⚙ Compute, Réseau & Stockage

Briques fondamentales du Cloud

Cloud Intro — B2 Informatique, IA & Data, Cybersécurité — 2025-2026

Formateur : Réda Bourebaba

🎯 Objectifs de cette séance

- Comprendre la différence entre **VM (Machine Virtuelle) et containers** en contexte Cloud
- Maîtriser les concepts de **réseau Cloud** : VPC, sous-réseaux, groupes de sécurité
- Distinguer les **types de stockage** : objet, bloc et fichier — quand utiliser quoi
- Mesurer les **impacts sur les coûts et les performances** de chaque brique
- Réaliser des **travaux pratiques (TP)** complets : service web containerisé + stockage objet MinIO

📌 Rappel séance 1 — Ce qu'on a vu

- ✅ **Apports du Cloud** : élasticité, OPEX (paiement à l'usage), haute disponibilité
- ✅ **Modèles de service** : IaaS → PaaS → SaaS (responsabilités croissantes vers le provider)
- ✅ **Types de déploiement** : public / privé / hybride — choix guidé par la conformité
- ✅ **Premier contact MinIO** : docker run + mc alias + bucket de démo

Aujourd'hui : on zoome sur les **briques techniques** et on met le projet en pratique.

💻 Architecture 3-tiers et pattern stateless

Architecture 3-tiers classique :

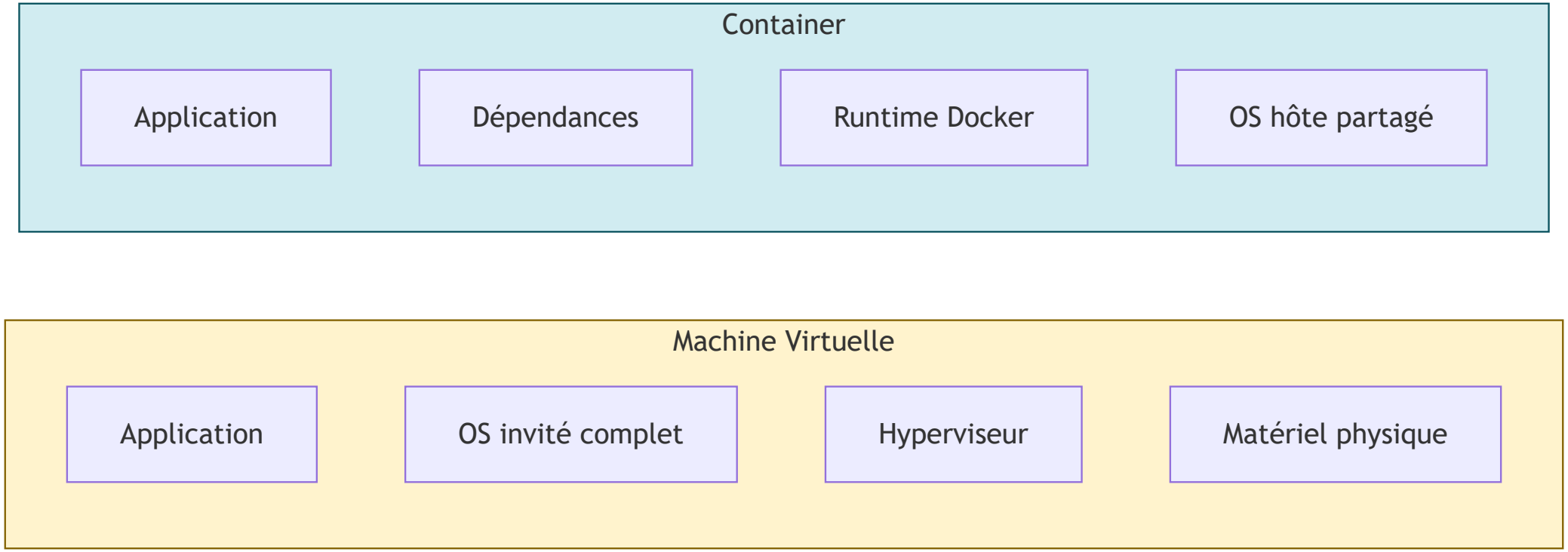
- **Couche présentation** : frontend (HTML/JS, app mobile)
- **Couche applicative** : API backend (logique métier)
- **Couche données** : base de données + stockage fichiers

Pattern stateless en Cloud :

- Chaque instance d'un service doit être **interchangeable** (pas d'état local)
- Les fichiers et sessions sont stockés en **stockage externe** (objet, cache distribué)
- Permet le **scaling horizontal** : ajouter / retirer des instances sans perte de données

> **Règle d'or :** ne jamais stocker de fichier permanent sur le disque local d'un container ou d'une instance — utiliser le stockage objet.

VM vs Containers



> Un container partage le noyau OS (système d'exploitation) de l'hôte : il est plus léger, démarre en secondes, et s'exécute identiquement sur tout environnement Docker.

🌐 Réseau Cloud — VPC et isolation

Réseau Privé Virtuel (VPC) :

- Espace réseau **logiquement isolé** dans l'infrastructure du provider
- Vous définissez les plages d'adresses IP (Internet Protocol) (ex. `10.0.0.0/16`)
- Aucune communication externe sans configuration explicite

Sous-réseaux :

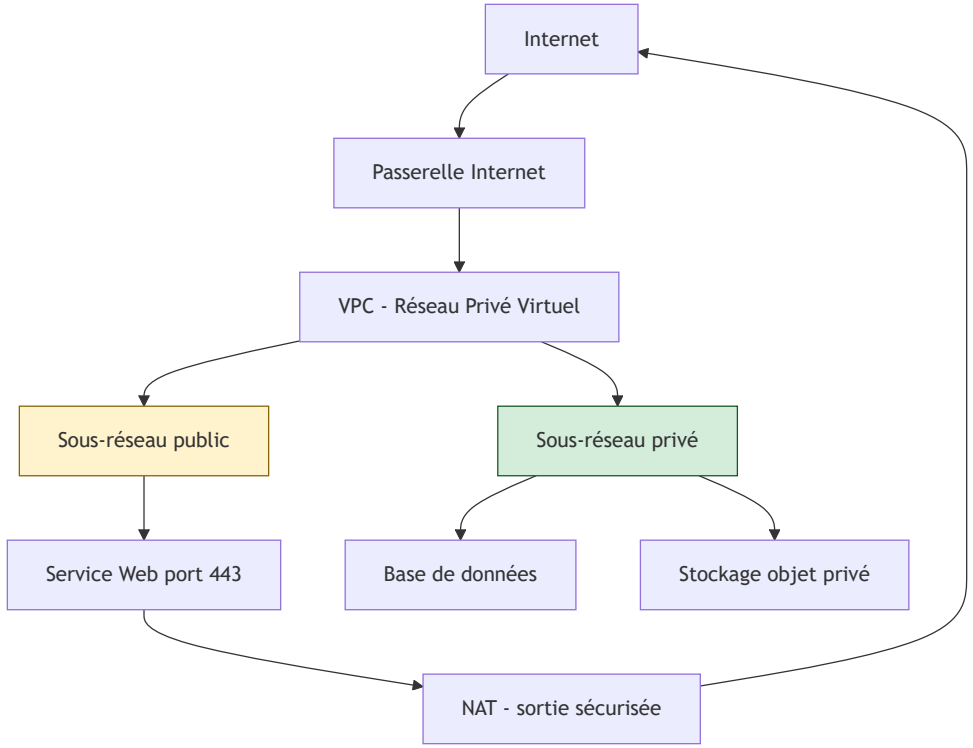
- **Sous-réseau public** : ressources accessibles depuis Internet (load balancer, bastion)
- **Sous-réseau privé** : ressources internes uniquement (base de données, stockage)

Groupe de sécurité (pare-feu virtuel) :

- Règles **entrée (ingress)** et **sortie (egress)** par port et protocole
- Appliqué à chaque instance individuelle (pas seulement au sous-réseau)

> **Bonne pratique :** placer les bases de données et le stockage en sous-réseau **privé**, jamais exposés directement à Internet.

📐 Topologie réseau Cloud



🗄️ Stockage Cloud — Trois types fondamentaux

Stockage Objet :

- Accès via **API** (interface de programmation applicative) **HTTP** (requêtes GET/PUT/DELETE)
- Idéal pour : images, vidéos, sauvegardes, logs, fichiers statiques
- Hautement durable (réplication automatique), tarif bas par Go stocké
- Exemples souverains : OVHcloud Object Storage, Scaleway Object Storage

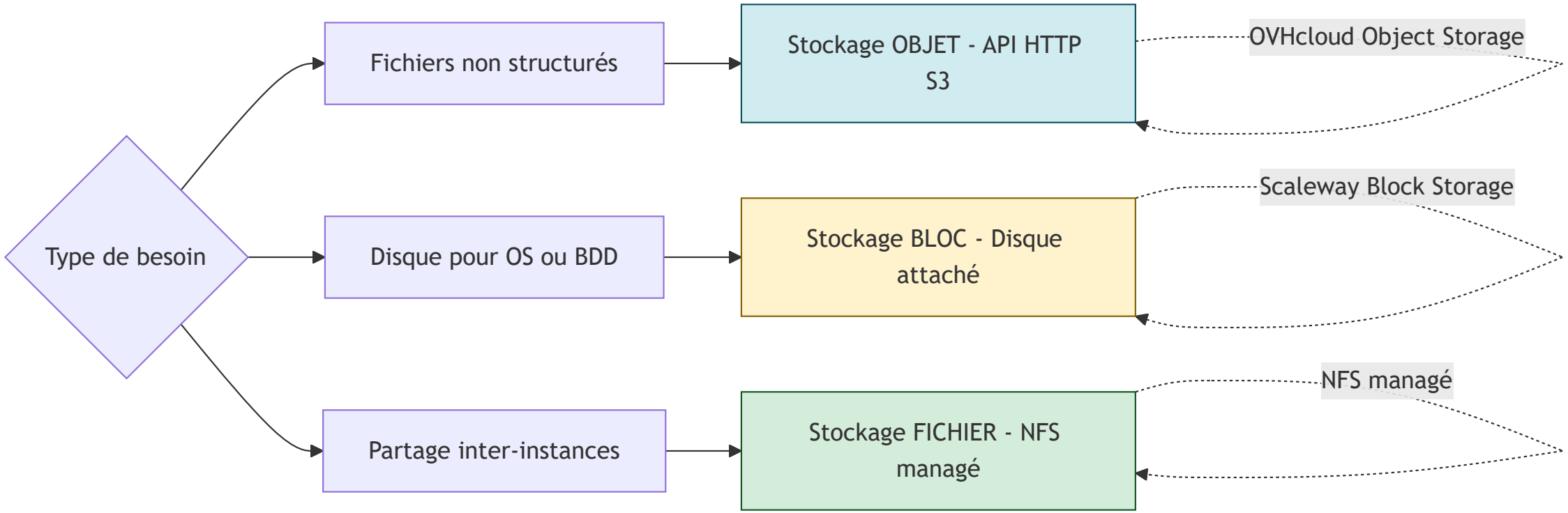
Stockage Bloc :

- Disque persistant **attaché à une instance** (comme un disque dur)
- Idéal pour : système d'exploitation, base de données, fichiers fréquemment modifiés
- Faible latence, accès aléatoire rapide
- Exemples souverains : Scaleway Block Storage, OVHcloud Block Storage

Stockage Fichier (NFS — Network File System — managé) :

- Système de fichiers **partagé entre plusieurs instances**
- Idéal pour : contenus partagés entre plusieurs services
- Exemples souverains : OVHcloud NAS-HA, Scaleway File Storage

📐 Types de stockage — Arbre de décision



💰 Impacts coûts & performances

Critère	Stockage Objet	Stockage Bloc	Stockage Fichier
Tarif stockage	Très bas (~0,01 €/Go/mois)	Moyen (~0,05–0,10 €/Go/mois)	Moyen-élevé
Tarif accès	Par requête + transfert	Inclus dans l'instance	Par Go transféré
Latence	Modérée (HTTP)	Très faible (µs–ms)	Modérée
Scalabilité	Illimitée automatiquement	Limitée à la taille du volume	Limitée
Usage optimal	Assets, backups	OS, base de données (BDD)	Partage inter-services

Conseil : pour un service web, stocker les **assets statiques en objet** (coût minimal) et la **BDD sur bloc** (performance maximale).

💻 Démarrer MinIO et un service web containerisé

```
# MinIO — stockage objet local (S3-compatible)
docker run -d --name minio \
  -p 9000:9000 -p 9001:9001 \
  -e MINIO_ROOT_USER=minioadmin \
  -e MINIO_ROOT_PASSWORD=minioadmin \
  minio/minio server /data --console-address ":9001"

# Service web Nginx sur port 8080
docker run -d --name webserver \
  -p 8080:80 \
  nginx:alpine

# Vérifier les deux containers
docker ps
```

> Les deux services tournent en isolation réseau — nous allons les connecter via l'URL signée MinIO.

🛠️ TP — Créer un bucket et uploader un fichier

Durée : 30 min | **Modalité :** binôme

```
# Configurer mc (une seule fois)
mc alias set local http://localhost:9000 minioadmin minioadmin

# Créer un bucket
mc mb local/mon-bucket

# Uploader un fichier de test
echo "Contenu de test" > test.txt
mc cp test.txt local/mon-bucket/test.txt

# Lister les objets
mc ls local/mon-bucket

# Vérifier via curl (accès temporairement public pour test)
mc anonymous set download local/mon-bucket
curl http://localhost:9000/mon-bucket/test.txt
```

Livrable : capture d'écran montrant le fichier listé dans `mc ls` et le résultat `curl`.

🔗 Connecter un service au stockage objet via URL signée

URL (Uniform Resource Locator) signée — ou presigned URL :

- URL temporaire permettant d'accéder à un objet **sans exposer les credentials**
- Validité configurable (minutes, heures, jours)
- Utilisée pour les téléchargements sécurisés depuis une application web

```
# Générer une URL signée valable 1 heure
mc share download --expire 1h local/mon-bucket/test.txt
```

```
# Remettre le bucket en mode privé (bonne pratique)
mc anonymous set none local/mon-bucket

# Tester l'URL signée générée ci-dessus
curl "https://localhost:9000/mon-bucket/test.txt?X-Amz-Algorithm=..."
```

> Le frontend appelle votre **API backend** → l'API génère l'URL signée → le client télécharge directement depuis MinIO. L'API ne transmet jamais les credentials.

📊 Observabilité basique

Logs Docker :

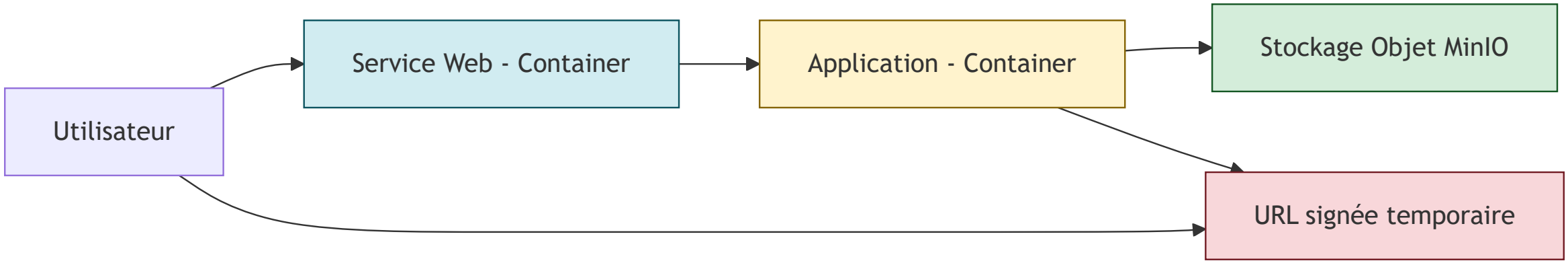
```
# Logs en temps réel d'un container
docker logs -f minio
docker logs -f webserver

# Dernières 50 lignes
docker logs --tail 50 minio
```

Console MinIO (métriques) :

- Accéder à `http://localhost:9001`
- Onglet **Metrics** : requêtes, bande passante, objets par bucket
- Onglet **Audit Log** : historique de chaque opération (PUT, GET, DELETE)

> En production Cloud : les providers proposent des services de monitoring (tableaux de bord, alertes, rétention des logs) intégrés à l'infrastructure.

📐 Architecture mini-projet fil rouge



> Architecture cible : un service web containerisé accède au stockage objet MinIO et génère des URL signées pour les téléchargements. Aucun credential exposé côté client.

🛠️ TP — Diagramme d'architecture en groupe

Durée : 50 min | **Modalité :** groupes de 3-4

Sujet

Concevoir l'architecture Cloud d'une application de partage de fichiers sécurisé (type WeTransfer minimal).

Contraintes

- Utiliser uniquement du vocabulaire générique (instance, VPC, stockage objet, groupe de sécurité)
- Justifier chaque brique choisie
- Identifier les données sensibles et leur emplacement

Livrables

- Diagramme draw.io ou Mermaid
- Note d'architecture 1 page : choix, justifications, providers souverains envisagés

✅ Critères de validation — TP Séance 2

Le TP est validé si les 4 critères suivants sont remplis :

Critère	Description	Commande de vérification
Service fonctionnel	Container Nginx répond sur le port 8080	`curl -s http://localhost:8080 | head -5`
Upload réussi	Fichier présent dans le bucket MinIO	`mc ls local/mon-bucket`
URL signée opérationnelle	Accès 200 via l'URL signée, 403 en direct	`curl -I http://localhost:9000/mon-bucket/test.txt`
Documentation	Diagramme d'architecture + endpoints décrits	Rendu draw.io / Mermaid

Modalité : individuel ou binôme | **Durée totale TP :** ~2h30

📌 Points clés à retenir

- ✅ **VM vs Container** : le container partage l'OS hôte, plus léger, plus rapide à démarrer
- ✅ **VPC** : réseau privé isolé — toujours séparer les ressources publiques et privées
- ✅ **Stockage objet** pour les assets, **bloc** pour les BDD, **fichier** pour le partage inter-services
- ✅ **URL signée** : la bonne pratique pour exposer temporairement un objet sans credentials
- ✅ **Stateless** : ne jamais stocker d'état dans le container — externaliser vers stockage ou cache

Changelog

Version	Date	Auteur	Modification
V0.0.3	18/05/2026	Réda Bourebaba	Fix overflow PDF : slide Rappel séance 1 raccourcie (bullets très longs)
V0.0.2	18/05/2026	Réda Bourebaba	Corrections scoring QA : slide Rappel séance 1 ajoutée ; slide Critères validation TP ajoutée ; alt text ajouté sur 4 SVGs
V0.0.1	18/05/2026	Réda Bourebaba	Création initiale du deck