

## Pilotage réseau avec des conteneurs

2 jours (14 heures)

Initiation

Présentiel ou classe à distance

à partir de

**1 650 € HT** par personne (TVA en vigueur)

Repas inclus en présentiel

Contenu mis à jour le 25/09/2026. Document téléchargé le 07/10/2026.

## Objectifs

A l'issue de cette formation, vous serez capable de :

- Décrire les concepts réseaux spécifiques aux conteneurs
- Configurer et gérer des réseaux Docker (bridge, macvlan, ipvlan, overlay)
- Déployer et configurer des réseaux Kubernetes (CNI, Calico, Flannel, Cilium)
- Sécuriser les communications entre conteneurs (TLS, Network Policies, service mesh)
- Diagnostiquer et résoudre les problèmes réseaux dans les environnements conteneurisés
- Optimiser les performances réseau (latence, débit, load balancing)
- Intégrer des conteneurs avec des infrastructures réseau existantes (VLAN, VPN, firewalls)
- Automatiser la gestion des réseaux conteneurisés (Ansible, Terraform, Helm).

Les compétences visées par cette formation font l'objet d'une évaluation.

## À qui s'adresse cette formation ?

Administrateurs système et réseau, développeurs travaillant avec des microservices, DevOps et SRE (Site Reliability Engineers), ingénieurs Cloud (AWS, GCP, Azure), architectes solutions.

## Prérequis

Avoir des connaissances de base en réseaux : modèle OSI, TCP/IP, adressage IP, sous-réseaux, NAT, DNS, DHCP, des notions de firewall, VLAN, VPN, load balancing, une expérience avec les conteneurs : utilisation basique de Docker (création d'images, lancement de conteneurs) et des notions d'orchestration (Kubernetes, Docker Swarm).  
Avoir des compétences Linux : ligne de commande (bash), gestion des processus, permissions, des notions de namespaces et cgroups (optionnel mais utile) et des bases en programmation / scripting (optionnel) : Python, YAML / JSON pour l'automatisation.

# Programme

## 1 Pendant votre formation

### Pilotage réseau avec des conteneurs

Jour 1 – Réseaux Docker et concepts fondamentaux

#### Introduction aux conteneurs et aux réseaux

- Rappels sur les conteneurs :
  - Différences entre conteneurs et machines virtuelles
  - Avantages des conteneurs (légèreté, portabilité, isolation)
  - Outils : Docker, containerd, CRI-O, Podman
- Isolation réseau dans les conteneurs :
  - Namespaces réseau (isolation des interfaces, tables de routage, règles iptables)
  - cgroups (limitation des ressources réseau)
  - iptables / nftables (filtrage et NAT pour les conteneurs)
- Comparaison avec les réseaux traditionnels :
  - Conteneurs vs VMs vs bare metal
  - Cas d'usage : Microservices, CI / CD, edge computing, IoT

#### Réseaux Docker : concepts et configuration

- Architecture réseau de Docker :
  - Docker daemon (dockerd) et Docker CLI (docker)
  - Drivers réseau : bridge, host, none, overlay, macvlan, ipvlan
- Réseau par défaut (bridge) :
  - Création automatique d'un réseau bridge (docker0)
  - Fonctionnement du NAT et de l'IP forwarding

- Limites : pas d'isolation entre conteneurs, problèmes de performance
- Réseaux personnalisés :
  - Création de réseaux bridge personnalisés
  - Avantages : isolation, résolution DNS entre conteneurs, sous-réseaux personnalisés
- Réseau host :
  - Partage de l'interface réseau de l'hôte
  - Avantages : meilleure performance, pas de NAT
  - Inconvénients : pas d'isolation, conflits de ports
- Réseau none :
  - Conteneur sans réseau
  - Utilisation : conteneurs isolés (ex : traitement de données)

#### Exemples de travaux pratiques (à titre indicatif)

- Lister les réseaux Docker existants et inspecter le réseau bridge par défaut
- Créer un réseau bridge personnalisé, lancer deux conteneurs sur ce réseau et tester la connectivité
- Tester le réseau host et vérifier son interface réseau
- Tester le réseau none et vérifier l'absence d'interfaces réseau

Après-midi : Réseaux avancés avec Docker

#### Réseaux macvlan et ipvlan

- Problématique : comment intégrer un conteneur directement au réseau physique (LAN) ?
- Réseau macvlan :
  - Principe : assigner une adresse MAC unique à chaque conteneur
  - Modes : bridge (par défaut), vepa, private, passthru
  - Avantages : conteneurs visibles comme des machines physiques sur le LAN
  - Inconvénients : nécessite une interface physique dédiée, problèmes de broadcast
- Réseau ipvlan :
  - Principe : partager une adresse MAC entre plusieurs conteneurs (utilise des adresses IP uniques)
  - Modes : L2 (couche 2), L3 (couche 3)
  - Avantages : moins de conflits de broadcast, meilleure scalabilité
  - Inconvénients : pas de support pour le multicast / broadcast en mode L3

#### Exemples de travaux pratiques (à titre indicatif)

- Créer un réseau macvlan en mode bridge, lancer un conteneur et vérifier qu'il a une IP dans le LAN et peut communiquer avec la gateway
- Créer un réseau ipvlan en mode L2, lancer un conteneur et vérifier la connectivité

#### Réseaux overlay pour les clusters Docker

- Problématique : comment connecter des conteneurs sur plusieurs hôtes Docker ?
- Réseau overlay :
  - Principe : création d'un réseau virtuel sur plusieurs hôtes (utilise VXLAN)
  - Utilisation : Docker Swarm, Kubernetes
  - Avantages : communication transparente entre conteneurs sur différents hôtes
  - Inconvénients : overhead dû à l'encapsulation VXLAN

#### Exemples de travaux pratiques (à titre indicatif)

- Initialiser un cluster Swarm sur le premier noeud et rejoindre le cluster avec un deuxième noeud
- Créer un réseau overlay et lancer un service sur ce réseau
- Vérifier que les conteneurs peuvent communiquer entre eux sur différents noeuds

Jour 2 – Réseaux Kubernetes et intégration avancée

### Introduction aux réseaux Kubernetes

- Architecture réseau de Kubernetes :
  - Pods : unité de déploiement (1 ou plusieurs conteneurs partageant le même réseau)
  - Services : exposition des pods (ClusterIP, NodePort, LoadBalancer, ExternalName)
  - Ingress : routage HTTP / HTTPS vers les services
  - CNI (Container Network Interface) : plugins pour la gestion des réseaux (Calico, Flannel, Cilium...)
- Exigences réseau de Kubernetes :
  - Tous les pods doivent pouvoir communiquer entre eux sans NAT
  - Tous les noeuds doivent pouvoir communiquer avec tous les pods
  - Les pods doivent avoir une IP unique dans le cluster

### CNI : plugins réseau pour Kubernetes

- Flannel : simple et léger, utilise VXLAN ou host-gw – idéal pour les petits clusters
- Calico : réseau L3 avec BGP, support des Network Policies – bonnes performances, scalable
- Cilium : utilise eBPF pour des performances optimales – support avancé des Network Policies et de la visibilité
- Multus : permet d'attacher plusieurs interfaces réseau à un pod – utile pour les cas d'usage avancés (ex : pods avec une interface dans un VLAN et une autre dans un overlay)
- Weave Net : simple à déployer, utilise VXLAN ou fastdp

#### Exemples de travaux pratiques (à titre indicatif)

- Installer un cluster Kubernetes (ex : avec kubeadm)
- Déployer Flannel et vérifier la connectivité entre les pods
- Supprimer Flannel, déployer Calico et vérifier la connectivité
- Supprimer Calico, déployer Cilium, vérifier la connectivité et explorer les fonctionnalités eBPF

Après-midi : Sécurité, performance et intégration

### Sécurité des réseaux conteneurisés

- Risques spécifiques :
  - Attaques entre pods (ex : pods malveillants)
  - Fuites de données (ex : trafic non chiffré)
  - Problèmes de configuration (ex : Network Policies trop permissives)
- Bonnes pratiques :
  - Network Policies : restriction du trafic entre pods
  - Chiffrement : TLS / mTLS pour les communications entre services
  - Service Mesh : Istio, Linkerd (pour le chiffrement, l'authentification, l'observabilité)
  - Isolation des noeuds : utilisation de VLANs ou de firewalls pour séparer les environnements (dev, prod)
  - Mises à jour : patch management pour les images de conteneurs et les CNI

#### *Exemples de travaux pratiques (à titre indicatif)*

- Déployer deux services (frontend et backend) et tester la connectivité entre les pods
- Créer une Network Policy pour restreindre l'accès au backend (seul le frontend peut accéder au backend)
- Appliquer la Network Policy et vérifier son efficacité

### **Optimisation des performances réseau**

- Problématiques : latence élevée, goulots d'étranglement, saturation de la bande passante
- Solutions :
  - Choix du CNI : Cilium (eBPF) pour des performances optimales
  - Load balancing : utilisation de services Kubernetes (ClusterIP, NodePort, LoadBalancer)
  - QoS (Quality of Service) : limitation de la bande passante pour certains pods
  - Observabilité : outils comme Prometheus, Grafana et les dashboards Kubernetes pour surveiller les performances

#### *Exemples de travaux pratiques (à titre indicatif)*

- Déployer un service avec plusieurs réplicas et configurer un load balancer pour répartir le trafic entre les pods
- Mettre en place des limites de bande passante pour certains pods
- Déployer des outils d'observabilité (Prometheus, Grafana) pour surveiller les performances

### **Intégration avec les infrastructures réseau existantes**

- Problématiques : comment connecter des conteneurs à des réseaux physiques ou virtuels existants ?
- Solutions :
  - Utilisation de macvlan / ipvlan pour connecter des conteneurs au LAN
  - Intégration avec des VLANs (via Multus ou des CNI spécifiques)
  - Connexion à des VPNs (ex : OpenVPN, WireGuard)
  - Utilisation de firewalls et de load balancers externes

#### *Exemples de travaux pratiques (à titre indicatif)*

- Configurer un réseau macvlan pour connecter des conteneurs au LAN

- Configurer un réseau VLAN pour isoler certains pods
- Configurer un VPN pour sécuriser les communications entre conteneurs et un réseau distant

### Automatisation de la gestion des réseaux conteneurisés

- Outils d'automatisation :
  - Ansible : automatisation des configurations réseau
  - Terraform : provisionnement d'infrastructures réseau
  - Helm : déploiement d'applications Kubernetes avec leurs configurations réseau
  - Scripts Python : automatisation via les APIs Kubernetes et Docker

### Exemples de travaux pratiques (à titre indicatif)

- Utiliser Ansible pour configurer des réseaux Docker sur plusieurs hôtes
- Utiliser Terraform pour provisionner un cluster Kubernetes avec un CNI spécifique
- Déployer une application Kubernetes avec Helm, incluant ses configurations réseau

## Financement

Plusieurs dispositifs de financement sont accessibles via les OPCO (Opérateurs de Compétences), organismes agréés par le ministère du Travail dont le rôle est d'accompagner, collecter et gérer les contributions des entreprises au titre du financement de la formation professionnelle.

Pour plus d'information, une équipe de gestionnaires SKOLAE Formation spécialisée vous accompagne dans le choix de vos formations et la gestion administrative.

[Nous contacter pour en savoir plus →](#)

## Modalités pédagogiques

Formation délivrée en présentiel ou distanciel\* (blended-learning, e-learning, classe virtuelle, présentiel à distance).

Le formateur alterne entre méthode\*\* démonstrative, interrogative et active (via des travaux pratiques et/ou des mises en situation).

Variables suivant les formations, les moyens pédagogiques mis en oeuvre sont :

- Ordinateurs Mac ou PC (sauf pour certains cours de l'offre Management), connexion internet fibre, tableau blanc ou paperboard, vidéoprojecteur ou écran tactile interactif (pour le distanciel)
- Environnements de formation installés sur les postes de travail ou en ligne
- Supports de cours et exercices

En cas de formation intra sur site externe à M2i, le client s'assure et s'engage également à avoir toutes les ressources matérielles pédagogiques nécessaires (équipements informatiques...) au bon déroulement de l'action de formation visée conformément aux prérequis indiqués dans le programme de formation communiqué.

\* nous consulter pour la faisabilité en distanciel

\*\* ratio variable selon le cours suivi

### Accessibilité aux personnes en situation de handicap

## Détails de la session

Du 8 octobre 2026 au 9 octobre 2026 (2 jours / 14 heures)

Paris La Défense

Du 8 octobre 2026 au 9 octobre 2026

Paris La Défense

- 8 octobre 2026 ..... 9h00 - 16h30
- 9 octobre 2026 ..... 9h00 - 16h30