metanext
by SII Group

Livre Blanc

Analyse Stratégique du Kubernetes as Code

Comparatif des approches Infra-Centric (Tanzu) et App-Centric (HyperShift) pour les Décideurs Techniques

Think I Build I Secure I Run the cloud

Introduction :
La Nouvelle Ère de la Gestion de Flottes Kubernetes

La croissance explosive des environnements multi-cloud et Edge rend les méthodes traditionnelles de provisionnement et de gouvernance des clusters Kubernetes de plus en plus insoutenables.

Alors que l'Infrastructure as Code (IaC) a déjà transformé la gestion des serveurs et des réseaux, une nouvelle évolution s'impose : **le Kubernetes as Code (KaC)**. Ce paradigme ne se contente pas de déployer des applications sur Kubernetes ; il traite le cluster Kubernetes lui-même comme une ressource déclarative, versionnée et entièrement pilotée par une API, au même titre qu'une application.

Deux philosophies dominantes incarnent aujourd'hui cette vision et proposent des approches architecturales distinctes pour y parvenir. D'un côté, l'approche "Infra-Centric" de **VMware Tanzu Supervisor**, qui intègre profondément Kubernetes dans l'hyperviseur. De l'autre, l'approche "App-Centric" de **Red Hat HyperShift**, qui traite le plan de contrôle du cluster comme une simple application conteneurisée.

L'objectif de ce livre blanc est de fournir aux architectes cloud, aux responsables de plateformes et aux décideurs techniques une analyse comparative approfondie de ces deux stratégies. En déconstruisant leurs fondements, leurs avantages et leurs implications opérationnelles, nous vous aiderons à identifier l'approche la plus alignée avec vos contraintes techniques, vos objectifs de gouvernance et votre stratégie cloud.

1.
Les Fondements du Kubernetes as Code (KaC)

Avant de comparer les solutions, il est essentiel de maîtriser les principes fondamentaux qui sous-tendent le Kubernetes as Code. Ce n'est pas simplement une nouvelle méthode d'automatisation, mais un véritable changement de paradigme. Le KaC repose sur un concept central : utiliser un cluster Kubernetes dédié, appelé Management Cluster, dont l'API sert à piloter le cycle de vie complet d'une flotte de clusters de production, les Workload Clusters.

Cette approche repose sur trois piliers technologiques.

L'API déclarative (ClusterAPI - CAPI)

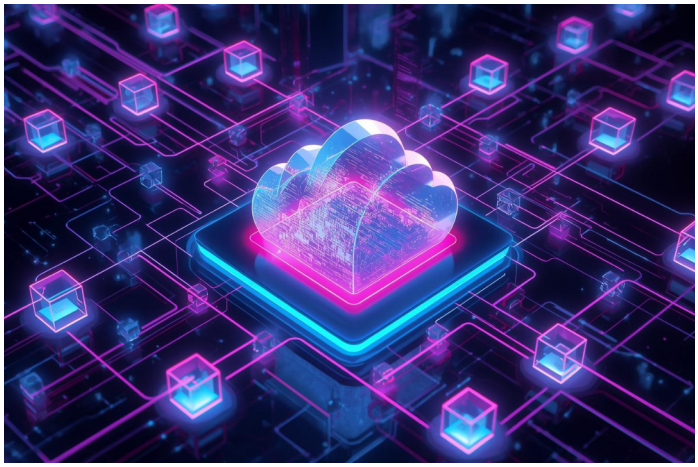
Le cœur du KaC est la capacité à définir un cluster Kubernetes entier via un simple fichier YAML. Grâce à des standards comme ClusterAPI (CAPI), le cluster devient un objet Kubernetes natif (par exemple, kind: Cluster). Toute sa configuration — version, taille des nœuds, réseau, stockage — est déclarée dans ce fichier. La gestion d'un cluster passe ainsi d'une série de commandes impératives à la maintenance d'un état désiré dans un fichier déclaratif.

Le GitOps Natif

L'association de l'API déclarative de CAPI avec des outils GitOps comme ArgoCD ou Flux représente l'aboutissement du KaC. En stockant les fichiers YAML de définition des clusters dans un dépôt Git, celui-ci devient la source unique de vérité pour toute l'infrastructure. Gérer une flotte de centaines de clusters se résume alors à modifier des fichiers et à soumettre des Pull Requests. L'infrastructure est traitée exactement comme du code applicatif, avec les mêmes bénéfices de versioning, de revue par les pairs et de traçabilité.

La Réconciliation Continue

Contrairement aux outils d'IaC traditionnels comme Terraform, souvent qualifiés de "Fire & Forget" (lancer et oublier), le KaC instaure une boucle de réconciliation permanente. Un contrôleur spécialisé sur le Management Cluster surveille en continu l'état réel de chaque Workload Cluster et le compare à l'état déclaré dans le fichier YAML. Si un écart est détecté, par exemple, un nœud est supprimé manuellement, le contrôleur intervient automatiquement pour le recréer et restaurer la conformité. Cette surveillance active garantit une résilience et une cohérence exceptionnelles de la flotte.



Maintenant que ces fondements théoriques sont posés, examinons comment ces principes sont mis en œuvre concrètement par les deux implémentations de référence : **Tanzu Supervisor et HyperShift.**

2.
Analyse des Deux Philosophies Architecturales

Bien que VMware Tanzu Supervisor et Red Hat HyperShift partagent l'objectif commun de fournir du "Cluster-as-a-Service", leurs philosophies architecturales fondamentales sont radicalement différentes. Elles ne représentent pas un simple choix d'outils, mais deux interprétations distinctes du KaC, chacune optimisée pour des contextes et des besoins spécifiques.

2.1. Tanzu Supervisor : L'Approche "Infra-Centric"

La philosophie de Tanzu consiste à fusionner le monde de la virtualisation traditionnelle avec celui de Kubernetes. Intégré nativement à vSphere, Tanzu Supervisor transforme l'hyperviseur lui-même en une API Kubernetes. Le Supervisor agit comme le Management Cluster, permettant aux administrateurs d'infrastructure de provisionner et de gérer des clusters Kubernetes de manière déclarative, en s'appuyant sur les primitives qu'ils maîtrisent déjà.

- **Architecture** : le Control Plane (API Server, Scheduler, etc.) de chaque cluster enfant est provisionné sous forme de machines virtuelles (VMs) dédiées sur l'infrastructure ESXi.
- **Intégration :** l'approche bénéficie d'une intégration profonde avec l'écosystème vSphere, notamment pour la gestion native du stockage et du réseau (NSX/VDS).
- **Le "Plus"** : l'avantage principal de cette approche est l'isolation forte. Chaque Control Plane étant encapsulé dans ses propres VMs, on obtient une séparation matérielle robuste entre les clusters. Cette caractéristique est cruciale pour les secteurs hautement régulés comme la banque ou la santé, où l'isolation par VM constitue une norme de sécurité éprouvée.

2.2. Red Hat HyperShift : L'Approche "App-Centric"

HyperShift adopte une approche purement cloud-native en traitant le Control Plane Kubernetes non pas comme une infrastructure lourde, mais comme une simple application conteneurisée. Cette technologie, également connue sous le nom de Hosted Control Planes, dématérialise les composants de gestion du cluster pour les exécuter de manière optimisée.

- **Architecture** : les composants du Control Plane de chaque cluster enfant (API Server, Scheduler, ETCD) sont déployés comme de simples Pods s'exécutant directement sur le Management Cluster. Seuls les nœuds de travail (Workers) sont des VMs ou des serveurs physiques.
- **Efficience** : aucune VM n'est consommée pour le Control Plane. Cette dissociation permet une réduction drastique de l'empreinte en ressources et des coûts associés.
- **Le "Plus"** : l'atout majeur de ce modèle est une densité et une vitesse de déploiement exceptionnelles. Il devient possible de démarrer une cinquantaine de clusters en quelques minutes sans saturer l'infrastructure sous-jacente, ouvrant la voie à des cas d'usage comme les environnements de développement et de test à la demande à très grande échelle.

Ces philosophies distinctes se traduisent par des caractéristiques techniques et des implications stratégiques qui méritent une comparaison directe.

3.
Comparatif Technique et Stratégique Détaillé

Les différences techniques entre Tanzu Supervisor et HyperShift ne sont pas de simples détails d'implémentation. Ce sont des choix d'ingénierie fondamentaux qui ont des implications directes sur les coûts, l'agilité opérationnelle, la scalabilité et les cas d'usage pour lesquels chaque solution est optimisée.

🧱☁️ **Nature du Control Plane :** Tanzu s'appuie sur une isolation matérielle robuste en dédiant des VMs à chaque Control Plane. Ce modèle est idéal pour les organisations dont les exigences de conformité et de sécurité imposent une séparation physique stricte entre les environnements. À l'inverse, HyperShift privilégie une isolation logique agile en exécutant les Control Planes comme des Pods. Cette approche cloud-native est parfaitement adaptée aux contextes où la densité et la rapidité priment sur la séparation matérielle, tout en garantissant une isolation sécurisée via les mécanismes natifs de Kubernetes.

⚡🚀 **Vitesse de Provisionnement** : la différence de nature du Control Plane impacte directement la vitesse de déploiement. Tanzu nécessite le démarrage complet de plusieurs VMs, un processus qui prend entre 10 et 15 minutes. HyperShift, en démarrant de simples Pods, peut provisionner un cluster entièrement fonctionnel en seulement 2 à 5 minutes. Pour une organisation, cette rapidité change la donne : elle permet la création d'environnements de test éphémères pour chaque Pull Request ou le scale-out dynamique de clusters en réponse à des pics de charge.

📈🌐 **Densité et Passage à l'Échelle** : le modèle VM-centric de Tanzu est par nature plus lourd en consommation de CPU et de RAM par cluster, ce qui limite la densité globale sur une infrastructure donnée. HyperShift a été conçu pour le multi-tenant massif. Un seul Management Cluster peut héberger des centaines de Control Planes avec une empreinte minimale, ce qui en fait la solution de choix pour les fournisseurs de services managés (MSP), les plateformes SaaS et les grandes entreprises gérant des parcs de clusters à très grande échelle.

🔄🧩 **Mécanisme de Réconciliation** : les deux solutions utilisent ClusterAPI comme standard, mais avec des "providers" spécifiques. Tanzu s'appuie sur le provider CAPV pour traduire les objets YAML en ressources vSphere (VMs, réseaux). HyperShift combine CAPI avec l'HyperShift Operator, qui traduit les objets YAML en cycle de vie de Control Planes hébergés (les Pods) et de pools de nœuds (les Workers), offrant une réconciliation fine et optimisée pour son architecture.

🏢☁️ **Cible et Cas d'Usage Idéaux** : le positionnement de chaque solution est clair. Tanzu Supervisor est l'outil de prédilection des organisations dont l'infrastructure est centrée sur un datacenter privé vSphere et qui cherchent à moderniser leurs opérations tout en capitalisant sur leurs investissements et compétences existants. HyperShift, quant à lui, est optimisé pour les déploiements sur les clouds publics, les plateformes SaaS et les environnements MSP où la vitesse, l'efficacité des coûts et la scalabilité horizontale sont les principaux moteurs.

💰📉 **Modèle de Coût** : les implications financières sont directes. Avec Tanzu, chaque cluster, même s'il est peu utilisé, engendre des coûts fixes liés à la consommation de ressources des VMs de son Control Plane. HyperShift élimine radicalement ce coût fixe. Le Control Plane ne nécessitant aucune VM dédiée, le modèle économique devient beaucoup plus efficient, en particulier lorsque l'on gère un grand nombre de clusters de tailles et d'utilisations variables.

Critères	Tanzu Supervisor (vSphere)	Red Hat HyperShift
Nature du Contrôle Place	VMs (Isolation matérielle)	Pods (Isolation logicielle)
Vitesse de provisionnement	~10-15 min (Boot VM)	~2-5 min (Boot Pods)
Densité (Scale)	Moyenne (Lourd en RAM/CPU)	Très élevée (Léger)
Réconciliation	ClusterAPI + CAPV	HyperShift Operator + CAPI
Cible idéale	Datacenter Privé / vSphere	Cloud Public / MSP / SaaS
Modèle de coût	Élevé (consommation ressources VM)	Optimisé (Zéro VM pour le CP)

En résumé, le choix entre ces deux approches n'est pas binaire. Il dépend fondamentalement du contexte d'infrastructure, des exigences de gouvernance et des objectifs stratégiques de l'organisation. Ces choix ont également des répercussions profondes sur le modèle de sécurité.

4.
Le Paradigme de Sécurité du Kubernetes as Code

La centralisation du pouvoir inhérente au Kubernetes as Code transforme le Management Cluster en un "cerveau" opérationnel, mais aussi en un périmètre de sécurité de la plus haute criticité. Il détient les clés de l'ensemble de l'infrastructure de clusters. Protéger ce cerveau n'est pas une option, c'est une obligation. Une automatisation puissante sans une sécurité renforcée peut rapidement devenir une faille majeure. Sécuriser ce composant central est donc non négociable et repose sur quatre règles d'or.

4.1. Appliquer le Principe du Moindre Privilège (RBAC)

Le premier principe est non négociable : il est hors de question de donner des droits cluster-admin à la légère. Il est impératif de segmenter les accès au Management Cluster. Chaque équipe de développement ou projet doit être isolée dans son propre Namespace (qu'il s'agisse des vSphere Namespaces dans Tanzu ou des namespaces Kubernetes standards dans HyperShift). De plus, les secrets critiques, tels que les kubeconfig des clusters enfants ou les clés d'API des fournisseurs de cloud, ne doivent jamais être exposés en clair. L'intégration avec des outils comme Sealed Secrets ou HashiCorp Vault est essentielle pour chiffrer ces informations sensibles au repos et en transit.

4.2. Durcir l'Accès à la Source de Vérité (Git)

Avec une approche GitOps, le dépôt Git devient de fait la console d'administration de l'infrastructure. Le protéger est aussi important que de protéger le Management Cluster lui-même. Des mesures strictes doivent être mises en place :

- **Protection des branches** : interdire toute modification directe sur les branches principales (main, master).
- **Revue de code obligatoire :** exiger qu'une seconde personne valide toute modification d'infrastructure (ajout de nœuds, mise à jour de version) via une Pull Request.
- **Signature des commits** : mettre en œuvre la signature cryptographique des commits pour garantir que seul le code provenant de développeurs autorisés puisse être synchronisé par les outils GitOps.

4.3. Instaurer la "Zero Trust" entre Management et Workload

La compromission d'un cluster enfant (Workload Cluster) ne doit jamais permettre à un attaquant de remonter jusqu'au Management Cluster. Chaque architecture propose une barrière efficace. HyperShift excelle dans ce domaine en isolant le Control Plane dans un namespace dédié, rendant le Management Cluster invisible depuis un nœud de travail compromis. Tanzu, de son côté, s'appuie sur l'isolation forte de l'hyperviseur ESXi, qui agit comme une barrière matérielle entre les VMs des différents clusters.

4.4. Mettre en Place un Audit et une Observabilité Stricts

Toute action sur l'infrastructure doit être traçable. Il est fondamental d'activer les logs d'audit sur l'API du Management Cluster pour savoir qui a demandé la création ou la modification de quel cluster, et quand. En complément, l'utilisation d'outils de Policy as Code (tels que Kyverno ou OPA Gatekeeper) est recommandée. Ces outils permettent de définir des règles qui empêchent le déploiement de clusters non conformes, par exemple en interdisant les versions de Kubernetes obsolètes ou en exigeant le chiffrement systématique des disques.

L'agilité conférée par le Kubernetes as Code doit impérativement s'accompagner d'une discipline de sécurité rigoureuse, appliquée à la fois au code et à la plateforme de gestion.

5.
Mise en pratique : Le YAML est « Roi »
Exemple Tanzu (Cluster & Node Pool Template)

Voici comment déclarer une topologie de cluster VKS (vks-cluster.yaml) qui sera interprétée par vSphere (Tanzu Supervisor).

1. La définition du Cluster (L'intention)

```
apiVersion: cluster.x-k8s.io/v1beta1
kind: Cluster
metadata:
  name: tkg-vks-prod
  namespace: finance-dept
spec:
  topology:
    class: tanzukubernetescluster
    version: v1.31.0
    controlPlane:
      replicas: 3
    workers:
      machineDeployments:
        - class: "custom-vks-node-pool"
          name: "frontend-pool"
          replicas: 3
```

2. Le Template de Node Pool (Hardware as Code)

```
apiVersion: infrastructure.cluster.x-k8s.io/v1beta1
kind: VSphereMachineTemplate
metadata:
  name: vks-node-hardware-spec
  namespace: finance-dept
spec:
  template:
    spec:
      vmClass: best-effort-xlarge
      storageClass: vks-gold-storage-policy
```

Exécution

Bash

Se connecter au Tanzu Supervisor

```
kubectl vsphere login --server=SUPERVISOR-IP
```

Appliquer la configuration

```
kubectl apply -f vks-cluster.yaml
```

Vérifier le provisionnement des VMs dans vCenter

```
kubectl get clusters -w
```

Exemple HyperShift (HostedCluster)

HyperShift permet de créer un cluster via une CLI qui génère le "HostedCluster" sur le management cluster.

Exécution

 # Création via la CLI (qui génère le YAML)

```
hypershift create cluster aws --name cluster-client-a --node-
pool-replicas 3
```

 # Vérifier l'état des Pods du Control Plane

```
oc get pods -n clusters-cluster-client-a
```

 # Récupérer le kubeconfig du nouveau cluster

```
hypershift create kubeconfig --name cluster-client-a > client-
a.kubeconfig
```

Le résultat est un objet HostedCluster dans K8s qui gère automatiquement le cycle de vie sur AWS.

Voici le fichier YAML généré par la commande hypershift create :

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: cluster-client-a
  namespace: clusters
spec:
  release:
    image: quay.io/openshift-release-dev/ocp-release:4.14.0-
x86_64
  region: eu-west-3
  platform:
    type: AWS
    aws:
      endpointAccess: Public
      roles: [...]
---
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  name: pool-1
  namespace: clusters
spec:
  clusterName: cluster-client-a
  replicas: 3
  platform:
    type: AWS
    aws:
      instanceType: m5.xlarge
```

6.
Conclusion : Définir Votre Stratégie Kubernetes as Code

Le Kubernetes as Code représente une avancée majeure, offrant une agilité et une reproductibilité sans précédent dans la gestion de flottes de clusters. Cependant, ce paradigme déplace la responsabilité : la sécurité et la gouvernance ne se gèrent plus uniquement au niveau du serveur, mais sont désormais intrinsèquement liées à la chaîne de déploiement (CI/CD) et à l'API de management. Le choix entre une approche "Infra-Centric" comme Tanzu et une approche "App-Centric" comme HyperShift dépend donc de la "Landing Zone" stratégique de votre organisation.

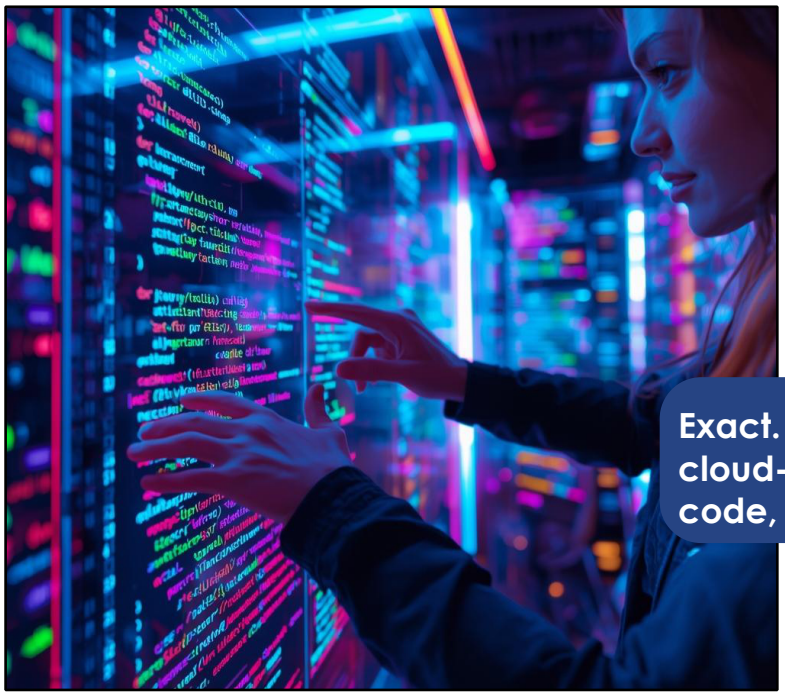
Voici nos recommandations finales pour guider votre décision :

- **Choisissez Tanzu Supervisor si...** votre organisation est fortement ancrée dans un datacenter On-Premise avec une culture vSphere mature. C'est le choix idéal si vos priorités sont la gouvernance stricte, l'intégration avec l'écosystème VMware existant et une isolation matérielle éprouvée pour répondre à des exigences de conformité rigoureuses.

- **Choisissez HyperShift si...** vous opérez à grande échelle, principalement sur le cloud public, ou si vous êtes un fournisseur de services (MSP) ou une plateforme SaaS. Cette approche est la plus pertinente lorsque vos principaux moteurs sont la réduction des coûts opérationnels, la densité maximale, la scalabilité massive et une agilité extrême pour provisionner des environnements à la demande.

En définitive, la question n'est plus de savoir si vous devez gérer vos clusters comme du code, mais comment le faire de la manière la plus alignée avec votre stratégie.

Le futur du Cloud-Native n'est pas de gérer des clusters,
mais de gérer le code qui les génère.



metanext
by SII Group

A PROPOS DE METANEXT

Spécialiste du conseil et de l'accompagnement à la transformation des infrastructures, Metanext est le facilitateur de la migration vers le cloud avec des solutions innovantes et pérennes. Partenaire privilégié des éditeurs majeurs du marché tels que Red Hat, Suse, VMware, Nutanix et des hyperscalers Google Cloud, Microsoft et
AWS, Metanext se positionne comme 1er acteur de confiance sur ces solutions.

VOTRE CONTACT METANEXT

commercial@metanext.com

Metanext.com

Think I Build I Secure I Run the cloud