

---

# Comment gérer des applications nécessitant la persistance des données avec Kubernetes

Devoxx France - 2018

---

# Florian Woerner

DevOps Engineer / Cloud Architect

Onmyown

@woernfl

[florian.woerner@onmyown.io](mailto:florian.woerner@onmyown.io)





---

# Agenda

- Contexte
- Volume provider
- Gestion le lifecycle de Volumes
- Demo
- Conclusion



---

# Pourquoi choisir Kubernetes plutôt que des VMs ?



**Parce que Kubernetes est un  
framework de gestion de  
système distribué**



---

# Pourquoi a-t-on besoin de gérer la persistance des données ?

Parce qu'une architecture  
stateless... ça n'existe pas



---

# Types de volumes

---

# Types de Volumes - Kubernetes Internals

- configMap
- downwardAPI
- emptyDir
- gitRepo
- hostPath
- local
- persistentVolumeClaim
- projected
- Secret
- csi



---

# Types de Volumes - Public Cloud

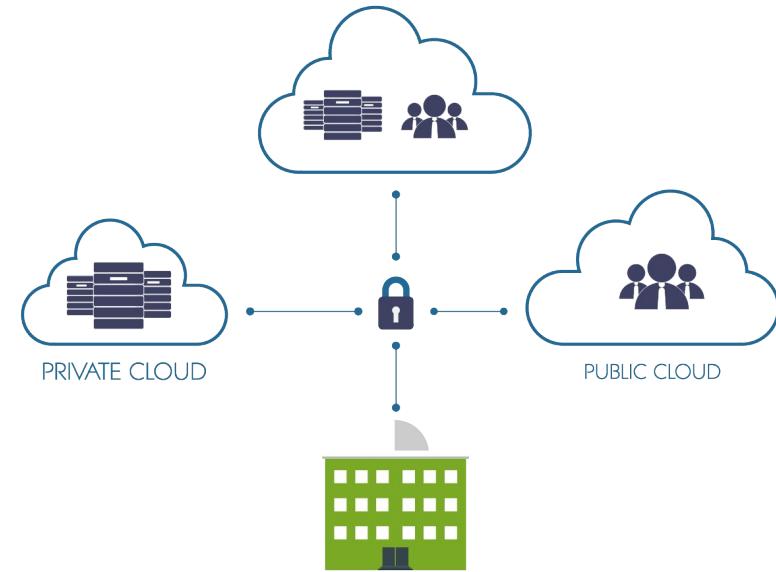
- awsElasticBlockStore
- azureDisk
- azureFile
- gcePersistentDisk



---

# Types de Volumes - Non Cloud Dependant

- cephfs
- fc (fibre channel)
- flocker
- glusterfs
- nfs
- iscsi
- portworxVolume
- quobyte
- rbd (Rados Block Device)
- scaleIO
- storageos
- vsphereVolume





---

# Un problème ?



# CSI à la rescousse



# CSI c'est quoi ?

- Container Storage Interface
- Adopté par Kubernetes, Mesos et Cloud Foundry (pour l'instant)
- Supporté en beta dans Kubernetes 1.10 (planifié en stable pour 1.12)
- Un moyen standard d'exposer du storage au container

# Pourquoi est ce important ?

- Interface commune à plusieurs plates-formes (standardisation)
- Permet l'évolution des storage provider en dehors des releases de Kubernetes

---

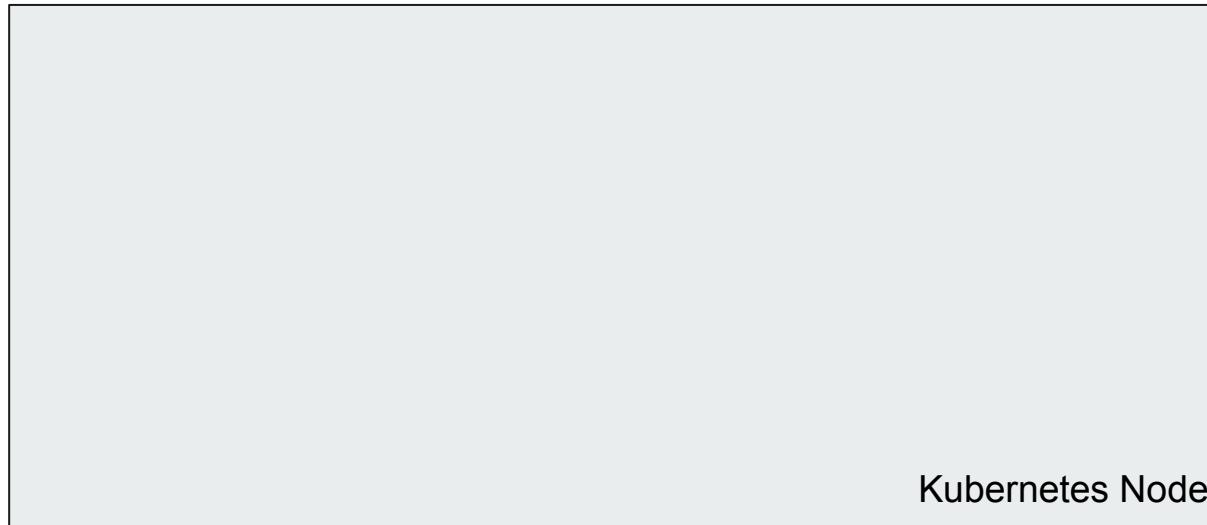
Mise en situation avec **Bob** le développeur et **John** le Sysadmin

---

# Scénario: Persistent Volumes et Claims

---

## Scénario: Persistent Volumes et Claims



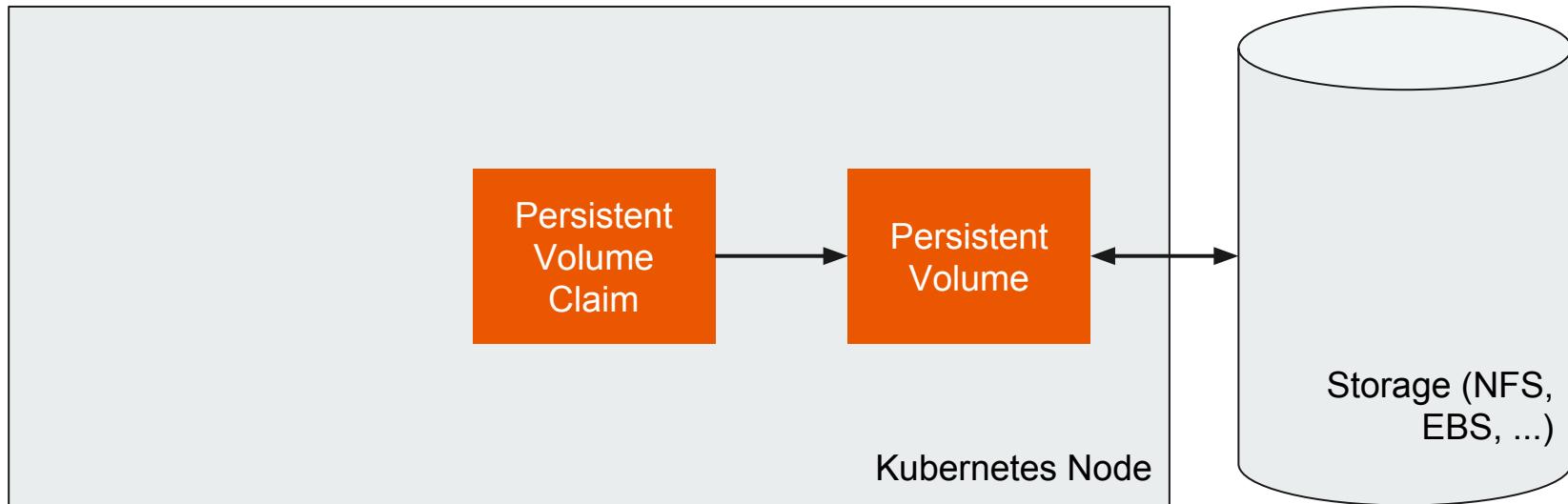
Storage (NFS,  
EBS, ...)

Kubernetes Node

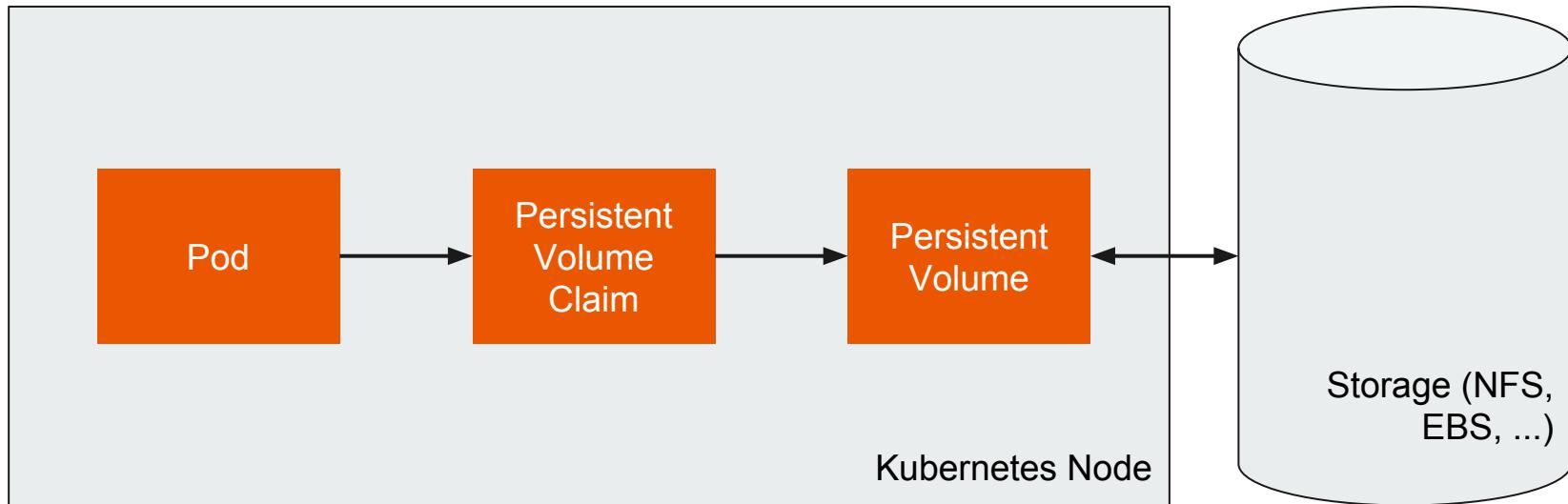
## Scénario: Persistent Volumes et Claims



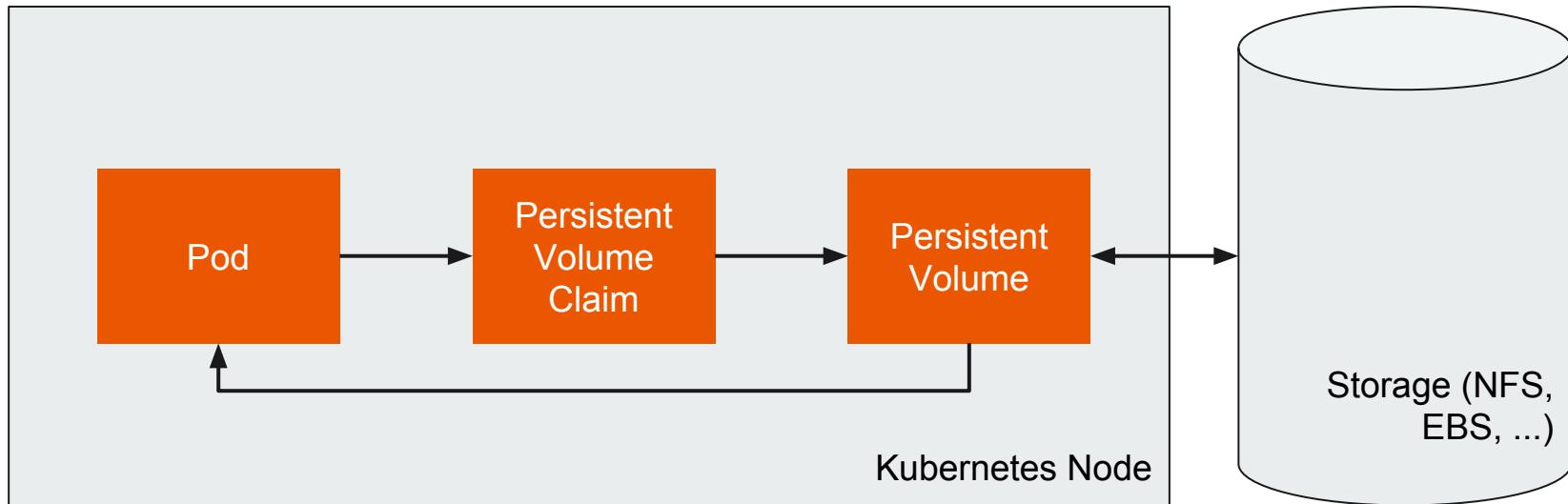
## Scénario: Persistent Volumes et Claims



## Scénario: Persistent Volumes et Claims



## Scénario: Persistent Volumes et Claims

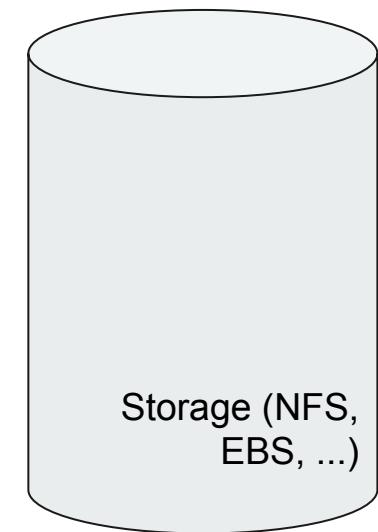
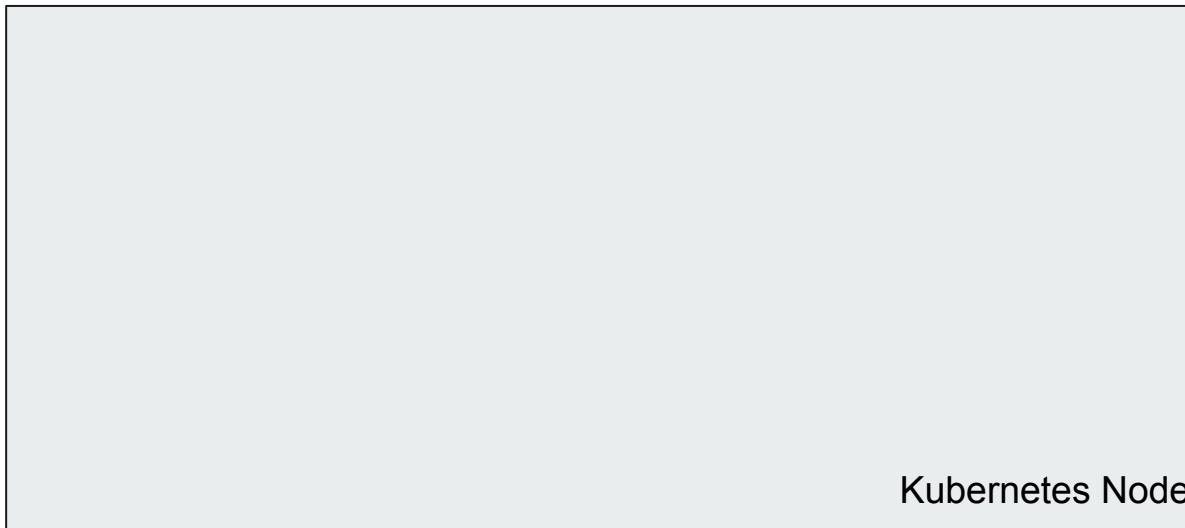


---

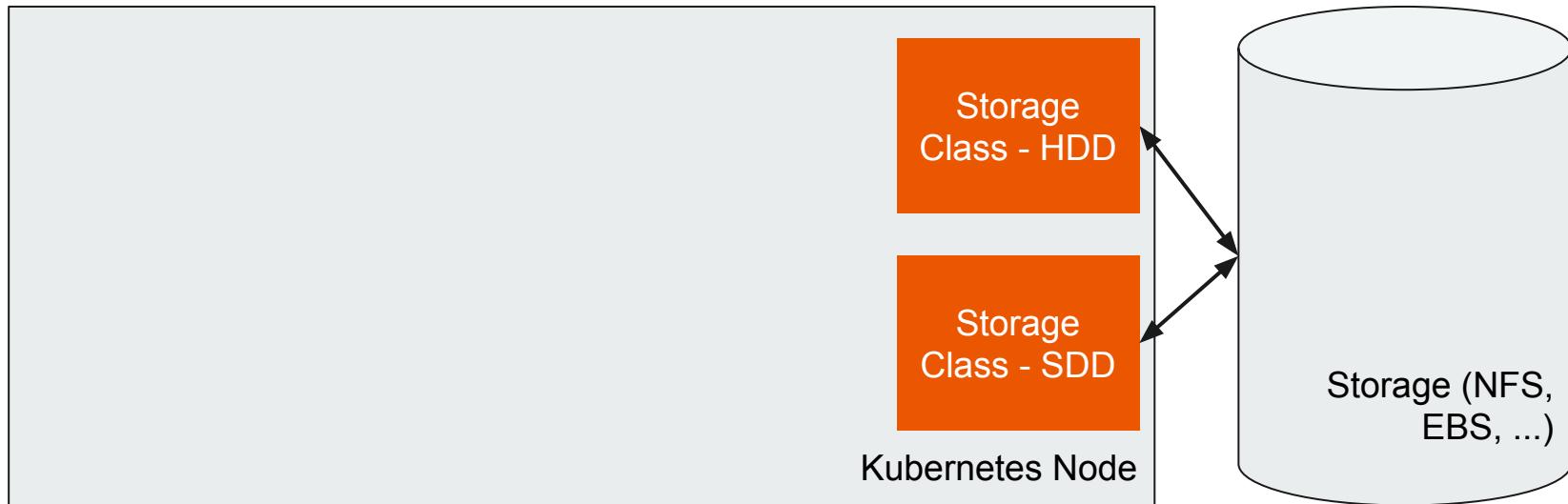
# Scénario: Dynamic Provisioning

---

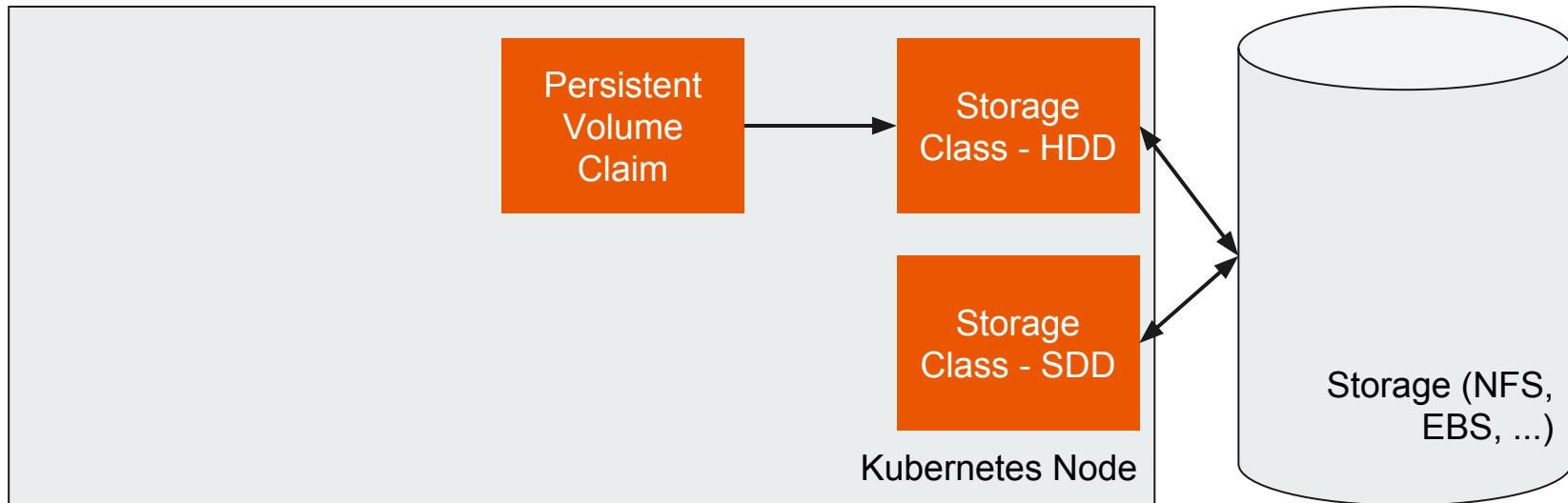
# Scénario: Dynamic Provisioning



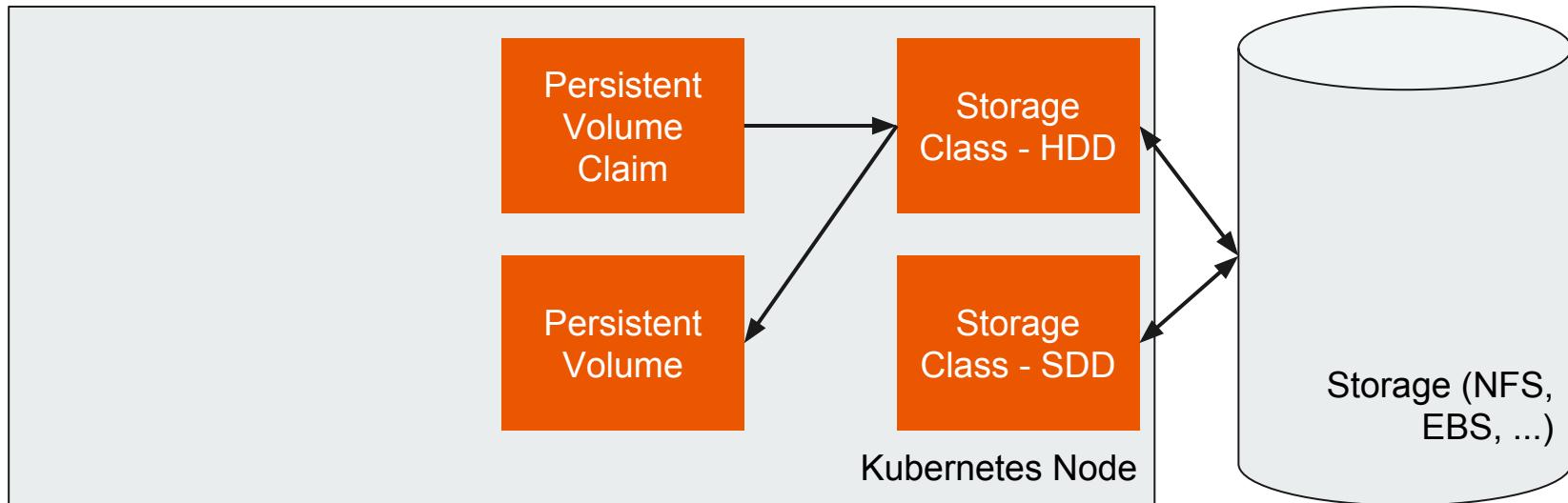
# Scénario: Dynamic Provisioning



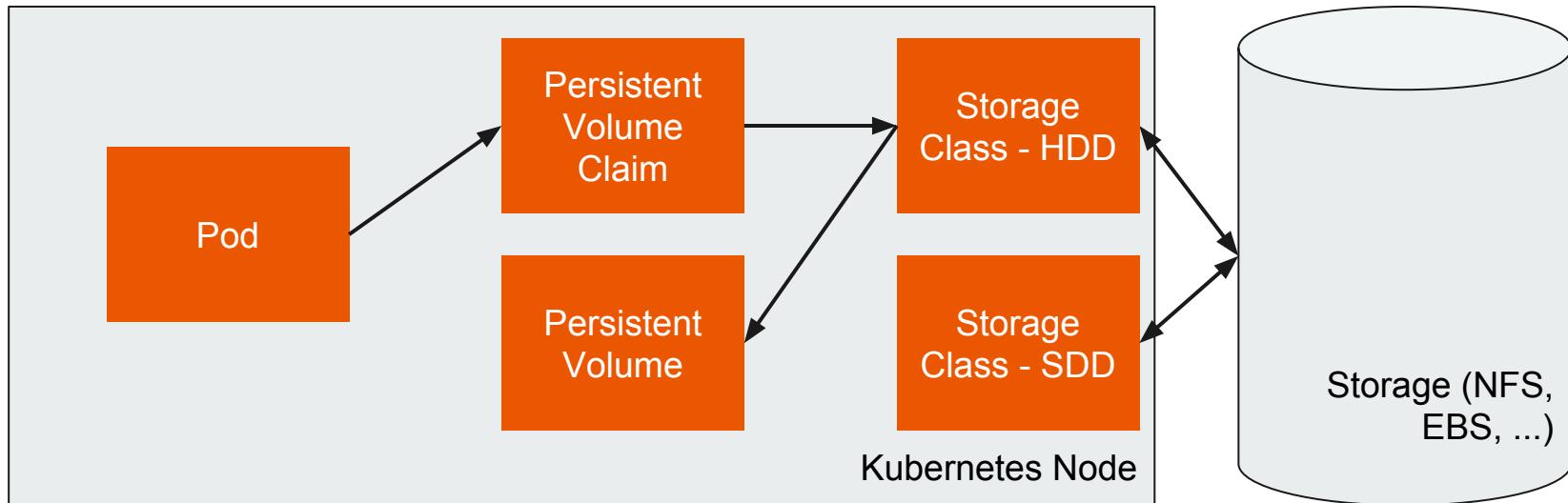
# Scénario: Dynamic Provisioning



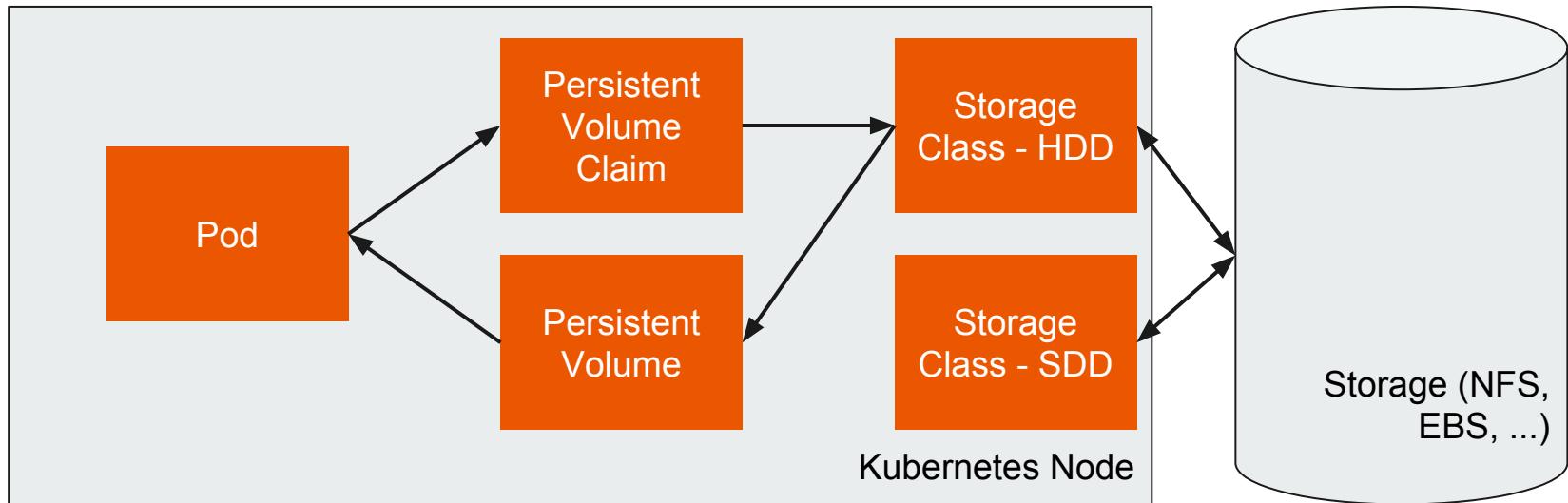
# Scénario: Dynamic Provisioning



# Scénario: Dynamic Provisioning



# Scénario: Dynamic Provisioning



---

# Demo



---

# Conclusion

---

# Référence

- Volumes Kubernetes Doc: <https://kubernetes.io/docs/concepts/storage/volumes/>
- Kubernetes Tasks: <https://kubernetes.io/docs/tasks/>
- Configure a Pod to Use a PersistentVolume for Storage:  
<https://kubernetes.io/docs/tasks/configure-pod-container/configure-persistent-volume-storage/>
- Run a Single-Instance Stateful Application:  
<https://kubernetes.io/docs/tasks/run-application/run-single-instance-stateful-application/>
- Run a Replicated Stateful Application:  
<https://kubernetes.io/docs/tasks/run-application/run-replicated-stateful-application/>
- StatefulSets Kubernetes Doc:  
<https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/>

---

# Merci !

Link vers le repos: <https://gitlab.com/woernfl/k8s-stateful-demo>

