D_la solution contacts et calendrier

Solution d'unification des contacts et du calendrier de Julien.

État au 3 août 2026 — déployé et opérationnel de bout en bout. Seul vdirsyncer (agrégation Google/Infomaniak) reste à faire.

---

1. Objectif

Unifier la base de contacts et de calendriers, dispersée entre plusieurs sources (Google, Infomaniak, téléphone), autour d'un point central auto-hébergé sur le VPS juxjux.ovh, synchronisé vers tous les clients existants.

Baïkal est la **source de vérité unique**. Roundcube, InfCloud et DAVx5 en sont des clients ; vdirsyncer sera le seul composant qui écrit depuis l'extérieur vers Baïkal.

2. Architecture réellement déployée

Composant	Rôle	Emplacement	État
Baïkal	Serveur CardDAV + CalDAV, stockage central	Docker, VPS juxjux	En service
InfCloud	Agenda **et** contacts web	Statique, servi par nginx	En service
Roundcube	Interface web des contacts	Docker, VPS juxjux	En service
Nginx	Reverse proxy HTTPS	Hôte VPS	En service
DAVx5	Pont de synchronisation vers Android	Téléphone	**Opérationnel**

Composant	Rôle	Emplacement	État
Fossify Agenda	Client calendrier	Téléphone (via DAVx5)	Opérationnel
vdirsyncer	Agrège Google et Infomaniak vers Baïkal	VPS, cron	**Non déployé**
Thunderbird	Client mail, indépendant de ce circuit	Téléphone	Inchangé

3. Accès

Service	URL	Identifiant
Agenda + contacts web	https://baikal.juxjux.ovh/infcloud/	`Julien` (compte Baïkal)
Contacts web (Roundcube)	https://secretariat.juxjux.ovh	`julien.bertrand@ik.me` (compte mail Infomaniak)
Administration Baïkal	https://baikal.juxjux.ovh/admin/	compte admin Baïkal
Découverte DAV (DAVx5)	`https://baikal.juxjux.ovh/dav.php/`	`Julien`

Utilisateur Baïkal : `Julien` — avec un J majuscule. La casse compte dans les URLs DAV.

- Principal : `https://baikal.juxjux.ovh/dav.php/principals/Julien/`
- Contacts : `https://baikal.juxjux.ovh/dav.php/addressbooks/Julien/default/`
- Calendrier : `https://baikal.juxjux.ovh/dav.php/calendars/Julien/default/`

Certificats Let's Encrypt valides jusqu'au **1er novembre 2026** pour les deux domaines. Celui de `baikal.juxjux.ovh` existait déjà mais avait **expiré le 8 février 2026** (vestige d'une tentative antérieure) ; il a été renouvelé le 3 août.

4. Déploiement

Stack sur le VPS : `/home/debian/baikal/docker-compose.yml`, projet compose `baikal`. Sources versionnées : `Syncthing/Jux_univers/Jux-scripts/Baikal-Contacts/`.

Fichier	Rôle
`docker-compose.yml`	Stack Baïkal + Roundcube
`baikal.juxjux.ovh.conf`	Vhost Baïkal (`.well-known` forcés en HTTPS + service d'InfCloud)
`secretariat.juxjux.ovh.conf`	Vhost Roundcube

Fichier	Rôle
`infcloud-config.js`	Configuration InfCloud
`finaliser_tls.sh`	Vérifie le DNS puis lance certbot
`vps_backup.sh`	Script de sauvegarde complet, Baïkal inclus
`push_bookstack.py`	Publication de cette page via l'API BookStack

```
cd /home/debian/baikal && sudo docker compose -p baikal up -d
```

Service	Image	Port local	Base	RAM
baikal	`ckulka/baikal:nginx`	127.0.0.1:8088	SQLite	~38 Mo
roundcube	`roundcube/roundcubemail:latest` (1.7.2)	127.0.0.1:8083	SQLite	~32 Mo
InfCloud	fichiers statiques (13 Mo sur disque)	—	aucune	**0**

Les deux containers pèsent **environ 70 Mo au total**, contre environ 600 Mo avec l'architecture MariaDB initialement prévue. Ce choix était important : le VPS était en pression mémoire au moment du déploiement (swap à 3,8/4,0 Go).

5. Écarts avec le plan initial

Le plan préparé prévoyait une configuration qui ne pouvait pas fonctionner en l'état. Écarts constatés et corrigés :

Point	Prévu	Réel	Raison
Port Baïkal	8081	**8088**	8081 est occupé par FreshRSS. 8088 correspond au vhost `baikal.juxjux.ovh` préexistant
Port Roundcube	8082	**8083**	8082 est ciblé par un vhost `nextcloud` mort
Domaine Roundcube	`mail.juxjux.ovh`	**`secretariat.juxjux.ovh`**	Aucun DNS n'existait ; `mail.` prêtait à confusion avec les MX OVH du domaine
Base Roundcube	MariaDB	**SQLite**	~400 Mo de RAM économisés, suffisant en mono-utilisateur

Point	Prévu	Réel	Raison
Vhost Baïkal	À créer	**Déjà présent**	Le fichier `baikal.conf` préparé n'a pas servi
Agenda web	Plugin calendar Roundcube	**InfCloud**	Plugin inutilisable, voir §6.2

6. Pièges rencontrés

6.1 Les plugins carddav et calendar ne sont pas dans l'image Roundcube

L'image officielle ne fournit ni `carddav` ni `calendar`. Les déclarer dans `ROUNDCUBEMAIL_PLUGINS` ne fait que les *activer* — s'ils sont absents, Roundcube part en erreur fatale.

Il faut les faire installer par composer au démarrage, via **`ROUNDCUBEMAIL_COMPOSER_PLUGINS`** (et non `ROUNDCUBEMAIL_INSTALL_PLUGINS`, qui n'existe pas dans cette image) :

```
ROUNDCUBEMAIL_COMPOSER_PLUGINS: "roundcube/carddav"
ROUNDCUBEMAIL_PLUGINS: "carddav"
```

Le premier démarrage est alors plus long (téléchargement composer).

6.2 Le plugin calendar de Roundcube est une impasse

Trois obstacles successifs, testés le 3 août :

a) Il casse le tout premier démarrage. Le script d'initialisation de base de `kolab/calendar` s'exécute **avant** que l'entrypoint n'écrive la configuration Roundcube : il tombe sur le DSN MySQL par défaut et échoue en `SQLSTATE[HY000] [2002] No such file or directory`. Cette erreur fatale empêche la régénération de l'autoloader composer, ce qui casse **aussi** le plugin carddav :

```
PHP Fatal error: Uncaught Error: Interface
"MStilkerich\RCMCardDAV\Frontend\RcmInterface" not found
in /var/www/html/plugins/carddav/carddav.php:43
```

Symptôme : HTTP 500 permanent, alors que les fichiers du plugin sont bien présents sur le disque — c'est `vendor/composer/autoload_psr4.php` qui ne contient aucune entrée carddav.

b) Ce premier obstacle est contournable. Réinstallé *après* que Roundcube ait été initialisé une première fois, la configuration existe déjà dans le volume et l'initialisation aboutit (`Creating database schema... [OK]`). Vérifié.

c) Mais le plugin ne sait pas parler à Baïkal. La seule version installable, `kolab/calendar` 3.2.9.1, ne livre que les pilotes `database`, `kolab` et `ldap` — **aucun pilote** `caldav`. Il ne créerait qu'un agenda **local** dans Roundcube, sans lien avec Baïkal ni le téléphone : pire qu'inutile, on pourrait y saisir des rendez-vous en croyant qu'ils se synchronisent.

Les versions qui embarquent le pilote CalDAV (3.5.7, 3.6.1) dépendent de `kolab/libkolab`, qui réclame `pear/http_request2` — paquet bloqué par composer pour **faille de sécurité** (`PKSA-jrt3-xndd-g4sz`). Ne pas contourner ce garde-fou sur un service exposé sur internet.

Conclusion : agenda web assuré par InfCloud (§7), plugin calendar définitivement écarté.

6.3 Baïkal ignore X-Forwarded-Proto — fuite HTTP sur la découverte

Baïkal ne tient pas compte de `X-Forwarded-Proto` et générait ses redirections de découverte en **HTTP nu** :

```
/.well-known/carddav -> http://baikal.juxjux.ovh/dav.php
```

C'est exactement le risque identifié dès la conception : CardDAV et CalDAV envoient les identifiants **à chaque synchronisation**. Un client suivant cette redirection part sur du HTTP.

Corrigé en court-circuitant Baïkal directement dans le vhost :

```
location = /.well-known/carddav {
    return 301 https://$host/dav.php/;
}
location = /.well-known/caldav {
    return 301 https://$host/dav.php/;
}
```

À vérifier après tout `reload nginx` : le rechargement n'est pas instantané, un test lancé dans la foulée peut encore montrer l'ancien comportement.

6.4 Roundcube exige un serveur IMAP pour authentifier

Roundcube **n'a pas de base d'utilisateurs propre** : son écran de login valide les identifiants contre le serveur IMAP configuré. Sans IMAP joignable, aucune connexion n'est possible — donc aucun accès aux contacts non plus.

L'idée d'un Roundcube « contacts et agenda seulement, sans IMAP » n'est pas réalisable. La configuration retenue pointe vers `ssl://mail.infomaniak.com:993`, ce qui couvre le compte `julien.bertrand@ik.me` (le domaine `ik.me` est servi par l'infrastructure Infomaniak).

Si un jour le compte passe en double authentification, il faudra générer un **mot de passe d'application** Infomaniak.

6.5 Baïkal doit être installé en SQLite

L'installeur Baïkal propose MySQL ou SQLite. **Choisir SQLite** : la stack ne contient aucun serveur MySQL, cocher « Enable MySQL » mène à une impasse. Base à `/var/www/baikal/Specific/db/db.sqlite`, dans le volume `baikal-data`.

7. InfCloud — l'agenda web

Client CalDAV/CardDAV **entièrement côté navigateur** : aucun backend, aucune base, du HTML/JS servi en statique par nginx. Affiche agenda et contacts.

- Source officielle : `https://www.inf-it.com/InfCloud_0.13.1.zip` (3,9 Mo, sha256 `9fa95edd2dcc2b864a10b503ab9220895ea28d4c5541ab02117de1511d5464d4`)
- Installé dans `/home/debian/baikal/infcloud`, propriétaire `www-data`
- **Servi sur le même domaine que Baïkal** (`/infcloud/`) et non sur un sous-domaine dédié : InfCloud interroge `/dav.php/` en direct depuis le navigateur ; depuis une autre origine il aurait fallu ouvrir du CORS sur les méthodes DAV de Baïkal, ce qui est fragile et l'affaiblirait. Même origine = aucun CORS.
- Dossier `auth/` **supprimé** : module PHP d'authentification par proxy inutilisé ici. nginx ne traite pas le PHP à cet emplacement — servis en statique, ces fichiers `.inc` auraient exposé leur contenu en clair.

Réserve assumée : projet figé depuis 2015, interface datée, embarque jQuery 2.1.4. Acceptable parce qu'il n'a aucun composant serveur : rien à maintenir, rien à patcher. La seule alternative maintenue serait SOGo, au prix d'environ 500 Mo de RAM et d'une base PostgreSQL.

7.1 Les trois pièges d'InfCloud

a) Le bloc regex du vhost avale tous les JS. Le vhost Baïkal contient déjà :

```
location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ { proxy_pass http://127.0.0.1:8088; }
```

En nginx, un bloc regex `~*` a priorité sur un bloc préfixe ordinaire. Tous les fichiers JavaScript d'InfCloud auraient donc été proxifiés vers Baïkal. Comme InfCloud est presque intégralement du JavaScript, la page serait restée blanche — symptôme difficile à diagnostiquer. **Il faut `^~`, qui passe devant les regex :**

```
location ^~ /infcloud/ {
    alias /home/debian/baikal/infcloud/;
    index index.html;
    location ~ /\. { deny all; }
}
```

b) L'endpoint par défaut vise DAViCal. Le `config.js` livré pointe `/caldav.php/` ; Baïkal expose `/dav.php/`.

c) Le href doit viser les *principals*, pas la racine DAV. InfCloud accole le nom d'utilisateur au `href`. Avec `/dav.php/` il demandait `/dav.php/Julien/` → **404**. La documentation du fichier le précise : *« globalNetworkCheckSettings : principal URL WITHOUT the USER/ part »*. Chez Baïkal, la bonne valeur est donc :

```
'/dav.php/principals/',
```

Après toute modification de `config.js`, **rechargement forcé du navigateur (Ctrl+Maj+R)** : le fichier est mis en cache.

7.2 Digest → Basic : le blocage d'authentification

Baïkal était configuré en `dav_auth_type: Digest` et annonçait `WWW-Authenticate: Digest realm="BaikalDAV"`.

InfCloud ne sait faire que du Basic. Ses requêtes partaient donc en 401 en boucle, alors que DAVx5 — qui gère Digest — fonctionnait parfaitement avec le même compte. Le symptôme prêtait à confusion : les identifiants étaient bons.

Diagnostic par les logs nginx, où le champ utilisateur est renseigné mais la réponse reste 401 :

```
83.195.45.93 - Julien "PROPFIND /dav.php/ HTTP/2.0" 401
```

Correctif : `dav_auth_type: Basic` dans `/var/www/baikal/config/baikal.yaml`, puis `docker restart baikal`.

Ce n'est pas un pis-aller. Digest repose sur MD5, est considéré comme obsolète, et impose au serveur de stocker une empreinte exploitable ; **Basic sur TLS est aujourd'hui la recommandation**, et tout est en HTTPS ici, y compris les redirections de découverte (§6.3).

Les mots de passe restent valides : Baïkal calcule la même empreinte `md5(username:auth_realm:password)` dans les deux modes (`Baikal\Core\PDOBasicAuth::validateUserPass`), avec `auth_realm: BaikalDAV` conservé en interne — même si le serveur annonce désormais `realm="sabre/dav"` côté client. Rien à redéfinir, DAVx5 bascule tout seul.

Sauvegarde : `/var/www/baikal/config/baikal.yaml.bak-260803-digest` (dans le container).

8. Sauvegarde

Baïkal a été ajouté à `/opt/backups/vps_backup.sh` le 3 août 2026 (cron quotidien à 2h UTC).

- Dossier kDrive de destination : **1467771** (`SYNC-pour_VPS/backups/baikal`)
- Le bilan de fin de log passe de 5 à **6 services** (le total est calculé dynamiquement, pas codé en dur)
- Sauvegarde de l'ancien script : `/opt/backups/vps_backup.sh.bak-260803`

La fonction archive **deux** volumes, via un `tar` intermédiaire puis 7z :

- `Specific/` — la base `db.sqlite` (contacts et calendriers)
- `config/` — `baikal.yaml` (paramètres et empreinte du mot de passe admin)

Sans le second, la restauration est incomplète. Testée le 3 août : upload HTTP 200, aucun résidu temporaire.

InfCloud n'a pas besoin d'être sauvegardé (fichiers statiques réinstallables), mais son `config.js` est versionné dans `Jux-scripts/Baikal-Contacts/`.

9. Vérifications effectuées le 3 août 2026

Test	Résultat
`GET /dav.php/`	HTTP 401 — authentification bien exigée
`PROPFIND /dav.php/`	HTTP 401 et non 405 — nginx laisse passer les méthodes DAV étendues
`.well-known/carddav` et `caldav`	301 vers HTTPS
Installation Baïkal	Utilisateur `Julien`, carnet et calendrier `default` créés
Login Roundcube	IMAP Infomaniak accepté
Découverte CardDAV Roundcube	Carnet résolu à `.../dav.php/addressbooks/Julien/default/`
Écriture d'un contact	vCard 3.0 écrite par RCMCardDAV v5.1.3, relue dans Baïkal, accents UTF-8 préservés
Synchronisation DAVx5	5 rendez-vous d'août écrits dans Baïkal par `DAVx5/4.5.18-ose`, `synctoken` du calendrier à 6
InfCloud	32 ressources servies en 200 ; `PROPFIND` sur principals, addressbooks et calendars tous en **207**
Dotfiles InfCloud	`.htaccess` bloqué en 403
Sauvegarde Baïkal	Archive uploadée sur kDrive, HTTP 200

Note : dans la table `carddav_addressbooks` de Roundcube, une seconde ligne nommée `%N` avec une URL vide n'est **pas** une erreur — c'est le *template* de rcmcarddav v5, qui porte les réglages appliqués aux carnets découverts.

10. Reste à faire

10.1 vdirsyncer — agrégation Google et Infomaniak

Seule brique du plan initial encore absente. Objectif : faire remonter dans Baïkal les contacts existants sur Google et Infomaniak, pour que Baïkal devienne le point central réel et pas un silo de plus.

- Google Contacts : CardDAV via `https://www.google.com/carddav/v1/principals/EMAIL/` — nécessite un **mot de passe d'application** (compte en 2FA)
- Infomaniak : CardDAV natif, identifiants standards du compte mail
- vdirsyncer : outil CLI léger, une paire de synchronisation par source, exécuté par cron sur le VPS

Décision à prendre avant déploiement :

- **one-way** (Google/Infomaniak → Baïkal, lecture seule) : sans risque, mais les modifications faites dans Baïkal ne remontent pas vers Google
- **two-way** : tout reste aligné, mais un conflit ou une erreur de configuration peut propager des suppressions dans les deux sens

Recommandation : démarrer en one-way le temps de vérifier que l'import est fidèle, puis basculer si besoin.

10.2 Nettoyage

Quatre vhosts morts traînent dans `/etc/nginx/sites-enabled/`, vestiges des tentatives précédentes : `radicale`, `agendav`, `infcloud`, `nextcloud`. Le dernier cible le port 8082, ce qui a contraint le choix du port de Roundcube. **Attention** : le vhost nommé `radicale.juxjux.ovh` sert en réalité Readeck — ne pas le supprimer sans vérifier.

11. Checklist sécurité

☑ Aucun mot de passe `CHANGE_ME_...` en production — la bascule vers SQLite a supprimé les identifiants MariaDB du compose

☑ HTTPS forcé sur les deux vhosts (redirection 80 vers 443)

☑ Certificats Let's Encrypt valides, renouvellement automatique certbot en place

☑ Découverte DAV forcée en HTTPS (§6.3)

☑ Authentification Basic **uniquement sur TLS** (§7.2)

☑ Les deux containers n'écoutent que sur `127.0.0.1` — seul nginx les expose

☑ Module PHP `auth/` d'InfCloud supprimé, dotfiles bloqués

☑ Sauvegarde quotidienne des volumes Baïkal vers kDrive

☐ Mot de passe d'application dédié pour l'accès CardDAV Google — à faire au moment de vdirsyncer

12. Commandes utiles

```
# État de la stack
sudo docker ps --filter name=baikal --filter name=roundcube

# Logs
sudo docker logs roundcube --tail 50
sudo docker logs baikal --tail 50

# Redéployer
cd /home/debian/baikal && sudo docker compose -p baikal up -d

# Inspecter la base Baïkal
sudo docker cp baikal:/var/www/baikal/Specific/db/db.sqlite /tmp/bk.sqlite
sudo sqlite3 /tmp/bk.sqlite "SELECT COUNT(*) FROM cards;"
sudo sqlite3 /tmp/bk.sqlite "SELECT COUNT(*) FROM calendarobjects;"
sudo sqlite3 /tmp/bk.sqlite "SELECT id, username FROM users;"

# Méthode d'authentification annoncée par Baïkal
curl -s -D - -o /dev/null -X PROPFIND -H 'Depth: 0' \
  https://baikal.juxjux.ovh/dav.php/ | grep -i www-authenticate

# Suivre les requêtes DAV des clients (DAVx5, InfCloud, Roundcube)
sudo tail -f /var/log/nginx/access.log | grep dav.php

# Vérifier la découverte DAV
curl -s -o /dev/null -w "%{http_code} -> %{redirect_url}\n" \
  https://baikal.juxjux.ovh/.well-known/carddav
```

13. Choix d'architecture — pourquoi ces outils

Radicale écarté : stockage fichier brut, pas de vraie gestion. La stack Radicale/InfCloud initialement envisagée n'a pas été mise en place — seul InfCloud a finalement été retenu, en client

de Baïkal.

Nextcloud écarté : trop lourd pour l'usage visé (PHP-FPM + base + Redis + écosystème complet), alors que le VPS fait déjà tourner PostGIS, Metabase, BookStack, FreshRSS.

Baïkal retenu : léger, CardDAV et CalDAV natifs, interface d'administration suffisante. Développé par des bénévoles sous l'organisation sabre-io (mainteneurs de SabreDAV, la librairie sous-jacente) — pas de société commerciale derrière, mais communauté active et outil robuste pour un usage personnel sur la durée.

Roundcube retenu comme interface de saisie des contacts, Baïkal n'ayant pas de vraie UI de consultation : léger, traduit en français, projet actif (version 1.7.2 en août 2026). A rejoint la famille Nextcloud en 2023 tout en restant indépendant, sous licence GPL. Réserves constatées à l'usage : il impose une authentification IMAP (§6.4) et ne couvre pas l'agenda (§6.2).

InfCloud retenu pour l'agenda web : sans backend, donc sans coût mémoire ni surface d'attaque serveur, et couvre agenda et contacts. Ancien mais fonctionnel.

SOGo envisagé et écarté : suite complète et maintenue, mais environ 500 Mo de RAM et une base PostgreSQL sur un VPS déjà en pression mémoire, et il aurait fait doublon avec Roundcube.

Thunderbird containerisé (VNC/noVNC) envisagé puis écarté : fonctionnel mais plus lourd et moins adapté au multi-accès qu'un vrai webmail.

Revision #4
Created 2026-08-03 02:23:16 UTC by Julien
Updated 2026-08-03 17:23:58 UTC by Julien