

# Retex HashiStack

12 et 13 juin 2024

---

Journées déploiement d'applications

Mickaël MASQUELIN

# Mickaël Masquelin

Ingénieur de Recherche CNRS @ CRISTAL



 <https://www.cristal.univ-lille.fr>

 [mickael.masquelin\\_at\\_cnrs.fr](mailto:mickael.masquelin_at_cnrs.fr)

# Agenda

- Présentation du contexte ;
- La genèse du projet ;
- HashiCorp ;
- Présentation générale de la pile technique ;
- Le projet d'évolution vers une architecture « cloud » ;
- Conclusion.

# Contexte



## ID Card (1/2)

Centre de **R**echerche en **I**nformatique **S**ignal et **A**utomatique de **L**ille (CRISTAL), UMR 9189

- **Tutelles principales :**

- Université de Lille ;
- CNRS (Sciences informatiques) ;
- Centrale Lille Institut.

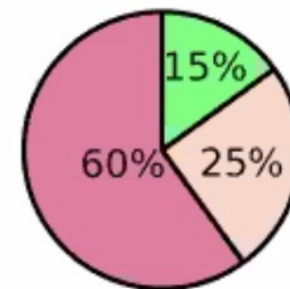
- **Place dans l'écosystème :**

- Approx. 500 membres ;
- L'une des plus grosses UMR de la région HdF.

- **Tutelle secondaire :**

- Centre Inria de l'Université de Lille

- **Profil d'activités :**



- Recherche académique
- Formation par la recherche
- Interactions socio-économiques

**Indust. :** Nombreux partenariats (par ex. Atos, Orange, Renault, Stormshield, Total, ...).



## ID Card (2/2)

### ▶ **Thématiques principales :**

- Intelligence artificielle ;
- Contrôle automatisé ;
- Bio-informatique ;
- Systèmes embarqués ;
- Interfaces Homme-Machine ;
- Calcul hautes perf. ;
- Images, signaux et leur utilisation ;
- Optimisation, recherche opérationnelle ;
- Ingénierie logicielle.

### ▶ **Champs applicatifs :**

- Cybersécurité ;
- Imagerie digitale ;
- Technologies de la santé ;
- Domotique, Internet des Objets ;
- Robotique ;
- Vente, commerce ;
- Industrie logicielle ;
- Transport et mobilité.

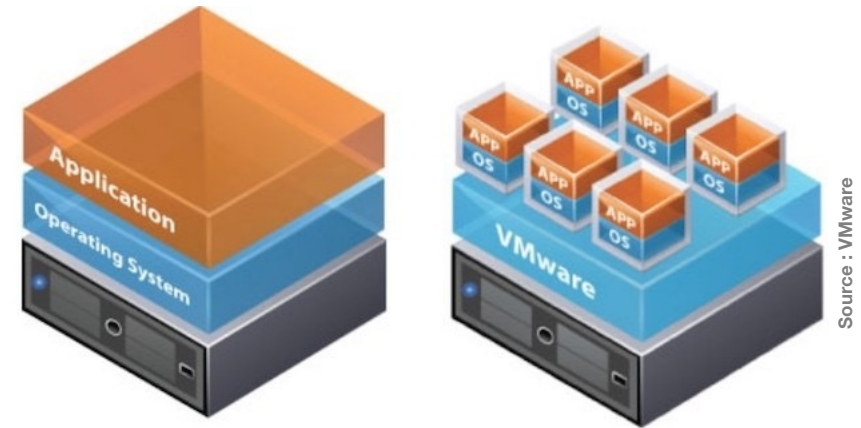
### ▶ **Champ disciplinaire :** Sciences du numérique.

# Aux origines du projet ...

# Pourquoi ce talk aujourd'hui ?



## Notre infrastructure ... avant ...



Architecture traditionnelle

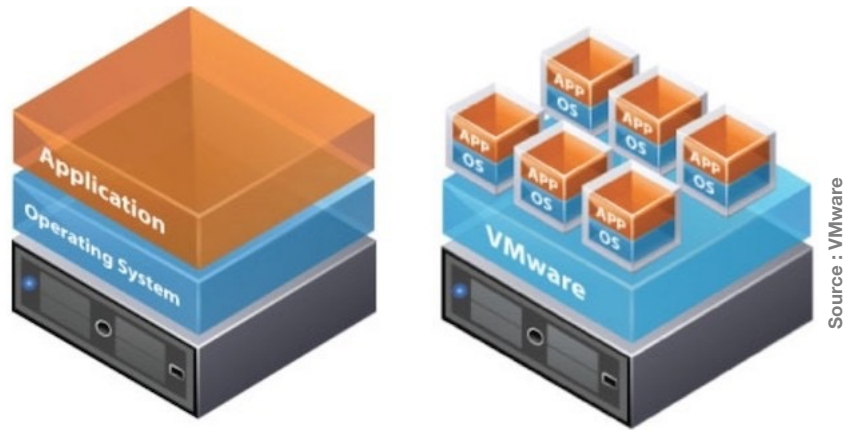
Approche via virtualisation

Hyperviseurs bare-metal VMware / Proxmox

(VMs **ET** conteneurs LXC)

# Pourquoi ce talk aujourd'hui ?

## Notre infrastructure ... avant ...



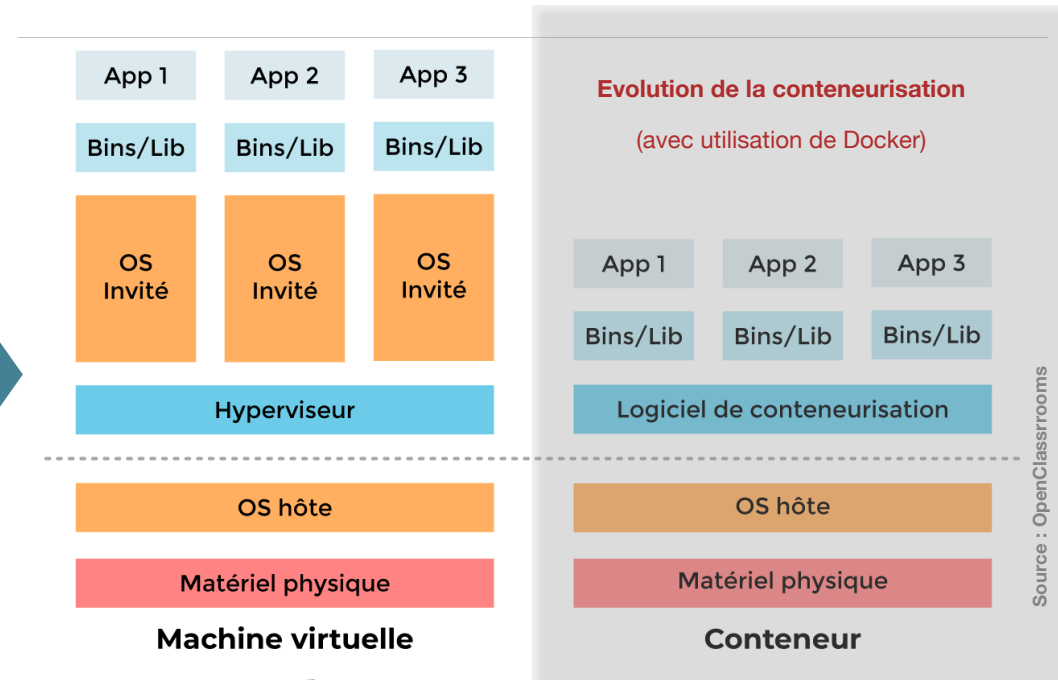
Architecture traditionnelle

Approche via virtualisation

Hyperviseurs bare-metal VMware / Proxmox

(VMs **ET** conteneurs LXC)

## ... puis progressivement ...



Machine virtuelle

Conteneur

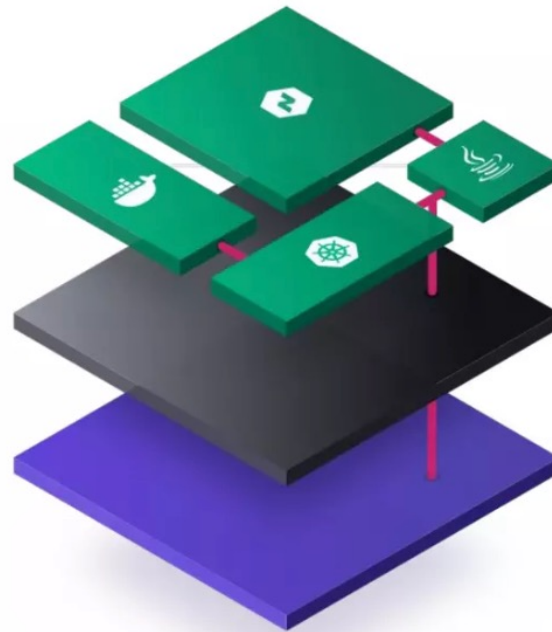
# Vers l'utilisation progressive de Docker

- Recours à des conteneurs Docker pour remplacer LXC ;
  - Gérer des « fermes » de conteneurs Docker unitairement est complexe ...
  - Comment faire fonctionner plusieurs conteneurs différents côte à côte ?
  - Comment gérer la création / suppression des conteneurs de manière automatique selon la charge ?
- ▶ Nécessité de déployer et d'utiliser **une solution technique plus adaptée** pour mieux **gérer nos flux de travail** !

# Pistes envisagées



**kubernetes**



 **Nomad**

 **Consul**

 **Vault**

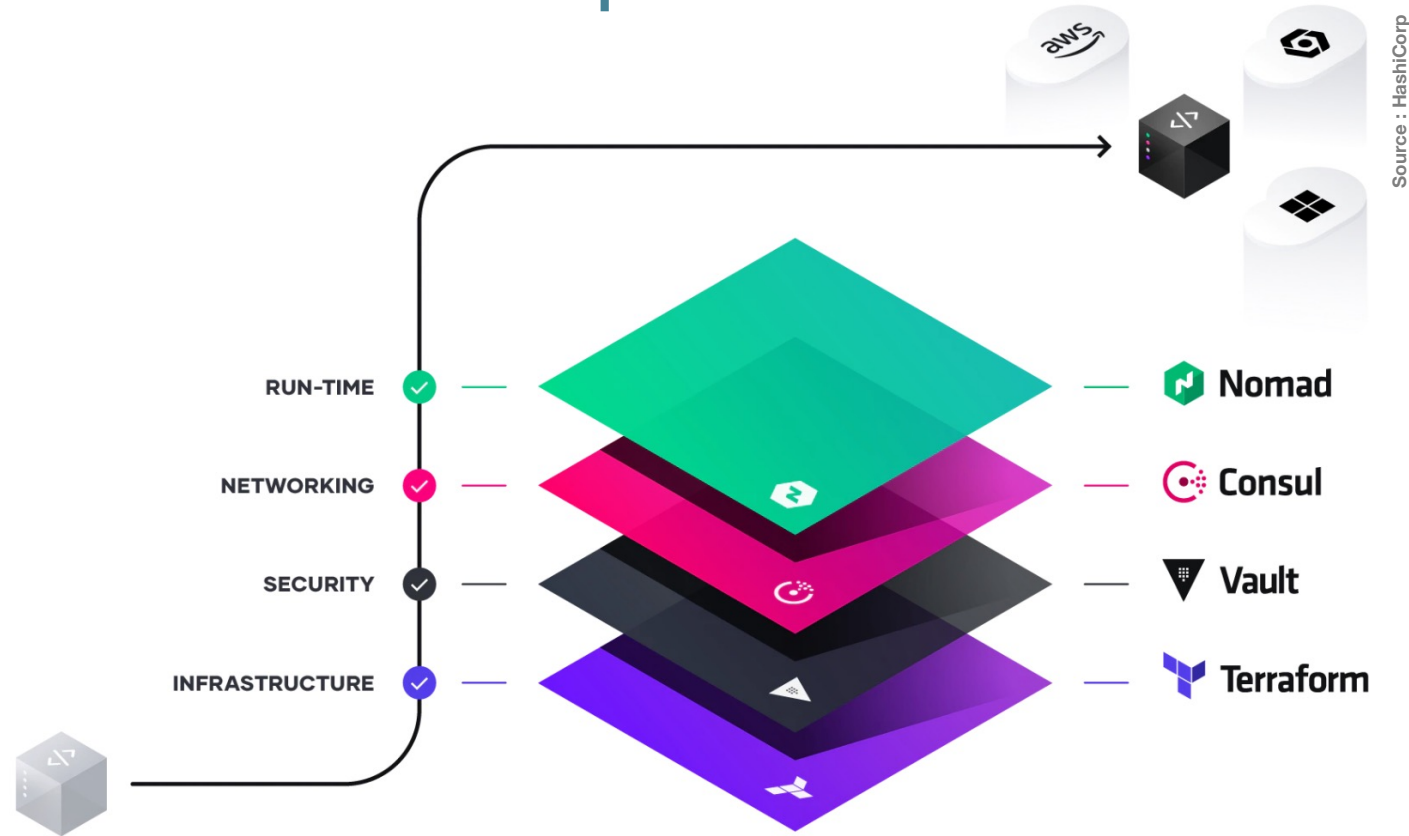
 **Terraform**

# HashiCorp

# HashiCorp

- Société américaine basée à San Francisco ;
- Fondée en 2012 ;
- Leader des solutions d'automatisation des infrastructures multi-clouds ;
- Propose des outils distribués gratuitement et pour lesquels le code source est accessible publiquement (licence BSL 1.1) ...
- ... et une version entreprise de ces mêmes outils (enrichie en fonctionnalités).


# La promesse HashiCorp



Permettre aux entreprises d'adopter des flux de travail cohérents pour provisionner, sécuriser, connecter et exécuter toute infrastructure pour toute application.

# K8s vs HashiStack

- Administration de K8s on-prem jugée « complexe » ;
- Pas fait pour les développeurs : nécessite des connaissances en systèmes et réseaux pour maîtriser K8s ;
- Sécurité des flux réseaux en mTLS native grâce à Consul Service Mesh ;
- Utilisation de Vault pour gérer les secrets ;
- Charge de travail pour Nomad : pas que des conteneurs !



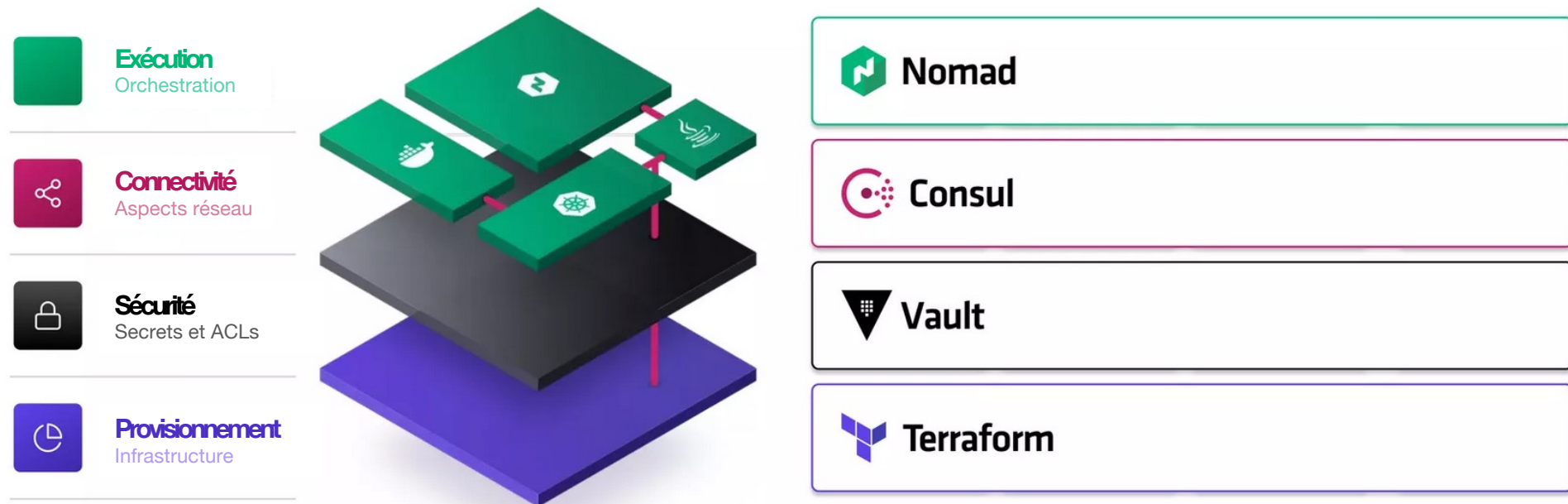
The screenshot shows a webpage titled "Kubernetes production best practices" from learnk8s.io. The page features a blue header with the title and a subtitle: "A curated checklist of best practices designed to help you release to production". Below the header, there is a section for "YOUR PROGRESS" showing "0% complete" and a list of categories: "1. Application development", "2. Governance", and "3. Cluster configuration". The "1. Application development" category is expanded, showing a list of items under "Health checks" and "Apps are independent".

YOUR PROGRESS	1. Application development
0% complete	<b>Health checks</b>
Categories	<input type="checkbox"/> Containers have Readiness probes
1. Application development	<input type="checkbox"/> Containers crash when there's a fatal error
2. Governance	<input type="checkbox"/> Configure a passive Liveness probe
3. Cluster configuration	<input type="checkbox"/> Liveness probes values aren't the same as the Readiness
	<b>Apps are independent</b>
	<input type="checkbox"/> The Readiness probes are independent

Liste de contrôles pour la prod. telle que proposée par <https://learnk8s.io>.

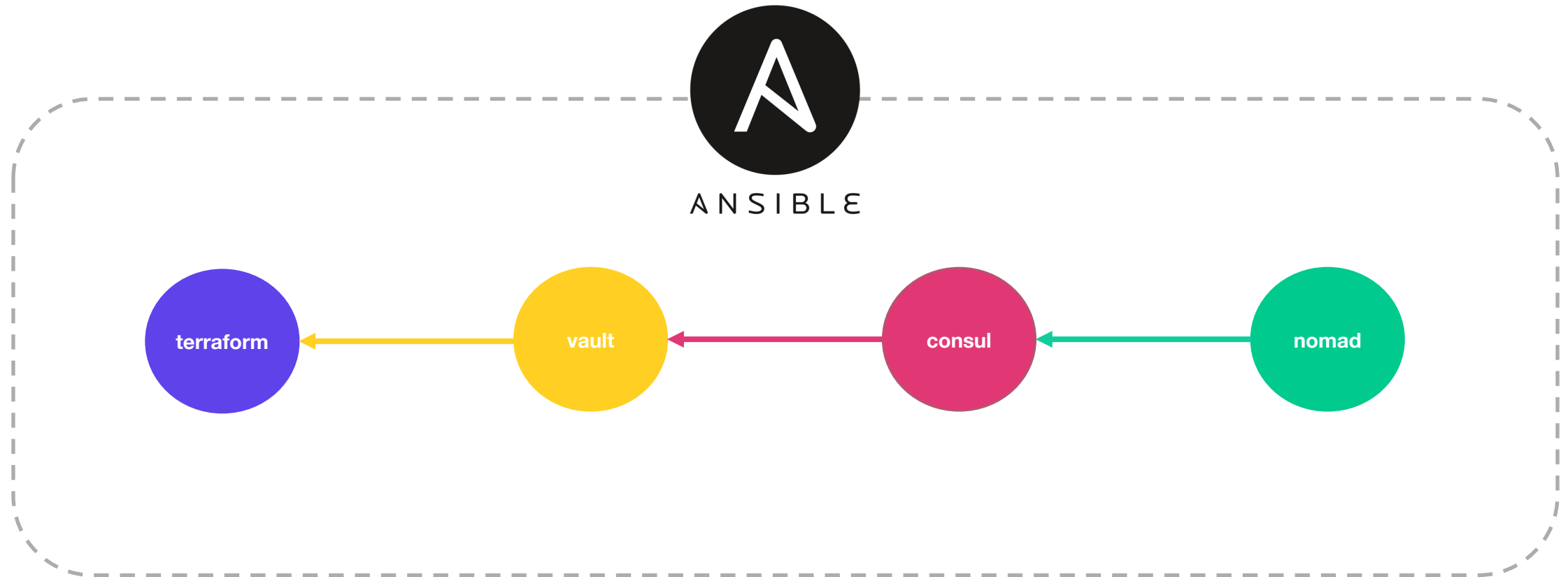
# Présentation des outils

# La pile technique



Un outil **dédié** et **adapté à chaque niveau** dans la mise en œuvre de notre « stratégie cloud »

# La chaîne des outils

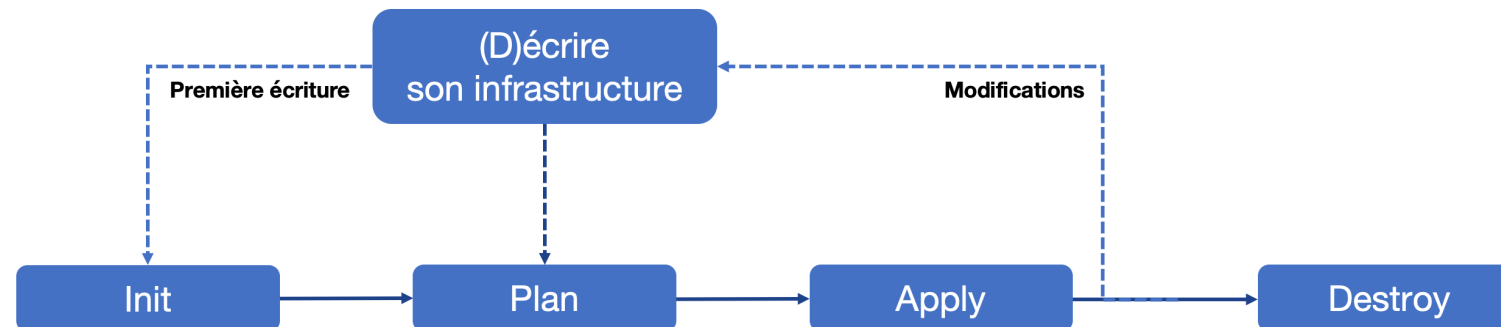


Recours à des playbooks Ansible pour réaliser la configuration de base

# Terraform

**Provisionner et administrer tous les services clouds existants et toutes les solutions d'infra on-premise**

- Solution d'IaC cross platforms, lancée en 2014 ;
- Possède une communauté open source très dynamique ;
- Utilise un langage, le **HCL**, pour définir l'infrastructure ;
- S'appuie sur la **gestion des états** pour assurer la cohérence de l'infrastructure et éviter les conflits.



# Vault

## Permettre le stockage et la gestion des secrets basée sur l'identité

- Fournit une interface qui permet d'accéder à n'importe quel secret grâce à des ACLs ainsi qu'un journal d'audit détaillé ;
- Gestion des certificats, des clefs ssh, etc.
- Révocation automatique des secrets à la fin du bail délivré par Vault ;
- Multiples intégrations (K8s, Nomad, Consul, etc.) ;
- 1 seul binaire pour toutes les opérations.

# Consul

**Découvrir, configurer et segmenter les services d'une infrastructure de manière distribuée**

- Outil de type service mesh complet qui offre un plan de contrôle complet
- Lancé en 2014 ;
- Permet de stocker des données de type clé/valeur ;
- 1 seule binaire pour toutes les opérations ;
- Excellente vidéo + article de G. Catteau (JRES 2022, Marseille).



Consul, l'outil magique pour une infrastructure dynamique :

<https://replay.jres.org/w/96qCNEzEnXrs4ZNTcYT5jd>

[https://conf-ng.jres.org/2021/document\\_revision\\_1565.html?download](https://conf-ng.jres.org/2021/document_revision_1565.html?download)

# Nomad

## Orchestrer ses conteneurs

- Développement démarré en 2015 ;
- Alternative à Kubernetes en tant qu'orchestrateur si combiné à Consul ;
- Permet d'utiliser d'autres ressources que des conteneurs : VMs QEMU, Firecracker, JAVA, GPU Nvidia, etc. ;
- Peut être déployé sur plusieurs régions et centres de données afin d'être hautement disponible et tolérant aux pannes ;
- Fourni sous la forme d'un seul binaire pour configurer des nœuds master ou piloter des minions.

# K8s vs Nomad (en tant qu'orchestrateur)

Kubernetes		Nomad	
Avantages	Inconvénients	Avantages	Inconvénients
Support communautaire important	Ecosystème assez « compliqué »	Facile à comprendre et à utiliser	Manque de maturité pour Nomad en comparaison avec Kubernetes
Plusieurs distributions disponibles	Peut entraîner des coûts « cachés »	Nécessite peu d'efforts en terme de configuration et est plutôt appropriée pour de petites équipes	Choix d'outils limités
Portabilité et flexibilité de la solution assurée	Nécessite la configuration de plusieurs composants interopérables qu'il faut gérer et installer	Fonctionne avec des modèles réseaux « courants »	Manque de contributeurs et d'un support communautaire large
Compatibilité multi-clouds	Nécessite une compréhension approfondie des concepts réseaux mis en œuvre dans K8s (MetalLB, Calico, eBPF par exemple)	Possibilité de scheduler, déployer et de gérer plusieurs types de charges serveurs (Docker, QEMU, Java, LXC, etc.)	
Solution open source		Agnostique : supporte Windows, Linux et macOS	

# Un exemple simple ...

# Un déploiement avec Nomad

```
01: job "appweb" {
02:   region = "fr"
03:   datacenters = ["principal", "secondaire"]
04:   type = "service"
05:   group "webserver" {
06:     network {
07:       mode = "bridge"
08:       port "portapp" {
09:         static = 8080
10:         to = 80
11:       }
12:     }
13:     count = 2
14:     task "server" {
15:       driver = "docker"
16:       config {
17:         image = "nginx:latest"
18:         ports = ["portapp"]
19:       }
20:     }
21:   }
22: }
```

Un extrait de manifeste Nomad au format HCL (notation JSON supportée)

## ► Job :

- Objet de haut niveau exécuté dans une région (region) donnée au sein d'un cluster (datacenter) ;
- Comprend n groupes (group) de tâches (task) ;
- Définition de la nature du job grâce à la directive type.

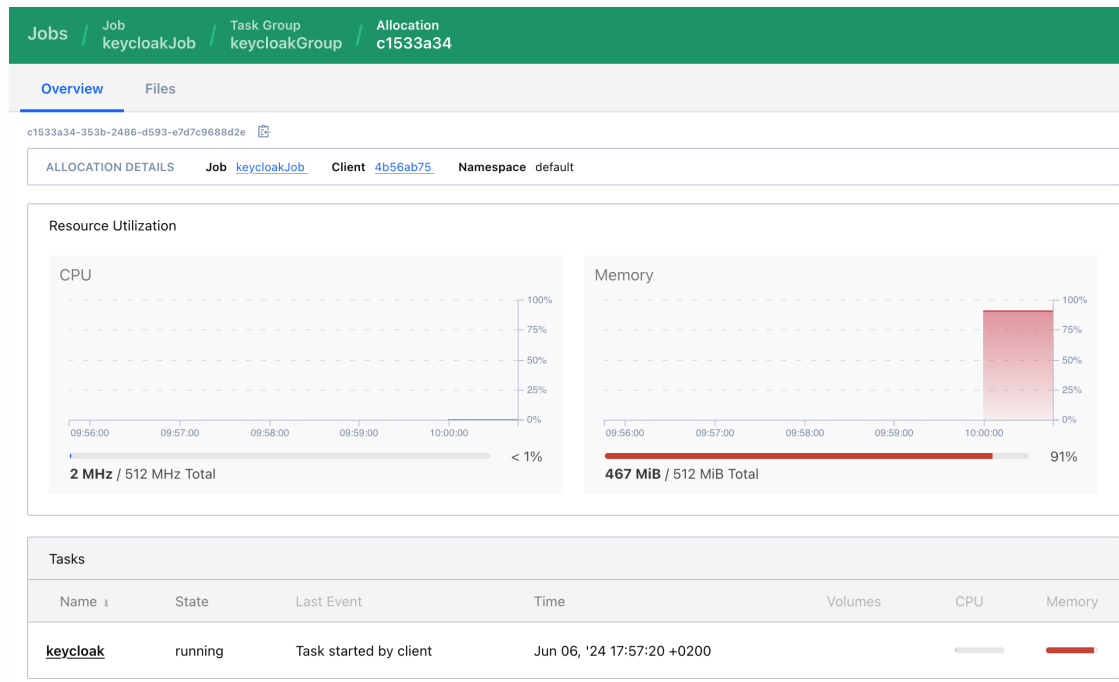
## ► Group :

- Unité de planification qui comprend n task ;
- Suppose que les tâches définies dans le group s'exécutent sur le même minion ;
- Possibilité de décrire la configuration réseau (network) par exemple.

## ► Task :

- Élément atomique ;
- Equivalent de l'objet de type pod en K8s ;
- Une task utilise un driver qui possède des propriétés particulières.

# Un déploiement avec Nomad



Ports		
Name	Host Address	Mapped Port
<code>www-test</code>		8080

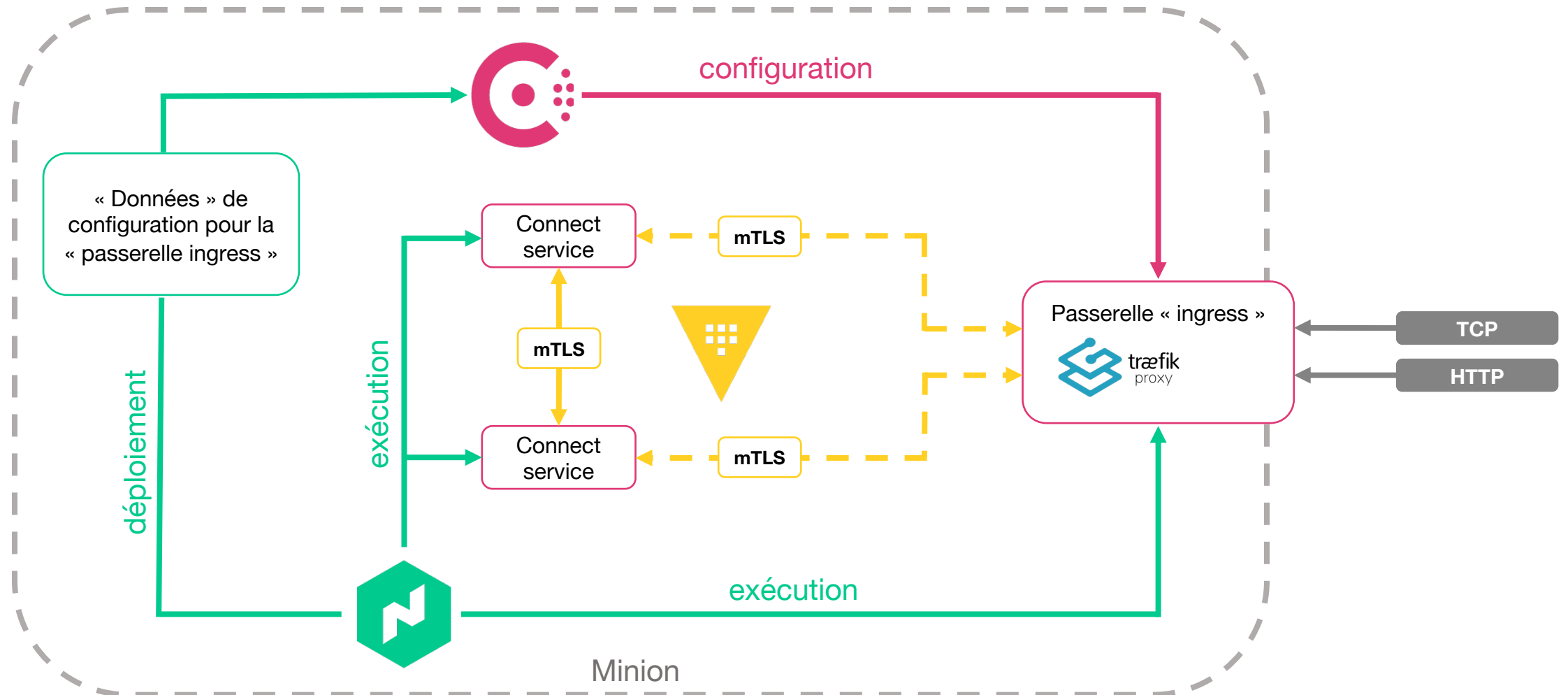
  

Services			
Name	Port	Tags	Health Check Status
<code>sso</code>	<code>www-test</code>	<code>traefik.enable=true</code> <code>traefik.http.routers.sso-nossl.entrypoints=web</code> <code>traefik.http.routers.sso-nossl.rule=Host('sso.cristal.fr')  Host(</code> <code>traefik.http.routers.sso-nossl.middlewares=http-to-https@file</code> <code>traefik.http.routers.sso.rule=Host('sso.cristal.fr')  H</code> <code>traefik.http.routers.sso.entrypoints=websecure</code> <code>traefik.http.routers.sso.tls=true</code> <code>traefik.http.routers.sso.tls.options=intermediate@file</code>	

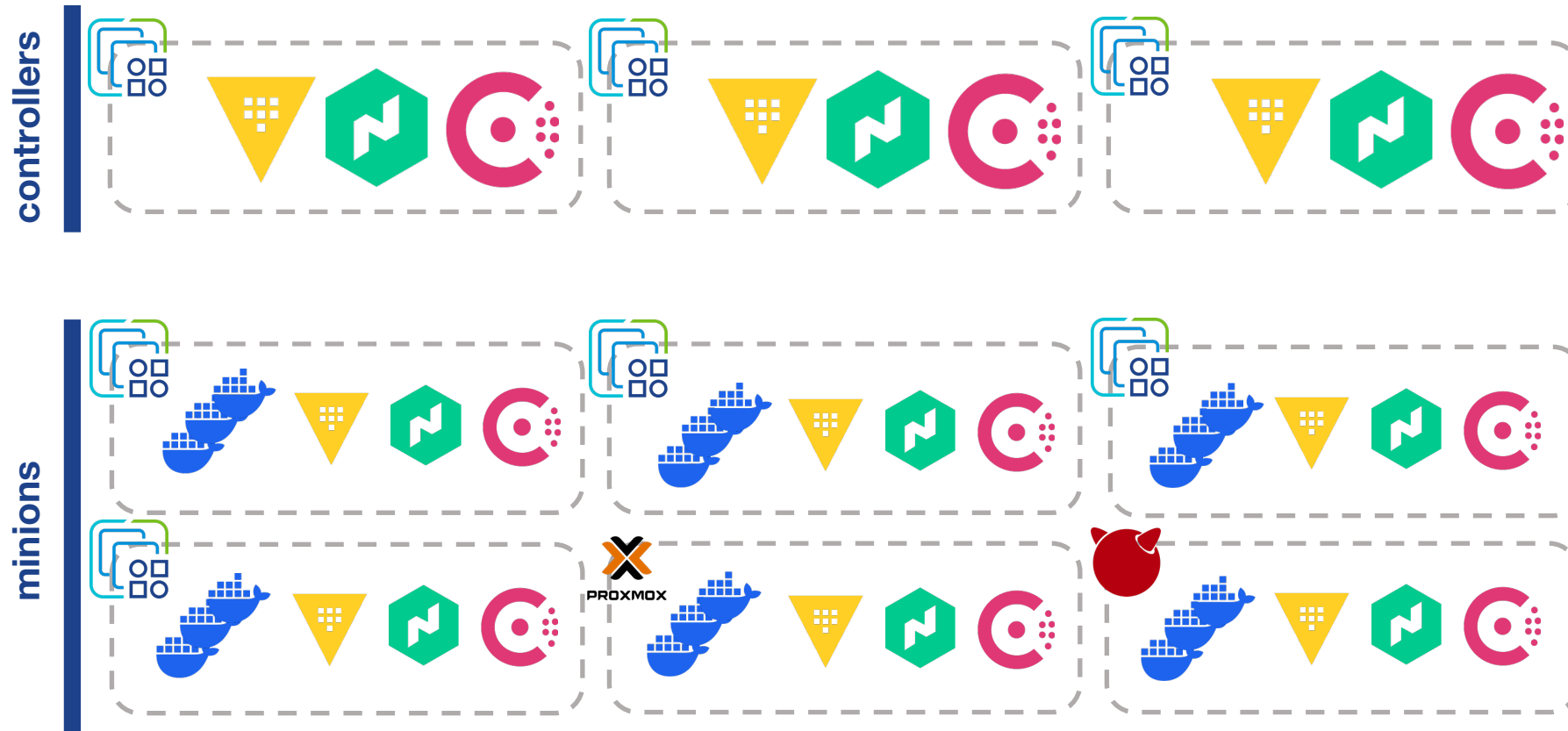
Vue graphique d'un déploiement opéré via Nomad

# Le projet d'évolution

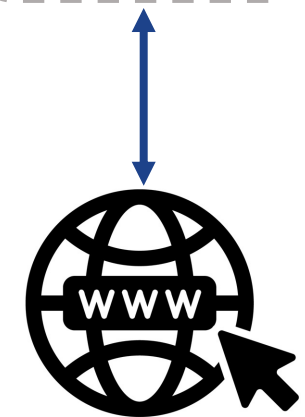
# Architecture cible souhaitée



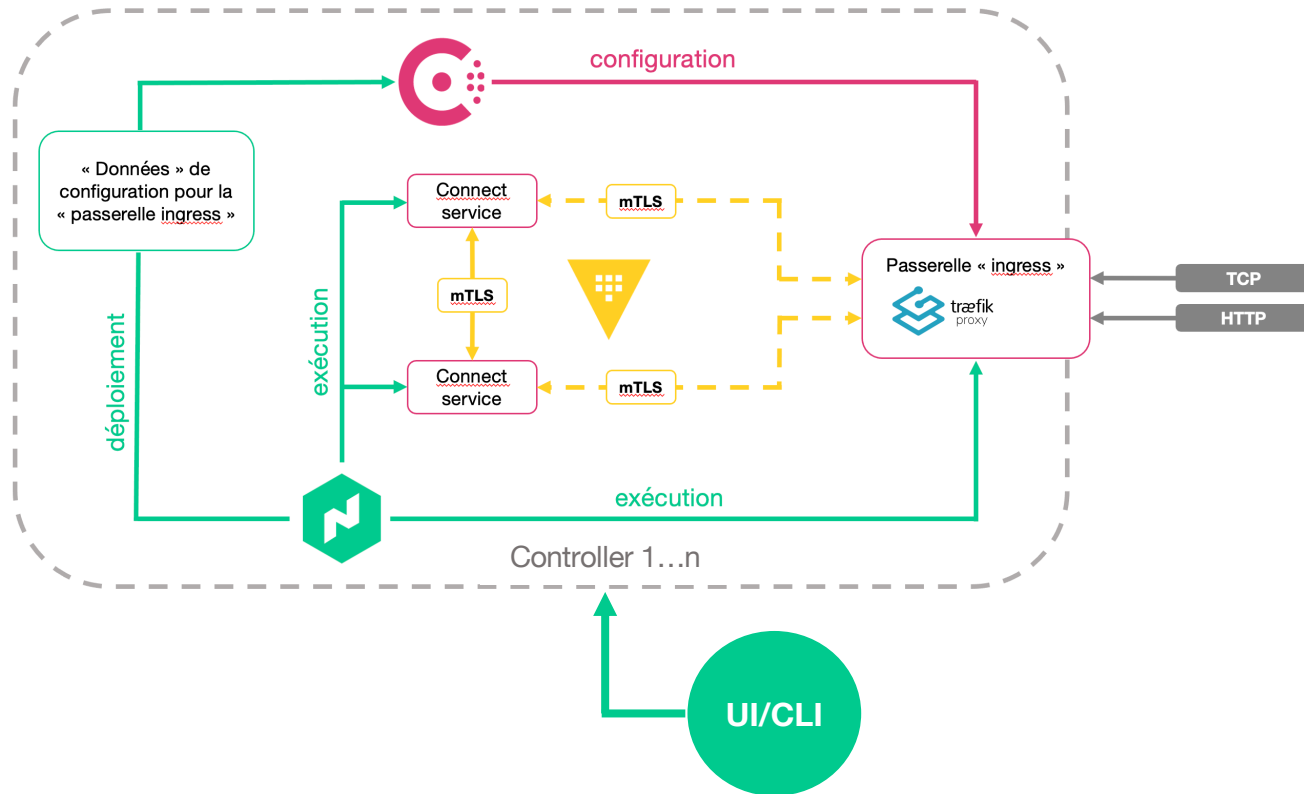
# Schéma « d'infrastructure cible » du cluster



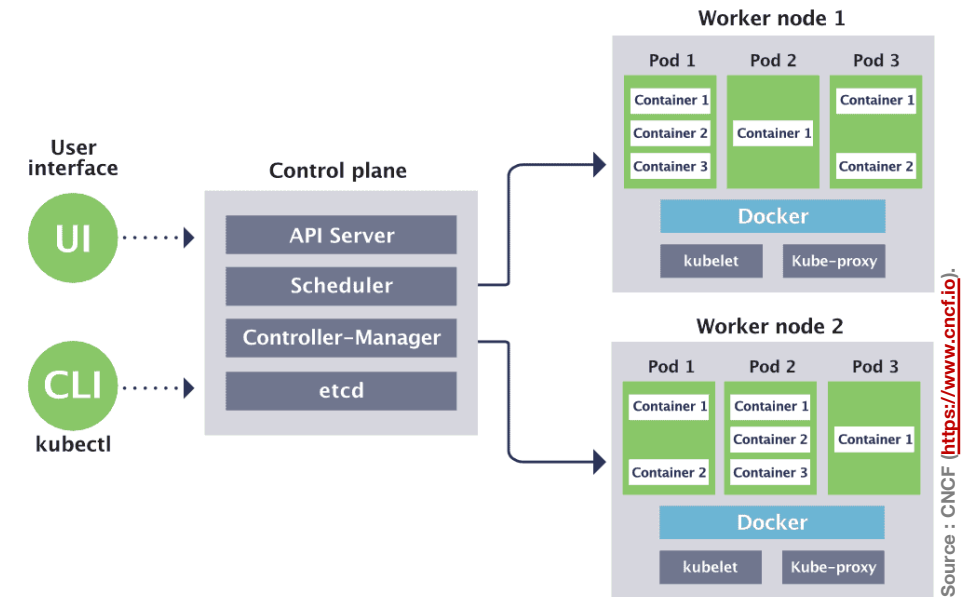
## networking télémetrie



# En comparaison ...

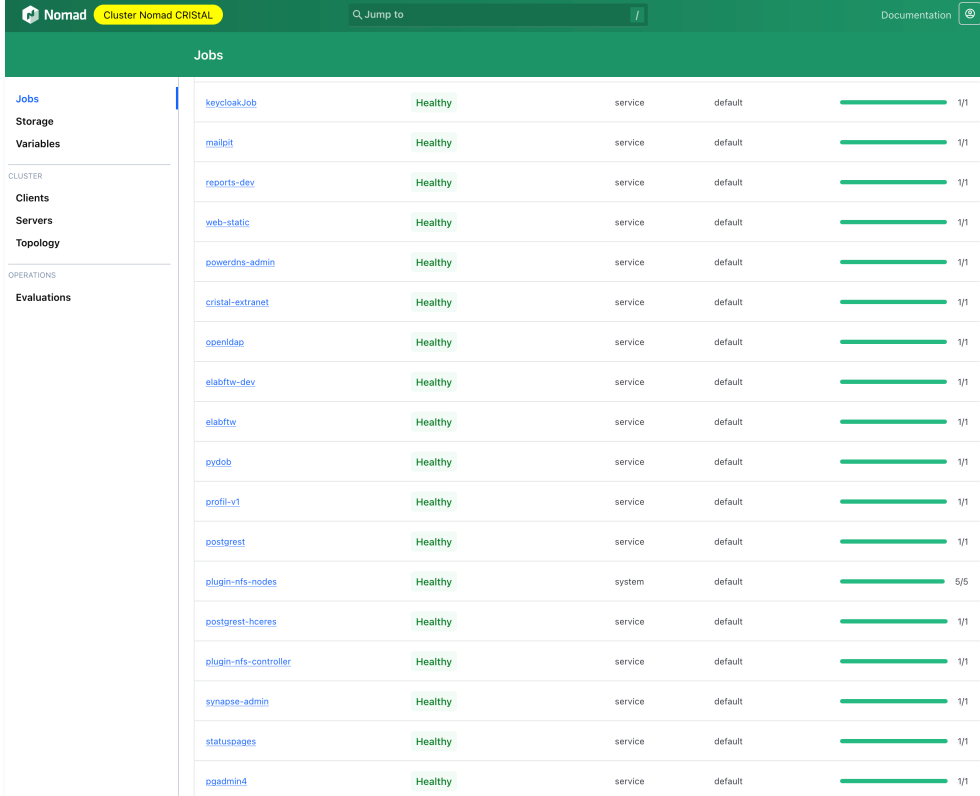


Interactions au sein de la HashiStack



Une architecture Kubernetes « classique »

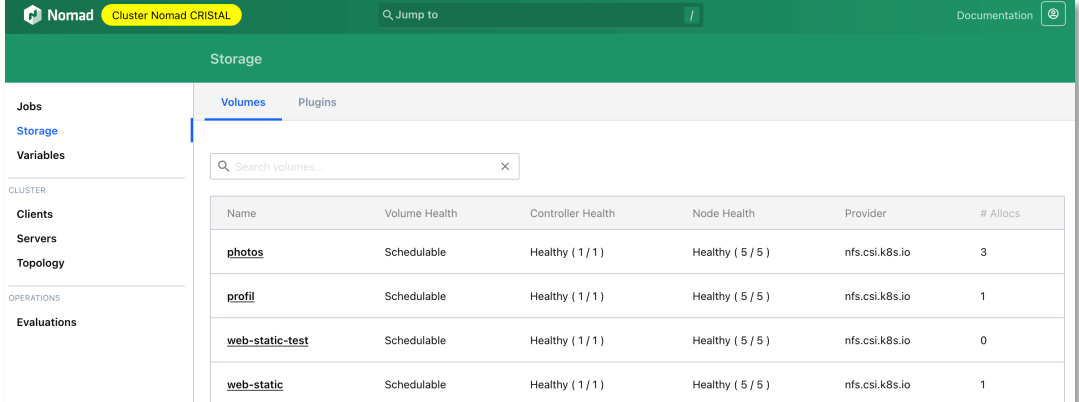
# L'interface graphique



The screenshot shows the 'Jobs' page in the Nomad web interface. The left sidebar contains navigation links for Jobs, Storage, Variables, CLUSTER (Clients, Servers, Topology), and OPERATIONS (Evaluations). The main content area displays a table of jobs with columns for job name, health status, type, default status, and progress.

Job Name	Health	Type	Default	Progress
keycloakJob	Healthy	service	default	1/1
mailpit	Healthy	service	default	1/1
reports-dev	Healthy	service	default	1/1
web-static	Healthy	service	default	1/1
powerdns-admin	Healthy	service	default	1/1
crystal-extranet	Healthy	service	default	1/1
openidap	Healthy	service	default	1/1
elabftw-dev	Healthy	service	default	1/1
elabftw	Healthy	service	default	1/1
pydob	Healthy	service	default	1/1
profil-v1	Healthy	service	default	1/1
postgrest	Healthy	service	default	1/1
plugin-nfs-nodes	Healthy	system	default	5/5
postgrest-hceres	Healthy	service	default	1/1
plugin-nfs-controller	Healthy	service	default	1/1
synapse-admin	Healthy	service	default	1/1
statuspages	Healthy	service	default	1/1
pgadmin4	Healthy	service	default	1/1

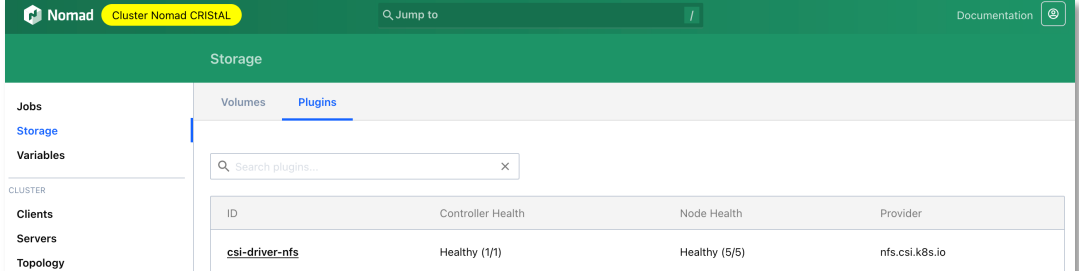
Tableau de bord web



The screenshot shows the 'Storage' page in the Nomad web interface, specifically the 'Volumes' tab. It features a search bar and a table listing storage volumes with columns for Name, Volume Health, Controller Health, Node Health, Provider, and # Allocs.

Name	Volume Health	Controller Health	Node Health	Provider	# Allocs
photos	Schedulable	Healthy (1/1)	Healthy (5/5)	nfs.csi.k8s.io	3
profil	Schedulable	Healthy (1/1)	Healthy (5/5)	nfs.csi.k8s.io	1
web-static-test	Schedulable	Healthy (1/1)	Healthy (5/5)	nfs.csi.k8s.io	0
web-static	Schedulable	Healthy (1/1)	Healthy (5/5)	nfs.csi.k8s.io	1

Aperçu de la gestion des backends de stockage

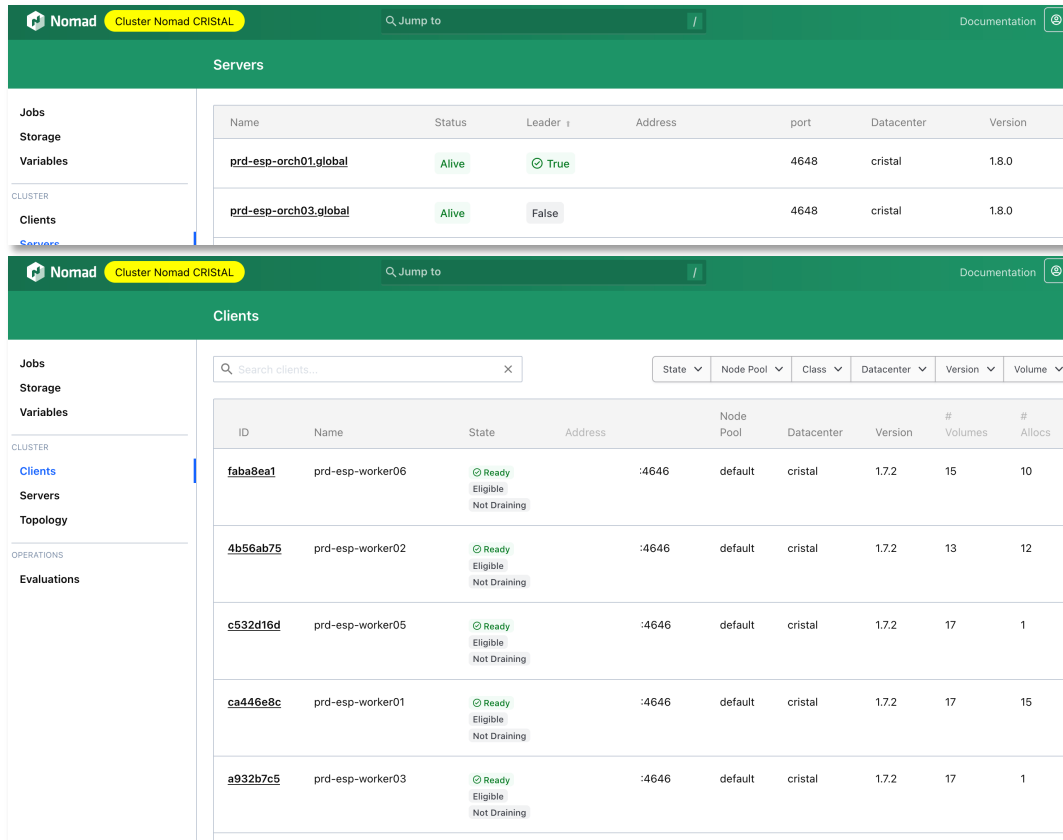


The screenshot shows the 'Storage' page in the Nomad web interface, specifically the 'Plugins' tab. It features a search bar and a table listing storage plugins with columns for ID, Controller Health, Node Health, and Provider.

ID	Controller Health	Node Health	Provider
csi-driver-nfs	Healthy (1/1)	Healthy (5/5)	nfs.csi.k8s.io

Possibilité d'utiliser les pilotes CSI

# La vue infrastructure dans Nomad

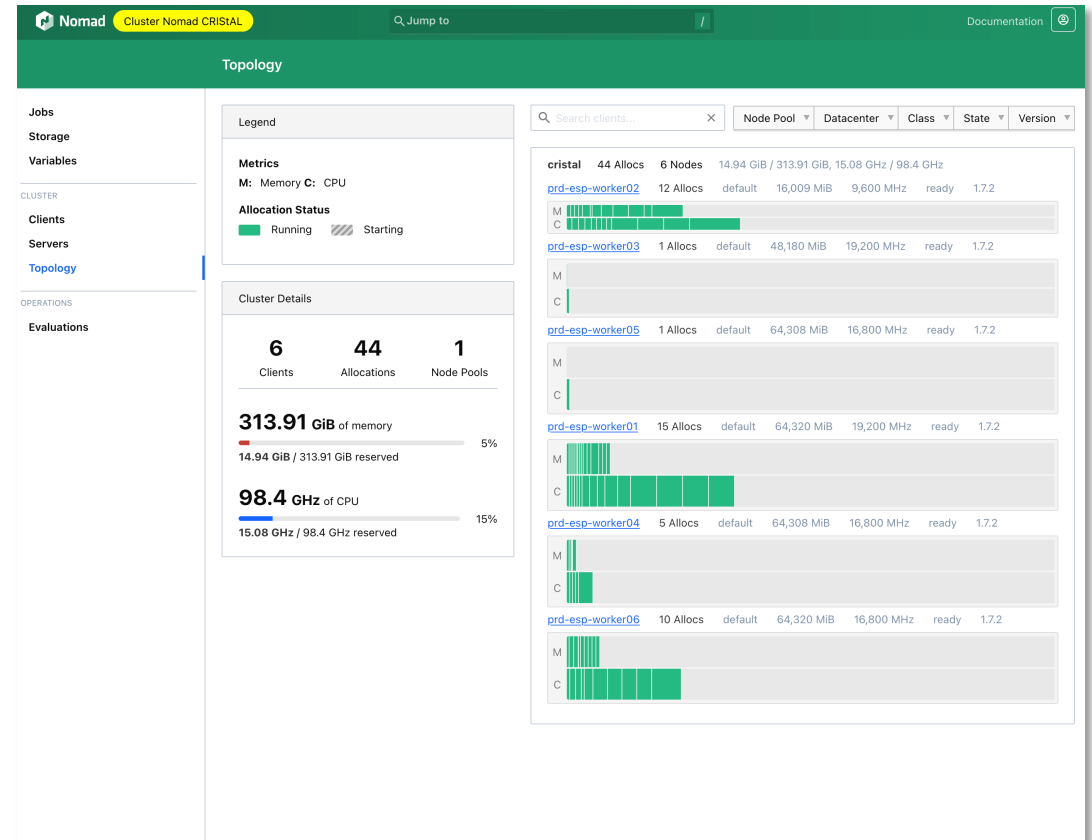


The screenshot shows two views of the Nomad infrastructure. The top view is 'Servers', showing a table with columns: Name, Status, Leader, Address, port, Datacenter, and Version. The bottom view is 'Clients', showing a table with columns: ID, Name, State, Address, Node Pool, Datacenter, Version, # Volumes, and # Allocs. The clients are listed as 'prd-esp-worker06', 'prd-esp-worker02', 'prd-esp-worker05', 'prd-esp-worker01', and 'prd-esp-worker03'.

Name	Status	Leader	Address	port	Datacenter	Version
prd-esp-orch01_global	Alive	True		4648	cristal	1.8.0
prd-esp-orch03_global	Alive	False		4648	cristal	1.8.0

ID	Name	State	Address	Node Pool	Datacenter	Version	# Volumes	# Allocs
faba8ea1	prd-esp-worker06	Ready Eligible Not Draining	:4646	default	cristal	1.7.2	15	10
4b56ab75	prd-esp-worker02	Ready Eligible Not Draining	:4646	default	cristal	1.7.2	13	12
c532d16d	prd-esp-worker05	Ready Eligible Not Draining	:4646	default	cristal	1.7.2	17	1
ca446e8c	prd-esp-worker01	Ready Eligible Not Draining	:4646	default	cristal	1.7.2	17	15
a932b7c5	prd-esp-worker03	Ready Eligible Not Draining	:4646	default	cristal	1.7.2	17	1

Vues des nœuds de contrôle (*servers*) et des minions (*clients*)



The screenshot shows the 'Topology' view in Nomad. It displays a summary of cluster metrics: 6 Clients, 44 Allocations, and 1 Node Pool. It also shows resource usage: 313.91 GiB of memory (14.94 GiB reserved, 5% usage) and 98.4 GHz of CPU (15.08 GHz reserved, 15% usage). The view includes a legend, allocation status, and a list of clients with their respective allocations and resource usage.

**Cluster Summary:**  
6 Clients, 44 Allocations, 1 Node Pools

**Resource Usage:**  
313.91 GiB of memory (14.94 GiB / 313.91 GiB reserved, 5%)  
98.4 GHz of CPU (15.08 GHz / 98.4 GHz reserved, 15%)

Client	Allocs	Node Pool	Memory	CPU	State	Version
prd-esp-worker02	12	default	16,009 MiB	9,600 MHz	ready	1.7.2
prd-esp-worker03	1	default	48,180 MiB	19,200 MHz	ready	1.7.2
prd-esp-worker05	1	default	64,308 MiB	16,800 MHz	ready	1.7.2
prd-esp-worker01	15	default	64,320 MiB	19,200 MHz	ready	1.7.2
prd-esp-worker04	5	default	64,308 MiB	16,800 MHz	ready	1.7.2
prd-esp-worker06	10	default	64,320 MiB	16,800 MHz	ready	1.7.2

Vue de synthèse avec les allocations effectuées et les ressources consommées

# Conclusion

# Perspectives

- Bilan positif : solution « cloud » on-premise polyvalente ;
- Une solution intéressante pour migrer progressivement d'une architecture classique vers une architecture plus « cloud » ;
- « Cohérence » avec le paradigme de la « *separation of concerns* » sur lequel repose Docker ;
- Des mises à jour et une maintenance quotidienne faciles ;
- Des axes d'améliorations identifiés (utilisation d'Envoy par exemple pour du mTLS sur toute la chaîne) ;
  
- Une inquiétude malgré tout ... avec le changement de licence (passage de la MPL à la BSL) ...

# Questions ?