Essentiels de l'infrastructure du SI

Wiki complet sur les infrastructures systèmes et réseaux : TCP/IP, serveurs (DNS, DHCP, annuaires), virtualisation, containerisation Docker, Infrastructure As Code (Terraform, Ansible) et Kubernetes

Systèmes 120 min de lecture Niveau Intermédiaire

Document généré le 27/06/2026 à 21h28 · nouvy.fr/wiki/essentiels-infrastructure-si

Sommaire

42 section(s) · 120 min de lecture

Wiki complet sur les infrastructures systèmes et réseaux pour concevoir des systèmes applicatifs et mettre en place des environnements de production.

Introduction

Ce wiki couvre les notions essentielles des infrastructures systèmes et réseaux nécessaires pour :

- Collaborer avec les opérateurs
- Développer des applications distribuées compatibles avec les infrastructures de production
- Mettre en œuvre des environnements de développement stables et cohérents
- Intervenir dans le choix des composants d'infrastructure hébergeant les applications

Objectifs d'apprentissage

- Comprendre les notions essentielles des infrastructures systèmes et réseaux pour concevoir des systèmes applicatifs
- Mettre en place un environnement simulant une infrastructure de production
- Utiliser les outils d'Infrastructure As Code

Prérequis

- Bases en réseaux

Partie 1 : Infrastructures réseaux

1.1 Modèle TCP/IP et protocoles associés

Architecture TCP/IP

Le modèle TCP/IP est composé de 4 couches :

Couche	Protocoles	Fonction
Application	HTTP, HTTPS, FTP, SSH, SMTP, DNS, DHCP	Interface avec les applications
Transport	TCP, UDP	Gestion de la communication entre applications
Internet	IP, ICMP, ARP	Routage et adressage logique
Accès réseau	Ethernet, Wi-Fi, PPP	Transmission physique des données

Protocole TCP (Transmission Control Protocol)

- **Connexion orientée** : Établit une connexion avant l'envoi de données
- **Fiabilité** : Garantit la livraison des données (acknowledgment)
- **Contrôle de flux** : Gère le débit pour éviter la saturation
- **Contrôle de congestion** : Adapte le débit selon la charge réseau

Protocole UDP (User Datagram Protocol)

- **Sans connexion** : Pas d'établissement de connexion
- **Non fiable** : Pas de garantie de livraison
- **Rapide** : Moins de surcharge que TCP
- **Utilisé pour** : DNS, DHCP, streaming vidéo, VoIP

Protocole IP (Internet Protocol)

IPv4 :

- Adresses 32 bits (4 octets)
- Format : `192.168.1.1`
- Environ 4,3 milliards d'adresses
- Classes : A (1-126), B (128-191), C (192-223)

Masque de sous-réseau :

- Permet de diviser un réseau en sous-réseaux
- Exemple : `255.255.255.0` = `/24` = 256 adresses

CIDR (Classless Inter-Domain Routing) :

- Notation : `192.168.1.0/24`
- Permet une allocation flexible des adresses

1.2 LAN / MAN / WAN

LAN (Local Area Network)

- **Portée** : Bâtiment ou campus
- **Technologie** : Ethernet (câble), Wi-Fi
- **Débit** : 100 Mbps à 10 Gbps
- **Gestion** : Locale, par l'organisation
- **Exemples** : Réseau d'entreprise, réseau domestique

MAN (Metropolitan Area Network)

- **Portée** : Ville ou région métropolitaine
- **Technologie** : Fibre optique, liaisons radio
- **Débit** : 1 Gbps à 100 Gbps
- **Gestion** : Opérateur télécom ou collectivité
- **Exemples** : Réseau de la ville, réseau universitaire

WAN (Wide Area Network)

- **Portée** : Pays, continent, monde
- **Technologie** : Fibre optique longue distance, satellite

- **Débit** : Variable selon la distance
- **Gestion** : Opérateurs télécom internationaux
- **Exemples** : Internet, réseau d'une multinationale

1.3 Focus sur IP (IPv4 & IPv6)

IPv4 (Internet Protocol version 4)

Structure d'une adresse IPv4 :

- 32 bits = 4 octets
- Format décimal : `192.168.1.1`
- Format binaire : `11000000.10101000.00000001.00000001`

Classes d'adresses :

Classe	Plage	Masque par défaut	Usage
A	1.0.0.0 - 126.255.255.255	/8	Grandes organisations
B	128.0.0.0 - 191.255.255.255	/16	Organisations moyennes
C	192.0.0.0 - 223.255.255.255	/24	Petites organisations
D	224.0.0.0 - 239.255.255.255	-	Multicast
E	240.0.0.0 - 255.255.255.255	-	Réservé

Adresses privées (RFC 1918) :

- `10.0.0.0/8` : 10.0.0.0 - 10.255.255.255
- `172.16.0.0/12` : 172.16.0.0 - 172.31.255.255
- `192.168.0.0/16` : 192.168.0.0 - 192.168.255.255

Adresses spéciales :

- `127.0.0.1` : Localhost (boucle locale)
- `0.0.0.0` : Toutes les interfaces
- `255.255.255.255` : Broadcast

IPv6 (Internet Protocol version 6)

Structure d'une adresse IPv6 :

- 128 bits = 16 octets
- Format hexadécimal : `2001:0db8:85a3:0000:0000:8a2e:0370:7334`
- Format abrégé : `2001:db8:85a3::8a2e:370:7334`

Avantages d'IPv6 :

- **Espace d'adressage** : 3,4 × 10^{38} adresses
- **Simplification** : En-têtes plus simples
- **Sécurité** : IPSec intégré
- **Auto-configuration** : SLAAC (Stateless Address Autoconfiguration)
- **Multicast** : Amélioré

Types d'adresses IPv6 :

- **Unicast** : Communication point à point
- **Multicast** : Communication un-à-plusieurs
- **Anycast** : Communication vers le plus proche

Adresses IPv6 spéciales :

- `::1` : Localhost
- `::` : Adresse non spécifiée
- `fe80::/10` : Link-local
- `fc00::/7` : Unique Local Address (ULA)

1.4 Éléments actifs du réseau

Hub (Concentrateur)

- **Fonction** : Répète le signal sur tous les ports
- **Domaine de collision** : Un seul pour tous les ports
- **Débit** : Partagé entre tous les ports
- **Utilisation** : Obsolète, remplacé par les switches

Switch (Commutateur)

- **Fonction** : Commute les trames entre ports selon l'adresse MAC
- **Domaine de collision** : Un par port (full-duplex)
- **Table MAC** : Mémorise les adresses MAC et leurs ports
- **Types** :
    - **Switch non managé** : Plug & Play
    - **Switch managé** : Configuration avancée (VLAN, STP, etc.)
    - **Switch Layer 3** : Routage intégré

Routeur

- **Fonction** : Route les paquets entre réseaux différents
- **Table de routage** : Contient les routes vers les réseaux
- **Interfaces** : Au moins 2 interfaces réseau
- **Protocoles de routage** :
    - **Statique** : Routes configurées manuellement
    - **Dynamique** : OSPF, EIGRP, BGP

Firewall

- **Fonction** : Filtre le trafic selon des règles de sécurité
- **Types** :
    - **Firewall réseau** : Filtre au niveau IP/TCP/UDP
    - **Firewall applicatif** : Filtre au niveau application
    - **Firewall stateful** : Suit l'état des connexions
- **Règles** : Autoriser/Refuser selon source, destination, port, protocole

Proxy

- **Fonction** : Intermédiaire entre clients et serveurs

- **Types** :
    - **Proxy HTTP** : Cache les pages web
    - **Proxy inverse** : Répartit la charge (load balancing)
    - **Proxy transparent** : Invisible pour les clients
- **Avantages** :
    - Cache (réduction de la bande passante)
    - Filtrage de contenu
    - Anonymat
    - Logs centralisés

1.5 Tolérance aux pannes des réseaux

Redondance des liens

LAG (Link Aggregation Group) :

- Agrégation de plusieurs liens physiques en un lien logique
- Augmente la bande passante
- Fournit la redondance
- Protocoles : LACP (Link Aggregation Control Protocol)

STP (Spanning Tree Protocol)

- **Fonction** : Évite les boucles dans un réseau commuté
- **Algorithme** : Crée un arbre couvrant sans boucles
- **Versions** :
    - **STP** : Standard (802.1D)
    - **RSTP** : Rapid STP (802.1w)
    - **MSTP** : Multiple STP (802.1s)

États des ports STP :

- **Blocking** : Bloqué, ne transmet pas
- **Listening** : Écoute les BPDU
- **Learning** : Apprend les adresses MAC
- **Forwarding** : Transmet les données
- **Disabled** : Désactivé

HSRP / VRRP (Redondance de passerelle)

HSRP (Hot Standby Router Protocol) :

- Routeur actif et routeur de secours
- Adresse IP virtuelle partagée
- Basculement automatique en cas de panne

VRRP (Virtual Router Redundancy Protocol) :

- Standard ouvert (RFC 3768)
- Similaire à HSRP mais standardisé

1.6 DMZ (Demilitarized Zone)

Concept :

- Zone réseau isolée entre Internet et le réseau interne
- Contient les serveurs accessibles depuis Internet
- Protégée par des firewalls

Architecture typique :

```
Internet → Firewall externe → DMZ → Firewall interne → Réseau interne
```

Copier

Serveurs en DMZ :

- Serveurs web
- Serveurs de messagerie (SMTP)
- Serveurs FTP
- Serveurs DNS publics

Règles de sécurité DMZ :

- Internet peut accéder à la DMZ (ports spécifiques)
- DMZ ne peut pas accéder au réseau interne
- Réseau interne peut accéder à la DMZ et Internet

Partie 2 : Architectures serveurs et rôles

2.1 OS serveurs d'entreprise

Windows Server

Versions récentes :

- Windows Server 2022
- Windows Server 2019
- Windows Server 2016

Éditions :

- **Standard** : Pour la plupart des environnements
- **Datacenter** : Virtualisation illimitée
- **Essentials** : Petites entreprises (max 25 utilisateurs)

Rôles principaux :

- Active Directory Domain Services (AD DS)
- DNS Server
- DHCP Server
- File Services

- Web Server (IIS)
- Hyper-V (virtualisation)

Linux serveur

Distributions populaires :

- **Red Hat Enterprise Linux (RHEL)** : Entreprise, support commercial
- **Ubuntu Server** : Facile à utiliser, support communautaire
- **Debian** : Stable, communautaire
- **CentOS** : Clone de RHEL (maintenant CentOS Stream)
- **SUSE Linux Enterprise Server** : Entreprise

Avantages Linux :

- Gratuit et open source
- Stable et sécurisé
- Performant
- Personnalisable

2.2 Serveurs de gestion

DNS (Domain Name System)

Fonction : Résout les noms de domaine en adresses IP

Types de serveurs DNS :

- **Serveur DNS récursif** : Résout les requêtes pour les clients
- **Serveur DNS autoritaire** : Détient les enregistrements d'un domaine
- **Serveur DNS racine** : Point d'entrée de la hiérarchie DNS

Types d'enregistrements DNS :

- **A** : IPv4 → `example.com → 192.168.1.1`
- **AAAA** : IPv6 → `example.com → 2001:db8::1`
- **CNAME** : Alias → `www.example.com → example.com`
- **MX** : Mail Exchange → `example.com → mail.example.com`
- **NS** : Name Server → `example.com → ns1.example.com`
- **TXT** : Texte (SPF, DKIM, etc.)

DHCP (Dynamic Host Configuration Protocol)

Fonction : Attribue automatiquement des adresses IP aux clients

Processus DHCP :

1. **DISCOVER** : Client demande une adresse IP
2. **OFFER** : Serveur propose une adresse IP
3. **REQUEST** : Client accepte l'offre
4. **ACK** : Serveur confirme l'attribution

Options DHCP :

- Adresse IP et masque de sous-réseau
- Passerelle par défaut (option 3)
- Serveurs DNS (option 6)
- Durée du bail (option 51)
- Nom de domaine (option 15)

Annuaires (LDAP / Active Directory)

LDAP (Lightweight Directory Access Protocol) :

- Protocole pour accéder aux annuaires
- Structure hiérarchique (arbre)
- Entrées : DN (Distinguished Name)

Active Directory (Windows) :

- Implémentation Microsoft de LDAP
- Domaine : Unité administrative
- Forêt : Collection de domaines
- Contrôleur de domaine : Serveur qui gère AD

Structure Active Directory :

```
Forêt (Forest)
└── Domaine (Domain)
    ├── OU (Organizational Unit)
    │   ├── Utilisateurs
    │   ├── Groupes
    │   └── Ordinateurs
    └── Conteneurs système
```

Copier

Commandes Active Directory :

```
# Créer un utilisateur
New-ADUser -Name "Jean Dupont" -SamAccountName "jdupont" -UserPrincipalName
"jdupont@example.com" -Path "OU=Users,DC=example,DC=com"

# Créer un groupe
New-ADGroup -Name "Developers" -GroupScope Global -Path "OU=Groups,DC=example,DC=com"

# Ajouter un utilisateur à un groupe
Add-ADGroupMember -Identity "Developers" -Members "jdupont"
```

Copier

2.3 Serveurs web

Serveurs HTTP

Apache HTTP Server :

- Le plus utilisé au monde
- Modules extensibles
- Configuration : fichiers `.htaccess` ou `httpd.conf`

Configuration Apache :

```
# /etc/apache2/sites-available/example.com.conf
<VirtualHost *:80>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com
    <Directory /var/www/example.com>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

Copier

Nginx :

- Performant et léger
- Gère bien la charge
- Utilisé comme reverse proxy

Configuration Nginx :

```
# /etc/nginx/sites-available/example.com
server {
    listen 80;
    server_name example.com www.example.com;
    root /var/www/example.com;
    index index.html index.php;
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Copier

IIS (Internet Information Services) :

- Serveur web Microsoft
- Intégré à Windows Server
- Gestion via interface graphique ou PowerShell

CDN (Content Delivery Network)

Fonction : Distribue le contenu depuis des serveurs proches des utilisateurs

Avantages :

- Réduction de la latence
- Réduction de la charge sur le serveur principal
- Disponibilité améliorée

Fournisseurs CDN :

- Cloudflare
- Amazon CloudFront
- Azure CDN

- Fastly

2.4 Serveurs d'applications / métier

Fonction : Hébergent les applications métier

Exemples :

- **Java EE** : WildFly, Tomcat, WebLogic
- **.NET** : IIS avec ASP.NET
- **Node.js** : Applications JavaScript côté serveur
- **Python** : Django, Flask avec Gunicorn/uWSGI

Architecture 3 tiers :

```
Présentation (Web) → Logique métier (App Server) → Données (Database)
```

Copier

2.5 Serveurs de bases de données

Architecture d'un SGBDR

SGBDR (Système de Gestion de Base de Données Relationnelle) :

- Stocke les données dans des tables
- Relations entre tables (clés étrangères)
- ACID : Atomicité, Cohérence, Isolation, Durabilité

SGBDR populaires :

- **MySQL / MariaDB** : Open source, très utilisé
- **PostgreSQL** : Avancé, open source
- **Oracle Database** : Entreprise, payant
- **SQL Server** : Microsoft, intégré à l'écosystème Windows
- **SQLite** : Léger, embarqué

NoSQL

Types de bases NoSQL :

- **Document** : MongoDB, CouchDB
- **Clé-valeur** : Redis, Memcached
- **Colonnes** : Cassandra, HBase
- **Graphes** : Neo4j

Quand utiliser NoSQL :

- Grandes quantités de données non structurées
- Scalabilité horizontale
- Performance élevée
- Flexibilité du schéma

2.6 Concept de cluster

Définition : Groupe de serveurs travaillant ensemble comme un seul système

Types de clusters :

- **Haute disponibilité (HA)** : Basculement automatique en cas de panne
- **Load balancing** : Répartition de la charge
- **Calcul haute performance (HPC)** : Traitement parallèle

Exemples :

- **Windows Server Failover Cluster** : HA pour Windows
- **Linux Pacemaker** : HA pour Linux
- **Kubernetes** : Orchestration de conteneurs
- **Apache Hadoop** : Traitement de données distribuées

2.7 Redondance, réplication, disponibilité

Redondance

Définition : Duplication des composants critiques

Niveaux de redondance :

- **N+1** : Un composant de secours pour N composants actifs
- **2N** : Doublement de tous les composants
- **2N+1** : Doublement + un composant de secours

Exemples :

- Disques en RAID
- Alimentations redondantes
- Liens réseau redondants

Réplication

Définition : Copie des données vers plusieurs emplacements

Types de réplication :

- **Synchrone** : Écriture simultanée sur tous les sites
- **Asynchrone** : Écriture différée sur les sites secondaires

Disponibilité

SLA (Service Level Agreement) :

- **99%** : 87,6 heures d'indisponibilité/an
- **99.9%** : 8,76 heures d'indisponibilité/an (3 nines)
- **99.99%** : 52,56 minutes d'indisponibilité/an (4 nines)
- **99.999%** : 5,26 minutes d'indisponibilité/an (5 nines)

Calcul de disponibilité :

```
Disponibilité = (Temps total - Temps d'indisponibilité) / Temps total × 100
```

Copier

Partie 3 : Systèmes de virtualisation

3.1 Types de virtualisation

Virtualisation complète

Hyperviseur de type 1 (Bare Metal) :

- S'exécute directement sur le matériel
- Exemples : VMware vSphere/ESXi, Hyper-V, Xen

Hyperviseur de type 2 (Hosted) :

- S'exécute sur un OS hôte
- Exemples : VMware Workstation, VirtualBox, Parallels

Virtualisation au niveau OS

- Conteneurs partageant le même noyau
- Plus léger que la virtualisation complète
- Exemples : Docker, LXC, OpenVZ

3.2 Outils de virtualisation pour les solutions d'entreprise

VMware vSphere

Composants :

- **ESXi** : Hyperviseur
- **vCenter** : Gestion centralisée
- **vMotion** : Migration à chaud des VM
- **HA** : Haute disponibilité
- **DRS** : Distribution des ressources

Avantages :

- Mature et stable
- Fonctionnalités avancées
- Support professionnel

Microsoft Hyper-V

Composants :

- **Hyper-V Manager** : Gestion locale
- **SCVMM** : Gestion centralisée
- **Live Migration** : Migration à chaud

- **Failover Cluster** : Haute disponibilité

Avantages :

- Intégré à Windows Server
- Gratuit avec Windows Server
- Bonne intégration avec Active Directory

KVM (Kernel-based Virtual Machine)

Caractéristiques :

- Intégré au noyau Linux
- Open source
- Performances élevées
- Gestion via libvirt

3.3 Outils pour les tests

VMware Workstation / Fusion

- Virtualisation de type 2
- Pour développeurs et testeurs
- Snapshots, clones, réseaux virtuels

VirtualBox

- Gratuit et open source
- Multi-plateforme
- Bon pour les tests et développement

Partie 4 : Containerisation

4.1 Différence avec la virtualisation

Aspect	Virtualisation	Containerisation
Isolation	OS complet	Processus
Taille	Go	Mo
Démarrage	Minutes	Secondes
Ressources	Plus élevées	Moins élevées
OS	Un par VM	Partage du noyau hôte

Avantages des conteneurs :

- Plus léger et rapide
- Consommation réduite
- Déploiement simplifié

- Scalabilité horizontale facile

4.2 Concepts de base

Image

Définition : Modèle en lecture seule pour créer des conteneurs

Création d'une image Docker :

```
# Dockerfile
FROM ubuntu:20.04
RUN apt-get update && apt-get install -y nginx
COPY index.html /var/www/html/
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Copier

Build de l'image :

```
docker build -t my-nginx:latest .
```

Copier

Container

Définition : Instance exécutable d'une image

Création et démarrage :

```
# Créer et démarrer un conteneur
docker run -d -p 8080:80 --name my-nginx my-nginx:latest

# Lister les conteneurs
docker ps

# Voir les logs
docker logs my-nginx

# Arrêter un conteneur
docker stop my-nginx

# Supprimer un conteneur
docker rm my-nginx
```

Copier

Client et hôte

- **Client Docker** : Interface en ligne de commande (`docker`)
- **Docker Engine** : Démon qui gère les conteneurs
- **Hôte** : Machine sur laquelle Docker s'exécute

Dockerfile

Instructions principales :

- `FROM` : Image de base
- `RUN` : Exécute une commande
- `COPY` / `ADD` : Copie des fichiers
- `WORKDIR` : Définit le répertoire de travail
- `EXPOSE` : Déclare les ports
- `CMD` / `ENTRYPOINT` : Commande par défaut

Exemple Dockerfile complet :

```
FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 8000

ENV PYTHONUNBUFFERED=1

CMD ["python", "app.py"]
```

Copier

Registres / Dépôts

Docker Hub : Registre public principal

```
# Pull une image
docker pull nginx:latest

# Push une image
docker tag my-app:latest username/my-app:latest
docker push username/my-app:latest
```

Copier

Autres registres :

- **GitHub Container Registry** : Intégré à GitHub
- **Amazon ECR** : AWS
- **Azure Container Registry** : Microsoft Azure
- **Google Container Registry** : Google Cloud

4.3 Gestion des réseaux

Réseaux Docker :

- **bridge** : Réseau par défaut
- **host** : Utilise le réseau de l'hôte
- **none** : Pas de réseau
- **overlay** : Pour Docker Swarm

Création d'un réseau :

```
# Créer un réseau
docker network create my-network

# Connecter un conteneur à un réseau
docker run -d --network my-network --name app1 my-app:latest
docker run -d --network my-network --name app2 my-app:latest

# Les conteneurs peuvent communiquer par leur nom
# app1 peut accéder à app2 via http://app2:port
```

Copier

Réseau personnalisé :

```
docker network create
  --driver bridge
  --subnet=172.20.0.0/16
  --gateway=172.20.0.1
  my-custom-network
```

Copier

4.4 Gestion des volumes

Types de volumes :

- **Volumes nommés** : Gérés par Docker
- **Bind mounts** : Pointent vers un chemin de l'hôte
- **tmpfs mounts** : En mémoire

Création et utilisation :

```
# Créer un volume
docker volume create my-data

# Utiliser un volume
docker run -d
  --name my-db
  -v my-data:/var/lib/mysql
  mysql:8.0

# Bind mount
docker run -d
  --name my-app
  -v /host/path:/container/path
  my-app:latest

# Volume anonyme
docker run -d
  --name my-app
  -v /container/path
  my-app:latest
```

Copier

Gestion des volumes :

```
# Lister les volumes
docker volume ls

# Inspecter un volume
docker volume inspect my-data

# Supprimer un volume
docker volume rm my-data
```

Copier

4.5 Docker Compose

Définition : Outil pour définir et exécuter des applications multi-conteneurs

Fichier docker-compose.yml :

```
version: '3.8'

services:
  web:
    build: .
    ports:
      - "8000:8000"
    volumes:
      - .:/app
    depends_on:
      - db
    environment:
      - DATABASE_URL=postgresql://user:password@db:5432/mydb

  db:
    image: postgres:13
    volumes:
      - db-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_USER=user
      - POSTGRES_PASSWORD=password
      - POSTGRES_DB=mydb

volumes:
  db-data:
```

Copier

Commandes Docker Compose :

```
# Démarrer les services
docker-compose up -d

# Arrêter les services
docker-compose down

# Voir les logs
docker-compose logs -f

# Reconstruire les images
docker-compose build

# Exécuter une commande dans un service
docker-comose exec web python manage.py migrate
```

Copier

4.6 Variables d'environnement

Définition dans Dockerfile :

```
ENV DATABASE_URL=postgresql://localhost/mydb
ENV DEBUG=false
```

Copier

Définition au runtime :

```
docker run -e DATABASE_URL=postgresql://db/mydb my-app
```

Copier

Fichier .env :

```
# .env
DATABASE_URL=postgresql://db/mydb
DEBUG=true
SECRET_KEY=my-secret-key
```

Copier

Utilisation dans docker-compose.yml :

```
services:
  web:
    env_file:
      - .env
    environment:
      - DATABASE_URL=${DATABASE_URL}
```

Copier

4.7 Sécurité des conteneurs

Bonnes pratiques :

- Utiliser des images officielles et à jour
- Exécuter en tant qu'utilisateur non-root
- Limiter les ressources (CPU, mémoire)
- Ne pas exposer de ports inutiles
- Utiliser des secrets pour les données sensibles

Exemple sécurisé :

```
FROM node:16-alpine

# Créer un utilisateur non-root
RUN addgroup -g 1001 -S nodejs &&
    adduser -S nodejs -u 1001

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .

# Changer de propriétaire
RUN chown -R nodejs:nodejs /app

USER nodejs

EXPOSE 3000

CMD ["node", "server.js"]
```

Copier

Limites de ressources :

```
services:
  web:
    deploy:
      resources:
        limits:
          cpus: '0.5'
          memory: 512M
        reservations:
          cpus: '0.25'
          memory: 256M
```

Copier

Partie 5 : Infrastructure As Code

5.1 Principe et concepts clés

Définition : Gestion de l'infrastructure via du code plutôt que manuellement

Avantages :

- **Versioning** : Historique des changements (Git)
- **Reproductibilité** : Même infrastructure à chaque déploiement
- **Automatisation** : Réduction des erreurs humaines
- **Documentation** : Le code documente l'infrastructure
- **Testabilité** : Tests automatisés de l'infrastructure

Concepts clés :

- **Idempotence** : Exécuter plusieurs fois donne le même résultat
- **Déclaratif vs Impératif** :
  - **Déclaratif** : Décrit l'état désiré (Terraform, Ansible)
  - **Impératif** : Décrit les étapes (Scripts shell)

5.2 Place de l'IaC dans une démarche DevOps

DevOps : Intégration du développement et des opérations

Cycle DevOps avec IaC :

```
Plan → Code → Build → Test → Release → Deploy → Operate → Monitor
         ↑                                                  ↓
         └──────────────────── IaC ──────────────────────┘
```

Copier

Bénéfices :

- Déploiements plus rapides
- Réduction des erreurs
- Collaboration améliorée
- Infrastructure versionnée

5.3 Principaux outils du marché

Terraform

Définition : Outil d'IaC déclaratif pour provisionner l'infrastructure

Caractéristiques :

- Multi-cloud (AWS, Azure, GCP, etc.)
- Langage HCL (HashiCorp Configuration Language)
- Plan d'exécution avant application
- State management

Exemple Terraform :

```
# main.tf
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.0"
    }
  }
}

provider "aws" {
  region = "us-east-1"
}

resource "aws_instance" "web" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"

  tags = {
    Name = "WebServer"
  }
}

output "instance_ip" {
  value = aws_instance.web.public_ip
}
```

Copier

Commandes Terraform :

```
# Initialiser
terraform init

# Planifier les changements
terraform plan

# Appliquer les changements
terraform apply

# Détruire l'infrastructure
terraform destroy
```

Copier

Ansible

Définition : Outil d'automatisation et de configuration management

Caractéristiques :

- Agentless (pas d'agent sur les serveurs)
- Utilise SSH
- Langage YAML
- Idempotent

Exemple playbook Ansible :

```
# playbook.yml
- name: Configure web server
  hosts: webservers
  become: yes
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present

    - name: Start nginx
      systemd:
        name: nginx
        state: started
        enabled: yes

    - name: Copy index.html
      copy:
        src: index.html
        dest: /var/www/html/index.html
```

Copier

Inventaire Ansible :

```
# inventory.ini
[webservers]
web1 ansible_host=192.168.1.10
web2 ansible_host=192.168.1.11

[databases]
db1 ansible_host=192.168.1.20
```

Copier

Exécution :

```
ansible-playbook -i inventory.ini playbook.yml
```

Copier

Puppet

Définition : Outil de configuration management déclaratif

Caractéristiques :

- Agent-based
- Langage déclaratif Puppet DSL
- Modèle client-serveur

Exemple manifest Puppet :

```
# nginx.pp
package { 'nginx':
  ensure => installed,
}

service { 'nginx':
  ensure => running,
  enable => true,
  require => Package['nginx'],
}

file { '/var/www/html/index.html':
  ensure  => present,
  content => '<h1>Hello World</h1>',
  require => Package['nginx'],
}
```

Copier

Chef

Définition : Plateforme d'automatisation d'infrastructure

Caractéristiques :

- Agent-based
- Langage Ruby
- Recettes et cookbooks

Vagrant

Définition : Outil pour créer et gérer des environnements de développement virtualisés

Exemple Vagrantfile :

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "private_network", ip: "192.168.33.10"
  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2048"
    vb.cpus = 2
  end
  config.vm.provision "shell", inline: <<-SHELL
    apt-get update
    apt-get install -y nginx
  SHELL
end
```

Copier

Commandes Vagrant :

```
# Démarrer
vagrant up

# Se connecter
vagrant ssh

# Arrêter
vagrant halt

# Détruire
vagrant destroy
```

Copier

Solutions cloud

AWS CloudFormation :

```
Resources:
  MyEC2Instance:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-0c55b159cbfafe1f0
      InstanceType: t2.micro
```

Copier

Azure Resource Manager (ARM) :

```
{
  "$schema":
"https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "resources": [
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-03-01",
      "name": "myVM"
    }
  ]
}
```

Copier

Google Cloud Deployment Manager :

```
resources:
- name: my-vm
  type: compute.v1.instance
  properties:
    zone: us-central1-a
    machineType: zones/us-central1-a/machineTypes/n1-standard-1
```

Copier

5.4 Bases du Scripting

Approche déclarative vs impérative

Déclaratif :

```
# Ansible - Déclare l'état désiré
- name: Ensure nginx is installed and running
  apt:
    name: nginx
    state: present
  systemd:
    name: nginx
    state: started
```

Copier

Impératif :

```
# Script shell - Décrit les étapes
if ! command -v nginx &> /dev/null; then
    apt-get update
    apt-get install -y nginx
fi
systemctl start nginx
systemctl enable nginx
```

Copier

Avantages déclaratif :

- Idempotence garantie
- Plus facile à comprendre
- Moins d'erreurs

Modèles (Templates)

Terraform modules :

```
# modules/web-server/main.tf
variable "instance_type" {
  type = string
}

resource "aws_instance" "web" {
  instance_type = var.instance_type
  # ...
}
```

Copier

Utilisation :

```
module "web_server" {
  source = "./modules/web-server"
  instance_type = "t2.micro"
}
```

Copier

Ansible roles :

```
# roles/nginx/tasks/main.yml
- name: Install nginx
  apt:
    name: nginx
    state: present

- name: Configure nginx
  template:
    src: nginx.conf.j2
    dest: /etc/nginx/nginx.conf
```

Copier

5.5 (Optionnel) Bases de l'orchestration avec Kubernetes

Kubernetes : Orchestrateur de conteneurs

Concepts de base :

- **Pod** : Plus petit déploiement (1 ou plusieurs conteneurs)
- **Deployment** : Gère les pods et leur réplication
- **Service** : Expose les pods (ClusterIP, NodePort, LoadBalancer)
- **Namespace** : Isolation logique

Exemple Deployment Kubernetes :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.21
        ports:
        - containerPort: 80
```

Copier

Service :

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - port: 80
    targetPort: 80
  type: LoadBalancer
```

Copier

Commandes Kubernetes :

```
# Appliquer un déploiement
kubectl apply -f deployment.yaml

# Voir les pods
kubectl get pods

# Voir les services
kubectl get services

# Scalabilité
kubectl scale deployment nginx-deployment --replicas=5
```

Copier

Conclusion

Ce wiki a couvert les essentiels de l'infrastructure du SI :

1. **Infrastructures réseaux** : TCP/IP, LAN/WAN/MAN, IPv4/IPv6, équipements réseau, tolérance aux pannes, DMZ
2. **Architectures serveurs** : OS serveurs, DNS, DHCP, annuaires, serveurs web/applications/bases de données, clusters, redondance
3. **Virtualisation** : Types, outils entreprise et tests
4. **Containerisation** : Docker, concepts, réseaux, volumes, Docker Compose, sécurité
5. **Infrastructure As Code** : Principes, outils (Terraform, Ansible, etc.), scripting, Kubernetes

Ces connaissances sont essentielles pour collaborer avec les opérateurs, développer des applications distribuées compatibles avec la production, et mettre en place des environnements stables et cohérents.

Ressources complémentaires

Téléchargements

- Windows 11 25H2 French x64 ISO
- Windows 11 24H2 French ARM64 ISO
- Windows Server Standard 2025 French ARM64 ISO

Documentation officielle

- Cisco Networking
- Docker Documentation
- Terraform Documentation
- Ansible Documentation
- Kubernetes Documentation

Plateformes d'apprentissage

- Cisco Packet Tracer (simulation réseau)
- Docker Playground (test Docker en ligne)
- Katacoda (tutoriels interactifs)

Livres recommandés

- "TCP/IP Illustrated" - W. Richard Stevens
- "Docker Deep Dive" - Nigel Poulton
- "Terraform Up & Running" - Yevgeniy Brikman
- "Kubernetes Up & Running" - Kelsey Hightower