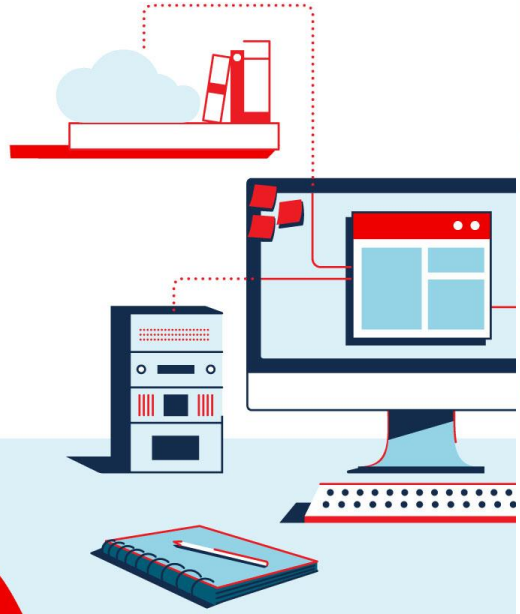


GitOps pour OpenShift

Martin Sauvé
Architecte principal
Mai 2020

Red Hat

GitOps

GitOps: l'état des applications et de l'infrastructure est en tout temps représenté dans un dépôt Git. Tout changement au dépôt est reflété dans l'infrastructure et les applications correspondance par des procédures complètement automatisées.

GitOps est une évolution naturelle au DevOps et aux méthodologies agiles

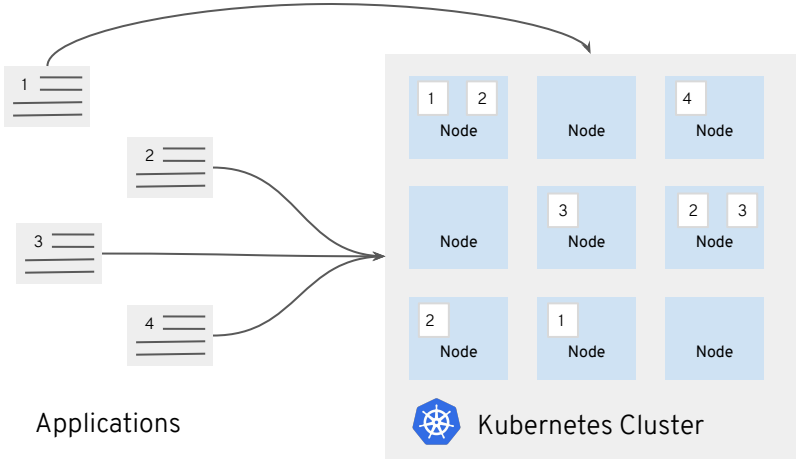
Pourquoi GitOps ?

- Tous les changements sont vérifiables et traçables.
- Processus standardisé pour retour en arrière ou déploiement d'une certaine version au besoin.
- Reprise en cas de désastre: Appliquer de nouveau tous les manifestes.
- Expérience utilisateur: Push et Pull Requests

KUBERNETES 101

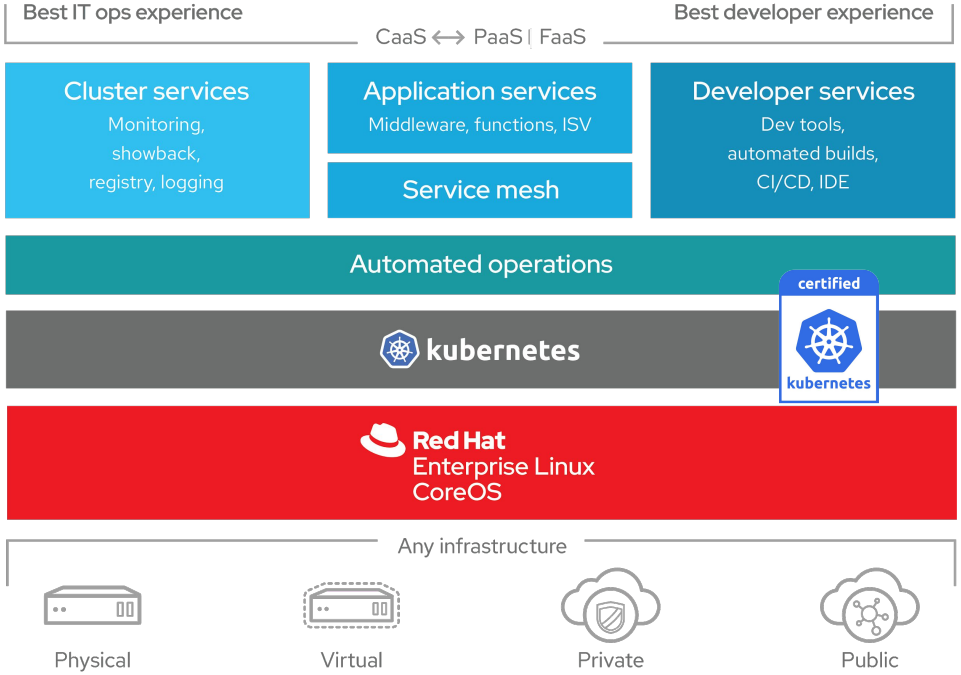
Kubernetes (K8s) est un ensemble de projets du logiciel libre pour automatiser, déployer et gérer les applications conteneurisées.

En gros…. C'est un céduleur de ressources….



Red Hat

OpenShift 4 - Une plateforme Kubernetes intelligente



Installation complète et automatisé: de la plateforme de gestion de conteneurs aux services applicatifs et opérations.

Déploiement automatisé de ressources Kubernetes: pour tous les clouds et dans vos centres de données.

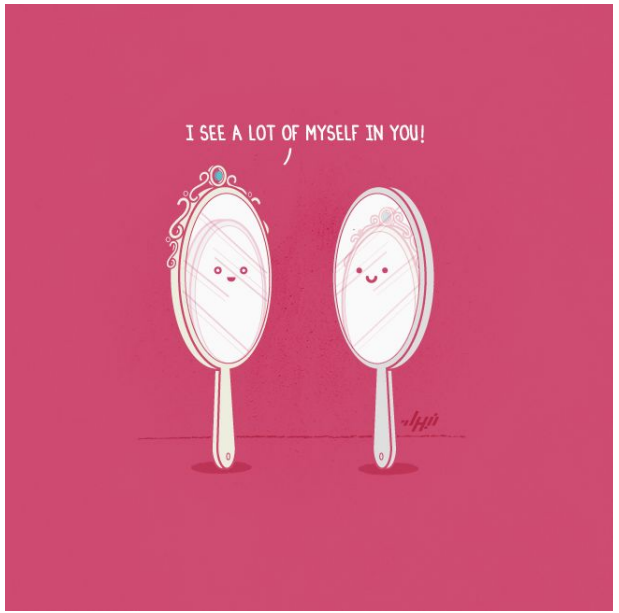
Auto Scaling: des ressources infonuagique

Mise-à-jour avec un bouton: pour la plateforme, les services et applications

Red Hat

OpenShift et GitOps

- OpenShift est un environnement déclaratif
  - La configuration est déclarée et les Operators l'appliquent
  - Les déploiements d'applications sont déclarés et le céduleur Kubernetes applique les déploiements.
- GitOps dans un environnement traditionnel demande du scripting. La nature déclarative d'OpenShift élimine ce requis.
- Les configurations pour les Operators sont de fichiers YAML facilement stockés et gérés par Git.
- Infrastructure-as-code, pipeline-as-code, configuration-as-code……..

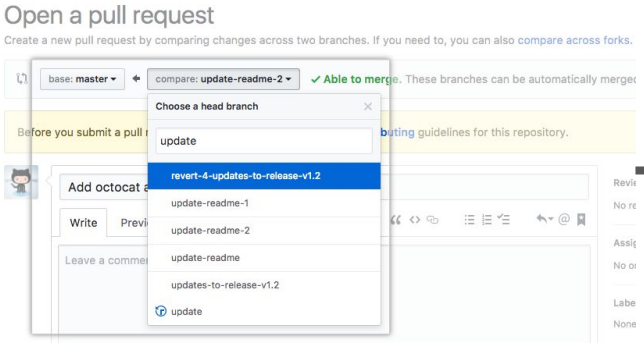


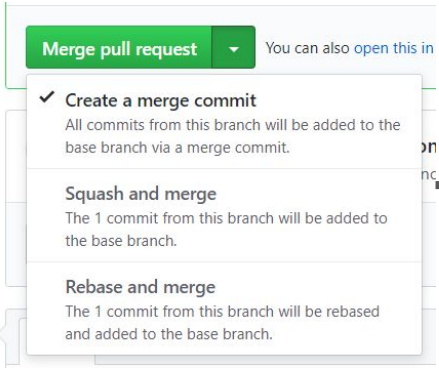
Red Hat

Principes GitOps pour OpenShift

- **Separation:** des manifestes (yaml) et du code source (Java/.Net/etc). Le comment et le quoi.
- Les manifestes sont des manifestes standards Kubernetes
- Éviter la **duplication** des fichiers yaml par environment.
- Les manifestes doivent être appliqués par un outillage **standard** Openshift et Kubernetes

Opérations (Jour 2) : Tous les changements à partir de Git






```
$ tkn pipelinerun logs update-from-master-run-g6s45

[run-kubectl] {"level":"info","ts":1580989837.3045664,"logger":"fallback-
logger","caller":"logging/config.go:69","msg":"Fetch GitHub commit ID from kodata failed:
\"KO_DATA_PATH\" does not exist or is empty"}
[run-kubectl] serviceaccount/demo-sa unchanged
[run-kubectl] clusterrolebinding.rbac.authorization.k8s.io/tekton-triggers-openshift-binding unchanged
[run-kubectl] eventlistener.tekton.dev/demo-event-listener configured
[run-kubectl] task.tekton.dev/deploy-from-source-task configured
[run-kubectl] triggerbinding.tekton.dev/update-from-master-binding unchanged
[run-kubectl] triggertemplate.tekton.dev/update-from-master-template unchanged
```

Red Hat

Outils

Argo CD

Kustomize

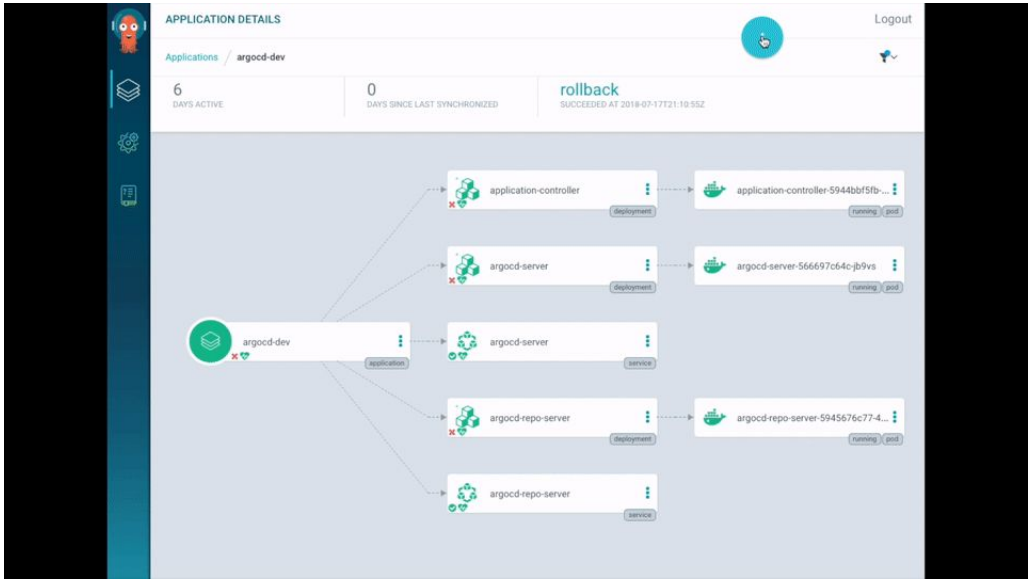
https://argoproj.github.io/argo-cd
https://kustomize.io/

Red Hat

Argo **CD** - Aperçu

Argo CD est un outil de livraison continue (CD) pour Kubernetes.

- Ressources Argo (Applications, Projets, Clusters) sont des ressources declarative Kubernetes
- Argo synchronize le cluster OpenShift avec des dépôt Git
- Compatible avec plusieurs outils Kubernetes
  - Helm
  - **Kustomize**
  - Ksonnet/Jsonnet
  - Directories of yaml
- **Pas un outil d'intégration continue (CI)**



Red Hat

Qu’est-ce qu’une application Argo ?

- Une configuration déclarative de type “Custom Resource” (CR) décrivant une application
- La configuration inclus:
  - Nom
  - Cluster
  - Dépôt Git
  - Politique de synchro.
- **CLI** ou interface utilisateur

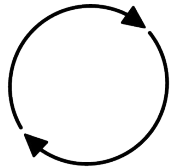
```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
 name: product-catalog-dev
 namespace: argocd
spec:
 destination:
  namespace: argocd
  server: https://kubernetes.default.svc
 project: product-catalog
 source:
  path: manifests/app/overlays/dev-quay
  repoURL: https://github.com/gnunn-gitops/product-catalog.git
  targetRevision: master
 syncPolicy:
  automated:
   prune: false
   selfHeal: false
```

Red Hat

Argo CD - Synchronisation

Changement dans Git

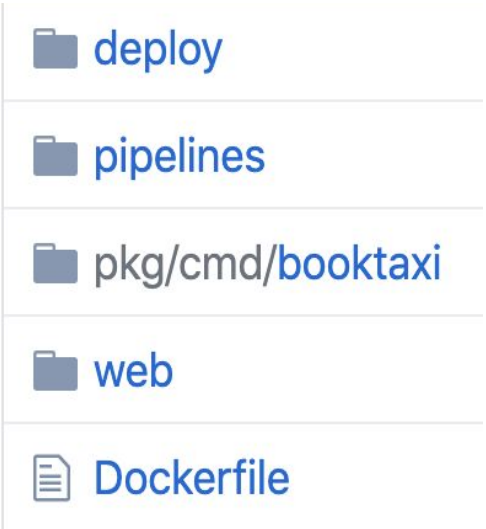
Réconcilliation



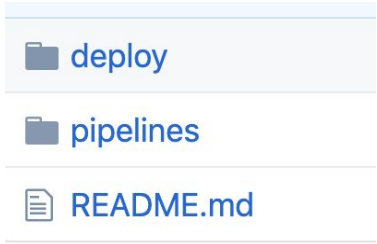
Synchronisation

Approche 1: Dépôts multiples

/taxi.git

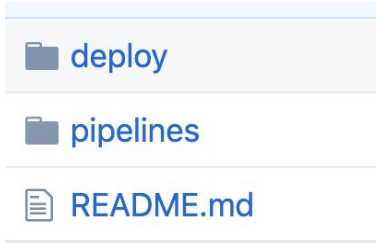


/taxi-config-**stage.git**



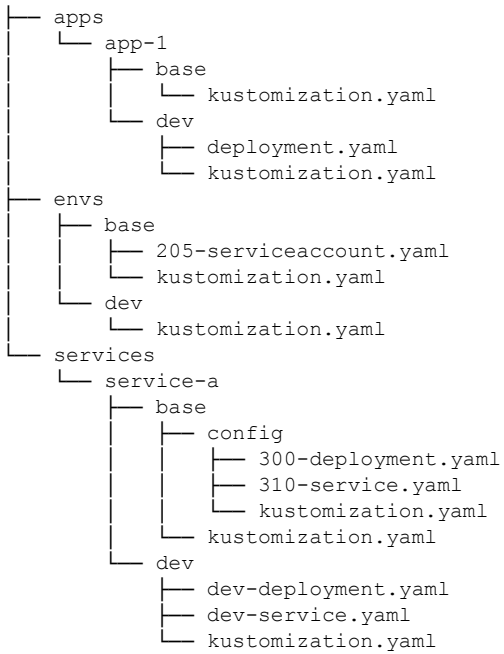
/taxi-config-**test.git**

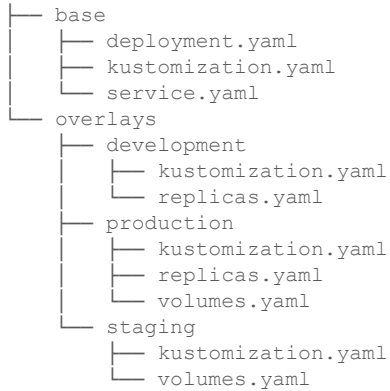
/taxi-config-**prod.git**

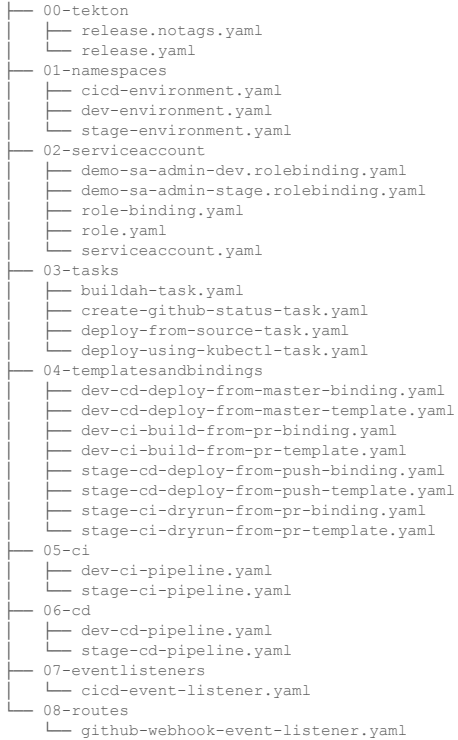


/taxi-config-**dev.git**

Approche 2 : Dépôt unique



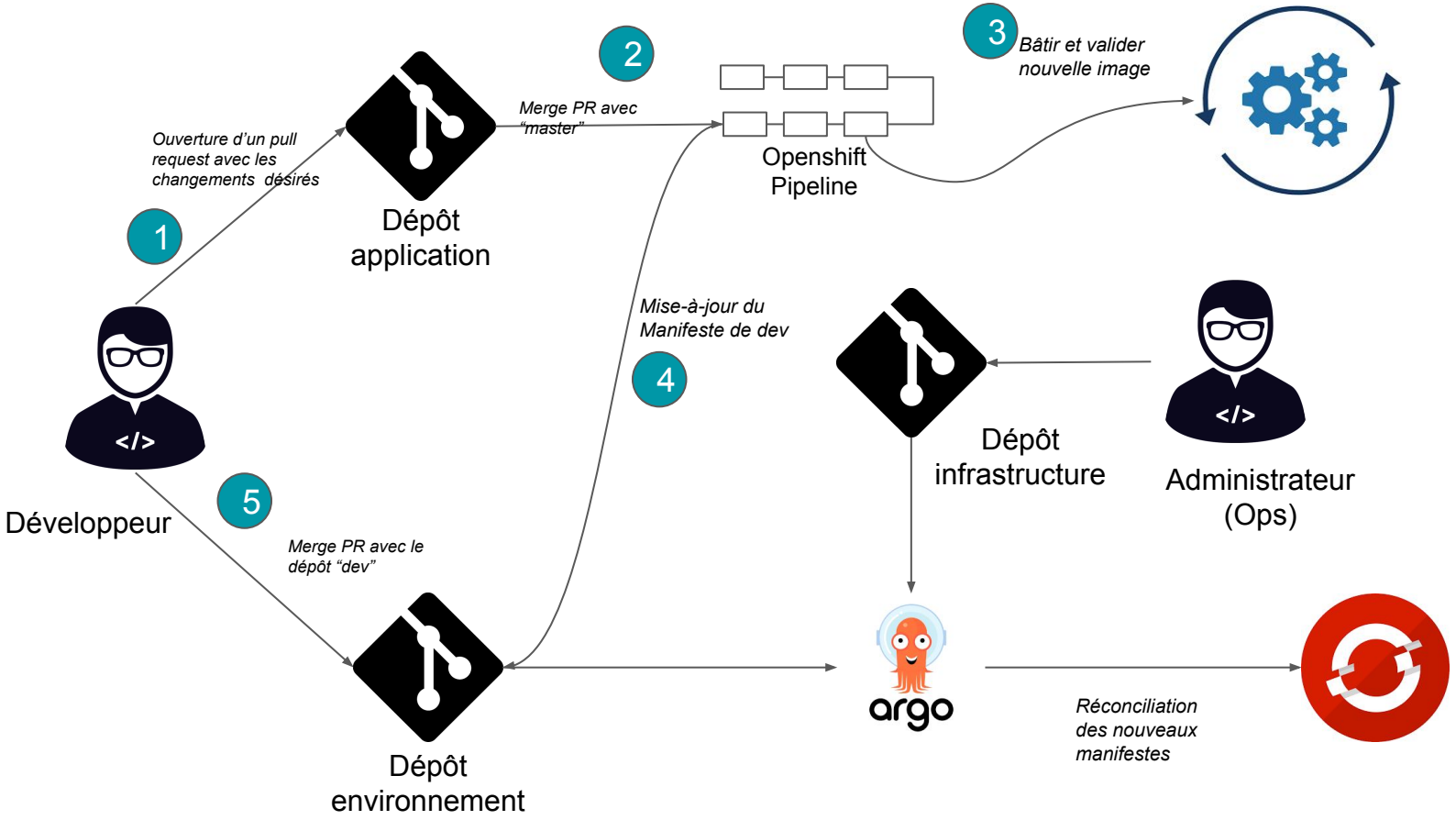




Argo CD - Gestion des secrets

Comment gérer les secrets (Encodage Base64 par Kubernetes) ?

- Utilisation d'une voûte externe.
- Encryption des secrets:
  - Bitnami Sealed Secrets
  - Mozilla SOPs/KSOPs
  - Plusieurs autres


1
Ouverture d'un pull request avec les changements désirés
Développeur
Dépôt application
2
Merge PR avec "master"
Openshift Pipeline
3
Bâtir et valider nouvelle image
4
Mise-à-jour du Manifeste de dev
Dépôt infrastructure
Administrateur (Ops)
5
Merge PR avec le dépôt "dev"
Dépôt environnement
argo
Réconciliation des nouveaux manifestes

Red Hat

Argo CD - Éviter la duplication

Argo CD permets le déploiement sur plusieurs clusters OpenShift!!!!

Comment gérer les manifestes sans dupliquer l'information partout ?

Red Hat

Kustomize - Aperçu

Kustomize permet de personnaliser les fichiers YAML sans mécanisme de gabarits (templates). Les YAML originaux sont utilisables directement

- Kustomize est un outils de ‘patches’
- Permet la modification de paramètres et configuration spécifique pour chaque environnement sans duplication.
- Contrairement à un approche par gabarits, tous les YAML sont utilisables directement.
- Kustomize est inclus avec kubectl et oc (à partir de la version Kubernetes 1.14)

```
oc apply -k apps/myapp/overlays/dev
```

Red Hat

Kustomize - Organisation

Kustomize est organisé en une structure de répertoires comprenant des “**bases**” et des “**overlays**”.

- Une **base** est un répertoire avec un fichier kustomization.yaml décrivant un ensemble de ressources.
    - Une base ne sait pas qu’il y a des “overlays”. Elle peut être utilisée avec plusieurs “overlays”
- Un **overlay** est un répertoire avec un fichier kustomization.yaml qui réfère à une base.
    - Un overlay peut avoir plusieurs bases et assembler des ressources de plusieurs bases.

https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/#bases-and-overlays

Kustomize

```
└── apps
    └── myapp
        ├── base
        │   ├── kustomization.yaml
        │   ├── service.yaml
        │   ├── route.yaml
        │   └── deployment.yaml
        │
        └── overlays
            ├── dev
            │   ├── patch-route.yaml
            │   ├── namespace.yaml
            │   └── kustomization.yaml
            └── test
                ├── patch-route.yaml
                ├── namespace.yaml
                └── kustomization.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
- service.yaml
- route.yaml
- deployment.yaml
```

```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

Namespace: dev
bases:
- ../../base
resources:
- namespace.yaml
patchesStrategicMerge:
- patch-route.yaml
```

Red Hat

Red Hat
Advanced Cluster Management
for Kubernetes

Red Hat Advanced Cluster Management for Kubernetes

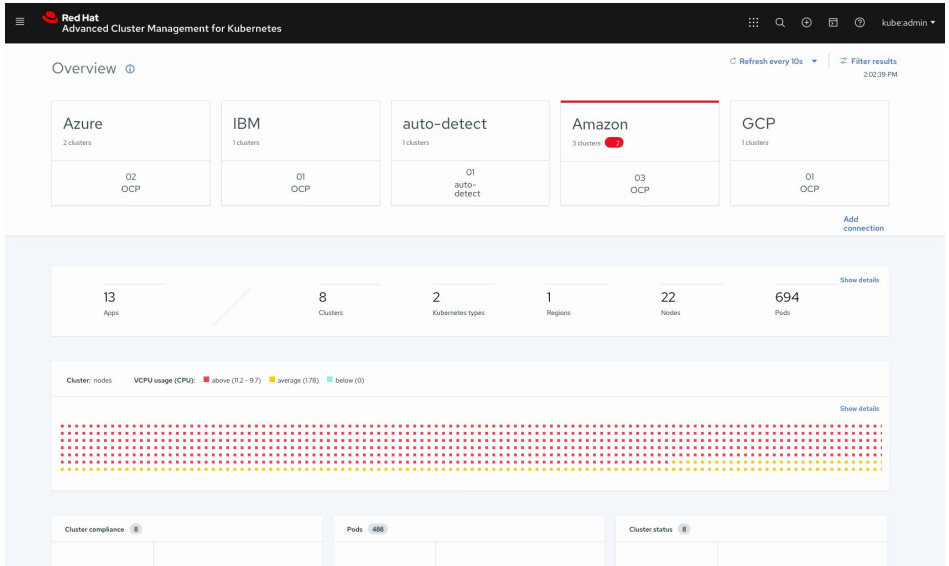
Gestion du cycle de vie multi-cluster

Aperçu

Operations TI

DevOps/SRE

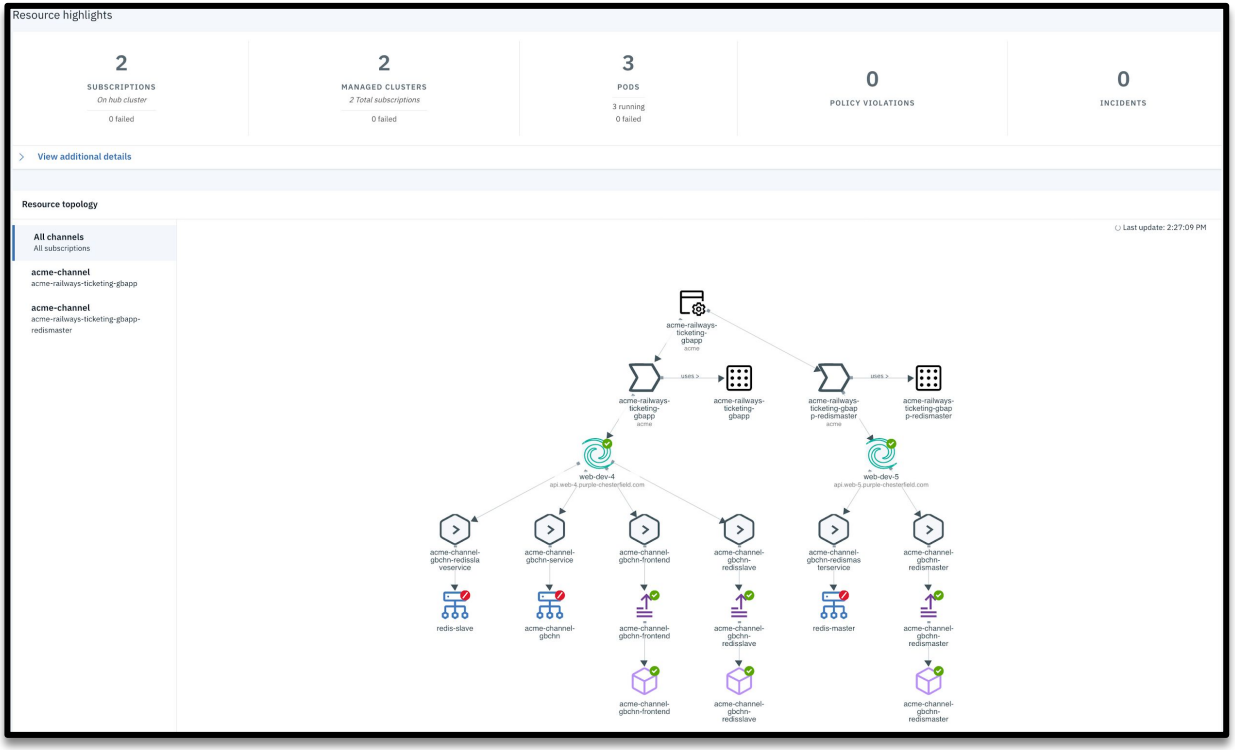
- Gestion de tout cluster Kubernetes
  - OpenShift 3.11, 4.1.x - 4.4.x
  - Cloud public: OCP
  - Kubernetes : EKS, AKS, GKE, IKS
- Recherchez, trouvez et modifiez les ressources kubernetes à partir d'un domaine de gestion.
- Gestion des TI en tant que code **YAML**
- Vue globale sur tous les clusters
  - Erreurs de configuration
  - État des Pods
  - Capacité et ressources
- Visualiser et résoudre les problèmes à partir du domaine fédéré
  - Tableau de bord, listes, tables
  - Balisage personnalisé
  - Régions
  - Cas d'affaire
  - Version



Red Hat

Gestion avancée du cycle de vie des applications

Simplification du cycle de vie



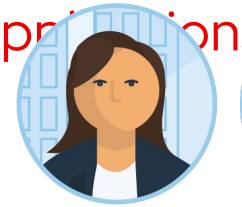
- **Déploiement** simplifié de vos applications sur plusieurs clusters
- Déploiement d'applications à partir de sources multiples
- Visualisez rapidement les relations entre les applications et clusters.

23

Red Hat

Gestion avancée du cycle de vie des applications

GitOps comme source unique de vérité



DevOps/SRE

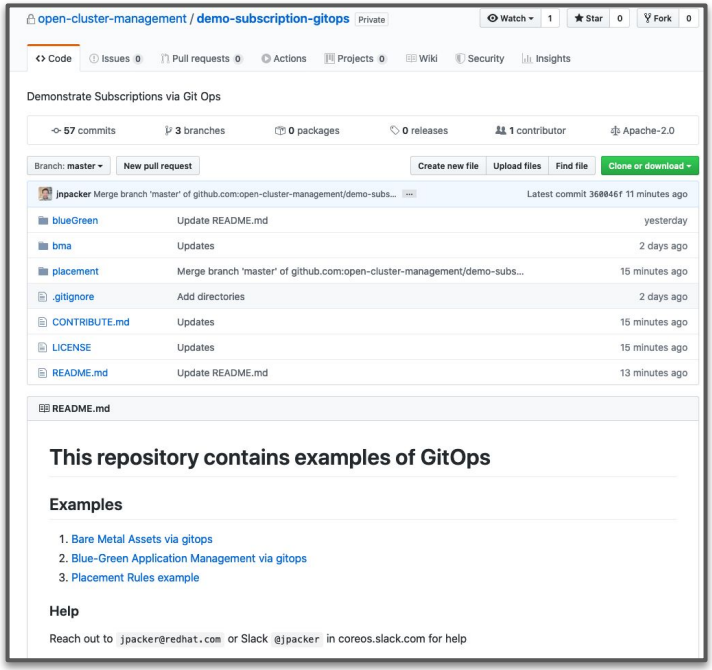
Operations TI

- Créez, modifiez et supprimez, comme vous le feriez pour n'importe quel code source. Git devient votre source de vérité contrôlant vos environnements Cloud ou OnPrem

- Ayez une liste de qui, quoi et quand pour chaque changement déployé dans vos environnements

- Grâce aux révisions et approbations de code, prenez le contrôle total de toutes les modifications apportées à environnements.

- Historique Git commit devient votre 'System of Record'



https://github.com/open-cluster-management/demo-subscription-gitops

Red Hat

Merci!

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Red Hat