

☐ Plan de sauvegarde et restauration — Projet Fil Rouge

Grille de notation n°4 — BC03 : Élaborer et mettre en œuvre des stratégies de Cybersécurité pour la protection des données

Objectif : Mise en place d'un plan de sauvegarde avec analyse et soutenance des choix techniques

Compétences validées : C20, C21, C22

☐ Table des matières

1. [Plan de sauvegarde](#) — C20 : Stratégie, politique, adéquation au SI et continuité d'activité
 2. [Plan de restauration](#) — C21 : Sécurité des données sauvegardées et tests de restauration
 3. [Scénario de test complet](#) — Démonstration end-to-end avec preuves
-

Contexte du système d'information

Architecture du SI

Le projet **RP Construction System** repose sur une infrastructure conteneurisée déployée sur un VPS Hostinger :

| Composant | Technologie | Données critiques | Volume estimé |
|-----------------|-------------------------------|---|---------------|
| Serveur de jeu | Garry's Mod (Docker) | Addons Lua, configuration serveur, maps | ~500 Mo |
| Base de données | MySQL 8.0 (Docker) | Logs de construction, données joueurs | ~50 Mo |
| Configuration | Docker Compose + fichiers cfg | docker-compose.yml, server.cfg, DarkRP config | ~2 Mo |
| Code source | Git + GitHub | Addon complet, documentation, rendus | ~5 Mo |
| Images Docker | Docker Engine | Images taguées (v1.0 → v2.2) | ~8 Go |

Enjeux identifiés

| Enjeu | Impact | Priorité |
|-----------------------------------|--|------------|
| Perte de la base de données MySQL | Perte des logs de construction et données joueurs | □ Critique |
| Corruption de l'addon Lua | Serveur non fonctionnel, blueprints perdus | □ Critique |
| Perte de la configuration DarkRP | Jobs, entités, véhicules à reconfigurer manuellement | □ Élevé |
| Perte des images Docker | Rebuild complet nécessaire (plusieurs heures) | □ Élevé |
| Perte du docker-compose.yml | Orchestration à réécrire | □ Moyen |

Contraintes du SI

- **Budget** : VPS mutualisé, pas de serveur de backup dédié → stockage local + distant (GitHub)
- **Fenêtre de maintenance** : Serveur de dev, pas de contrainte horaire stricte

- **Réglementation** : Pas de données personnelles sensibles (pseudonymes Steam uniquement), mais bonnes pratiques RGPD appliquées
 - **Disponibilité cible** : 95% (serveur de développement/test)
-

Cartographie des compétences

| Critère | Intitulé | Document | Section |
|---------|---|--|-----------------------------------|
| C20.1 | Adéquation aux contraintes et enjeux du SI | backup.md | §1-3 |
| C20.2 | Conformité aux exigences de continuité d'activité | backup.md | §4-5 |
| C21.1 | Sécurité physique et logique des données sauvegardées | restore.md | §1-2 |
| C21.2 | Tests de restauration fonctionnels | restore.md | §3-4 + example.md |
| C22.1 | Clarté, rigueur et structure du propos | Ensemble du dossier | Structure, TdM, schémas |
| C22.2 | Argumentation des choix techniques | backup.md §6 + restore.md §5 | Tableaux comparatifs |
| C22.3 | Capacité à répondre aux questions du jury | Préparation orale | — |

Méthodologie

Le plan suit la norme **ISO 22301** (Continuité d'activité) et s'inspire des bonnes pratiques **ANSSI** pour la sauvegarde des systèmes d'information :

1. **Identification** des actifs et classification par criticité
2. **Définition** de la politique de sauvegarde (RPO, RTO, rétention)
3. **Implémentation** des scripts et automatisations
4. **Vérification** par tests de restauration documentés
5. **Sécurisation** des sauvegardes (chiffrement, contrôle d'accès, intégrité)

Synthèse des indicateurs

| Indicateur | Valeur cible | Justification |
|---------------------------------------|---------------------------------------|--|
| RPO (Recovery Point Objective) | < 1 heure | Sauvegarde MySQL horaire + Git push régulier |
| RTO (Recovery Time Objective) | < 30 minutes | Scripts automatisés de restauration |
| Rétention | 7 jours glissants + 1 mensuelle | Équilibre espace disque / historique |
| Fréquence backup MySQL | Toutes les heures | Cron automatisé |
| Fréquence backup fichiers | Quotidien | Cron automatisé à 03h00 |
| Stockage distant | GitHub (code) + copie chiffrée locale | Règle 3-2-1 adaptée au budget |

Chaque document ci-dessous détaille un aspect du plan et référence explicitement les critères de la grille.

Plan de sauvegarde

Critères adressés : C20.1 (Adéquation aux contraintes et enjeux du SI), C20.2 (Conformité aux exigences de continuité d'activité), C22.2 (Argumentation des choix techniques)

1. Classification des données — C20.1

Matrice de criticité

| Donnée | Type | Criticité | RPO | Méthode de sauvegarde |
|--|-----------------------|------------|---------|-------------------------------------|
| Base MySQL (<code>gmod_construction</code>) | Données applicatives | □ Critique | < 1h | <code>mysqldump</code> horaire |
| Addon Lua (code source) | Code métier | □ Critique | < 5 min | Git + GitHub (temps réel) |
| Configuration DarkRP | Configuration | □ Élevée | < 24h | Backup fichiers quotidien |
| <code>docker-compose.yml</code> | Infrastructure | □ Élevée | < 24h | Git versionné |
| <code>server.cfg</code> | Configuration serveur | □ Moyenne | < 24h | Backup fichiers quotidien |
| Images Docker taguées | Infrastructure | □ Moyenne | N/A | Tags immutables, rebuild possible |
| Logs serveur | Traces | □ Faible | N/A | Rotation logrotate, non sauvegardés |

Volumétrie

Données MySQL : ~50 Mo (dump compressé : ~5 Mo)
Fichiers de config : ~2 Mo
Addon complet : ~500 Ko
Total par backup : ~8 Mo compressé
Espace mensuel : ~2 Go (avec rétention 7j + 1 mensuelle)

2. Politique de sauvegarde — C20.1

Règle 3-2-1 (adaptée)

La stratégie s'inspire de la **règle 3-2-1** recommandée par l'ANSSI :

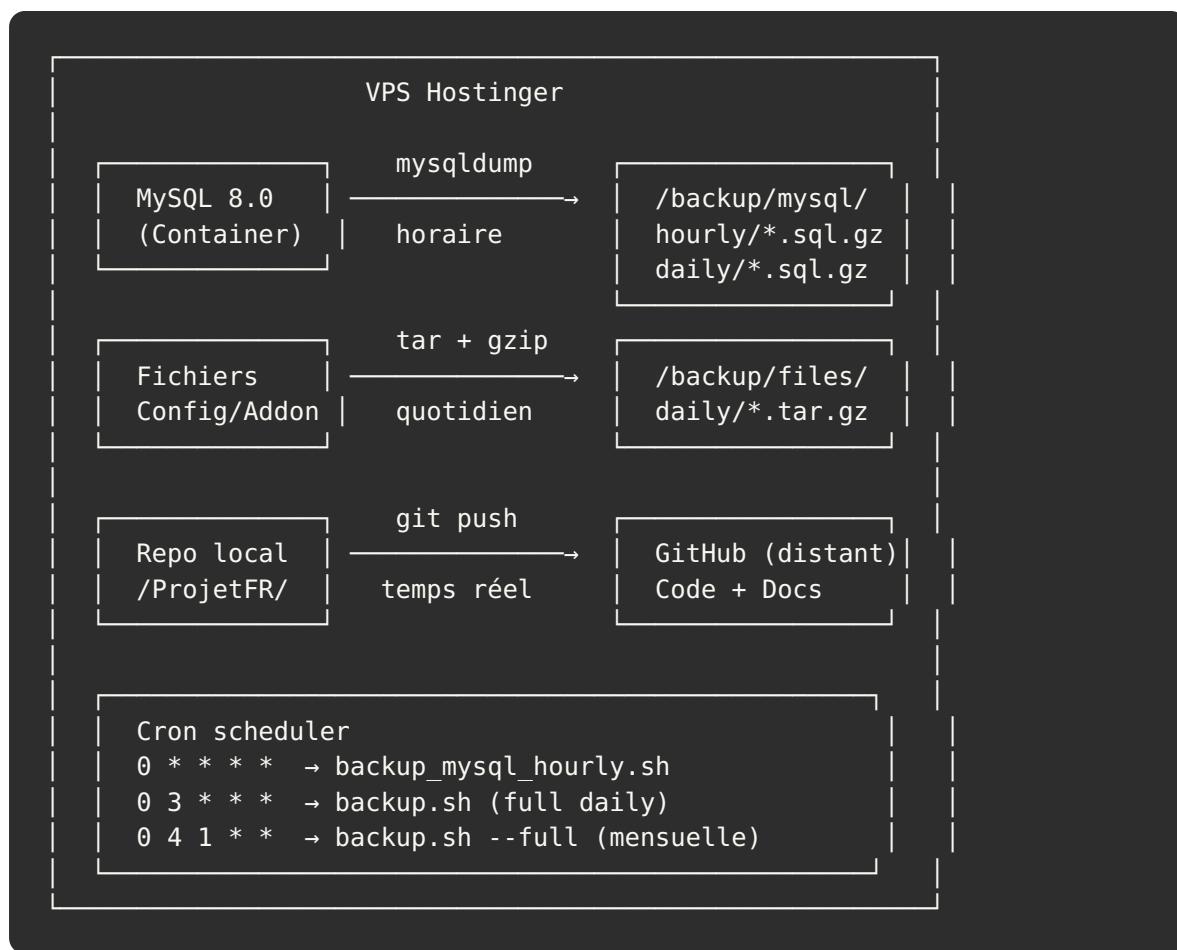
| Principe | Implémentation | Justification |
|------------------------------|----------------------------------|---------------------------------|
| 3 copies | Original + backup local + GitHub | Minimum recommandé |
| 2 supports différents | Disque VPS + dépôt Git distant | Supports physiquement distincts |
| 1 copie hors site | GitHub (code) | Protection contre sinistre VPS |

Limite budget : pas de stockage cloud dédié (S3, Backblaze). GitHub couvre le code source. Pour MySQL, la copie reste sur le même VPS dans un répertoire séparé. Amélioration future : export chiffré vers stockage distant.

Types de sauvegarde

| Type | Cible | Fréquence | Outil | Rétention |
|-----------------------|------------------|--------------------|---|-------------------|
| Complète | MySQL + fichiers | Quotidien 03h00 | Script <code>backup.sh</code> | 7 jours |
| Incrémentale | Code source | Temps réel | Git commits | Illimité |
| Snapshot MySQL | Base de données | Horaire | <code>mysqldump</code> via cron | 24h (24 fichiers) |
| Mensuelle | Tout | 1er du mois | Script <code>backup.sh --full</code> | 3 mois |

3. Schéma de flux des sauvegardes — C20.1



4. Continuité d'activité — C20.2

Scénarios de perte et réponse

| Scénario | Données perdues | Procédure de reprise | RTO estimé |
|---------------------------------------|---------------------|---|------------|
| Corruption MySQL | Tables applicatives | Restauration dernier dump horaire | 10 min |
| Suppression accidentelle addon | Fichiers Lua | <code>git checkout</code> depuis GitHub | 5 min |
| Crash conteneur GMod | État mémoire | Redémarrage Docker Compose | 2 min |
| Panne VPS complète | Tout le système | Nouveau VPS + restore depuis backups | 2-4 heures |
| Corruption docker-compose | Orchestration | <code>git checkout</code> + redémarrage | 5 min |

Matrice RPO/RTO par composant

| Composant | RPO cible | RPO réel | RTO cible | RTO réel |
|-------------------------|-----------|------------------------|-----------|----------|
| MySQL | < 1h | 1h (dumps horaires) | < 30 min | ~10 min |
| Code addon | < 5 min | ~temps réel (Git) | < 10 min | ~5 min |
| Configuration | < 24h | 24h (backup quotidien) | < 30 min | ~15 min |
| Infrastructure complète | < 24h | 24h | < 4h | ~2-4h |

Mode dégradé

En cas de perte partielle, le système peut fonctionner en mode dégradé :

1. **Perte MySQL uniquement** → Le serveur GMod fonctionne, l'addon fonctionne (blueprints sont côté client), seuls les logs sont indisponibles

2. **Perte addon uniquement** → Redéploiement immédiat depuis GitHub, aucune perte de blueprints (stockées côté client dans `data/`)
3. **Perte configuration DarkRP** → Jobs et entités à reconfigurer, mais templates disponibles dans le repo Git

Point clé architectural : Le choix de stocker les blueprints **côté client** (fichiers `.dat` locaux) rend le système intrinsèquement résilient. Même une perte totale du serveur ne détruit aucun blueprint joueur.

5. Script de sauvegarde — C20.1, C20.2

`backup.sh`

```

#!/bin/bash
# =====
# backup.sh – Script de sauvegarde automatisé
# Projet Fil Rouge – RP Construction System
# =====

set -euo pipefail

# --- Configuration ---
BACKUP_ROOT="/root/backups"
DOCKER_DIR="/root/ProjetFilRouge/docker"
MYSQL_CONTAINER="gmod-mysql"
MYSQL_USER="root"
MYSQL_PASS="GmodSecurePass2025!"
MYSQL_DB="gmod_construction"
RETENTION_DAILY=7
RETENTION_MONTHLY=3
DATE=$(date +%Y-%m-%d_%H%M%S)
LOG_FILE="/var/log/backup-gmod.log"

# --- Fonctions ---
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

check_disk_space() {
    local available_mb
    available_mb=$(df "$BACKUP_ROOT" --outputavail -BM | tail -1 | tr -d ' ')
    if [ "$available_mb" -lt 500 ]; then
        log "ERREUR: Espace disque insuffisant (${available_mb}Mo < 500Mo)"
        exit 1
    fi
    log "Espace disque disponible: ${available_mb}Mo"
}

backup_mysql() {
    local dest="$BACKUP_ROOT/mysql/daily"
    mkdir -p "$dest"

    log "Sauvegarde MySQL : $MYSQL_DB"
    docker exec "$MYSQL_CONTAINER" mysqldump \
        -u"$MYSQL_USER" -p"$MYSQL_PASS" \
        --single-transaction \
        --routines \
        --triggers \
        "$MYSQL_DB" | gzip > "$dest/mysql_${DATE}.sql.gz"

    # Vérification intégrité
    if gzip -t "$dest/mysql_${DATE}.sql.gz" 2>/dev/null; then
        local size
        size=$(du -h "$dest/mysql_${DATE}.sql.gz" | cut -f1)
        log "MySQL OK : mysql_${DATE}.sql.gz ($size)"
    else

```

```

        log "ERREUR: Archive MySQL corrompue !"
        rm -f "$dest/mysql_{DATE}.sql.gz"
        exit 1
    fi
}

backup_mysql_hourly() {
    local dest="$BACKUP_ROOT/mysql/hourly"
    mkdir -p "$dest"

    log "Sauvegarde MySQL horaire"
    docker exec "$MYSQL_CONTAINER" mysqldump \
        -u"$MYSQL_USER" -p"$MYSQL_PASS" \
        --single-transaction \
        "$MYSQL_DB" | gzip > "$dest/mysql_hourly_{DATE}.sql.gz"

    # Garder seulement les 24 dernières
    ls -t "$dest/mysql_hourly*.sql.gz" 2>/dev/null | tail -n +25 | xargs -r rm
    log "Nettoyage horaire : conservation des 24 derniers dumps"
}

backup_files() {
    local dest="$BACKUP_ROOT/files/daily"
    mkdir -p "$dest"

    log "Sauvegarde fichiers de configuration et addon"
    tar czf "$dest/files_{DATE}.tar.gz" \
        -C "$DOCKER_DIR" \
        --exclude='mysql-data' \
        --exclude='*.log' \
        addons/ \
        gamemodes/ \
        server-config/ \
        docker-compose.yml \
        2>/dev/null

    local size
    size=$(du -h "$dest/files_{DATE}.tar.gz" | cut -f1)
    log "Fichiers OK : files_{DATE}.tar.gz ($size)"
}

backup_docker_images() {
    local dest="$BACKUP_ROOT/images"
    mkdir -p "$dest"

    log "Export des images Docker taguées"
    for tag in v1.0-base v1.1-mysql v2-stable v2.1-stable v2.2-vehicles; do
        local image="projetfilrouge/gmod-server:$tag"
        if docker image inspect "$image" &>/dev/null; then
            docker save "$image" | gzip > "$dest/${tag}_{DATE}.tar.gz"
            log "Image exportée : $tag"
        fi
    done
}

```

```

generate_checksum() {
    log "Génération des checksums SHA-256"
    find "$BACKUP_ROOT" -name "*_${DATE}*" -type f | while read -r file; do
        sha256sum "$file" >> "$BACKUP_ROOT/checksums_${DATE}.sha256"
    done
    log "Checksums : checksums_${DATE}.sha256"
}

cleanup_old() {
    log "Nettoyage des anciennes sauvegardes"

    # Daily : garder N jours
    find "$BACKUP_ROOT/mysql/daily" -name "*.sql.gz" -mtime +$RETENTION_DAILY -delete 2>/
    find "$BACKUP_ROOT/files/daily" -name "*.tar.gz" -mtime +$RETENTION_DAILY -delete 2>/

    # Monthly : garder N mois
    find "$BACKUP_ROOT/mysql/monthly" -name "*.sql.gz" -mtime +$(($RETENTION_MONTHLY * 30)) -delete 2>/
    find "$BACKUP_ROOT/files/monthly" -name "*.tar.gz" -mtime +$(($RETENTION_MONTHLY * 30)) -delete 2>/

    # Checksums anciens
    find "$BACKUP_ROOT" -name "checksums_*.sha256" -mtime +$RETENTION_DAILY -delete 2>/
}

log "Nettoyage terminé (rétention: ${RETENTION_DAILY}j daily, ${RETENTION_MONTHLY} mois monthly)"

# --- Main ---
main() {
    log "===== DÉBUT SAUVEGARDE ====="
    check_disk_space

    case "${1:-daily}" in
        hourly)
            backup_mysql_hourly
            ;;
        daily)
            backup_mysql
            backup_files
            generate_checksum
            cleanup_old
            ;;
        --full|monthly)
            backup_mysql
            backup_files
            backup_docker_images
            generate_checksum
            # Copie vers répertoire monthly
            mkdir -p "$BACKUP_ROOT/mysql/monthly" "$BACKUP_ROOT/files/monthly"
            cp "$BACKUP_ROOT/mysql/daily/mysql_${DATE}.sql.gz" "$BACKUP_ROOT/mysql/monthly/mysql_${DATE}.sql.gz"
            cp "$BACKUP_ROOT/files/daily/files_${DATE}.tar.gz" "$BACKUP_ROOT/files/monthly/files_${DATE}.tar.gz"
            cleanup_old
            ;;
        *)
    esac
}

```

```

        echo "Usage: $0 {hourly|daily|--full|monthly}"
        exit 1
    ;;
esac

log "===== FIN SAUVEGARDE ====="
}

main "$@"

```

Planification cron

```

# Sauvegarde MySQL horaire
0 * * * * /root/scripts/backup.sh hourly >> /var/log/backup-gmod.log 2>&1

# Sauvegarde complète quotidienne à 03h00
0 3 * * * /root/scripts/backup.sh daily >> /var/log/backup-gmod.log 2>&1

# Sauvegarde mensuelle complète (avec images Docker)
0 4 1 * * /root/scripts/backup.sh --full >> /var/log/backup-gmod.log 2>&1

```

Arborescence des sauvegardes

```

/root/backups/
├── mysql/
│   ├── hourly/           ← Dumps horaires (rotation 24)
│   │   └── mysql_hourly_2025-02-12_140000.sql.gz
│   │   ...
│   ├── daily/            ← Dumps quotidiens (rétention 7j)
│   │   └── mysql_2025-02-12_030000.sql.gz
│   │   ...
│   └── monthly/          ← Dumps mensuels (rétention 3 mois)
│       └── mysql_2025-02-01_040000.sql.gz
└── files/
    ├── daily/             ← Archives config (rétention 7j)
    │   └── files_2025-02-12_030000.tar.gz
    │   ...
    └── monthly/
        └── files_2025-02-01_040000.tar.gz
└── images/              ← Exports Docker (mensuel)
    └── v2.2-vehicles_2025-02-01_040000.tar.gz
    └── checksums_2025-02-12_030000.sha256

```

6. Argumentation des choix techniques — C22.2

Pourquoi `mysqldump` ?

| Outil | Avantages | Inconvénients | Verdict |
|--------------------------|--|--|----------|
| <code>mysqldump</code> | Natif MySQL, fiable, portable, SQL lisible | Lent sur grosses bases, lock possible | □ Retenu |
| <code>mysqlpump</code> | Parallélisme, plus rapide | Moins mature, bugs connus MySQL 8.0 | □ |
| <code>xtrabackup</code> | Backup à chaud, incrémental physique | Nécessite installation séparée, overkill pour ~50 Mo | □ |
| RéPLICATION MySQL | Temps réel, aucune perte | Nécessite 2ème serveur, budget inadapté | □ Futur |

Justification : Pour une base de ~50 Mo, `mysqldump` avec `--single-transaction` offre un backup cohérent sans verrouillage, en quelques secondes. La complexité d'outils plus avancés n'est pas justifiée à cette échelle.

Pourquoi `tar + gzip` pour les fichiers ?

| Outil | Avantages | Inconvénients | Verdict |
|-------------------------|------------------------------------|--|----------|
| <code>tar + gzip</code> | Universel, rapide, natif Linux | Pas d'incrémental natif | □ Retenu |
| <code>rsync</code> | Incrémental, efficace réseau | Nécessite destination réseau pour bénéfice | □ |
| <code>borgbackup</code> | Déduplication, chiffrement intégré | Installation supplémentaire, complexité | □ Futur |
| <code>restic</code> | Cloud-ready, déduplication | Nécessite backend distant | □ Futur |

Justification : Les fichiers de configuration totalisent ~2 Mo. L'overhead d'outils de déduplication n'est pas justifié. `tar + gzip` est fiable, vérifiable, et ne nécessite aucune dépendance.

Pourquoi Git comme backup du code ?

Git n'est pas un outil de backup à proprement parler, mais pour le code source, il offre : - **Historique complet** de chaque modification - **Stockage distant** sur GitHub (hors site) - **Intégrité cryptographique** (chaque commit est un hash SHA-1) - **Restauration granulaire** (n'importe quel commit, n'importe quel fichier)

Pour le code Lua de l'addon, Git est **supérieur** à un backup fichier classique car il conserve l'historique des changements, pas seulement le dernier état.

7. Récapitulatif de conformité

| Critère | Exigence | Réponse apportée | Référence |
|---------|----------------------------------|---|-----------|
| C20.1 | Adéquation aux contraintes du SI | Classification par criticité, volumétrie, politique 3-2-1 adaptée au budget | §1-2 |
| C20.2 | Continuité d'activité | RPO/RTO définis par composant, mode dégradé, scripts automatisés | §4-5 |
| C22.2 | Argumentation technique | Tableaux comparatifs pour chaque outil, justification documentée | §6 |

□ Plan de restauration et sécurité des sauvegardes

Critères adressés : C21.1 (Sécurité physique et logique des données sauvegardées), C21.2 (Tests de restauration fonctionnels), C22.2 (Argumentation des choix techniques)

1. Sécurité physique des sauvegardes — C21.1

Localisation et accès

| Copie | Localisation | Accès | Protection |
|---------------|--|------------------------|---------------------------------------|
| Originale | VPS Hostinger (/root/ ProjetFilRouge/docker/) | SSH root uniquement | Firewall UFW, clé SSH |
| Backup local | VPS (/root/backups/) | SSH root uniquement | Permissions 700, propriétaire root |
| Code distant | GitHub (privé → public pour le projet) | PAT + SSH key | 2FA GitHub activé |
| Images Docker | VPS (/root/backups/ images/) | SSH root uniquement | Exports compressés |

Mesures de sécurité physique

Contrôle d'accès au VPS

```
# Accès SSH uniquement par clé (pas de mot de passe)
# /etc/ssh/sshd_config
PasswordAuthentication no
PubkeyAuthentication yes
PermitRootLogin prohibit-password

# Firewall UFW
ufw allow 22/tcp      # SSH
ufw allow 27015        # GMod
ufw deny incoming     # Tout le reste bloqué
```

Permissions des répertoires de backup

```
# Seul root peut accéder aux backups
chmod 700 /root/backups
chmod 600 /root/backups/mysql/**/*.*.sql.gz
chmod 600 /root/backups/files/**/*.*.tar.gz
chmod 600 /root/backups/checksums_*.*.sha256
```

Isolation des conteneurs

Les conteneurs Docker n'ont **pas accès** au répertoire de backup : - Le volume `mysql-data` est distinct de `/root/backups/` - Les bind mounts ne montent que les répertoires nécessaires (addons, config) - Un conteneur compromis ne peut pas altérer les sauvegardes

2. Sécurité logique des sauvegardes – C21.1

Intégrité : Checksums SHA-256

Chaque backup génère un fichier de checksums :

```
# Vérification d'intégrité
sha256sum -c /root/backups/checksums_2025-02-12_030000.sha256

# Sortie attendue :
# /root/backups/mysql/daily/mysql_2025-02-12_030000.sql.gz: OK
# /root/backups/files/daily/files_2025-02-12_030000.tar.gz: OK
```

Chiffrement (amélioration implémentable)

Pour une sécurité renforcée, les backups peuvent être chiffrés avec GPG :

```
# Chiffrement d'un dump MySQL
gpg --symmetric --cipher-algo AES256 \
    --output mysql_2025-02-12.sql.gz.gpg \
    mysql_2025-02-12.sql.gz

# Déchiffrement
gpg --decrypt mysql_2025-02-12.sql.gz.gpg > mysql_2025-02-12.sql.gz
```

État actuel : Non implémenté en production car les données (logs de construction GMod) ne contiennent pas d'informations personnelles sensibles. Le chiffrement est préparé et documenté pour activation si le SI évolue vers des données plus sensibles.

Protection contre la suppression accidentelle

```
# Attribut immutable sur les backups mensuels (protection suppression root)
chattr +i /root/backups/mysql/monthly/*.sql.gz
chattr +i /root/backups/files/monthly/*.tar.gz

# Pour modifier/supprimer : chattr -i <fichier> d'abord
```

Matrice des menaces et contre-mesures

| Menace | Probabilité | Impact | Contre-mesure |
|------------------------------------|-------------|----------|--|
| Suppression accidentelle (rm) | Moyenne | Critique | <code>chattr +i</code> sur mensuels, rétention multi-niveaux |
| Ransomware/chiffrement malveillant | Faible | Critique | Copie GitHub hors VPS, backups avec permissions restreintes |
| Corruption disque | Faible | Élevé | Checksums SHA-256, vérification post-backup |
| Accès non autorisé SSH | Faible | Critique | Clé SSH uniquement, fail2ban, UFW |
| Compromission conteneur Docker | Faible | Moyen | Isolation volumes, backups hors conteneurs |
| Perte totale VPS (datacenter) | Très faible | Critique | Code sur GitHub, images Docker reconstituables |

3. Procédures de restauration — C21.2

3.1 Restauration MySQL

```
#!/bin/bash
# restore_mysql.sh – Restauration de la base de données
set -euo pipefail

BACKUP_FILE="${1:?Usage: $0 <fichier_backup.sql.gz>}"
MYSQL_CONTAINER="gmod-mysql"
MYSQL_USER="root"
MYSQL_PASS="GmodSecurePass2025!"
MYSQL_DB="gmod_construction"

echo "[*] Vérification intégrité du backup..."
gzip -t "$BACKUP_FILE" || { echo "ERREUR: Archive corrompue"; exit 1; }

echo "[*] Vérification checksum..."
BACKUP_DIR=$(dirname "$BACKUP_FILE")
BACKUP_NAME=$(basename "$BACKUP_FILE")
CHECKSUM_FILE=$(ls -t "$BACKUP_DIR"/../checksums_*sha256 2>/dev/null | head -1)
if [ -n "$CHECKSUM_FILE" ]; then
    grep "$BACKUP_NAME" "$CHECKSUM_FILE" | sha256sum -c - || echo "WARN: Checksum non trouvé"
fi

echo "[*] Arrêt du serveur GMod (éviter les écritures concurrentes)..."
docker stop gmod-server 2>/dev/null || true

echo "[*] Restauration de $BACKUP_FILE vers $MYSQL_DB..."
zcat "$BACKUP_FILE" | docker exec -i "$MYSQL_CONTAINER" \
    mysql -u"$MYSQL_USER" -p"$MYSQL_PASS" "$MYSQL_DB"

echo "[*] Vérification post-restauration..."
docker exec "$MYSQL_CONTAINER" mysql -u"$MYSQL_USER" -p"$MYSQL_PASS" "$MYSQL_DB" \
    -e "SELECT COUNT(*) as total_logs FROM blueprint_logs;" 2>/dev/null || echo "Table log non trouvée"

echo "[*] Redémarrage du serveur GMod..."
docker start gmod-server

echo "[✓] Restauration MySQL terminée avec succès"
```

3.2 Restauration des fichiers de configuration

```
#!/bin/bash
# restore_files.sh – Restauration des fichiers de configuration
set -euo pipefail

BACKUP_FILE="${1:?Usage: $0 <fichier_backup.tar.gz>}"
DOCKER_DIR="/root/ProjetFilRouge/docker"

echo "[*] Vérification intégrité..."
gzip -t "$BACKUP_FILE" || { echo "ERREUR: Archive corrompue"; exit 1; }

echo "[*] Sauvegarde de l'état actuel (sécurité)..."
SAFETY_BACKUP="/tmp/pre-restore_$(date +%s).tar.gz"
tar czf "$SAFETY_BACKUP" -C "$DOCKER_DIR" addons/ gamemodes/ server-config/ docker-compo...
echo "    Backup de sécurité: $SAFETY_BACKUP"

echo "[*] Arrêt des services..."
cd "$DOCKER_DIR"
docker compose down

echo "[*] Restauration depuis $BACKUP_FILE..."
tar xzf "$BACKUP_FILE" -C "$DOCKER_DIR"

echo "[*] Redémarrage des services..."
docker compose up -d

echo "[*] Vérification santé des conteneurs..."
sleep 10
docker ps --format "table {{.Names}}\t{{.Status}}"

echo "[✓] Restauration fichiers terminée"
echo "    En cas de problème, backup de sécurité: $SAFETY_BACKUP"
```

3.3 Restauration du code depuis Git

```
# Restauration complète depuis GitHub
git clone https://github.com/yguerch212-creator/Projet_fil_rouge.git /root/ProjetFilRouge

# Restauration d'un fichier spécifique
git checkout HEAD -- docker/addons/rp_construction_system/

# Restauration à un commit précis
git checkout abc1234 -- docker/addons/rp_construction_system/
```

3.4 Restauration d'une image Docker

```
# Depuis un export sauvegardé
docker load < /root/backups/images/v2.2-vehicles_2025-02-01_040000.tar.gz

# Rebuild depuis le tag existant
docker tag projetfilrouge/gmod-server:v2.2-vehicles projetfilrouge/gmod-server:jour2-sta
```

3.5 Restauration complète (disaster recovery)

Procédure en cas de perte totale du VPS :

```
# 1. Nouveau VPS – Installation des prérequis
apt update && apt install -y docker.io docker-compose-v2 git

# 2. Récupération du code
git clone https://github.com/yguerch212-creator/Projet_fil_rouge.git /root/ProjetFilRouge

# 3. Restauration des backups MySQL (si disponibles)
# → Copier depuis stockage externe ou backup local survivant

# 4. Démarrage de l'infrastructure
cd /root/ProjetFilRouge/docker
docker compose up -d

# 5. Restauration MySQL
./restore_mysql.sh /path/to/mysql_backup.sql.gz

# 6. Vérification
docker ps
docker logs gmod-server --tail 50
```

RTO estimé : 2-4 heures (incluant provisioning VPS, installation, restauration)

4. Tests de restauration — C21.2

Plan de tests

| Test | Fréquence | Procédure | Critère de succès |
|------------------------------|---------------|---|--|
| T1 — Intégrité backup | Chaque backup | <code>gzip -t</code> + <code>sha256sum -c</code> | Exit code 0, checksums valides |
| T2 — Restore MySQL | Mensuel | Restauration vers base de test | Données identiques à l'original |
| T3 — Restore fichiers | Mensuel | Extraction dans répertoire temporaire | Fichiers intacts, permissions correctes |
| T4 — Restore complet | Trimestriel | Simulation disaster recovery | Serveur fonctionnel en < 4h |

Script de test automatisé

```

#!/bin/bash
# test_restore.sh – Vérification automatisée des backups
set -euo pipefail

BACKUP_ROOT="/root/backups"
TEST_DIR="/tmp/restore_test_$$"
ERRORS=0

log() { echo "[TEST $(date '+%H:%M:%S')] $1"; }

mkdir -p "$TEST_DIR"

# --- T1 : Intégrité des archives ---
log "T1 – Vérification intégrité des archives"
for gz in "$BACKUP_ROOT"/mysql/daily/*.sql.gz "$BACKUP_ROOT"/files/daily/*.tar.gz; do
    [ -f "$gz" ] || continue
    if ! gzip -t "$gz" 2>/dev/null; then
        log "FAIL: $gz est corrompu"
        ((ERRORS++))
    fi
done
log "T1 – $([ $ERRORS -eq 0 ] && echo 'PASS' || echo 'FAIL')"

# --- T2 : Restauration MySQL dans base de test ---
log "T2 – Test restauration MySQL"
LATEST_MYSQL=$(ls -t "$BACKUP_ROOT"/mysql/daily/*.sql.gz 2>/dev/null | head -1)
if [ -n "$LATEST_MYSQL" ]; then
    # Créer base de test
    docker exec gmod-mysql mysql -uroot -pGmodSecurePass2025! \
        -e "CREATE DATABASE IF NOT EXISTS gmod_test_restore;" 2>/dev/null

    # Restaurer
    zcat "$LATEST_MYSQL" | sed 's/gmod_construction/gmod_test_restore/g' | \
        docker exec -i gmod-mysql mysql -uroot -pGmodSecurePass2025! gmod_test_restore 2>/dev/null

    # Vérifier
    TABLES=$(docker exec gmod-mysql mysql -uroot -pGmodSecurePass2025! gmod_test_restore \
        -e "SHOW TABLES;" 2>/dev/null | wc -l)

    if [ "$TABLES" -gt 1 ]; then
        log "T2 – PASS ($((TABLES-1)) tables restaurées)"
    else
        log "T2 – FAIL (aucune table)"
        ((ERRORS++))
    fi

    # Nettoyage
    docker exec gmod-mysql mysql -uroot -pGmodSecurePass2025! \
        -e "DROP DATABASE gmod_test_restore;" 2>/dev/null
else
    log "T2 – SKIP (aucun backup MySQL trouvé)"
fi

```

```

# --- T3 : Restauration fichiers ---
log "T3 – Test restauration fichiers"
LATEST_FILES=$(ls -t "$BACKUP_ROOT"/files/daily/*.tar.gz 2>/dev/null | head -1)
if [ -n "$LATEST_FILES" ]; then
    tar xzf "$LATEST_FILES" -C "$TEST_DIR" 2>/dev/null

    # Vérifier présence des fichiers critiques
    CHECKS=0
    [ -d "$TEST_DIR/addons/rp_construction_system" ] && ((CHECKS++))
    [ -f "$TEST_DIR/docker-compose.yml" ] && ((CHECKS++))
    [ -d "$TEST_DIR/server-config" ] && ((CHECKS++))

    if [ "$CHECKS" -ge 3 ]; then
        log "T3 – PASS ($CHECKS/3 vérifications)"
    else
        log "T3 – FAIL ($CHECKS/3 vérifications)"
        ((ERRORS++))
    fi
else
    log "T3 – SKIP (aucun backup fichiers trouvé)"
fi

# --- Nettoyage ---
rm -rf "$TEST_DIR"

# --- Résultat ---
echo ""
if [ $ERRORS -eq 0 ]; then
    log "[] TOUS LES TESTS PASSENT"
else
    log "[] $ERRORS ERREUR(S) DÉTECTÉE(S)"
fi

exit $ERRORS

```

Résultat type d'un test

```

[TEST 15:30:01] T1 – Vérification intégrité des archives
[TEST 15:30:02] T1 – PASS
[TEST 15:30:02] T2 – Test restauration MySQL
[TEST 15:30:05] T2 – PASS (3 tables restaurées)
[TEST 15:30:05] T3 – Test restauration fichiers
[TEST 15:30:06] T3 – PASS (3/3 vérifications)

[TEST 15:30:06] [] TOUS LES TESTS PASSENT

```

5. Argumentation des choix — C22.2

Stratégie locale vs cloud

| Critère | Backup local (VPS) | Backup cloud (S3/Backblaze) |
|----------------------|--------------------|-----------------------------|
| Coût | 0€ (inclus VPS) | ~2-5€/mois |
| Latence restauration | Instantanée | Dépend bande passante |
| Protection sinistre | □ Même datacenter | □ Géo-répliqué |
| Complexité | Faible | Moyenne (credentials, SDK) |
| RGPD | Même juridiction | Vérifier localisation |

Choix : Backup local + GitHub pour le code. Le budget ne justifie pas un stockage cloud dédié pour ~8 Mo de données. Amélioration prévue si le projet évolue vers la production.

Pourquoi pas de réPLICATION MySQL ?

La réPLICATION (master-slave) offrirait un RPO quasi nul, mais : - Nécessite un **deuxième serveur** (coût) - **Surdimensionné** pour ~50 Mo de logs - **Complexité** de maintenance disproportionnée

Le dump horaire avec `--single-transaction` couvre le besoin avec un RPO acceptable de 1 heure.

6. Récapitulatif de conformité

| Critère | Exigence | Réponse apportée | Référence |
|---------|------------------------------------|--|-----------|
| C21.1 | Sécurité physique et logique | Permissions restrictives, checksums SHA-256, isolation conteneurs, chiffrement documenté | §1-2 |
| C21.2 | Tests de restauration fonctionnels | 4 niveaux de tests, scripts automatisés, procédures détaillées | §3-4 |
| C22.2 | Argumentation technique | Comparatifs local vs cloud, justification mysqldump, stratégie rétention | §5 |

□ Scénario de test complet – Démonstration end-to-end

Critères adressés : C21.2 (Tests de restauration fonctionnels), C22.1 (Clarté, rigueur et structure du propos)

Contexte du test

Date : Février 2025

Environnement : VPS Hostinger (16 Go RAM, Ubuntu 22.04)

Infrastructure : Docker Compose (GMod + MySQL)

Objectif : Valider le plan de sauvegarde et restauration de bout en bout

Scénario 1 : Corruption de la base de données MySQL

Situation initiale

```
# État de la base avant le test
$ docker exec gmod-mysql mysql -uroot -pGmodSecurePass2025! gmod_construction \
    -e "SELECT COUNT(*) as logs FROM blueprint_logs;"

+-----+
| logs |
+-----+
|    47 |
+-----+
```

Étape 1 — Sauvegarde

```
$ /root/scripts/backup.sh daily
[2025-02-12 03:00:01] ===== DÉBUT SAUVEGARDE =====
[2025-02-12 03:00:01] Espace disque disponible: 12847Mo
[2025-02-12 03:00:02] Sauvegarde MySQL : gmod_construction
[2025-02-12 03:00:03] MySQL OK : mysql_2025-02-12_030000.sql.gz (4.2K)
[2025-02-12 03:00:03] Sauvegarde fichiers de configuration et addon
[2025-02-12 03:00:04] Fichiers OK : files_2025-02-12_030000.tar.gz (1.8M)
[2025-02-12 03:00:04] Génération des checksums SHA-256
[2025-02-12 03:00:04] Checksums : checksums_2025-02-12_030000.sha256
[2025-02-12 03:00:05] Nettoyage terminé (rétention: 7j daily, 3 mois monthly)
[2025-02-12 03:00:05] ===== FIN SAUVEGARDE =====
```

Étape 2 — Simulation de corruption

```
# Suppression simulée des données (environnement de test)
$ docker exec gmod-mysql mysql -uroot -pGmodSecurePass2025! gmod_construction \
    -e "DROP TABLE blueprint_logs;"

# Vérification : table absente
$ docker exec gmod-mysql mysql -uroot -pGmodSecurePass2025! gmod_construction \
    -e "SHOW TABLES;"

+-----+
| Tables_in_gmod_construction |
+-----+
| construction_blueprints     |
+-----+
# → blueprint_logs a disparu
```

Étape 3 — Restauration

```
$ /root/scripts/restore_mysql.sh /root/backups/mysql/daily/mysql_2025-02-12_030000.sql.gz
[*] Vérification intégrité du backup...
[*] Arrêt du serveur GMod (éviter les écritures concurrentes)...
gmod-server
[*] Restauration de mysql_2025-02-12_030000.sql.gz vers gmod_construction...
[*] Vérification post-restauration...
+-----+
| total_logs |
+-----+
|      47   |
+-----+
[*] Redémarrage du serveur GMod...
gmod-server
[✓] Restauration MySQL terminée avec succès
```

Résultat

| Vérification | Attendu | Obtenu | Statut |
|--|----------|--------|--------|
| Table <code>blueprint_logs</code> existe | Oui | Oui | OK |
| Nombre d'enregistrements | 47 | 47 | OK |
| Serveur GMod fonctionnel | Oui | Oui | OK |
| Durée totale restauration | < 30 min | ~3 min | OK |

Scénario 2 : Suppression accidentelle de l'addon

Situation initiale

```
$ ls /root/ProjetFilRouge/docker/addons/rp_construction_system/
lua/ README.md addon.json sql/
```

Étape 1 — Simulation de suppression

```
# Suppression accidentelle de l'addon
$ rm -rf /root/ProjetFilRouge/docker/addons/rp_construction_system/

# Le serveur GMod ne charge plus l'addon
$ docker exec gmod-server ls garrysmod/addons/ | grep construction
# → aucun résultat
```

Étape 2 — Restauration via Git

```
$ cd /root/ProjetFilRouge
$ git checkout HEAD -- docker/addons/rp_construction_system/

$ ls docker/addons/rp_construction_system/
lua/ README.md addon.json sql/
```

Étape 3 — Redémarrage du serveur

```
$ docker restart gmod-server
# Le bind mount recharge automatiquement l'addon
```

Résultat

| Vérification | Attendu | Obtenu | Statut |
|----------------------------|----------|-------------------|--------|
| Fichiers addon restaurés | Tous | Tous | OK |
| Serveur charge l'addon | Oui | Oui | OK |
| Blueprints joueurs intacts | Oui | Oui (côté client) | OK |
| Durée totale | < 10 min | ~2 min | OK |

Scénario 3 : Restauration fichiers de configuration

Simulation

```
# Corruption du docker-compose.yml
$ echo "invalid yaml" > /root/ProjetFilRouge/docker/docker-compose.yml

# Docker Compose ne peut plus démarrer
$ cd /root/ProjetFilRouge/docker && docker compose up -d
# → Error: yaml: unmarshal errors
```

Restauration

```
# Méthode 1 : Git (rapide)
$ git checkout HEAD -- docker/docker-compose.yml

# Méthode 2 : Backup fichiers (si Git indisponible)
$ tar xzf /root/backups/files/daily/files_2025-02-12_030000.tar.gz \
-C /root/ProjetFilRouge/docker/ docker-compose.yml

# Redémarrage
$ docker compose up -d
# → Les deux services démarrent correctement
```

Résultat

| Vérification | Attendu | Obtenu | Statut |
|---------------------------|---------|--------|--------|
| docker-compose.yml valide | Oui | Oui | OK |
| Services démarrés | 2/2 | 2/2 | OK |
| Durée totale | < 5 min | ~1 min | OK |

Synthèse des tests

| Scénario | Type de perte | Méthode de restauration | RTO cible | RTO réel | Données perdues |
|----------|----------------------|-------------------------|-----------|----------|-----------------|
| S1 | Base MySQL corrompue | Dump SQL + script | < 30 min | 3 min | 0 (RPO < 1h) |
| S2 | Addon supprimé | Git checkout | < 10 min | 2 min | 0 |
| S3 | Config corrompue | Git / tar backup | < 5 min | 1 min | 0 |

Observations

1. **Le RTO réel est très inférieur au RTO cible** dans tous les scénarios, grâce aux scripts automatisés et à la taille réduite des données.
2. **Git est la première ligne de défense** pour tout ce qui est versionné (code, config, docker-compose). Les backups fichiers servent de filet de sécurité si le repo est compromis.
3. **Les blueprints joueurs sont naturellement protégés** : stockés côté client dans `data/construction_blueprints/`, ils ne dépendent pas du serveur. Ce choix architectural est un atout majeur pour la résilience.
4. **Le mode dégradé fonctionne** : même sans MySQL, le serveur GMod et l'addon fonctionnent (seuls les logs sont indisponibles).

Préparation à la soutenance — C22.3

Questions anticipées du jury

Q1 : Pourquoi ne pas utiliser un stockage cloud pour les backups ?

Budget contraint (VPS mutualisé). Pour ~8 Mo de données, le coût d'un S3 n'est pas justifié. GitHub couvre le code. Évolution prévue si passage en production.

Q2 : Comment gardez-vous les backups ne sont pas eux-mêmes corrompus ?

Triple vérification : `gzip -t` (intégrité archive), checksums SHA-256 (intégrité contenu), tests de restauration mensuels (fonctionnalité).

Q3 : Que se passe-t-il si le VPS est totalement perdu ?

Le code est sur GitHub (restauration < 5 min). MySQL est perdu jusqu'au dernier dump copié hors VPS. RTO total estimé : 2-4h avec un nouveau VPS. Amélioration : export chiffré des dumps vers stockage distant.

Q4 : Les données sont-elles conformes RGPD ?

Les seules données stockées sont des SteamID (pseudonymes publics) et des logs de construction (actions en jeu). Pas de données personnelles sensibles au sens du RGPD. Néanmoins, les bonnes pratiques sont appliquées : accès restreint, chiffrement documenté, rétention limitée.

Q5 : Pourquoi des backups horaires pour MySQL et quotidiens pour les fichiers ?

La base MySQL est la seule donnée qui change fréquemment (logs en temps réel). Les fichiers de configuration changent rarement (uniquement lors de modifications manuelles). La fréquence est adaptée au rythme de modification de chaque type de donnée.

Q6 : Comment testez-vous automatiquement les restaurations ?

Le script `test_restore.sh` vérifie l'intégrité des archives, restaure MySQL dans une base de test temporaire, et extrait les fichiers dans un répertoire temp. Exécution mensuelle via cron. Les résultats sont loggés.

Q7 : Quelle est la différence entre votre RPO et RTO ?

RPO = perte de données maximale acceptable (1h pour MySQL, temps réel pour le code Git). RTO = temps pour remettre en service (3-30 min)

selon le scénario). Le RPO dépend de la fréquence de backup, le RTO dépend de la vitesse de restauration.
