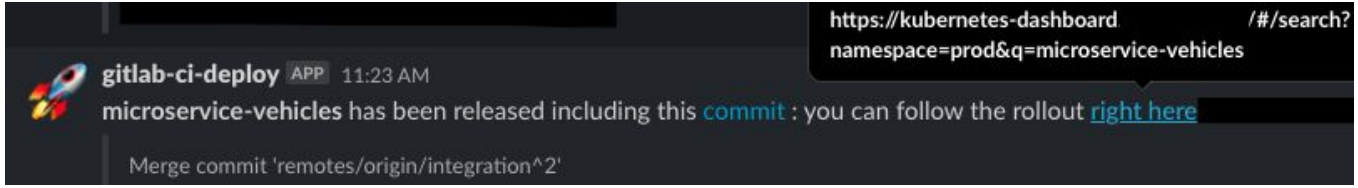



```
...  ...  @@ -22,7 +22,7 @@ spec:
22   22       spec:
23   23         containers:
24   24           - name: *appName
25        -        image: [redacted]/microservice-vehicles:release-1175-bc23d9df
     25   +        image: [redacted]/microservice-vehicles:release-1179-344b5907
26   26           ports: [{containerPort: 80}]
27   27           resources: {requests: {memory: 256Mi, cpu: 150m}, limits:
28   28           livenessProbe: {httpGet: {path: /health, port: 80, scheme
     failureThreshold: 10, timeoutSeconds: 5}
...  ...  @@ -41,7 +41,7 @@ spec:
```

2022 : réflexion sur un script maison

Compromis :

- Push fail si plusieurs déploiements live (1 retry suffit)
- Plus d'attente du rollout
  - développeurs habitués au vert = done
  - utile certes mais pas sécuritaire comme on l'a vu
- Plutôt une notification Slack avec un lien k8s dashboard



2023 : upgrade à FluxV2

- Meilleure architecture
  - On peut surveiller plusieurs repos (Sources)
    - Utile pour teamwork sur le même cluster
  - Notifications (Slack, Teams, Discord, etc.)

- Documentation abondante
  - Pour un test rapide : flux2-lite (open source)

Conclusion / Takeaways

Conclusion / Takeaways

- GitOps est une philosophie (meilleures pratiques) pour gérer, déployer, monitorer des containers

- Change également la façon de travailler en entreprise
  - Les développeurs ont le ownership du cycle de vie de leurs applications
  - On est obligés de communiquer (MR ou PR)
  - Collaboration (review), partage du savoir, cohésion

Conclusion / Takeaways

- Meilleure productivité
  - Moins d'erreurs, moins de coûts
  - Plus besoin d'impliquer une autre équipe pour déployer / rollback, - de procédures / runbooks

- Meilleure sécurité
  - Audit, tout est traçable
  - Permissions simples à gérer (GitLab, GitHub, etc.)

Conclusion / Takeaways

- Au final : bonne balance entre le contrôle et la vitesse de livraison

- Quelques bonnes pratiques
  - commit id dans le nom de l'image buildée
  - le pipeline n'a pas d'accès direct au système
  - le pipeline roule des tests de syntaxe
  - les secrets sont stockés dans un Vault

Aller plus loin

Aller plus loin

Auto-Image Update

- L'agent scan le registry suivant nos règles
  - Minor release pour une dépendance (nginx, etc.)
- Si nouvelle image uploadée, on la déploie
  - Consistera en un commit / push bien sûr

Aller plus loin

Explorer ArgoCD pour comparer

- + centré développeur que Flux (qui est + SRE)
  - UI out of the box

- Auto-sync ou bien il peut “retenir” les commits / push
  - Les sync peuvent être manuels
  - Ou appliqués pendant une maintenance window

Aller plus loin

Explorer l'Infrastructure As Code (**IaC**) en général

- Philosophie : plus aucun setup manuel
- Apporte fiabilité, visibilité, réduit les coûts

On peut gérer aussi avec un repo Git les changements :

- d'infra (instances) avec Terraform, Ansible
- réseaux (règles de routing, firewalls)
- d'accés (liste des comptes autorisés)
- de monitoring

Merci!

Avez-vous des questions?

N’hésitez pas à partager votre expérience : #confoo

https://www.linkedin.com/in/lucienboix/

@lboix

Crédits

Ce template de slides a été réalisé et mis à disposition gratuitement par SlidesCarnival