

SDN-NFV - 2

ESIR

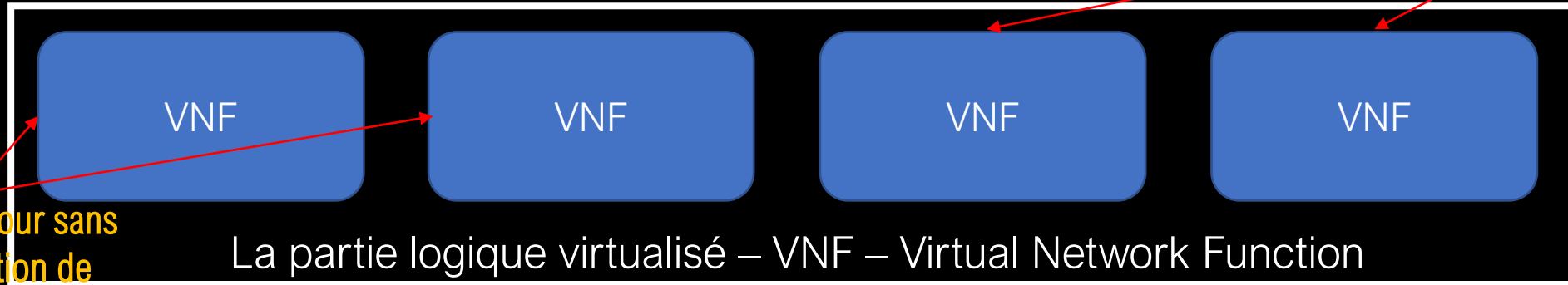
Djob Mvondo

NFV: Network Function Virtualization

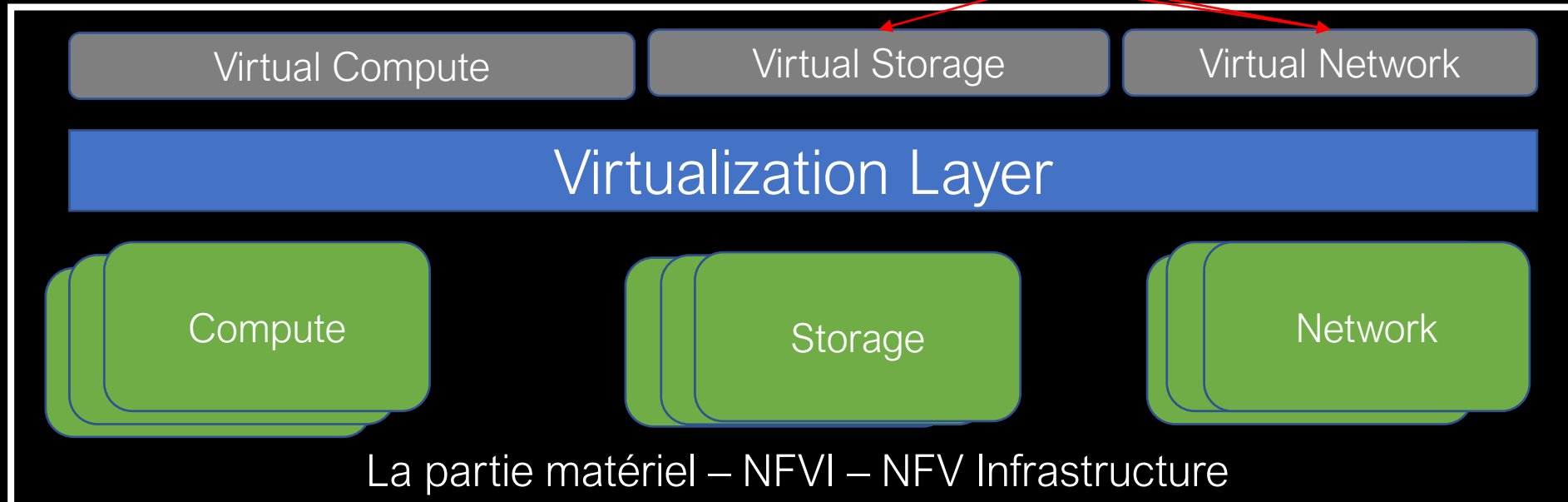
Trois composants forment les NFVs: VNF, NFVI, et MONA.

Optimiser le démarrage
des VNFs

Mise à jour sans
interruption de
service



Optimisation la couche réseau et stockage –
passage à l'échelle



Surveillance
interopérable et
performante

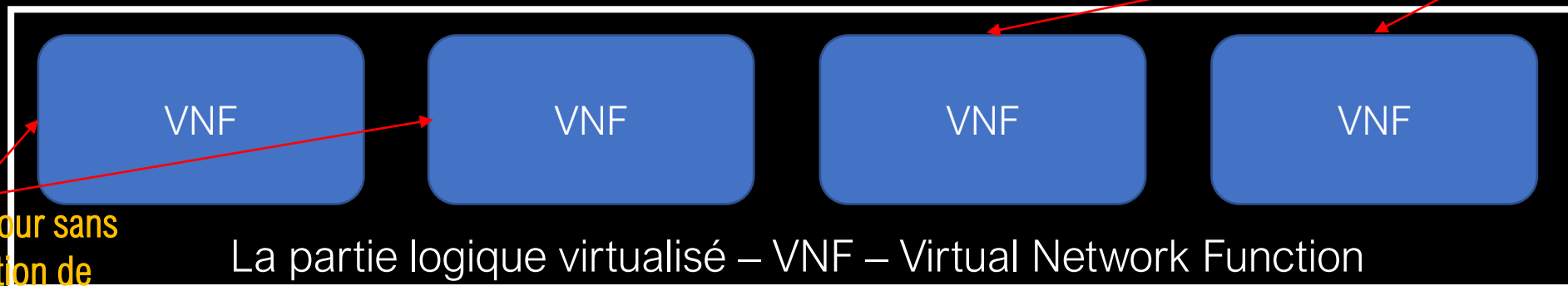
NFV MONA –
Management
and
Orchestration

NFV: Network Function Virtualization

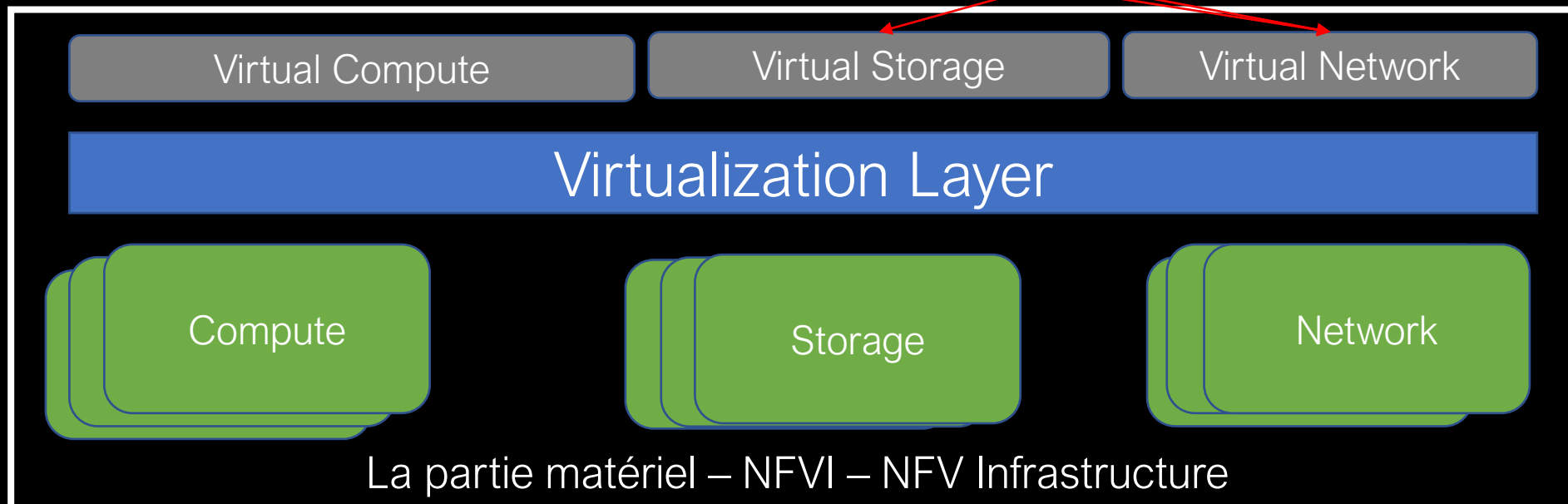
Commençons par détailler la partie MONA

Optimiser le démarrage
des VNFs

Mise à jour sans
interruption de
service



Optimisation la couche réseau et stockage –
passage à l'échelle

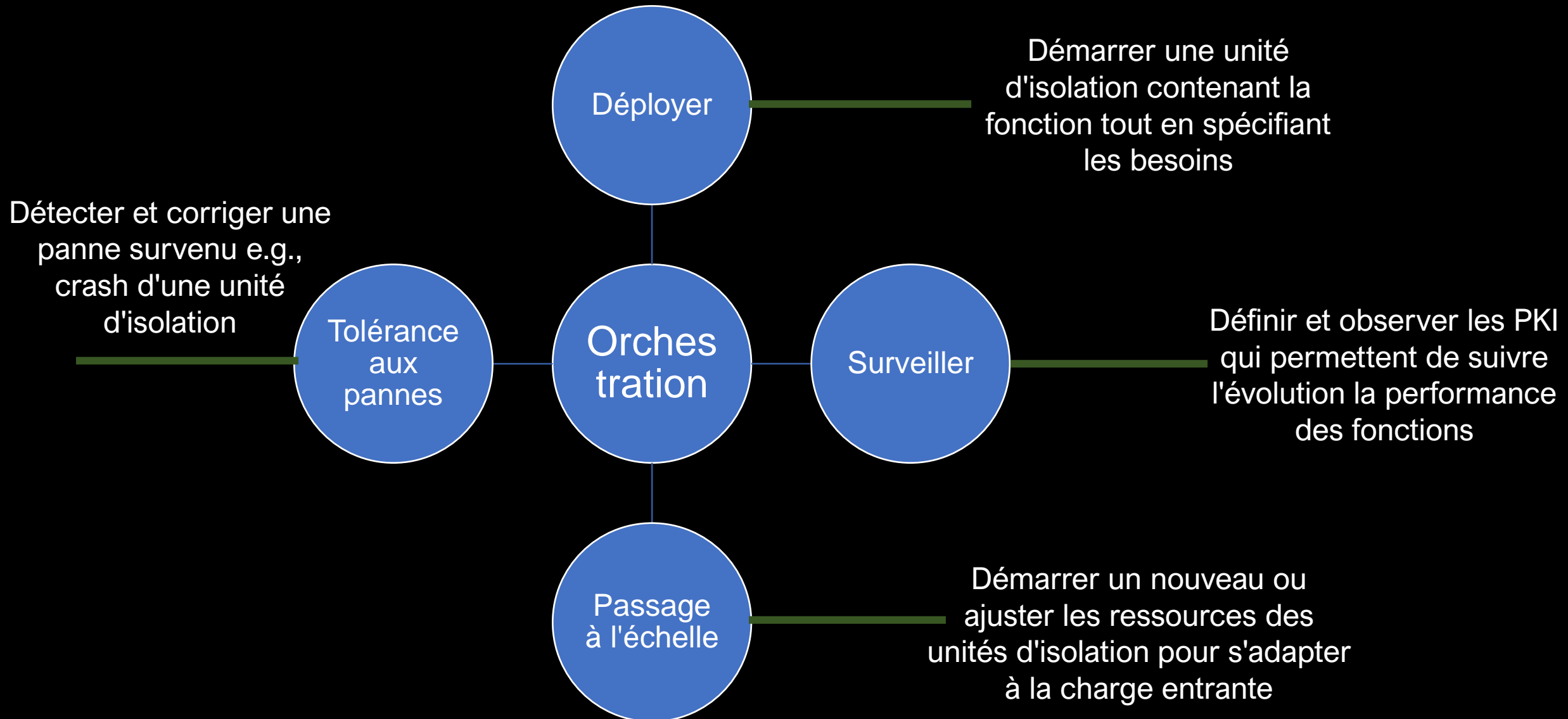


Surveillance
interopérable et
performante

NFV MONA –
Management
and
Orchestration

MONA: Management and Orchestration

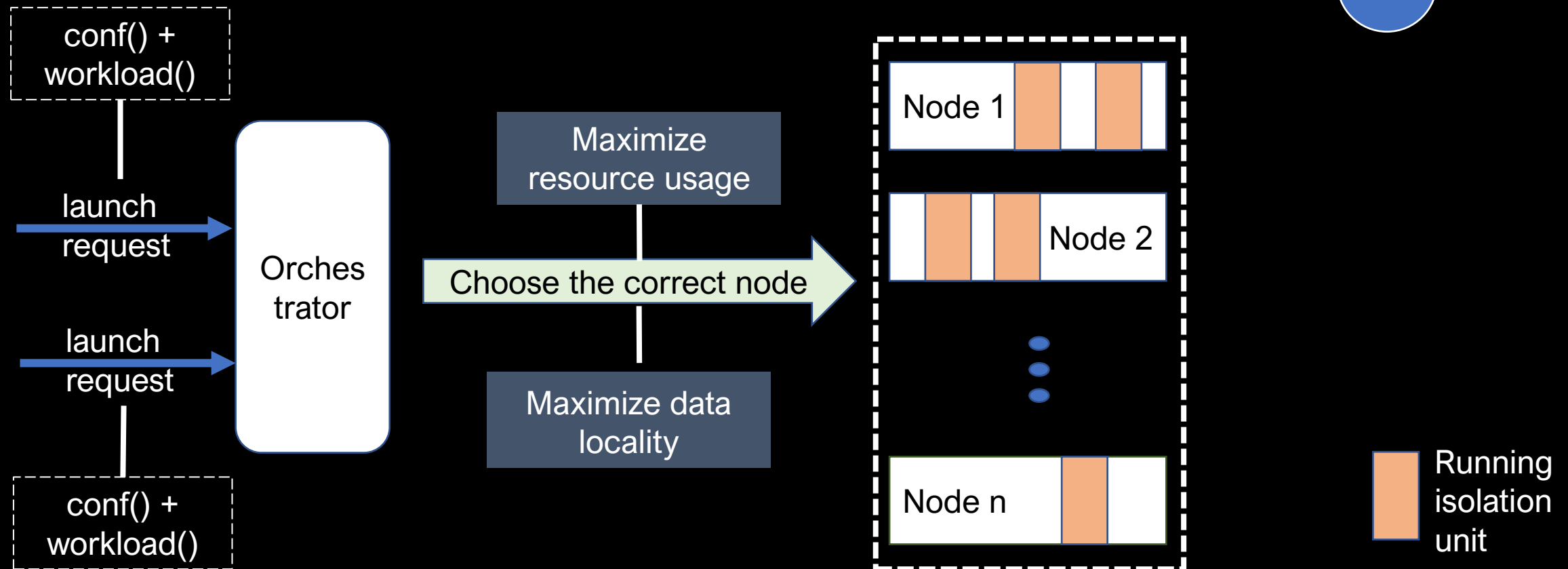
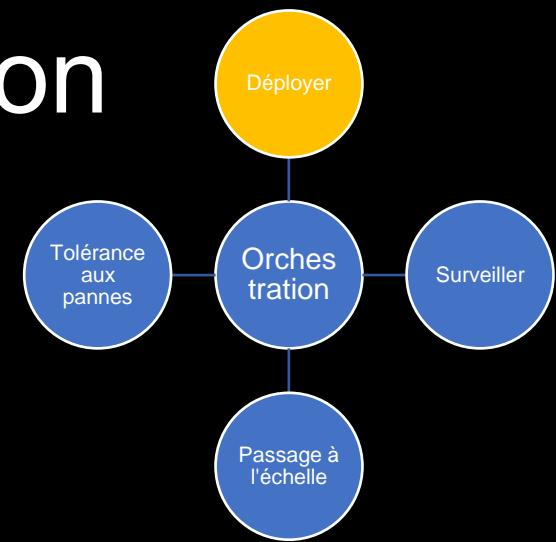
Que signifie **orchestrer** ?



MONA: Management and Orchestration

Orchestrer est **dur**. Pourquoi ?

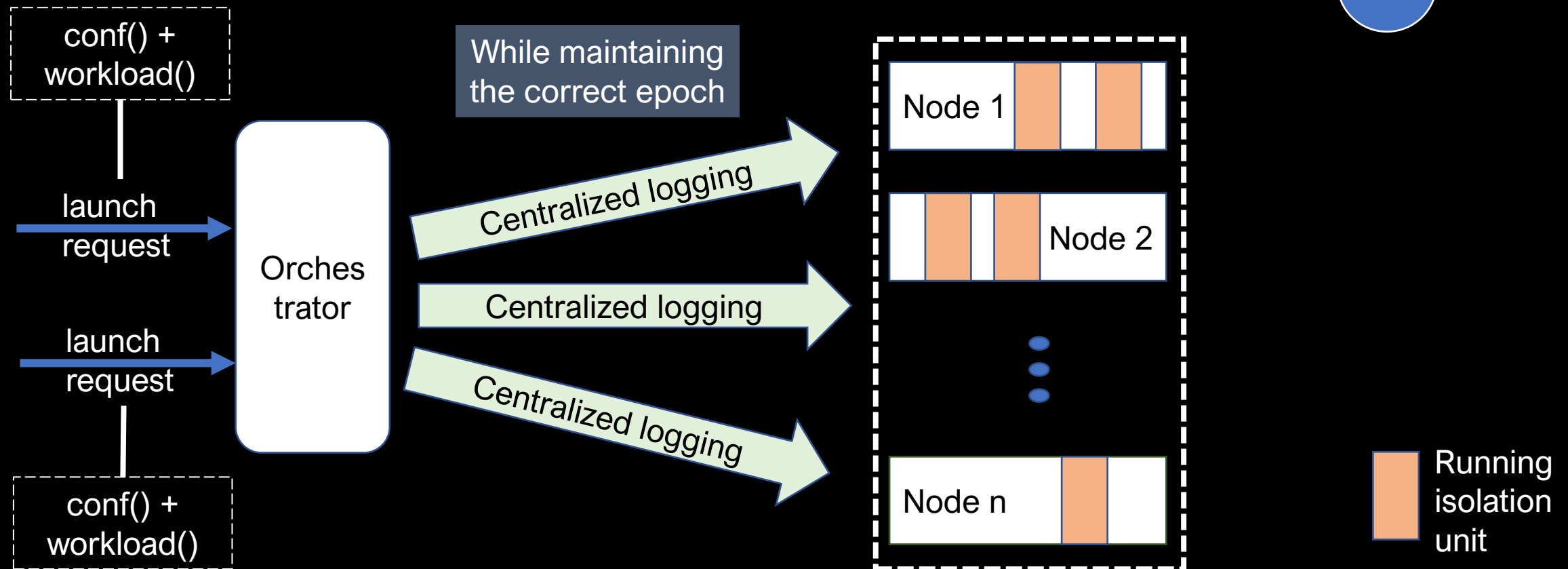
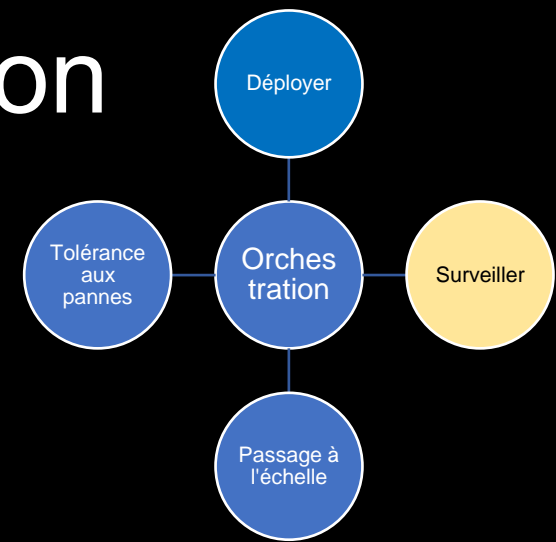
Le schéma du cas d'utilisation "Déploiement"



MONA: Management and Orchestration

Orchestrer est **dur**. Pourquoi ?

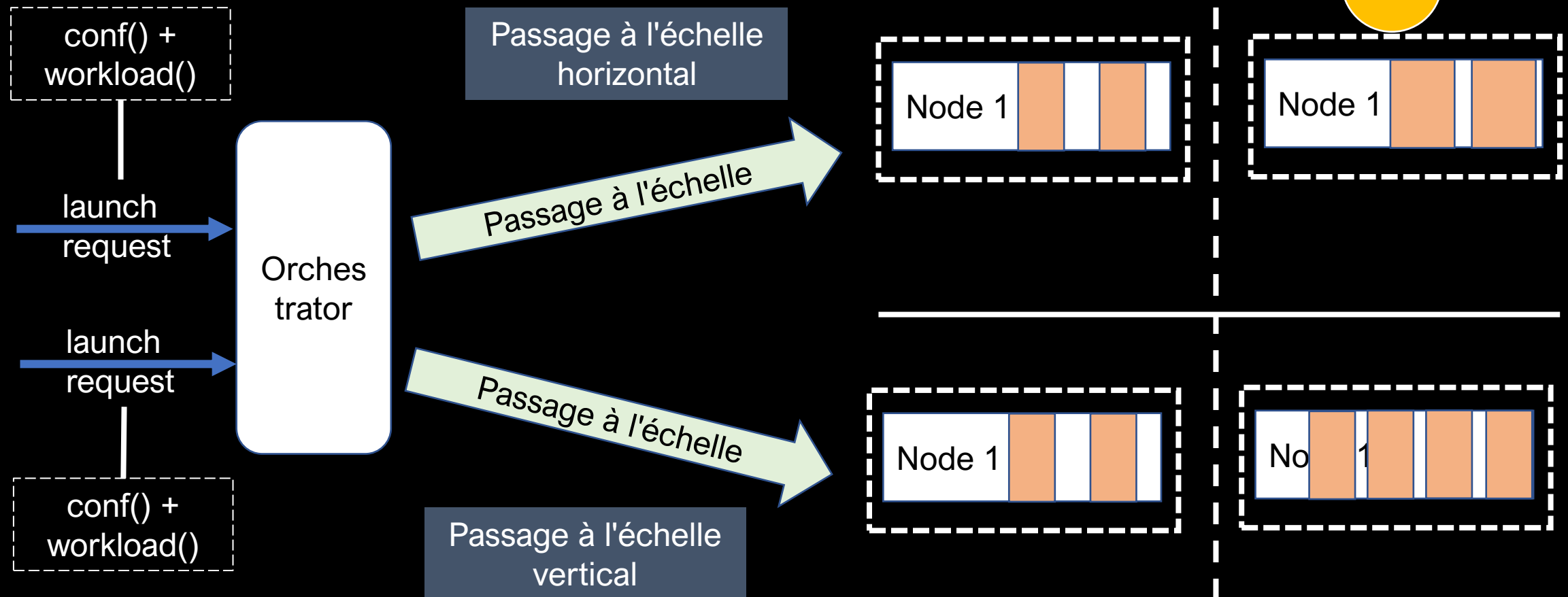
Le schéma du cas d'utilisation "Surveillance"



MONA: Management and Orchestration

Orchestrer est **dur**. Pourquoi ?

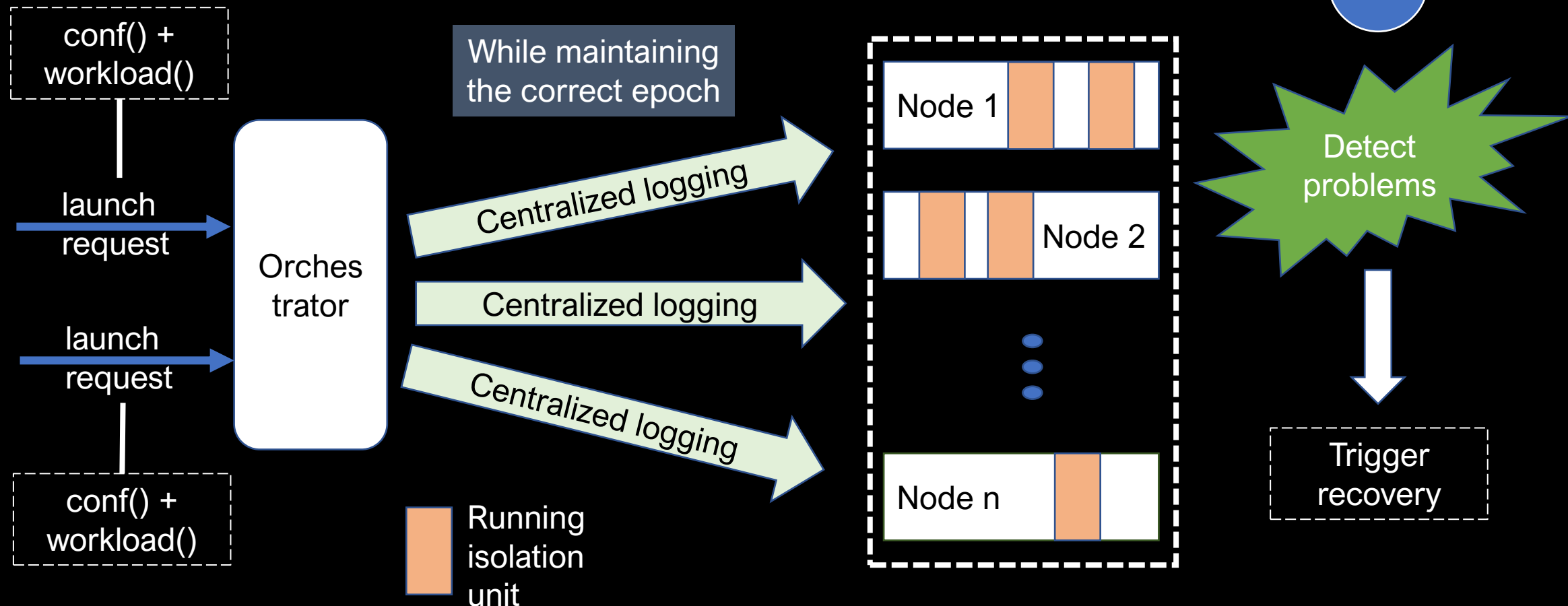
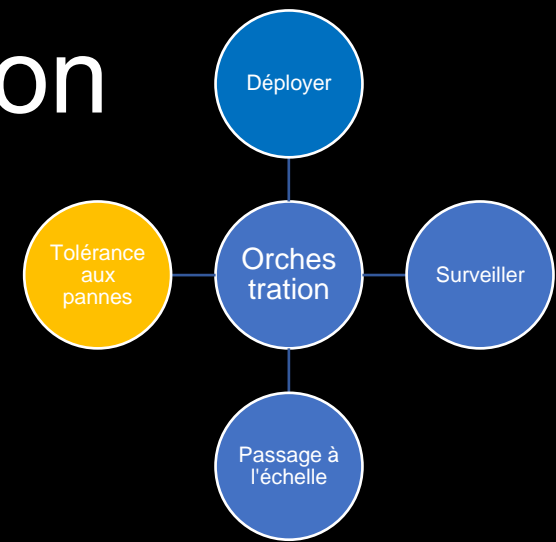
Le schéma du cas d'utilisation "**Passage à l'échelle**"



MONA: Management and Orchestration

Orchestrer est **dur**. Pourquoi ?

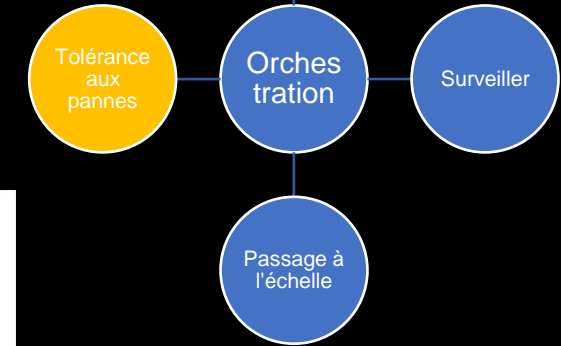
Le schéma du cas d'utilisation "Surveillance"



MONA: Management and Orchestration

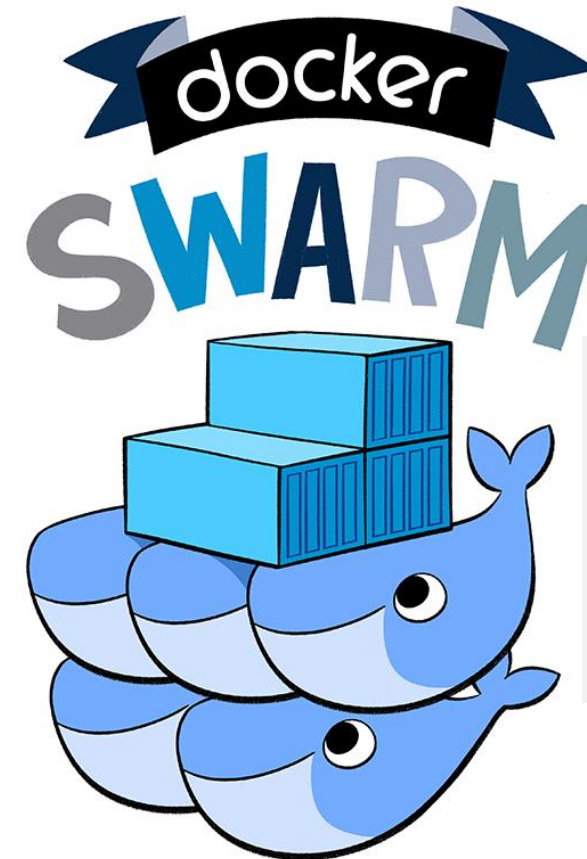
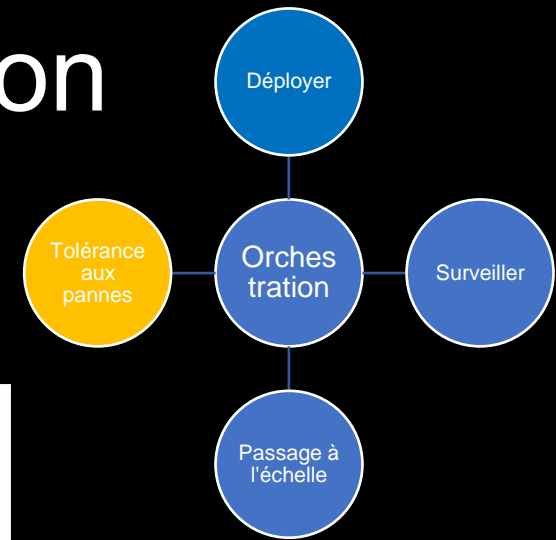
Orchestrer est **dur**.

Mieux vaut utiliser un outil tout fait.



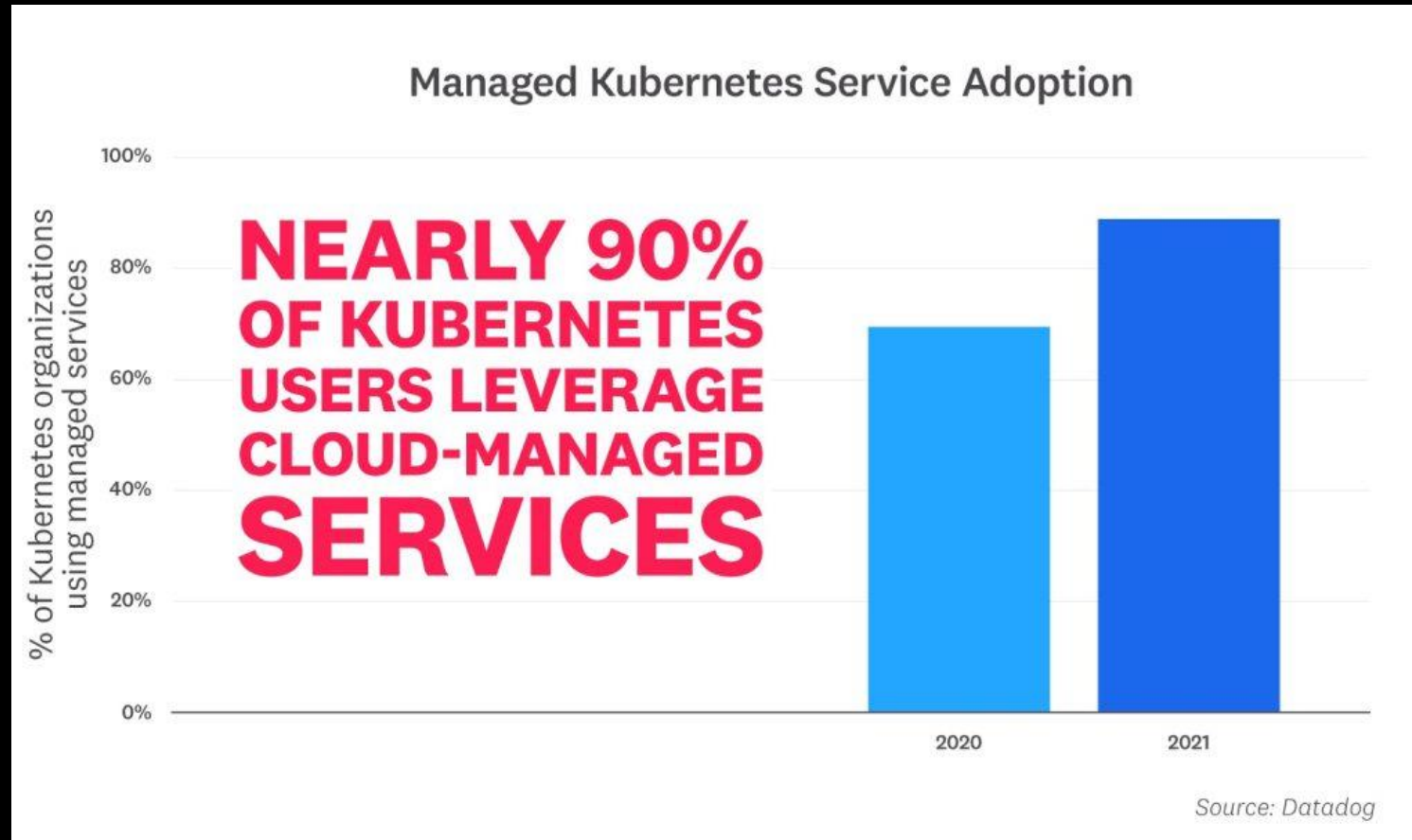
MONA: Management and Orchestration

Nous allons utiliser **KUBERNETES** comme illustration.
Mais l'architecture de base reste semblable aux autres.

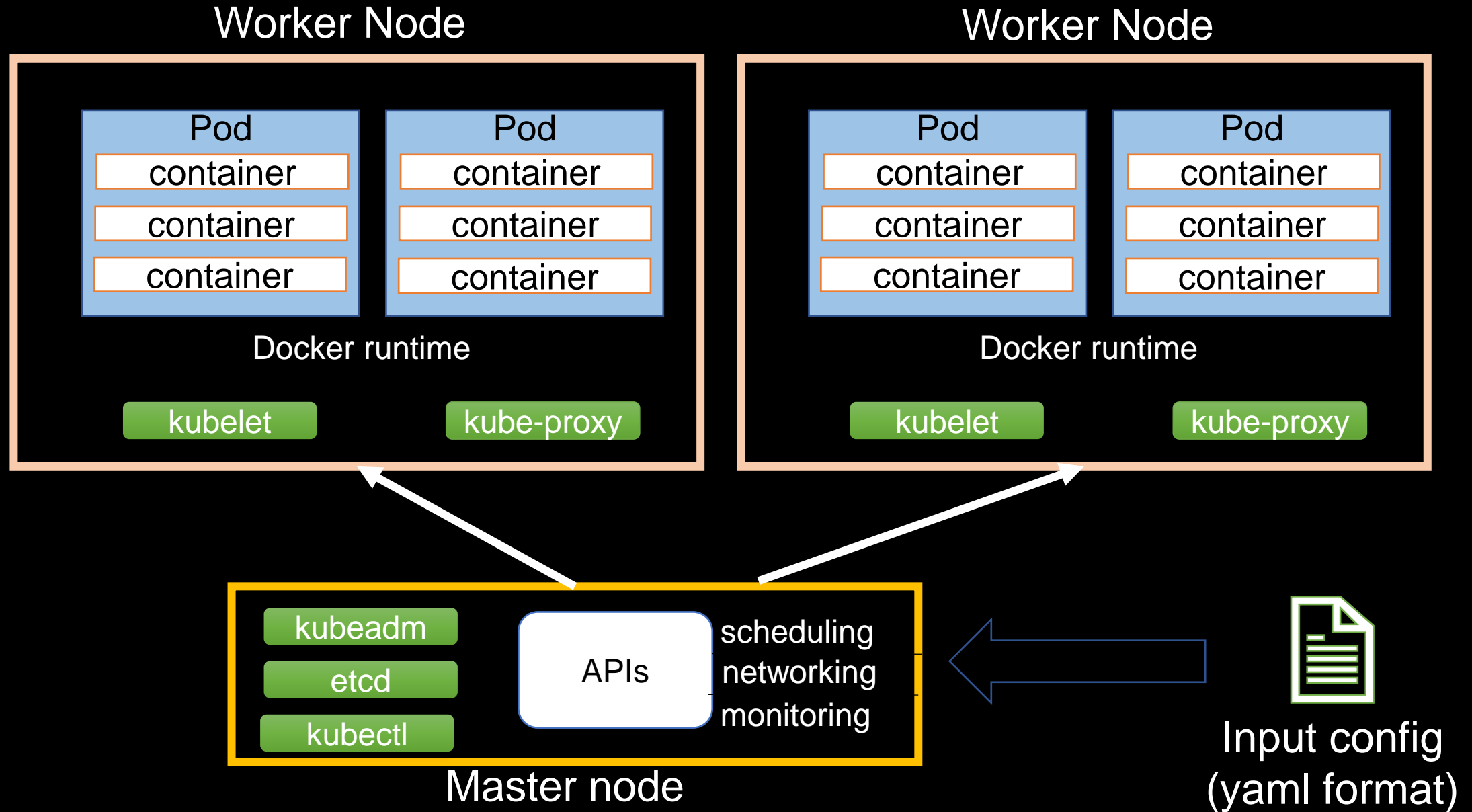


MONA: Management and Orchestration

Nous allons utiliser **KUBERNETES** comme illustration.
Pourquoi ?

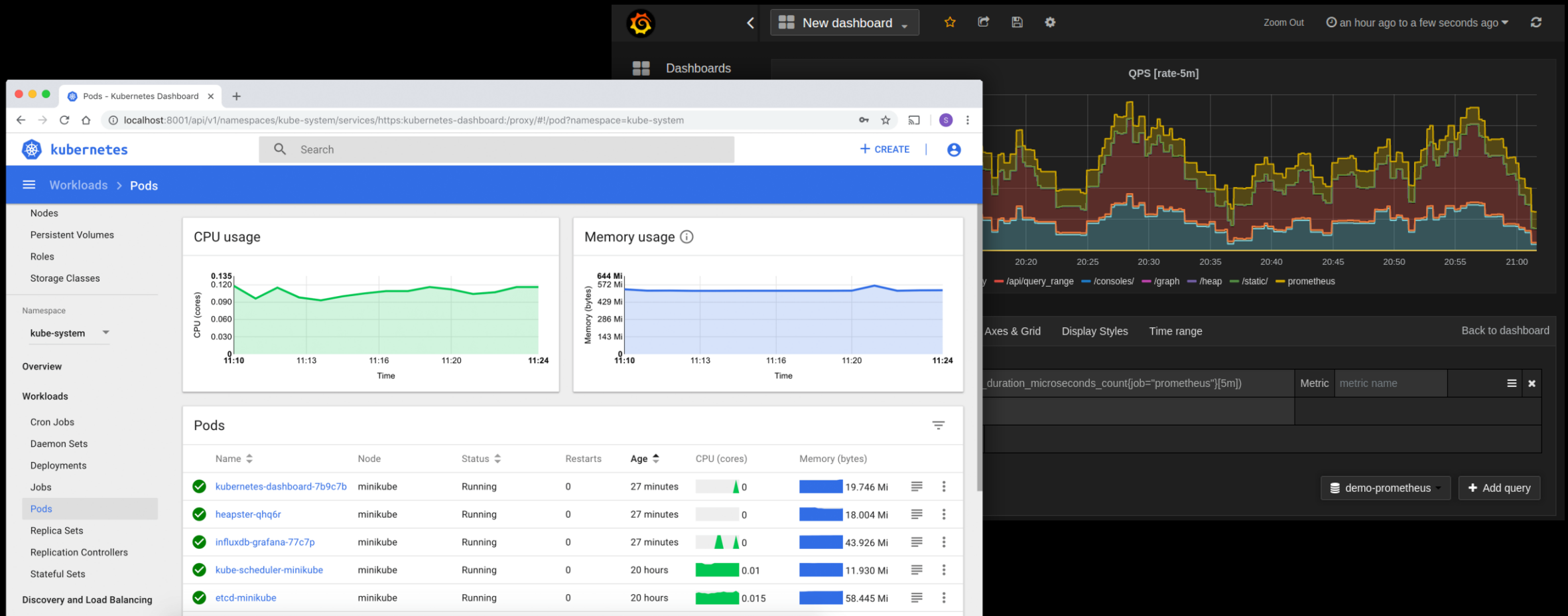


Orchestrateur – Kubernetes (k8s)



Orchestrateur – Kubernetes (k8s)

Visualisation de plusieurs métriques grâce à une interface web natif ou Prometheus/Grafana



Orchestrateur – Kubernetes (k8s)

Kubernetes repose sur une architecture master/workers.

Les nœuds workers possède l'environnement pour l'exécution des containers.

Le nœud master communique avec les workers pour surveiller l'état des containers grâce au service **kubelet** et persiste les données sur **etcd**.

Les nœuds workers créent des containers dans une abstraction appelé **pods** qui symbolise un groupe de containers qui réalisent un objectif précis.

L'isolation réseau est assuré par le composant **kube-proxy**

Orchestrateur – Kubernetes (k8s)

On Ubuntu

```
sudo apt-get update
```

```
sudo apt-get install -y apt-transport-https ca-certificates  
curl
```

```
sudo curl -fsSL /usr/share/keyrings/kubernetes-archive-  
keyring.gpg  
https://packages.cloud.google.com/apt/doc/apt-key.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-  
archive-keyring.gpg] https://apt.kubernetes.io/  
kubernetes-xenial main" | sudo tee  
/etc/apt/sources.list.d/kubernetes.list
```

```
sudo apt-get update  
sudo apt-get install -y kubelet kubeadm kubectl  
sudo apt-mark hold kubelet kubeadm kubectl
```

On Windows (kubectl, kubelet)

Installer docker-desktop puis aller dans les paramètres de Docker et assurer vous que Kubernetes est activé (diode verte)

Kubeadm ne supporte pas les hôtes Windows mais les worker peuvent tourner sous Windows

Tester votre installation avec :

```
kubectl --version  
kubectl get all
```

Orchestrateur – Kubernetes (k8s)

Format du fichier de configuration

Télécharger le fichier deployment.yaml à l'adresse suivante :

https://github.com/djobiii2078/cloud_course_resources/blob/main/deployment.yaml

Pouvez-vous expliquer ce qu'il décrit ?

Orchestrateur – Kubernetes (k8s)

Déploiement continu

```
minReadySeconds: 10
strategy:
  rollingUpdate:
    maxSurge: 1
    maxUnavailable: 0
type: RollingUpdate
```

Définit la stratégie de mise à jour pour notre déploiement.

MaxSurge conditionne le nombre de containers à instancier lors du passage à l'échelle.

Mettez à jour le déploiement :

```
kubectl apply -f deployment.yaml
```

Orchestrateur – Kubernetes (k8s)

Déploiement continu

```
spec:  
  containers:  
  - name: bb-site:v2  
    image: getting-started
```

Le fichier peut être mis à jour pour faire évoluer l'architecture.

Kubernetes redéploie les composants modifiés pour s'ajuster aux nouvelles directives.

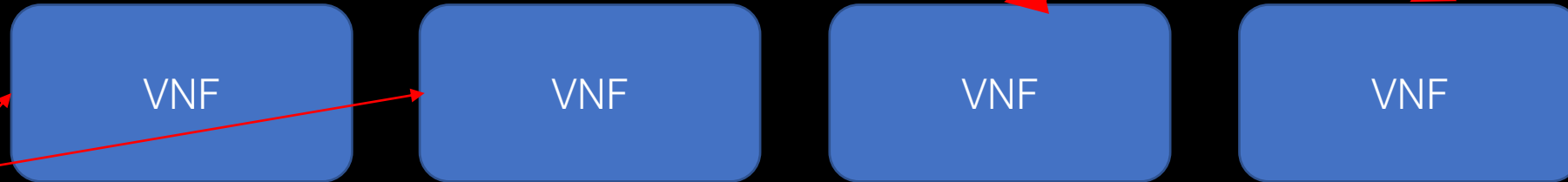
Mettez à jour le déploiement :

```
kubectl apply -f deployment.yaml
```

NFV: Network Function Virtualization

Commençons par détailler la partie MONA

Optimiser le démarrage
des VNFs



La partie logique virtualisé – VNF – Virtual Network Function

Mise à jour sans
interruption de
service

Optimisation la couche réseau et stockage –
passage à l'échelle



La partie matériel – NFVI – NFV Infrastructure

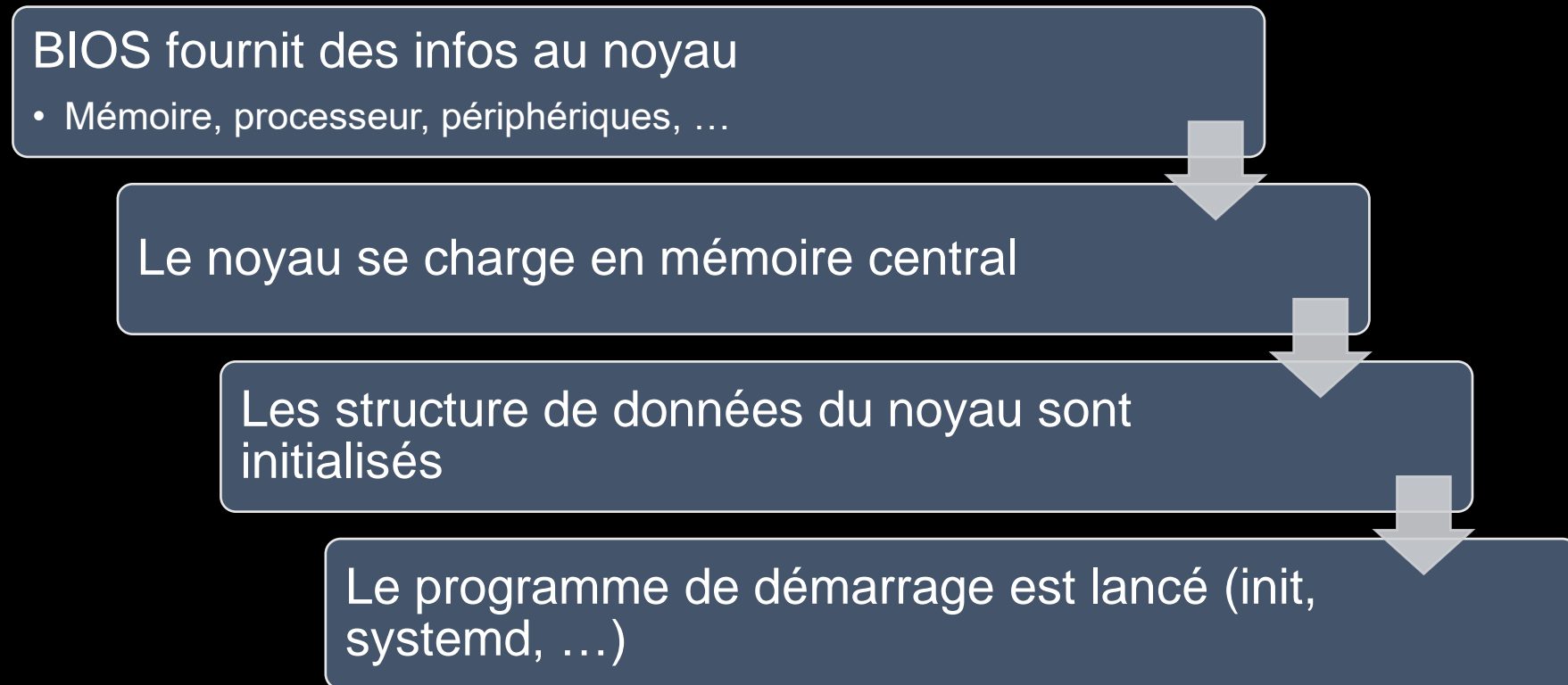
Surveillance
interopérable et
performante

NFV MONA –
Management
and
Orchestration

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Rappel sur le démarrage d'une unité d'isolation: Plusieurs composants doivent charger avant le lancement de la fonction applicative.



NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Rappel sur le démarrage d'une unité d'isolation: Plusieurs composants doivent charger avant le lancement de la fonction applicative.

BIOS fournit des infos au noyau

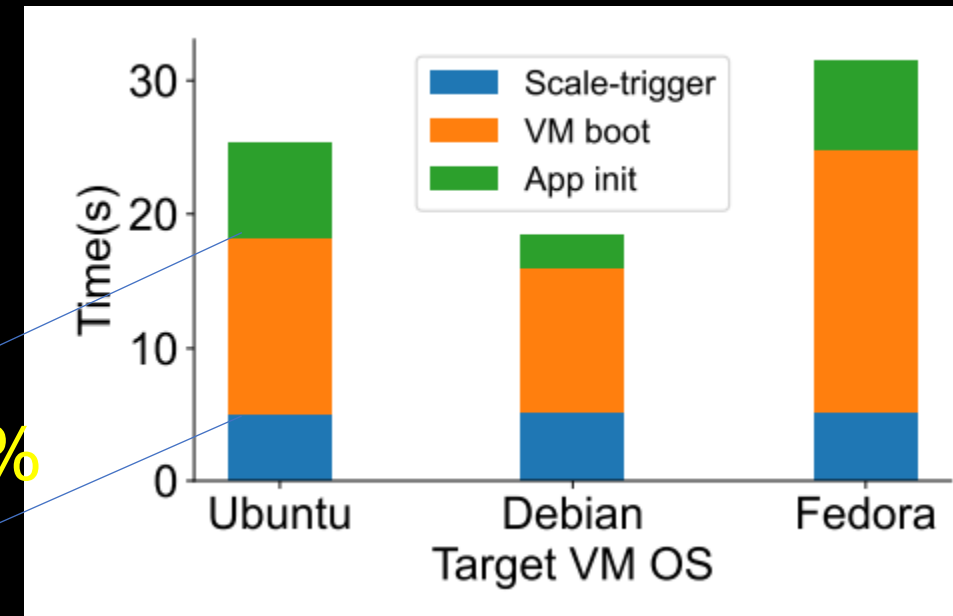
- Mémoire, processeur, périphériques, ...

Le noyau se charge en mémoire central

Les structure de données du noyau sont initialisés

Le programme de démarrage est lancé (init, systemd, ...)

~ 75%



Réduire la phase d'initialisation du noyau

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Ce temps est encore plus couteux lorsque les taches qui s'executent sont très **courtes**, de l'ordre de la milliseconde **ex: routage d'un paquet**

BIOS fournit des infos au noyau

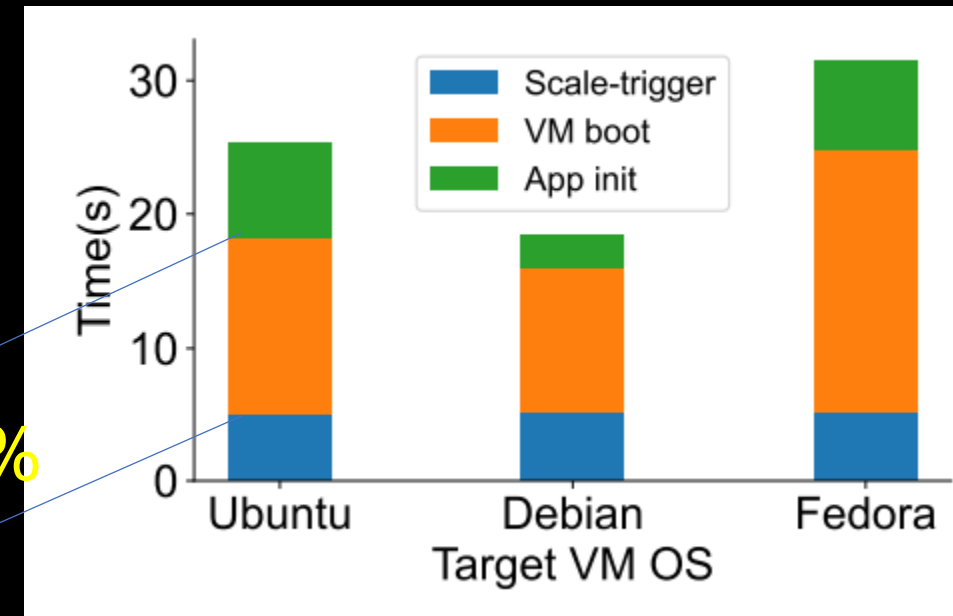
- Mémoire, processeur, périphériques, ...

Le noyau se charge en mémoire central

Les structure de données du noyau sont initialisés

Le programme de démarrage est lancé (init, systemd, ...)

~ 75%



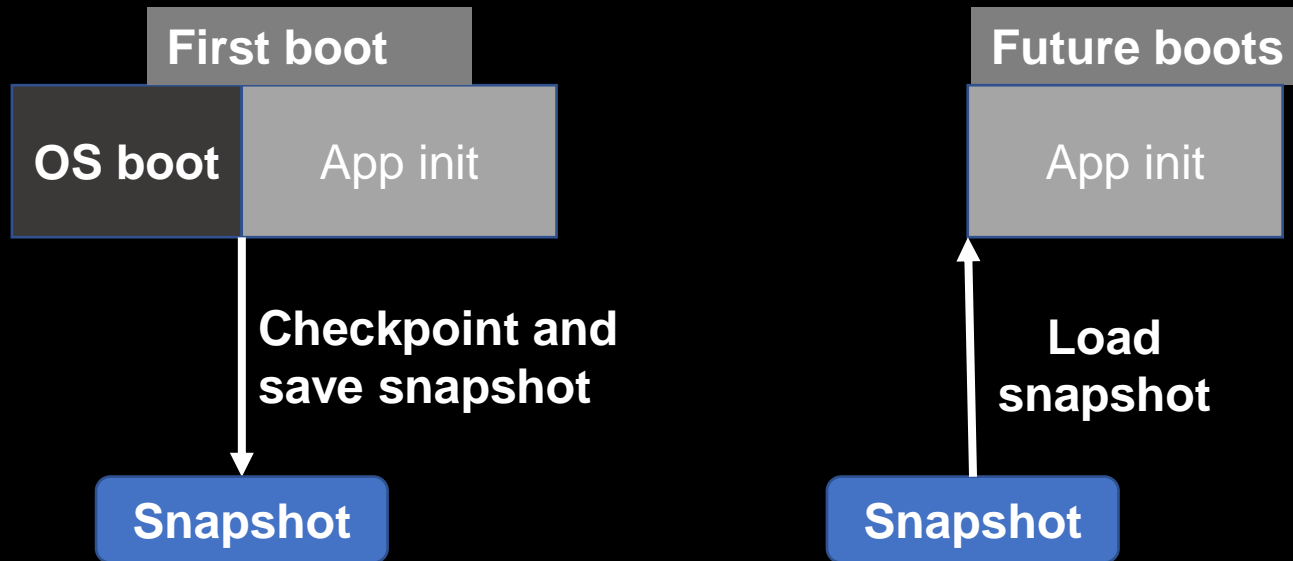
Réduire la phase d'initialisation du noyau

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Plusieurs approches dans l'état existent pour réduire la phase d'initialisation

A) Checkpointing/restore



*Potkemin-SOSP'05,
SnowFlock-EUROSYS'09,
ShadowReboot-
VEE'13/DSN'11,
Halite-ATC'13, Agamotto-
USENIX SECURITY'20,
Catalyzer-ASPLOS'20,
FaasM-ATC'20,
REAP-ASPLOS'21,
FaaSnap-EUROSYS'22
AWS SnapStart*

Space constraints and potentially infinite states

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Plusieurs approches dans l'état existent pour réduire la phase d'initialisation

A) Specialization



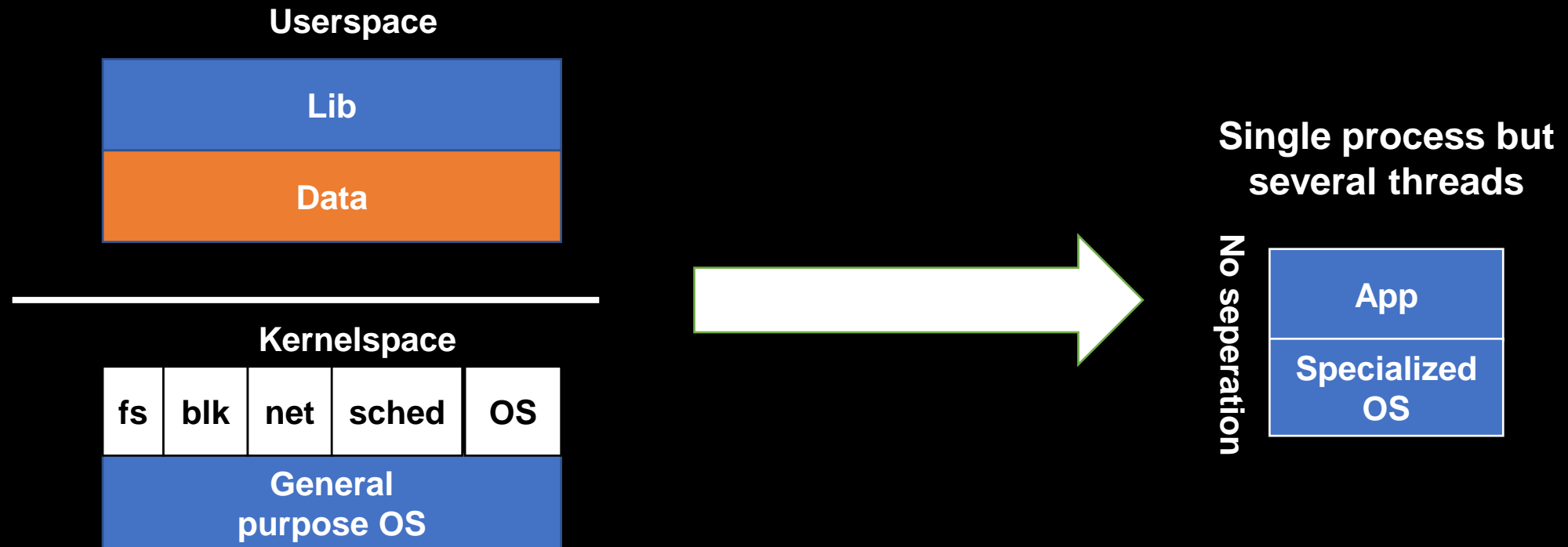
*ClickOS-NSDI'14,
OsV-ATC'14, LightVM-
SOSP'17,
Yolo-EUROPAR'19,
Firecracker-EUROSYS'20,
Unikraft-EUROSYS'21,
FlexOS-HotOS'21/ASPLOS'22*

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Utiliser des noyau optimisé pour chaque fonction

→→ **Unikernels**



NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Technologie assez récente mais plusieurs projets
par domaine

Mini-OS

IncludeOS

Unikraft
(Toolchain)

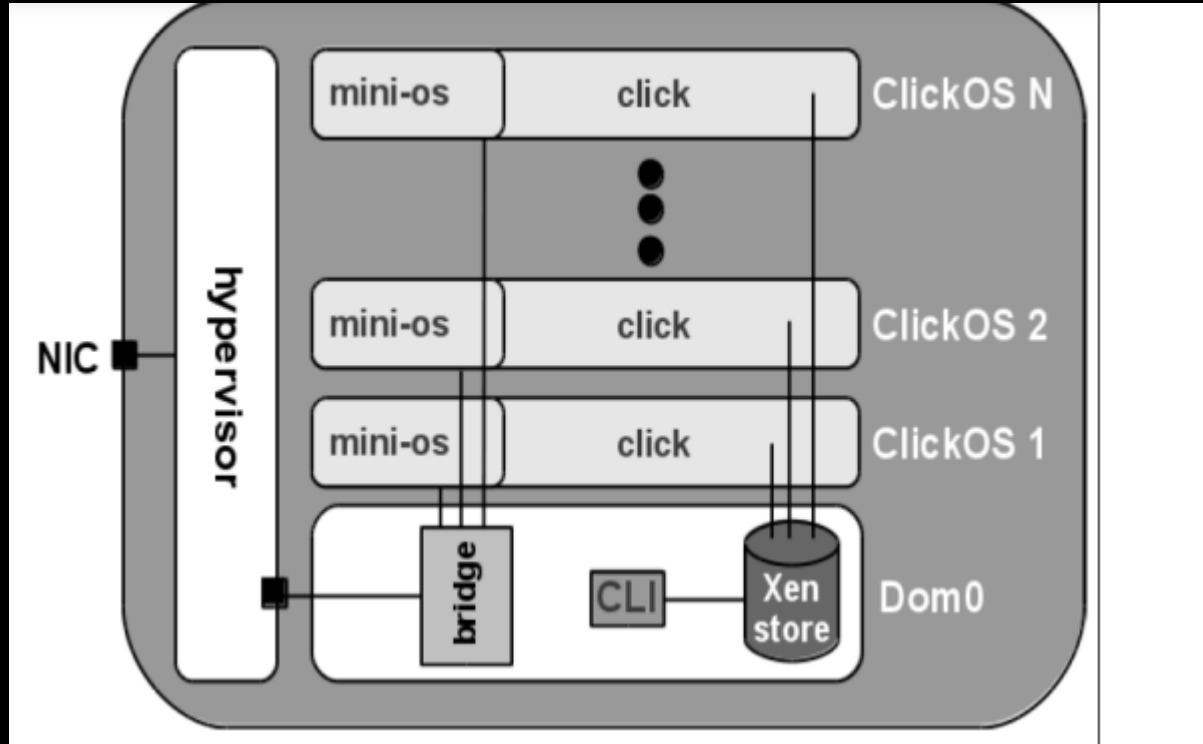
Rumprun

ClickOS

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Cas pratique: ClickOS



Se base sur miniOS (5MB de taille)

Propose une optimisation des couches "bridge" et l'interface d'administration pour retirer les fonctionnalités inutiles

Près de **30x** le temps de démarrage comparé à un OS classique.

NFV: Network Function Virtualization

Optimiser le démarrage des VNFs

Cas pratique: ClickOS

Plusieurs alternatives ont émergé au fil des années, mais la logique reste la même et insiste sur la spécialisation.

D'autres approches plus complexes insistent sur la génériqueité et peuvent aussi être exploitées pour générer des images spécialisées mais facilement reconfigurables en fonction des besoins.

Unikraft: fast, specialized unikernels the easy way. Simon Kuenzer et al. EUROSYS'21
FlexOS: towards flexible OS isolation. Hugo Lefeuvre et al. ASPLOS'22