🔒 Sécurité & Gouvernance Cloud

IAM, conformité et soutenance

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

Formateur : Réda Bourebaba

🎯 Objectifs de cette séance

- Comprendre le modèle **IAM** : utilisateurs, rôles, politiques et ressources
- Appliquer le principe du **moindre privilège** dans la gestion des accès
- Connaître les **bonnes pratiques** de gestion des secrets et clés d'API (interface de programmation applicative)
- Maîtriser les mécanismes de **chiffrement** (en transit et au repos)
- Situer les enjeux de **souveraineté et conformité** (RGPD, SecNumCloud)
- Soutenir le **mini-projet fil rouge** avec un retour structuré

📌 Rappel séance 2 — Ce qu'on a vu

- ✅ **VM vs Containers** : le container partage l'OS hôte — plus léger, démarre en secondes
- ✅ **VPC** : réseau privé isolé — sous-réseaux public / privé, groupes de sécurité
- ✅ **Stockage** : objet (assets, API HTTP) — bloc (BDD, OS) — fichier (partage inter-services)
- ✅ **TP MinIO complet** : bucket créé, fichier uploadé, URL signée générée, service web connecté

Aujourd'hui : on **sécurise** l'architecture et on **soutient** le projet fil rouge.

🔑 IAM — Identité et contrôle d'accès

IAM (Identity and Access Management) : système qui contrôle **qui** peut faire **quoi** sur **quelles ressources**.

Concepts clés :

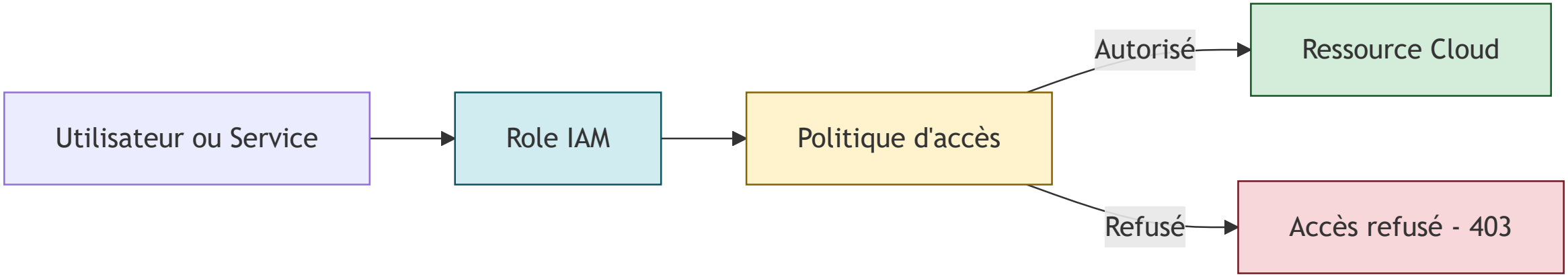
Concept	Description
Utilisateur	Identité humaine ou applicative (nom + credentials)
Rôle	Ensemble de permissions pouvant être endossé temporairement
Politique	Document décrivant les actions autorisées ou refusées
Ressource	Élément Cloud ciblé (bucket, instance, base de données...)

Principe du moindre privilège :

> Accorder **uniquement** les permissions strictement nécessaires à l'accomplissement d'une tâche. Rien de plus.

Compte root : ne jamais utiliser le compte administrateur initial pour les opérations courantes.

📐 Modèle IAM — Flux d'autorisation



🚫 Gestion des secrets & clés d'API

Anti-patterns à éviter absolument :

- ❌ Clé d'API en clair dans le code source (git histoire publique = fuite permanente)
- ❌ Credentials partagés dans un fichier `.env` commité
- ❌ Clé de production utilisée sur un poste de développement
- ❌ Clé sans date d'expiration ni rotation prévue

Bonnes pratiques :

- ✅ **Variables d'environnement** injectées au démarrage (jamais dans le code)
- ✅ **Gestionnaire de secrets** (HashiCorp Vault, ou service managé du provider)
- ✅ **Rotation régulière** des credentials (automatisée si possible)
- ✅ **Audit des accès** : toute utilisation d'une clé doit être journalisée
- ✅ **Clés scoped** : une clé par usage, avec des permissions minimales

> Si vous poussez une clé sur GitHub par erreur : **révoquez immédiatement**, considérez-la compromise — même si le repo est privé.

🔐 Chiffrement — En transit et au repos

En transit (TLS — Transport Layer Security) :

- Toute communication entre clients et services doit utiliser **HTTPS** (HyperText Transfer Protocol Sécurisé) **/TLS**
- Empêche l'interception (man-in-the-middle) sur le réseau
- MinIO : activer TLS avec un certificat (Let's Encrypt ou PKI — Infrastructure à Clé Publique — interne)

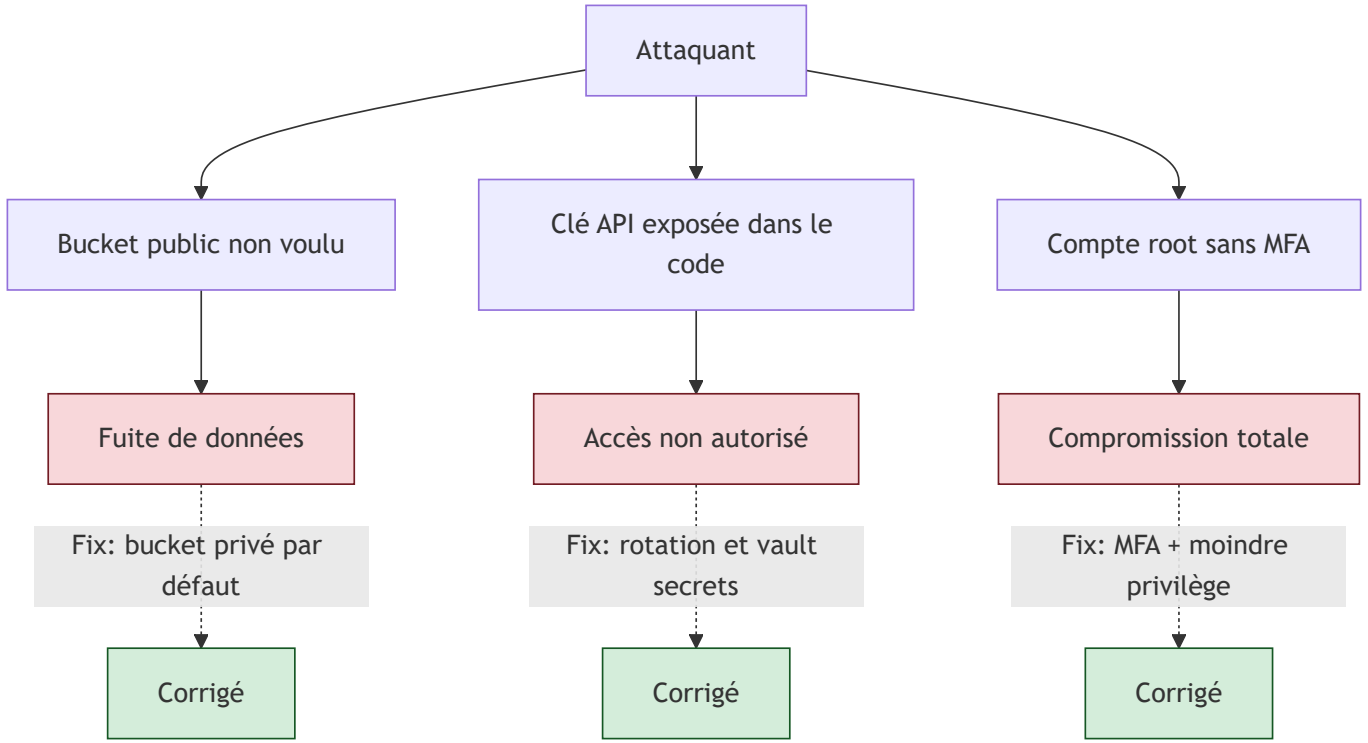
Au repos (SSE — Server-Side Encryption) :

- Les données stockées sont chiffrées **automatiquement** par le service
- MinIO supporte SSE-S3 (clé gérée par MinIO) et SSE-KMS (KMS — Key Management Service — clé externe)
- Les buckets et volumes doivent avoir le chiffrement activé **par défaut**

Clé de chiffrement :

- Ne jamais stocker la clé de chiffrement au même endroit que les données chiffrées
- Utiliser un service de gestion de clés (KMS) distinct

📐 Surface d'attaque Cloud — Vecteurs et contre-mesures



✅ Checklist sécurité Cloud

Configuration des accès :

- [ ] Compte root sans accès quotidien — MFA (Multi-Factor Authentication — authentification multi-facteur) activé obligatoirement
- [ ] Principe du moindre privilège appliqué à chaque rôle
- [ ] Revue des accès tous les 3 mois minimum

Stockage :

- [ ] Buckets **privés par défaut** — aucun bucket public sans justification
- [ ] Chiffrement au repos activé sur tous les buckets
- [ ] Journalisation des accès activée

Secrets :

- [ ] Aucune clé en clair dans le code ou les fichiers de configuration
- [ ] Rotation des credentials planifiée
- [ ] Alerts sur utilisation inhabituelle (IP inconnue, volume anormal)

Réseau :

- [ ] Groupes de sécurité : règle `deny all` par défaut, ouvertures explicites uniquement
- [ ] Base de données (BDD) et stockage en sous-réseau privé

🌍 Souveraineté & Conformité

RGPD (Règlement Général sur la Protection des Données) :

- Données personnelles des résidents EU → traitement et stockage encadrés
- Le responsable de traitement doit **savoir où** sont hébergées les données
- Transfert hors UE : clauses contractuelles spécifiques requises

Contexte marché 2026 — pourquoi les providers souverains progressent :

- Les *egress fees* (frais de sortie de données) des hyperscalers US rendent coûteux tout retour en arrière
- Plus de **80 % des entreprises** étudient ou ont initié un rapatriement partiel vers des infras locales
- OVHcloud et Scaleway bénéficient directement de cette dynamique sur les charges IaaS sensibles

📋 Certifications & Labels Cloud FR/EU

Certifications et labels FR/EU :

Label	Description	Providers
SecNumCloud	Qualification ANSSI (Agence Nationale de la Sécurité des Systèmes d'Information) — niveau de sécurité élevé	Outscale/Numspot
HDS	Hébergeur Données de Santé — obligatoire secteur médical	OVHcloud, Scaleway
ISO 27001 (Organisation Internationale de Normalisation)	Système de management de la sécurité de l'information	Tous grands providers

Choix de région :

- Données sensibles → région France ou EU explicitement sélectionnée
- Vérifier la chaîne de sous-traitance du provider (hébergeur utilisé par le PaaS)

💻 Rendre un bucket MinIO privé

```
# Vérifier la politique actuelle du bucket
mc anonymous get local/mon-bucket

# Forcer le bucket en mode privé (aucun accès anonyme)
mc anonymous set none local/mon-bucket

# Appliquer une politique fine via JSON
cat > policy-readonly-user.json << 'EOF'
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam:::user/readonlyuser"]},
      "Action": ["s3:GetObject"],
      "Resource": ["arn:aws:s3:::mon-bucket/*"]
    }
  ]
}
EOF

mc admin policy add local readonly-mon-bucket policy-readonly-user.json
```

🛠️ TP — Générer et tester une URL signée

Durée : 25 min | **Modalité :** individuel ou binôme

```
# Vérifier que le bucket est bien privé
mc anonymous get local/mon-bucket
# Résultat attendu : none

# Tenter un accès direct (doit échouer en 403)
curl -I http://localhost:9000/mon-bucket/test.txt

# Générer une URL signée valable 1 heure
mc share download --expire 1h local/mon-bucket/test.txt
# → Copier l'URL générée

# Accéder via l'URL signée (doit réussir en 200)
curl "<URL-signee-copiee>"

# Attendre l'expiration (ou modifier --expire 10s) et retester → 403
```

Livrable : captures d'écran montrant : 403 en accès direct, 200 via URL signée.

💻 Créer un compte de service restreint dans MinIO

```
# Créer un utilisateur restreint (lecture seule sur un bucket)
mc admin user add local readonlyuser secretpassword123

# Appliquer la politique readonly
mc admin policy attach local readonly-mon-bucket --user readonlyuser

# Tester avec le compte restreint
mc alias set restricted http://localhost:9000 readonlyuser secretpassword123

# Lister (autorisé)
mc ls restricted/mon-bucket

# Tenter d'écrire (doit échouer)
mc cp autre.txt restricted/mon-bucket/
# → Erreur: Access Denied
```

🛠️ TP — Vérifier les logs d'audit MinIO

Durée : 25 min | **Modalité :** binôme

1. Accéder à la console MinIO → `http://localhost:9001`
2. Naviguer dans **Monitoring** → **Audit Logs**
3. Identifier dans les logs :
   - Les **accès autorisés** (PUT, GET avec credentials valides)
   - Les **accès refusés** (403 sur bucket privé, URL expirée)
4. Filtrer par utilisateur `readonlyuser` — vérifier les tentatives d'écriture refusées

Question de réflexion :

> En production, combien de temps faut-il conserver les logs d'audit pour être conforme RGPD ?
> **Réponse :** le RGPD n'impose pas de durée fixe pour les logs techniques — mais la CNIL (Commission Nationale de l'Informatique et des Libertés) recommande **6 mois** minimum, **1 an** pour les incidents de sécurité.

🎓 Barème de soutenance — Mini-projet fil rouge

Total : /20 points

Critère	Barème	Description
Fonctionnalité	/6	Service web opérationnel + stockage objet MinIO fonctionnel
Architecture	/5	Diagramme clair, briques justifiées, flux de données cohérents
Sécurité	/5	Bucket privé, URL signée, compte restreint, secrets non exposés
Soutenance	/4	Clarté de la présentation, réponses aux questions (5-7 min)

Bonus : déploiement en Cloud souverain réel (+2 pts)

📋 Préparer sa soutenance (5-7 minutes)

Structure recommandée :

1. **Contexte** (30 s) : quel problème résout votre application ?
2. **Démonstration** (2 min) : montrer l'application qui fonctionne
3. **Architecture** (2 min) : expliquer le diagramme — chaque brique et pourquoi
4. **Sécurité** (1 min) : bucket privé + URL signée + compte restreint
5. **Ce que j'ai appris** (30 s) : une chose technique concrète retenue

Questions fréquentes à préparer :

- Pourquoi avoir choisi le stockage objet plutôt que bloc pour les fichiers ?
- Comment gérez-vous la rotation des credentials dans votre architecture ?
- Quel provider souverain choisiriez-vous en production et pourquoi ?

🛠️ Soutenance — Fil rouge groupe par groupe

Modalité : passage en groupe devant la classe

Déroulement :

- **5-7 min** de présentation + démo
- **3-5 min** de questions du formateur et des pairs

Critères d'évaluation en direct :

- La démo fonctionne-t-elle ? (bucket privé, upload, URL signée)
- Le diagramme est-il compréhensible sans explication ?
- Les choix de sécurité sont-ils justifiés ?
- Le groupe maîtrise-t-il le vocabulaire générique Cloud ?

📌 Points clés à retenir

- ✅ **IAM** : chaque entité (utilisateur, service) ne doit avoir que les permissions nécessaires
- ✅ Aucune clé d'API dans le code — **variables d'environnement** et **rotation régulière**
- ✅ **Chiffrement** : TLS en transit, SSE au repos — activer par défaut, sans exception
- ✅ **Bucket privé par défaut** : toute ouverture doit être explicite et justifiée
- ✅ **RGPD + SecNumCloud** : choisir un provider dont vous maîtrisez la localisation et le droit applicable

Changelog

Version	Date	Auteur	Modification
V0.0.4	18/05/2026	Réda Bourebaba	Corrections scoring QA : slide Rappel séance 2 ajoutée ; alt text ajouté sur 2 SVGs
V0.0.2	18/05/2026	Réda Bourebaba	Enrichissement slide Souveraineté (contexte marché 2026, egress fees, cloud repatriation, dynamique OVHcloud/Scaleway)
V0.0.1	18/05/2026	Réda Bourebaba	Création initiale du deck