

Dibab

Manuel utilisateur

Classer les papiers administratifs d'un foyer, sur vos propres machines

Version 0.9.0-alpha

2026-11-26

Traduction. La version anglaise fait foi.

`dibab.davalan.fr`

Table des matières

Introduction **3**

Ce que fait Dibab . . . 3
Ce qu'il ne fait pas . . . 3
Un outil de foyer . . . 3
Organisation de ce manuel . . . 4
Conventions . . . 4

Installation **5**

Ce qu'il vous faut . . . 5
Les modèles, par rôle . . . 5
L'arborescence finale . . . 5
Installation depuis le paquet Debian . . . 6
Installation sur Fedora, openSUSE Leap et Arch . . . 7
Installer depuis les sources . . . 7
Lancer les tests (facultatif) . . . 8
Où la configuration est cherchée . . . 8

Configuration **9**

Les clés à écrire . . . 9
`destinations` : le coffre de chaque domaine . . . 9
`destination`, `arborescence`, `executer_sur` . . . 10
`suivi` : état, journal et rapports . . . 10
`ollama` : les modèles . . . 11
`extraction` et `classification` . . . 12
`categories`, `personnes`, `emetteurs_connus`, `pieces_communes` . . . 13
`nommage` : le nom d'un document classé . . . 13
`extensions_cibles` et `scan` . . . 14
`sources_locales` . . . 14
`ssh`, `sources_distantes`, `imap`, `oauth2`, `sources_imap` . . . 14
`mesure` et `inventaire` . . . 15

Premier passage **16**

Simuler : `--dry-run` . . . 16
Vérifier l'acheminement sans les modèles . . . 18
Classer : `--execute` . . . 19
Annuler : `--rollback` . . . 21
Codes de sortie . . . 21

Coffre et chiffrement **22**

Ce qu'est un coffre . . . 22
La vérification du montage . . . 22
Un exemple avec gocryptfs . . . 23
Où vit l'état . . . 23
Place sur le disque . . . 24

Sources par ssh **25**
Ce que la machine distante doit offrir . . . 25
Déclarer la machine . . . 25
Un premier passage : lister, puis récupérer . . . 26
Relancer . . . 27
Quand la connexion échoue . . . 28

Boîtes IMAP et comptes Google **29**
Ce qui est demandé au serveur . . . 29
Déclarer une boîte . . . 29
Le mot de passe . . . 30
Un premier passage : lister, puis déposer . . . 30
Quand la connexion échoue . . . 32
Comptes Google avec OAuth2 . . . 33

Inventaire et rotation **35**
Lire ce qu'un coffre contient déjà : `--inventaire` . . . 35
Garder le journal court : la rotation . . . 36

Diagnostic **37**
Les états . . . 37
Les contrôles, dans l'ordre . . . 38
Lire un échec . . . 39
Une installation toute neuve . . . 40
Dans une autre langue . . . 40

Souveraineté et limites **41**
Ce que l'outil garantit . . . 41
Le respect du foyer . . . 41
Les limites . . . 41
Signaler un problème . . . 42

Introduction

Dibab (« scribe » en breton) collecte, analyse et classe les documents administratifs personnels d'un foyer : factures, courriers, contrats, avis d'organismes. Il déplace chaque document dans une arborescence normalisée, sous un nom qui dit la date, l'émetteur et la catégorie, et il tient un journal de tout ce qu'il fait, de sorte que tout classement peut être annulé. Ce manuel décrit la version 0.9.0-alpha.

Il tourne en mode souverain. Aucune donnée ne quitte votre réseau local : pas de service tiers, pas de modèle distant, pas de nuage. Les modèles de langage sont ceux d'Ollama, qui tourne sur votre propre machine.

Ce que fait Dibab

- Il scanne des répertoires locaux, des machines distantes par `ssh` et des boîtes mail par IMAP. Un fichier distant est copié et vérifié par empreinte, et l'original reste intact. Une boîte mail est lue sans être modifiée : seules les pièces jointes sont récupérées, et aucun message n'est marqué comme lu.
- Il extrait le texte de chaque fichier (la couche texte d'un PDF, sinon la lecture des images des pages par un modèle de vision), met de côté ce qui n'est pas administratif, puis classe le reste par catégorie et par domaine, personnel ou professionnel, avec l'accord de plusieurs modèles.
- Il renomme et déplace chaque document dans le coffre de son domaine, écrit un journal horodaté, et peut annuler toute une session avec `--rollback`.

Ce qu'il ne fait pas

- Il ne déplace pas un fichier dont il n'est pas sûr. Le doute mène à un dossier nommé `À_VÉRIFIER/`, jamais à un classement hasardeux.
- Il ne supprime jamais un document. Il copie, vérifie, puis ne retire l'original qu'une fois la copie établie identique.
- Il ne modifie ni une boîte mail ni une machine distante.
- Il ne tourne pas en tâche de fond. Vous le lancez à la main, et son mode par défaut est la simulation, `--dry-run`.
- Il ne monte, ne chiffre et ne synchronise aucun coffre. C'est votre affaire, et le chapitre 5 explique comment l'outil le vérifie.

Un outil de foyer

Dibab est conçu pour la vie administrative d'un foyer, par le foyer. Une machine, un répertoire ou un compte mail n'entre dans son périmètre qu'avec l'accord, explicite ou implicite, de la personne concernée ; en cas de doute, laissez-le de côté. Le logiciel ne peut pas garantir ce respect : c'est l'administrateur de l'instance qui s'y engage.

Le journal contient des noms de fichiers et des émetteurs, qui révèlent déjà beaucoup (un courrier d'un organisme de santé suffit). Il a donc sa place dans le coffre, pas à côté : placez les clés `suivi.*` dans le coffre, sinon il est écrit en clair sous `~/.local/state/dibab/`.

Organisation de ce manuel

1. Introduction (ce chapitre).
2. Installation : ce qu'il faut installer, où tout se range, et une première vérification.
3. Configuration : le fichier de configuration, clé par clé.
4. Premier passage : simuler, lire le rapport, classer pour de bon, annuler.
5. Coffre et chiffrement : comment l'outil établit qu'un coffre est monté, et où garder l'état.
6. Sources par ssh : les machines distantes.
7. Boîtes IMAP et comptes Google.
8. Inventaire et rotation : lire ce qu'un coffre contient déjà, et garder le journal court.
9. Diagnostic : `--diagnose`, état par état.
10. Souveraineté et limites : ce que l'outil garantit, et ce qu'il ne garantit pas.

Conventions

Les commandes sont montrées comme on les tape, dans un terminal, pour l'utilisateur de l'installation décrite au chapitre 2. La commande est `dibab`. Le texte qui suit une commande est sa sortie réelle, produite en rejouant la commande sur une aire d'essai faite de documents fictifs ; les modèles nommés dans ces sorties (`essai-ocr`, `essai-tri`, `essai-classification`) sont les doublures de cette aire d'essai, et les vôtres sont ceux que vous nommez dans votre configuration. Chaque document, personne, émetteur et société de ce manuel est inventé.

La langue des messages suit votre session : Dibab lit la première variable non vide parmi `LC_ALL`, `LC_MESSAGES` et `LANG`, et parle anglais, français, allemand, espagnol ou chinois simplifié en conséquence (toute autre valeur donne l'anglais). Les noms d'options, les jetons d'état comme `OK` et `ÉCHEC`, les identifiants, les journaux et les fichiers de plan ne changent jamais avec la langue, de sorte qu'un journal écrit dans une langue est rejoué par `--rollback` dans n'importe quelle autre. Les sorties de ce manuel sont en anglais.

Installation

Ce qu'il vous faut

- Un système Linux avec Python 3.10 ou plus récent. Dibab n'utilise que la bibliothèque standard de Python : rien d'autre n'est à installer côté Python.
- `findmnt` (paquet `util-linux`), qui établit qu'un coffre est monté.
- `pdftotext` et `pdftoppm` (paquet `poppler-utils`), qui lisent les fichiers PDF.
- `ssh` (paquet `openssh-client`), seulement si vous scannez des machines distantes, avec une clé qui ouvre une session sans mot de passe.
- Ollama, installé et en marche, avec au moins un modèle pour chaque rôle (ci-dessous). Dibab n'installe jamais Ollama et ne télécharge jamais un modèle.
- Un répertoire de destination par domaine, personnel et professionnel, de préférence sur un volume chiffré (chapitre 5).
- `git`, pour récupérer les sources.

Sur Debian ou Ubuntu, les paquets s'installent en une commande :

```
sudo apt install python3 git util-linux poppler-utils openssh-client
```

Les modèles, par rôle

Trois rôles emploient chacun un modèle. Le rôle `ocr` lit les pages numérisées, `tri_rapide` (« tri rapide ») met de côté ce qui n'est pas administratif, et `classification` classe. Les modèles du rôle `classification` votent : un document n'est classé que si leur accord atteint `ollama.seuil_confiance` (chapitre 3). Deux rôles facultatifs, `tri_decision` et `objet`, confient le tri à un modèle `decision` en un passage et l'objet à un petit modèle ; sans eux, `tri_rapide` trie (chapitre 3).

```
ollama pull glm-ocr
ollama pull mistral-nemo:12b-instruct-2407-q6_K
ollama pull gemma4
ollama pull gemma3:4b
ollama pull qwen2.5
ollama pull granite4.2:8b-Q5_K_M
```

Ce sont les modèles de l'instance de référence ; n'importe quels autres conviennent, pourvu que vous les nommiez dans `ollama.roles`. Listez ce qui est installé avec `ollama list`.

L'arborescence finale

~ est votre répertoire personnel. Adaptez les chemins à votre installation.

```
~/applications/dibab/              le dépôt cloné
├── dibab.py                       le point d'entrée
├── modules/                            un fichier par responsabilité
│   └── langues/                        les dictionnaires de messages : en, fr, de, es, zh
├── docs/                               la page de manuel et les README traduits
├── tests/                              les tests
```

```
└── config.exemple.json                la configuration d'exemple
~/.local/bin/dibab                  un lien vers dibab.py : la commande
~/.config/dibab/config.json         votre configuration, jamais versionnée
~/.local/state/dibab/               l'état, le journal et les rapports de session (emplacement
 ↪  par défaut)
~/Classement/Personnel/                 le coffre du domaine personnel
~/Classement/Professionnel/             le coffre du domaine professionnel
```

Installation depuis le paquet Debian

Sur Debian 12 et 13 et sur Ubuntu 22.04 et 24.04, un seul fichier installe le programme et ses dépendances (`python3`, `poppler-utils`, `util-linux`, `openssh-client`). Il ne contient ni Ollama, ni modèle, ni configuration. Téléchargez `dibab_0.9.0-alpha_all.deb` depuis la page de téléchargement de dibab.davalan.fr, puis, depuis le dossier qui le contient :

```
sudo apt install ./dibab_0.9.0-alpha_all.deb
```

Écrivez la configuration, dans la langue de votre session, puis suivez les étapes 4 et 5 de l'installation depuis les sources ci-dessous (le coffre de chaque domaine, les clés de la configuration) :

```
dibab --initialiser
```

`--initialiser` écrit la configuration dans la langue de votre session : le dossier de vérification, son préfixe et le dossier des doublons prennent les noms de cette langue. Il crée le fichier en `600` dans un dossier en `700` et ne réécrit jamais un fichier qui existe ; une session ultérieure dans une autre langue ne les renomme pas.

Vérifiez l'installation, sans rien écrire :

```
dibab --version
dibab --diagnose
```

Le paquet place la commande dans `/usr/bin/dibab`, le programme dans `/usr/lib/dibab/`, et les README, la configuration d'exemple et la page de manuel (`man dibab`, en cinq langues) sous `/usr/share/`. Le retrait garde vos données : `sudo apt remove dibab` et `sudo apt purge dibab` suppriment les fichiers du paquet et ne touchent jamais à votre configuration, à votre état, à votre journal ni à vos documents ; la purge nomme ce qu'elle laisse.

Niveau : installation et diagnostic validés en conteneur sur Debian 12 et 13 et sur Ubuntu 22.04 et 24.04 (installation, `--version`, `--diagnose`, le refus d'écrire sans configuration et celui d'écrire dans un coffre non monté, un classement simulé, retrait et purge). Un conteneur ne montre ni le montage d'un coffre chiffré, ni un passage avec votre propre Ollama et vos propres documents.

Installation sur Fedora, openSUSE Leap et Arch

Sur Fedora 44, openSUSE Leap 16.0 et Arch Linux, un seul fichier installe aussi le programme et ses dépendances. Téléchargez le fichier de votre système depuis la page de téléchargement de dibab.davalan.fr, puis, depuis le dossier qui le contient, lancez la ligne de votre système :

```
sudo dnf install ./dibab-*.fc44.noarch.rpm
sudo zypper install --allow-unsigned-rpm ./dibab-*.leap160.noarch.rpm
sudo pacman -U ./dibab-*-any.pkg.tar.zst
```

Les paquets ne sont pas signés. La suite est la même qu'avec le paquet Debian : `dibab --initialiser`, puis `dibab --version` et `dibab --diagnose`. Le retrait du paquet (`sudo dnf remove dibab`, `sudo zypper remove dibab`, `sudo pacman -R dibab`) ne touche jamais à votre configuration, à votre état, à votre journal ni à vos documents.

Niveau : installation et diagnostic validés en conteneur sur Fedora 44, openSUSE Leap 16.0 et Arch (installation, `--version`, `--diagnose`, les deux refus d'écrire, un classement simulé, réinstallation par-dessus une installation cassée, retrait). Un conteneur ne montre ni le montage d'un coffre chiffré, ni un passage avec votre propre Ollama et vos propres documents.

Installer depuis les sources

1. Récupérez les sources :

```
mkdir -p ~/applications
git clone https://github.com/RonanDavalan/dibab.git ~/applications/dibab
```

2. Rendez la commande disponible. Le point d'entrée est exécutable ; un lien dans `~/.local/bin` lui donne le nom `dibab` (sur la plupart des distributions, ce répertoire est déjà dans le `PATH` d'un shell de connexion) :

```
mkdir -p ~/.local/bin
ln -s ~/applications/dibab/dibab.py ~/.local/bin/dibab
```

3. Écrivez la configuration dans la langue de votre session (le chapitre 3 explique chaque clé) :

```
dibab --initialiser
```

`--initialiser` écrit la configuration dans la langue de votre session : le dossier de vérification, son préfixe et le dossier des doublons prennent les noms de cette langue. Il crée le fichier en `600` dans un dossier en `700` et ne réécrit jamais un fichier qui existe ; une session ultérieure dans une autre langue ne les renomme pas.

4. Créez le coffre de chaque domaine et lisez le type de son système de fichiers (le chapitre 5 explique pourquoi) :

```
mkdir -p ~/Classement/Personnel ~/Classement/Professionnel
findmnt -n -T ~/Classement/Personnel -o FSTYPE
```

```
ext4
```

La commande affiche le type du volume qui contient le répertoire, par exemple `ext4`. Écrivez ce type dans `destinations.<domain>.systemes_fichiers_acceptes` pour un répertoire ordinaire, ou ouvrez d'abord votre volume chiffré et écrivez le type que `findmnt` rapporte pour lui.

5. Éditez `~/.config/dibab/config.json`, au moins les clés du chapitre 3 marquées comme obligatoires, puis vérifiez l'installation sans rien écrire :

```
dibab --version
```

```
dibab 0.9.0-alpha
```

La commande affiche une ligne et s'arrête. Elle ne lit aucune configuration, ne touche à aucun fichier et ne parle à aucun service.

Lancer les tests (facultatif)

Les sources portent leurs propres tests, qui demandent `pytest` dans un environnement virtuel. Ils ne sont pas nécessaires pour utiliser Dibab ; lancez-les après une mise à jour, ou si vous doutez de l'installation :

```
cd ~/applications/dibab && \
  python3 -m venv venv && \
  venv/bin/pip install -r requirements-dev.txt && \
  venv/bin/python -m pytest -q tests
```

La dernière ligne rapporte les tests réussis et les tests ignorés ; les ignorés sont les essais contre un Ollama en marche, que la suite de tests n'exige pas.

Où la configuration est cherchée

Dibab lit le premier fichier trouvé parmi, dans cet ordre : l'option `--config <path>`, la variable d'environnement `DIBAB_CONFIG`, `$XDG_CONFIG_HOME/dibab/config.json` (`~/.config/dibab/config.json` quand la variable est vide), puis `config.json` à côté de `dibab.py`. Il n'y a aucune fusion : le premier fichier trouvé est le seul lu. Une option ou une variable qui désigne un fichier absent arrête le programme avec le code de sortie 1. Il ne se rabat jamais sur un autre fichier, car il fonctionnerait alors sur une configuration que vous n'avez pas désignée, c'est-à-dire sur un autre coffre.

Configuration

Rien n'est écrit en dur dans Dibab. Chaque chemin, nom de modèle, seuil, délai et règle de nommage vit dans un seul fichier JSON, `config.json`. Changer le comportement de l'outil, c'est changer ce fichier, pas le code. Les sources portent un exemple complet et réaliste, `config.exemple.json` ; ce chapitre l'explique bloc par bloc. Le chapitre 2 dit où le programme cherche le fichier.

Les chemins écrits avec ~ ou `$HOME` sont développés. L'exemple ne contient aucun chemin absolu. Aucun secret n'est jamais écrit dans ce fichier : un mot de passe de messagerie va dans une variable d'environnement, un jeton Google dans un fichier séparé en mode `600` (chapitre 7).

Une clé absente n'est pas remplacée en silence par une supposition. Le programme s'arrête avec le code de sortie 1 et nomme les clés manquantes ; `dibab --diagnose` (chapitre 9) est donc la première commande à lancer après une modification.

Les clés à écrire

Ouvrez le fichier que `dibab --initialiser` a écrit, puis changez au moins celles-ci :

Clé	Ce qu'il faut écrire
`destinations.personnel.racine`, `destinations.professionnel.racine`	la racine du coffre de chaque domaine
`destinations.*.systemes_fichiers_acceptes`	le type de système de fichiers qui établit que le coffre est monté (chapitre 5)
`personnes`	les prénoms des membres du foyer ; ils nomment les dossiers du domaine personnel
`sources_locales[].chemin`	les répertoires à scanner
`ollama.roles`	pour chaque rôle, les modèles préférés, dans l'ordre (`config.exemple.json` laisse les trois listes vides)
`suivi.fichier_date`, `suivi.journal`, `suivi.sessions`	facultatives, mais recommandées à l'intérieur du coffre (chapitre 5)

Supprimez les entrées d'exemple de `sources_distantes` et de `sources_imap` que vous n'utilisez pas : une source déclarée est une source lue.

`destinations` : le coffre de chaque domaine

Une entrée par domaine, `personnel` et `professionnel`.

Clé	Signification
`racine`	la racine du coffre de ce domaine
`systemes_fichiers_acceptes`	les types de système de fichiers qui établissent que le coffre est monté ; l'exemple attend `fuse.gocryptfs`

Clé	Signification
sous_repertoire_classement	le dossier, sous la racine, où les documents de ce domaine sont classés (Administratif pour le domaine personnel, Classement pour le professionnel)
sous_repertoire_a_verifier	le dossier des documents dont Dibab n'est pas sûr (À_VÉRIFIER/)

Le nom du dossier à vérifier et son préfixe sont écrits à l'initialisation dans la langue de la session : le dossier est réglable par sous_repertoire_a_verifier et le préfixe par nommage.prefixe_a_verifier. Le dossier à la racine d'un coffre qui reçoit les doublons est réglable par destination.repertoire_doublon.

destination, arborescence, executer_sur

Clé	Signification
destination.sous_repertoire_logs	le dossier des journaux métier à l'intérieur d'un coffre (logs) ; les plans d'inventaire y sont écrits
destination.vide_poches_prefixe	le préfixe des « vide-poches » (vide-poches-), où les fichiers récupérés attendent
destination.repertoire_doublon	le dossier, à la racine d'un coffre, qui reçoit un document dont l'empreinte est déjà classée (doublon/)
arborescence.personnel	le sous-chemin d'un document personnel sous son dossier de classement : {personne}/{annee} donne Alice/2026
arborescence.professionnel	la même chose pour un document professionnel : {annee}/{categorie} donne 2026/Factures
arborescence.personne_inconnue	le dossier employé quand la personne n'est pas dans personnes (Commun)
executer_sur	les noms de machine (hostname) où --execute est autorisé ; une liste vide signifie n'importe quelle machine, et --execute ailleurs s'arrête avec le code de sortie 4

suivi : état, journal et rapports

Clé	Signification
fichier_date	le fichier dernier_scan.json, qui retient la date du dernier scan de chaque source
journal	le journal des actions journal.jsonl, une ligne JSON par action, relu par --rollback ; il porte des chemins, jamais le texte d'un document
sessions	le dossier des rapports de session
logs_systeme	réservé à un journal technique ; rien ne l'écrit aujourd'hui

Clé	Signification
`securite_jours`	de combien de jours avant le dernier scan un nouveau scan repart (`1`), afin qu'un fichier ajouté en parallèle ne soit pas manqué
`premier_scan_jours`	jusqu'où remonte le tout premier scan d'une source (`90` jours)
`rotation.journal_taille_max_mo`	la taille au-delà de laquelle le journal est tourné (`20` Mo)
`rotation.journal_archives_max`	combien de journaux tournés sont conservés (`5`)
`rotation.sessions_max`	combien de fichiers de session sont conservés (`100`)

Les quatre clés de chemin (`fichier_date`, `journal`, `sessions`, `logs_systeme`) sont facultatives. Sans elles, les fichiers vont sous `$XDG_STATE_HOME/dibab/`, soit `~/.local/state/dibab/`, sous les noms `dernier_scan.json`, `journal.jsonl`, `sessions/` et `dibab.log`, avec le mode `600` pour les fichiers et `700` pour les dossiers. Ces emplacements par défaut sont en clair sur le disque : le chapitre 5 explique comment les déplacer dans un coffre. Trois autres fichiers suivent `fichier_date`, où qu'il soit : `rapatries.json`, `imap_deposes.json` et `dibab.verrou`.

`ollama` : les modèles

Clé	Signification
`url`	l'adresse d'Ollama (`http://localhost:11434`) ; chaque appel est une simple requête HTTP vers elle
`roles.ocr`	les préférences ordonnées du modèle qui lit les pages numérisées
`roles.tri_rapide`	les préférences ordonnées du modèle qui met de côté ce qui n'est pas administratif
`roles.classification`	les modèles qui classent ; ils tournent tous sur chaque document et votent
`roles.tri_decision`	facultatif : les préférences ordonnées d'un modèle `decision` (`ollama show` en donne la capacité), qui trie en un passage, avec des seuils, à la place de `tri_rapide`
`roles.objet`	facultatif : le petit modèle de génération qui donne l'objet d'un document quand le modèle decision le trie
`seuil_confiance`	entre 0 et 1 ; un domaine n'est retenu que si la part des modèles de classification d'accord l'atteint (`0.4` dans l'exemple)
`timeout_secondes`	le délai de chaque requête vers Ollama (`120`)
`keep_alive_entre_pages`	combien de temps le modèle d'OCR reste chargé entre deux pages d'un document (`2m`)

Au démarrage, Dibab demande à Ollama les modèles installés et retient, pour `ocr` et `tri_rapide`, la première préférence qui est installée ; un nom sans étiquette vaut `:latest`. Pour `classification`, il retient toutes

les préférences installées, et elles votent. Un rôle sans aucun modèle installé arrête le programme avec le code de sortie 3, liste les modèles présents et nomme la ligne à corriger, avant que rien ne soit écrit. Les deux rôles facultatifs ne l'arrêtent jamais : quand `tri_decision` est déclaré mais inutilisable (aucun modèle installé, pas de capacité `decision`, pas de seuils, pas de modèle pour `objet`, ou un Ollama qui ne connaît pas `/v1/systemone`), le tri passe par `tri_rapide` et l'en-tête de la session dit pourquoi.

`extraction` et `classification`

Clé	Signification
`extraction.seuil_caracteres_alphanumeriques`	en dessous de ce nombre de lettres et de chiffres, un PDF est lu par le modèle d'OCR plutôt que par sa couche texte (`200`)
`extraction.pages_ocr`	combien de pages l'OCR lit (`3`)
`extraction.resolution_ocr`	la résolution des pages rendues, en ppp (`200`)
`extraction.taille_max_texte`	le texte extrait est tronqué à ce nombre de caractères (`20000`)
`extraction.repertoire_temporaire`	où les pages sont rendues ; gardez-le en mémoire (`/dev/shm`)
`extraction.timeout_outils_secondes`	le délai de `pdftotext` et de `pdftoppm` (`60`)
`extraction.consigne_ocr`	la consigne envoyée au modèle d'OCR avec chaque image
`classification.temperature`, `classification.num_ctx`	les options envoyées avec chaque requête (`0` et `8192`) ; un contexte trop court coupe le début de la consigne
`classification.caracteres_max_prompt`	quelle part du texte est envoyée aux modèles de classification (`12000`)
`classification.caracteres_max_tri`	quelle part du texte est envoyée au modèle de tri (`3000`) ; le début d'un document dit de quoi il s'agit
`classification.keep_alive_entre_documents`	combien de temps un modèle reste chargé entre deux documents (`2m`)
`classification.longueur_max_champ`, `longueur_max_citation`	la longueur maximale de l'émetteur, du destinataire et de l'objet retenus, et d'un extrait cité (`80` et `120`)
`classification.marqueurs_mentions`	les mots dont la présence est un signal professionnel faible (`TVA`, `SIRET`, …)
`classification.seuils_tri_decision`	facultatif : pour chaque modèle decision, `non_administratif` et `administratif` (entre 0 et 1) et `caracteres_min` ; la probabilité de non administratif rendue par le modèle vaut non administratif à partir du premier, administratif sous le second, incertain entre les deux, et un texte de moins de `caracteres_min` lettres et chiffres est incertain. Les seuils valent pour un modèle et un `caracteres_max_tri` : ils se remesurent quand l'un des deux change

Le texte d'un document est traité comme une donnée, jamais comme une instruction : il est délimité par

un marqueur aléatoire, le modèle répond par un JSON fermé que Dibab valide, et toute réponse invalide compte comme « incertain ». Un modèle ne peut jamais déclencher une action.

categories, **personnes**, **emetteurs_connus**, **pieces_communes**

Clé	Signification
`categories`	la liste fermée des catégories (`Factures`, `Aides`, `Impôts`, …) ; un modèle ne peut répondre que par l'une d'elles, ou « aucune »
`personnes`	les membres du foyer ; un modèle ne peut répondre que par l'un d'eux, ou « aucun »
`emetteurs_connus`	les émetteurs connus par catégorie, chacun `{"nom": ..., "domaine": "personnel" | "professionnel" | "mixte"}` ; un émetteur connu est un signal de domaine, et une liste vide ne donne aucun signal
`pieces_communes`	les catégories copiées dans le coffre de chaque domaine, sous le même nom et avec la même empreinte (`Identité`, `Domicile`, `RIB`), seulement si chaque modèle de classification rend cette même catégorie

Les catégories et les émetteurs connus de l'exemple sont génériques ; adaptez-les à votre pays.

nommage : le nom d'un document classé

Clé	Signification
`format`	le modèle d'un document classé, avec `{date}`, `{emetteur}`, `{categorie}`, `{personne}`, `{objet}` (l'objet rendu par le tri), `{ext}` (`{date}_{emetteur}_{categorie}{ext}`)
`separateur`, `espaces_vers`	le séparateur et le remplacement des espaces (_ et -)
`date_format`	le format de la date (`%Y-%m-%d`)
`date_inconnue_prefixe`	le préfixe de la date de modification employée quand aucune date n'est trouvée dans le document (~)
`emetteur_inconnu`	le nom employé quand l'émetteur est inconnu (`Inconnu`)
`prefixe_a_verifier`	le préfixe d'un document placé dans le dossier à vérifier (`[À_VÉRIFIER]`)
`format_a_verifier`	le modèle qui suit le préfixe, avec les mêmes variables et `{origine}`, le nom d'origine ; une partie vide ou inconnue est retirée avec son séparateur, et le nom entier tient en 180 caractères (`{date}_{emetteur}_{objet}_{origine}{ext}`) ; sans cette clé, le document garde son nom d'origine
`suffixe_collision`	le suffixe d'un nom déjà pris, à partir de 2 (`_{n}`)

Avec la configuration que `dibab --initialiser` écrit, un avis d'un organisme public daté du 15 mars 2026 devient `2026-03-15_Organisme-public_Aides.pdf`, et un courrier sans en-tête placé dans le dossier à vérifier devient `[À_VÉRIFIER]courrier-de-suivi-de-dossier_courrier-sans-en-tete.pdf`.

`extensions_cibles` et `scan`

Clé	Signification
`extensions_cibles`	les extensions retenues (`.pdf`, `.png`, `.jpg`, `.jpeg`), comparées sans tenir compte de la casse
`scan.recursif_par_defaut`	scanner les sous-dossiers (`true`) ; chaque source peut le modifier
`scan.exclusions_automatiques`	les sous-dossiers jamais scannés, pour toutes les sources (`rebut`, et le dossier nommé par `destination.repertoire_doublon`)
`scan.suivre_symlinks`	suivre les liens symboliques (`false`, afin que le scan ne puisse pas sortir de son périmètre)
`scan.ignorer_dotfiles`	ignorer les fichiers et dossiers cachés (`true`)

`sources_locales`

Chaque entrée est un répertoire à scanner.

Clé	Signification
`id`	un nom unique ; c'est la clé de la source dans `dernier_scan.json`
`chemin`	le répertoire
`date_reference`	`modification` (recommandé) ou `creation` ; la date employée pour garder les fichiers récents
`recursif`	remplace `scan.recursif_par_defaut` pour cette source
`exclusions`	les noms de sous-dossiers à ignorer pour cette source seulement

Le premier scan d'une source remonte de `suivi.premier_scan_jours` jours ; les suivants repartent de la date du dernier scan complet, moins `suivi.securite_jours`.

`ssh`, `sources_distantes`, `imap`, `oauth2`, `sources_imap`

Ces blocs sont expliqués là où ils servent : `ssh` et `sources_distantes` au chapitre 6, `imap`, `oauth2` et `sources_imap` au chapitre 7.

`mesure` et `inventaire`

Clé	Signification
`inventaire.exclusions`	les noms de dossiers que `--inventaire` ne lit pas (chapitre 8)
`mesure.graine`, `documents_par_coffre`, `exclusions`, `taux_erreur_max`	lues seulement par l'outil de mesure des sources (`_outils/mesurer_domaines.py`), qui échantillonne des documents classés et confronte les modèles au coffre ; les valeurs de l'exemple sont un point de départ

Premier passage

Un premier passage a quatre temps : poser les documents dans un répertoire source, simuler avec `--dry-run`, lire ce que la simulation propose, puis classer pour de bon avec `--execute`. Tout classement peut être annulé avec `--rollback`. Commencez toujours par la simulation : elle ne déplace rien, n'écrit rien dans aucun coffre, et montre exactement où irait chaque document.

Ce chapitre emploie le répertoire source `~/Classement/Entrant` de la configuration de l'aire d'essai, qui contient quatre documents fictifs : un avis d'allocations familiales, une facture d'hébergement, une recette de gâteau et une lettre sans aucun en-tête.

```
ls ~/Classement/Entrant

courrier-sans-en-tete.pdf
facture-hebergement.pdf
notification-organisme.pdf
recette-gateau.pdf
```

Simuler : `--dry-run`

`--dry-run` est le mode par défaut : `dibab` seul fait la même chose. Il a besoin d'Ollama, car toute la classification s'exécute, mais il ne déplace aucun fichier et n'écrit ni `dernier_scan.json` ni aucun document classé.

```
dibab --dry-run

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : dry-run
  Configuration      : ~/.config/dibab/config.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

Source: entrant (~/Classement/Entrant)
  - notification-organisme.pdf                        (928 B, mtime 2026-11-23)
  - facture-hebergement.pdf                           (922 B, mtime 2026-11-23)
  - recette-gateau.pdf                                (895 B, mtime 2026-11-23)
  - courrier-sans-en-tete.pdf                         (901 B, mtime 2026-11-23)
  Source total: 4 file(s), 3.6 KB

============================================================
Scan total: 4 candidate file(s), 3.6 KB
============================================================
Text extraction (the text is neither shown nor written)
  - notification-organisme.pdf: text layer, 258 character(s), 1 page(s), 0.02 s
  - facture-hebergement.pdf: text layer, 246 character(s), 1 page(s), 0.02 s
```

```
  - recette-gateau.pdf: text layer, 226 character(s), 1 page(s), 0.02 s
  - courrier-sans-en-tete.pdf: text layer, 236 character(s), 1 page(s), 0.02 s

Classification (dry-run, nothing is moved; the text is neither shown nor written)
model 1/1, document 1/3
model 1/1, document 2/3
model 1/1, document 3/3
  - notification-organisme.pdf: triage administrative, domain personnel, agreement 1/1
      subject: notification de paiement d'allocation
      -> vault personnel, classement: ~/Classement/Personnel/Administratif/Alice/2026/2026-03-1
       ↪  5_Organisme-public_Aides.pdf
  - facture-hebergement.pdf: triage administrative, domain professionnel, agreement 1/1
      subject: facture d'hébergement
      -> vault professionnel, classement: ~/Classement/Professionnel/Classement/2026/Factures/2
       ↪  026-03-15_Hebergeur-Exemple_Factures.pdf
  - recette-gateau.pdf: triage not administrative, domain not applicable
      subject: recette de gâteau
      -> stays in place (not administrative)
  - courrier-sans-en-tete.pdf: triage administrative, domain incertain, agreement 1/1
      subject: courrier de suivi de dossier
      reason: no recipient quoted and no known issuer
      -> vault personnel, a_verifier: ~/Classement/Personnel/À_VÉRIFIER/[À_VÉRIFIER]courrier-de
       ↪  -suivi-de-dossier_courrier-sans-en-tete.pdf (not moved)
```

Les lignes de progression (`model 1/1, document 1/3`) vont sur la sortie d'erreur, une par modèle et par document. Le premier passage est lent, puisque chaque modèle de classification lit chaque document, et ces lignes vous disent que le programme n'est pas figé.

Lire le rapport

Le rapport a quatre parties, dans cet ordre.

1. L'en-tête : la version et l'heure de la session, le mode, le fichier de configuration lu, chaque coffre avec la vérification de son montage, le chemin du journal de session, le modèle retenu pour chaque rôle et la règle de consensus.
2. Le scan : pour chaque source, les fichiers candidats avec leur taille et leur date, puis les totaux. Un fichier est candidat quand son extension figure dans `extensions_cibles` et que sa date est assez récente.
3. L'extraction du texte : pour chaque candidat, comment son texte a été obtenu (`text layer`, ou `OCR` avec le modèle qui l'a lu), combien de caractères et de pages, et combien de temps cela a pris. Le texte lui-même n'est ni montré ni écrit.
4. La classification : un bloc par document. La première ligne donne le tri (administratif, non administratif ou incertain), le domaine que Dibab retient et l'accord, qui compte les modèles de classification qui rendent ce domaine parmi ceux interrogés (`agreement 2/3` : deux modèles sur trois). La ligne `subject:` dit en quelques mots ce qu'est le document, tel que le modèle de tri l'a lu ; ce n'est jamais un extrait du texte. Pour un document envoyé au dossier à vérifier, la ligne `reason:` dit pourquoi, d'après une liste fermée. La dernière ligne, après `->`, donne la destination.

En exécution, une dernière ligne nomme le rapport de session.

Une destination se lit `-> vault <domain>, <zone>: <path>`. La zone dit ce qui va se passer :

Zone ou texte	Signification
`classement`	le document est classé dans la zone de classement de ce coffre, sous son nom normalisé
`a_verifier`	Dibab n'est pas sûr : le document va dans le dossier à vérifier, `À_VÉRIFIER/`, sous le préfixe `[À_VÉRIFIER]` suivi de sa date, de son émetteur et de son objet quand le document les donne, et de son nom d'origine ; la simulation ajoute `(not moved)` car rien n'est classé dans la zone de classement
`stays in place (not administrative)`	le tri a mis le document de côté ; il n'est ni déplacé ni renommé, et il est listé de nouveau à chaque passage
`(duplicate: would go to doublon/)`	l'empreinte du document est déjà classée ; il va dans le dossier `doublon/` au lieu d'être classé deux fois

Le domaine `incertain` mène toujours au dossier à vérifier, de même qu'un document sur lequel les modèles ne s'accordent pas. Une pièce commune (une pièce d'identité, un justificatif de domicile, un relevé d'identité bancaire) n'est proposée dans le coffre de chaque domaine que si chaque modèle de classification rend la même catégorie ; sinon elle va dans le dossier à vérifier. Dans le doute, Dibab préfère une décision humaine à un classement hasardeux.

Vérifier l'acheminement sans les modèles

Pour voir quelle destination mène un verdict sans attendre aucun modèle, donnez à `--dry-run` un fichier JSON qui associe un nom de fichier à un verdict. La table d'acheminement lui est appliquée et la destination s'affiche. L'option n'existe qu'avec `--dry-run` : le programme la refuse, avec le code de sortie 2, avec `--execute`.

```
cat > verdicts.json <<'EOF'
{
  "notification-organisme.pdf": {"administratif": true, "domaine": "personnel", "categorie": "A
   ↪  ides", "personne": "Alice", "date": "2026-03-15", "emetteur": "Organisme public"},
  "recette-gateau.pdf": {"administratif": false, "categorie": "Inconnue"}
}
EOF
dibab --dry-run --verdicts-simules verdicts.json | sed -n '/Simulated routing/,$p'

model 1/1, document 1/3
model 1/1, document 2/3
model 1/1, document 3/3
Simulated routing (verdicts supplied, nothing is moved)
  - notification-organisme.pdf: domain personnel
      -> vault personnel, classement: ~/Classement/Personnel/Administratif/Alice/2026/2026-03-1
       ↪  5_Organisme-public_Aides.pdf
  - facture-hebergement.pdf: no simulated verdict
```

```
  - recette-gateau.pdf: not administrative, stays in place
  - courrier-sans-en-tete.pdf: no simulated verdict
```

Classer : `--execute`

Quand la simulation vous convient, lancez la même commande avec `--execute`. Chaque document est copié vers sa destination, la copie est vérifiée par son empreinte SHA256, et seulement alors l'original est retiré de la source. Rien n'est retiré avant que la copie soit établie identique.

```
dibab --execute

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : execute
  Configuration      : ~/.config/dibab/config.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

Source: entrant (~/Classement/Entrant)
  - notification-organisme.pdf                        (928 B, mtime 2026-11-23)
  - facture-hebergement.pdf                           (922 B, mtime 2026-11-23)
  - recette-gateau.pdf                                (895 B, mtime 2026-11-23)
  - courrier-sans-en-tete.pdf                         (901 B, mtime 2026-11-23)
  Source total: 4 file(s), 3.6 KB

============================================================
Scan total: 4 candidate file(s), 3.6 KB
============================================================
Text extraction (the text is neither shown nor written)
  - notification-organisme.pdf: text layer, 258 character(s), 1 page(s), 0.02 s
  - facture-hebergement.pdf: text layer, 246 character(s), 1 page(s), 0.02 s
  - recette-gateau.pdf: text layer, 226 character(s), 1 page(s), 0.02 s
  - courrier-sans-en-tete.pdf: text layer, 236 character(s), 1 page(s), 0.02 s

Filing (execution: each document is copied, checked by SHA256, then removed from its source)
model 1/1, document 1/3
model 1/1, document 2/3
model 1/1, document 3/3
  - notification-organisme.pdf: classe
      -> ~/Classement/Personnel/Administratif/Alice/2026/2026-03-15_Organisme-public_Aides.pdf
  - facture-hebergement.pdf: classe
      -> ~/Classement/Professionnel/Classement/2026/Factures/2026-03-15_Hebergeur-Exemple_Factu⌋
       ↪ res.pdf
  - recette-gateau.pdf: not administrative, stays in place
  - courrier-sans-en-tete.pdf: a_verifier
      -> ~/Classement/Personnel/À_VÉRIFIER/[À_VÉRIFIER]courrier-de-suivi-de-dossier_courrier-sa⌋
       ↪ ns-en-tete.pdf
```

```
Session report: ~/.local/state/dibab/sessions/rapport_2026-11-23T12:09:03.json
```

En exécution, chaque ligne se termine par l'action accomplie : `classe` (classé dans la zone de classement), `a_verifier` (placé dans le dossier à vérifier) ou `doublon` (placé dans le dossier `doublon/`). Ces mots sont les mêmes dans toutes les langues, comme les actions écrites dans le journal.

Les documents sont maintenant là où le rapport l'annonçait :

```
find ~/Classement -type f | sort
```

```
~/Classement/Entrant/recette-gateau.pdf
~/Classement/Personnel/Administratif/Alice/2026/2026-03-15_Organisme-public_Aides.pdf
~/Classement/Personnel/À_VÉRIFIER/[À_VÉRIFIER]courrier-de-suivi-de-dossier_courrier-sans-en-tet
↪ e.pdf
~/Classement/Professionnel/Classement/2026/Factures/2026-03-15_Hebergeur-Exemple_Factures.pdf
```

Trois fichiers ont quitté `Entrant`. La recette est restée : elle n'est pas administrative, et Dibab ne touche pas à ce qu'on ne lui a pas demandé de classer.

Ce qu'écrit `--execute`

Chaque déplacement est écrit dans le journal des actions, `suivi.journal`, une ligne JSON par action :

```
head -n 1 ~/.local/state/dibab/journal.jsonl
```

```
{"schema_version": 1, "session": "2026-11-23T12:09:03", "timestamp": "2026-11-23T12:09:03",
↪ "evenement": "action", "id_action": "<identifiant>", "action": "classe", "domaine": "person
↪ nel", "origine": "~/Classement/Entrant/notification-organisme.pdf", "empreinte_avant":
↪ "6850987e722babfbca9f474ce8971afc532762ac07cd767308c8294fc1383d28", "destinations":
↪ [{"chemin": "~/Classement/Personnel/Administratif/Alice/2026/2026-03-15_Organisme-public_Ai
↪ des.pdf", "empreinte_apres":
↪ "6850987e722babfbca9f474ce8971afc532762ac07cd767308c8294fc1383d28"}]}
```

La ligne contient l'action et son identifiant, le domaine, les chemins et les empreintes avant et après, et aucun texte du document. Chaque session laisse aussi deux fichiers dans le dossier `suivi.sessions` : un journal de session (`session_<time>.jsonl`, les mêmes événements que l'écran) et un rapport (`rapport_<time>.json`) avec le compte de chaque action, les échecs et la durée, sans aucun nom de document. La date du scan de chaque source n'avance dans `dernier_scan.json` que pour les sources lues sans échec. Le chapitre 8 explique comment ces fichiers restent courts.

Une seule exécution tourne à la fois. Un second `--execute` ou `--rollback` qui trouve le verrou `dibab.verrou` pris s'arrête avec le code de sortie 5 et ne touche à rien.

Annuler : `--rollback`

`--rollback` annule la dernière session de classement. `--rollback --session <time>` annule une session donnée, dont l'heure est celle imprimée dans son en-tête et employée dans les noms de ses fichiers.

```
dibab --rollback

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : rollback
  Configuration      : ~/.config/dibab/config.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl

Rollback of session 2026-11-23T12:09:03: 3 undone, 0 failed
```

Chaque document retourne à sa place d'origine, la copie vérifiée par empreinte, et sa destination est retirée. Une ligne `annulation` est ajoutée au journal, qui n'est jamais réécrit. Une pièce commune est retirée du coffre de chaque domaine en même temps. Une action déjà annulée n'est pas rejouée deux fois.

```
find ~/Classement/Entrant -type f | sort

~/Classement/Entrant/courrier-sans-en-tete.pdf
~/Classement/Entrant/facture-hebergement.pdf
~/Classement/Entrant/notification-organisme.pdf
~/Classement/Entrant/recette-gateau.pdf
```

Quand une action ne peut pas être annulée, rien n'est retiré pour elle et la raison s'affiche, avec le code de sortie 1. La raison est `destination modified since the filing` ou `destination not found` quand le document a été modifié ou déplacé à la main (remettez-le ou laissez-le), ou `source occupied by another file` quand un autre fichier a pris la place d'origine (déplacez-le, puis relancez la commande).

Codes de sortie

Code	Signification
`0`	succès
`1`	configuration introuvable, invalide ou privée d'une clé ; au moins une source ou un document en échec ; annulation seulement partielle ; consentement OAuth2 échoué ; `--diagnose` avec au moins un `ÉCHEC`
`2`	montage du coffre non vérifié avant un mode qui écrit ; ligne de commande invalide
`3`	aucun modèle installé pour un rôle, ou Ollama ne répond pas
`4`	`--execute` refusé sur cette machine (clé `executer_sur`)
`5`	une autre exécution détient le verrou `dibab.verrou`

`--version` rend `0`, ou `2` pour une ligne de commande invalide.

Coffre et chiffrement

Ce qu'est un coffre

Un coffre est le répertoire qui reçoit les documents d'un domaine : un pour le domaine personnel, un pour le domaine professionnel (`destinations.personnel.racine` et `destinations.professionnel.racine`). Dibab ne le crée pas, ne le monte pas, ne le chiffre pas et ne le synchronise pas. Il attend que le répertoire soit là et, comme les documents sont sensibles, il attend qu'il se trouve sur un volume chiffré que vous ouvrez avant le passage.

Le danger est facile à décrire. Quand un volume chiffré est fermé, son point de montage n'est qu'un répertoire vide sur le disque en clair. Un programme qui y écrit y écrit les documents en clair, en dessous du chiffrement, sans aucune erreur. Dibab établit donc, avant chaque écriture, que chaque coffre est réellement monté.

La vérification du montage

Avant `--execute`, `--rollback`, `--inventaire` ou `--autoriser-google`, Dibab lance `findmnt -n -T <root> -o FSTYPE` pour la racine de chaque coffre et compare le type de système de fichiers à la liste `systemes_fichiers_acceptes` de ce coffre. Il vérifie aussi que la racine est accessible en écriture. Si le type n'est pas dans la liste, le montage du coffre n'est pas établi et le programme s'arrête avec le code de sortie 2 sans rien toucher.

La configuration d'exemple attend `fuse.gocryptfs`, le type que `findmnt` rapporte pour un volume gocryptfs monté. Avec un répertoire ordinaire à sa place, ce qui est l'aspect d'un coffre fermé, le refus est immédiat :

```
dibab --execute --config ~/.config/dibab/config-ferme.json; echo "exit code: $?"

ERROR: vault personnel: vault not mounted (type ext4, expected fuse.gocryptfs) (~/Classement/Pe⌋
↪  rsonnel)
ERROR: vault professionnel: vault not mounted (type ext4, expected fuse.gocryptfs) (~/Classemen⌋
↪  t/Professionnel)
        Nothing was touched. Mount the destination directory, then run again.
exit code: 2
```

La simulation ne s'arrête pas sur un coffre dont le montage n'est pas établi : elle le signale par `[!]` dans l'en-tête et poursuit, de sorte que vous puissiez lire les propositions avant d'ouvrir le coffre. Elle n'écrit pas son journal de session dans un chemin que le coffre contient, car cela mettrait le journal en clair sur le disque.

Choisir les types à accepter

Lisez le type de votre propre volume une fois ouvert, avec la commande du chapitre 2, et écrivez-le dans la liste :

```
"systemes_fichiers_acceptes": ["fuse.gocryptfs"]
```

Si vous chiffrez autrement (LUKS, eCryptfs, un volume d'un autre outil), ouvrez le volume et écrivez le type que `findmnt` rend pour lui. Sur un volume non chiffré, la vérification ne distingue plus un montage absent d'un montage présent : vous renoncez alors à cette protection, et le diagnostic du chapitre 9 vous rappelle que l'état est en clair.

Un exemple avec gocryptfs

gocryptfs ne fait pas partie de Dibab, et rien dans Dibab ne l'exige. C'est une façon de construire le coffre de chaque domaine. Créez chaque répertoire chiffré une fois, en choisissant une phrase secrète que vous gardez hors de la machine ; la commande la demande.

```
mkdir -p ~/Chiffre/Personnel ~/Chiffre/Professionnel
gocryptfs -init ~/Chiffre/Personnel
gocryptfs -init ~/Chiffre/Professionnel
```

Ouvrez chaque coffre avant un passage. La commande redemande la phrase secrète et monte la vue en clair sur le répertoire nommé dans la configuration :

```
gocryptfs ~/Chiffre/Personnel ~/Classement/Personnel
gocryptfs ~/Chiffre/Professionnel ~/Classement/Professionnel
```

Fermez-les après le passage :

```
fusermount3 -u ~/Classement/Personnel
fusermount3 -u ~/Classement/Professionnel
```

Les répertoires chiffrés `~/Chiffre/...` sont ce que vous sauvegardez et synchronisez. Dibab n'en sait rien.

Où vit l'état

À côté des documents, Dibab tient un état : la date du dernier scan de chaque source, le journal des actions, les journaux et rapports de session, le registre des fichiers récupérés et le verrou. Sans aucun chemin `suivi .*` dans la configuration, tout cela va sous `~/.local/state/dibab/` (ou `$XDG_STATE_HOME/dibab/`), en clair, avec le mode `700` pour les dossiers et `600` pour les fichiers. Le journal contient des noms de fichiers et des émetteurs ; c'est déjà révélateur. Le diagnostic le signale par un avertissement : `AVERTISSEMENT emplacement_etat`.

Pour chiffrer l'état, placez-le dans un coffre, avec trois clés du bloc `suivi` :

```
"suivi": {
  "fichier_date": "~/Classement/Personnel/logs/dernier_scan.json",
  "journal": "~/Classement/Personnel/logs/journal.jsonl",
  "sessions": "~/Classement/Personnel/logs/sessions"
}
```

Les autres fichiers de l'état (`rapatries.json`, `imap_deposes.json`, `dibab.verrou`) suivent `fichier_date`. Avec ce réglage, le diagnostic ne signale plus d'avertissement.

Place sur le disque

Dibab déplace et copie, et ne supprime jamais. Un document classé est copié vers sa destination, vérifié, puis retiré de la source : la place sur le disque ne croît donc pas avec un classement. Un document qui est un doublon d'un document déjà classé va dans le dossier `doublon/` du coffre au lieu d'être classé deux fois ; vider ce dossier est votre décision, jamais celle de l'outil.

Sources par ssh

Une machine distante du foyer se lit par `ssh`. Dibab liste les fichiers des répertoires que vous nommez, les copie dans un vide-poches du coffre personnel, vérifie chaque copie par son empreinte, puis les classe comme des documents locaux. Les fichiers d'origine restent intacts sur la machine distante : Dibab n'y retire ni n'y modifie jamais rien.

Ce que la machine distante doit offrir

- Un système Linux avec GNU `find` (la liste emploie `-printf`), `sha256sum` et `cat`.
- Un alias `ssh` qui ouvre une session sans mot de passe. Tout ce qui concerne la connexion, l'adresse, l'utilisateur et la clé, vit dans le `~/.ssh/config` de la machine locale, sous l'alias ; Dibab n'en sait rien. Par exemple, avec une machine inventée :

```
Host machine1
    HostName 192.0.2.10
    User archive
    IdentityFile ~/.ssh/id_ed25519
```

Vérifiez l'alias à la main avant le premier passage. La commande ne doit rien afficher et rendre `0` :

```
ssh machine1 true
```

Déclarer la machine

Deux blocs de `config.json` sont concernés : `ssh`, qui donne les délais, et `sources_distantes`, qui liste les machines.

```
"ssh": {
  "delai_connexion_secondes": 10,
  "delai_commande_secondes": 120,
  "delai_copie_secondes": 900
},

"sources_distantes": [
  {
    "id": "ssh-machine1",
    "alias_ssh": "machine1",
    "vide_poches_destination": "vide-poches-machine1",
    "repertoires": ["~/Téléchargements"],
    "date_reference": "modification",
    "recursif": true,
    "exclusions": []
  }
]
```

Clé	Signification
`ssh.delai_connexion_secondes`	le délai pour ouvrir la connexion (`10`)
`ssh.delai_commande_secondes`	le délai d'une liste ou d'une empreinte (`120`)

Clé	Signification
`ssh.delai_copie_secondes`	le délai pour copier un fichier (`900`)
`id`	un nom unique ; c'est la clé de la machine dans `dernier_scan.json`
`alias_ssh`	l'alias de `~/.ssh/config`
`vide_poches_destination`	le nom du vide-poches, sans aucun séparateur, créé sous la racine du coffre personnel ; les fichiers récupérés y arrivent
`repertoires`	les répertoires à lire sur la machine distante ; ~ est le répertoire personnel de l'utilisateur distant
`date_reference`, `recursif`, `exclusions`	comme pour une source locale (chapitre 3)

Les trois délais de `ssh` sont obligatoires dès qu'une source distante est déclarée.

Un premier passage : lister, puis récupérer

La configuration de ce chapitre déclare la machine et rien d'autre. La simulation liste les fichiers sans rien copier :

```
dibab --dry-run --config ~/.config/dibab/config-ssh.json

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : dry-run
  Configuration      : ~/.config/dibab/config-ssh.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

Remote source: ssh-machine1 (nothing is fetched in dry-run)
  - releve-bancaire.pdf                              (remote, 920 B, mtime 2026-11-23)

============================================================
Scan total: 0 candidate file(s), 0 B
============================================================
Text extraction (the text is neither shown nor written)

Classification (dry-run, nothing is moved; the text is neither shown nor written)
```

Le bloc `Remote source` montre chaque fichier avec le mot `remote`. Rien n'est écrit dans le vide-poches. Lors d'un premier scan, seuls les fichiers modifiés pendant les `suivi.premier_scan_jours` derniers jours (90 par défaut) sont listés, et seulement ceux dont l'extension figure dans `extensions_cibles`.

L'exécution récupère les fichiers, puis les classe :

```
dibab --execute --config ~/.config/dibab/config-ssh.json

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : execute
  Configuration      : ~/.config/dibab/config-ssh.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

Transfer of remote sources (copy checked by fingerprint, nothing is removed on the remote side)
  ssh-machine1: 1 candidate(s)

Source: ssh-machine1-vide-poches (~/Classement/Personnel/vide-poches-machine1)
  - releve-bancaire.pdf                              (920 B, mtime 2026-11-23)
  Source total: 1 file(s), 920 B

============================================================
Scan total: 1 candidate file(s), 920 B
============================================================
Text extraction (the text is neither shown nor written)
  - releve-bancaire.pdf: text layer, 242 character(s), 1 page(s), 0.02 s

Filing (execution: each document is copied, checked by SHA256, then removed from its source)
model 1/1, document 1/1
  - releve-bancaire.pdf: classe
      -> ~/Classement/Personnel/Administratif/Bob/2026/2026-03-15_Banque-Exemple_Banques.pdf

Session report: ~/.local/state/dibab/sessions/rapport_2026-11-23T12:09:03.json
```

Chaque fichier est copié dans `vide-poches-machine1`, relu, et comparé par SHA256 à l'empreinte calculée sur la machine distante. Une copie qui ne correspond pas n'est pas gardée. Le scan traite ensuite le vide-poches comme une source locale, sans aucun filtre de date, car un vide-poches se vide à mesure du classement : tout ce qu'il contient attend. Le registre `rapatries.json`, à côté de `dernier_scan.json`, retient les fichiers récupérés par empreinte.

Relancer

Une seconde exécution ne récupère rien de nouveau. Le fichier distant est de nouveau listé, mais son empreinte est dans le registre : il n'est donc ni copié ni classé une seconde fois :

```
dibab --execute --config ~/.config/dibab/config-ssh.json

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : execute
  Configuration      : ~/.config/dibab/config-ssh.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
```

```
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

Transfer of remote sources (copy checked by fingerprint, nothing is removed on the remote side)
  ssh-machine1: 1 candidate(s)

Source: ssh-machine1-vide-poches (~/Classement/Personnel/vide-poches-machine1)
  (no candidate file)
  Source total: 0 file(s), 0 B

============================================================
Scan total: 0 candidate file(s), 0 B
============================================================
Text extraction (the text is neither shown nor written)

Filing (execution: each document is copied, checked by SHA256, then removed from its source)

Session report: ~/.local/state/dibab/sessions/rapport_2026-11-23T12:09:03.json
```

La date de la machine dans `dernier_scan.json` n'avance qu'après une liste complète où chaque copie a été vérifiée. Un fichier en échec est récupéré de nouveau à l'exécution suivante, sans doublon. Un fichier qui existe à la fois sur la machine distante et dans un scan local va dans le dossier `doublon/` : il est classé une seule fois.

Quand la connexion échoue

Un refus arrête la machine concernée, et elle seule ; les autres sources continuent. Dibab ne réessaie pas en boucle, et vous non plus.

Message	Cause probable
`Could not resolve hostname` ou `Permission denied`	la machine est éteinte, ou l'alias ou la clé est faux ; corrigez, puis relancez une seule fois
un délai dépassé	augmentez `ssh.delai_commande_secondes` (liste) ou `ssh.delai_copie_secondes` (copie)
`remote directory partly readable or absent`	corrigez `repertoires`
aucun candidat alors que la machine a des fichiers	les fichiers sont plus anciens que `suivi.premier_scan_jours`, ou leur extension n'est pas dans `extensions_cibles`

Boîtes IMAP et comptes Google

Dibab lit les pièces jointes d'une boîte mail par IMAP, avec un mot de passe ou, pour un compte Google, avec un jeton OAuth2. Il lit sans modifier : la boîte est ouverte en lecture seule, aucun message n'est marqué comme lu, aucun drapeau n'est posé, et rien n'est copié, déplacé ni supprimé sur le serveur.

Ce qui est demandé au serveur

Le programme n'envoie au serveur que `EXAMINE` (l'ouverture d'un dossier en lecture seule), `UID SEARCH ... SINCE` (les messages reçus depuis une date), et `UID FETCH` de la date de réception, de la structure du message et d'une pièce jointe à la fois avec `BODY.PEEK`, la forme qui ne marque pas le message comme lu. Le corps d'un message n'est jamais demandé. La connexion est toujours en TLS, et le certificat est vérifié contre le magasin du système : une source avec `"ssl": false` est refusée.

Une pièce jointe est une partie du message qui porte un nom et dont la disposition est `attachment`, ou `inline` sans `Content-ID` (les images d'une signature en portent un et restent dans le message). Son extension doit figurer dans `extensions_cibles`. Son nom est une donnée fournie par l'expéditeur : il est décodé et nettoyé comme le nom d'un fichier récupéré (aucun séparateur, aucun caractère de contrôle, jamais caché, longueur bornée), et il n'entre ni dans le journal de session ni dans le rapport.

Déclarer une boîte

```
"imap": {
  "delai_secondes": 60,
  "taille_max_piece_mo": 25
},

"sources_imap": [
  {
    "id": "imap-foyer",
    "nom": "foyer",
    "serveur": "imap.example.com",
    "port": 993,
    "ssl": true,
    "authentification": "login",
    "utilisateur": "utilisateur@example.com",
    "mot_de_passe_env": "DIBAB_IMAP_FOYER",
    "dossiers": ["INBOX"],
    "vide_poches_destination": "vide-poches-foyer",
    "date_reference": "reception"
  }
]
```

Clé	Signification
`imap.delai_secondes`	le délai de chaque échange réseau, connexion comprise (`60`)
`imap.taille_max_piece_mo`	une pièce jointe plus grande que cela, d'après la taille qu'annonce le serveur, n'est pas récupérée (`25`)

Clé	Signification
`id`	un nom unique ; c'est la clé de la source dans `dernier_scan.json`
`nom`	une étiquette
`serveur`, `port`, `utilisateur`	le compte
`ssl`	doit être `true`
`authentification`	`login` (mot de passe dans une variable d'environnement) ou `oauth2` (compte Google) ; toute autre valeur arrête la source
`mot_de_passe_env`	le nom de la variable d'environnement qui contient le mot de passe (`login` seulement)
`dossiers`	les dossiers à lire, chacun ouvert en lecture seule
`vide_poches_destination`	le vide-poches, un nom unique, sous la racine du coffre personnel
`date_reference`	`reception` : la date de réception enregistrée par le serveur, jamais l'en-tête `Date`, que l'expéditeur écrit

Les deux blocs `imap` et `sources_imap` sont obligatoires ensemble ; une clé absente arrête le programme avec le code de sortie 1 et la nomme.

Le mot de passe

Le mot de passe n'est jamais dans `config.json`. Dibab le lit dans la variable d'environnement nommée par `mot_de_passe_env`, et ne l'écrit, ne le journalise ni ne l'affiche jamais. Définissez-la dans le terminal qui lance Dibab, sans l'écrire nulle part :

```
read -rs DIBAB_IMAP_FOYER && export DIBAB_IMAP_FOYER
```

Une source dont la variable n'est pas définie est ignorée avec un avertissement, sans aucune connexion. C'est aussi ce qui arrive à l'entrée d'exemple de `config.exemple.json` tant que vous la laissez déclarée.

Un premier passage : lister, puis déposer

La configuration de ce chapitre déclare la boîte et rien d'autre. La simulation liste les pièces jointes qui seraient récupérées, avec le mot `imap`, et n'en télécharge aucune :

```
dibab --dry-run --config ~/.config/dibab/config-imap.json
```

```
Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : dry-run
  Configuration      : ~/.config/dibab/config-imap.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
```

```
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

IMAP source: imap-foyer (nothing is downloaded in dry-run)
  - attestation-assurance.pdf                          (imap, 1274 B, received 2026-11-23)

============================================================
Scan total: 0 candidate file(s), 0 B
============================================================
Text extraction (the text is neither shown nor written)

Classification (dry-run, nothing is moved; the text is neither shown nor written)
```

L'exécution télécharge chaque pièce jointe, vérifie son empreinte, la dépose dans le vide-poches et la classe comme un document local :

```
dibab --execute --config ~/.config/dibab/config-imap.json

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : execute
  Configuration      : ~/.config/dibab/config-imap.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

IMAP sources (read-only: nothing is modified or deleted in the mailbox)
IMAP source: imap-foyer
  - attestation-assurance.pdf                          (imap, 1274 B, received 2026-11-23)

Source: imap-foyer-vide-poches (~/Classement/Personnel/vide-poches-foyer)
  - attestation-assurance.pdf                          (928 B, mtime 2026-11-23)
  Source total: 1 file(s), 928 B

============================================================
Scan total: 1 candidate file(s), 928 B
============================================================
Text extraction (the text is neither shown nor written)
  - attestation-assurance.pdf: text layer, 258 character(s), 1 page(s), 0.02 s

Filing (execution: each document is copied, checked by SHA256, then removed from its source)
model 1/1, document 1/1
  - attestation-assurance.pdf: classe
      -> ~/Classement/Personnel/Administratif/Alice/2026/2026-03-15_Assurance-Exemple_Assurance
       ↪  s.pdf

Session report: ~/.local/state/dibab/sessions/rapport_2026-11-23T12:09:03.json
```

Le fichier `imap_deposes.json`, à côté de `dernier_scan.json`, retient ce qui a été déposé (dossier, `UIDVALIDITY`, `UID`, partie et empreinte). Une pièce jointe dont l'empreinte y figure déjà, ou figure dans le journal des actions, n'est pas déposée de nouveau :

```
dibab --execute --config ~/.config/dibab/config-imap.json

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : execute
  Configuration      : ~/.config/dibab/config-imap.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl
  Model ocr           : essai-ocr:latest
  Model tri_rapide    : essai-tri:latest
  Model classification: essai-classification:latest
  Consensus          : 1 classification model(s), threshold 0.4

IMAP sources (read-only: nothing is modified or deleted in the mailbox)
IMAP source: imap-foyer
  - attestation-assurance.pdf                          (imap, 1274 B, received 2026-11-23)

Source: imap-foyer-vide-poches (~/Classement/Personnel/vide-poches-foyer)
  (no candidate file)
  Source total: 0 file(s), 0 B

============================================================
Scan total: 0 candidate file(s), 0 B
============================================================
Text extraction (the text is neither shown nor written)

Filing (execution: each document is copied, checked by SHA256, then removed from its source)

Session report: ~/.local/state/dibab/sessions/rapport_2026-11-23T12:09:03.json
```

La date de la source dans `dernier_scan.json` n'avance qu'après une lecture complète où chaque dépôt a réussi.

Quand la connexion échoue

Un refus d'authentification ou une connexion échouée arrête la source, sans seconde tentative. Les autres sources continuent.

Message	Cause probable
`environment variable ... not defined, source skipped`	exportez le mot de passe dans le terminal qui lance Dibab
`authentication refused by the server`	le mot de passe ou l'utilisateur est faux ; corrigez, puis relancez une fois, pas en boucle
`connection impossible`	le serveur, le port ou le réseau
`folder ... not found`	corrigez `dossiers`

Message	Cause probable
`attachment(s) above imap.taille_max_piece_mo`	augmentez cette valeur

Comptes Google avec OAuth2

Un compte Google se lit avec un jeton OAuth2 au lieu d'un mot de passe. Les étapes dans la console Google sont à faire par vous, avec le compte Google à lire, et les libellés de la console changent avec le temps : les notions sont celles de sa documentation OAuth2.

1. Dans la console Google Cloud, créez un projet, activez l'API Gmail et configurez l'écran de consentement OAuth (type « externe », votre adresse comme contact). Publiez ensuite l'application, afin que son état soit « En production ». Avec l'état « Test », l'autorisation expire après sept jours.
2. Créez un identifiant OAuth du type « Application de bureau » et téléchargez son fichier JSON.
3. Gardez ce fichier dans le coffre, dans un dossier en mode `700`, avec le mode `600`. Dibab ne le modifie jamais. C'est le `client_fichier`.
4. Déclarez la source avec `"authentification": "oauth2"`, `client_fichier` et `token_fichier` à la place de `mot_de_passe_env`, et ajoutez le bloc `oauth2` :

```
"oauth2": {
  "url_autorisation": "https://accounts.google.com/o/oauth2/v2/auth",
  "url_jeton": "https://oauth2.googleapis.com/token",
  "portee": "https://mail.google.com/",
  "marge_renouvellement_secondes": 300,
  "delai_secondes": 30,
  "delai_consentement_secondes": 300,
  "chemins_interdits": ["~/Documents"]
},

"sources_imap": [
  {
    "id": "imap-google",
    "nom": "foyer-google",
    "serveur": "imap.gmail.com",
    "port": 993,
    "ssl": true,
    "authentification": "oauth2",
    "utilisateur": "utilisateur@example.com",
    "client_fichier": "~/Classement/Personnel/logs/client-google.json",
    "token_fichier": "~/Classement/Personnel/logs/jeton-google.json",
    "dossiers": ["INBOX"],
    "vide_poches_destination": "vide-poches-google",
    "date_reference": "reception"
  }
]
```

Clé de `oauth2`	Signification
`url_autorisation`, `url_jeton`	les points d'accès de Google ; ils doivent être en `https`, sauf sur la boucle locale
`portee`	la portée OAuth (`https://mail.google.com/`)
`marge_renouvellement_secondes`	le jeton d'accès est renouvelé quand il lui reste moins que cela (`300`)
`delai_secondes`	le délai de chaque échange avec le serveur d'autorisation (`30`)
`delai_consentement_secondes`	combien de temps `--autoriser-google` attend votre réponse (`300`)
`chemins_interdits`	les dossiers où un jeton est refusé (`~/Documents`)

5. Consentez une fois, depuis un terminal de la machine, avec l'`id` de la source :

```
dibab --autoriser-google imap-google
```

Dibab affiche une adresse qui commence par `https://accounts.google.com/`. Ouvrez-la dans un navigateur. Acceptez l'avertissement « application non vérifiée », affiché parce que l'application est la vôtre, puis le consentement. Le programme reçoit la réponse sur la boucle locale, la vérifie, l'échange contre un jeton et écrit le fichier de jeton en mode `600` dans un dossier en mode `700`. Le fichier de jeton doit se trouver hors de tout dépôt git et hors de `chemins_interdits` ; un fichier de jeton lisible par le groupe ou par les autres arrête la source. Si le consentement est refusé, si aucune réponse n'arrive à temps, ou si Google ne rend aucun jeton de renouvellement, rien n'est écrit ; relancez la commande.

Dès lors, le scan tourne comme pour une source à mot de passe, sans aucune variable d'environnement : le jeton d'accès se renouvelle seul, et le jeton n'apparaît ni dans le journal ni à l'écran. Un fichier de jeton qui n'existe pas n'est pas une erreur : la source est ignorée, avec la commande à lancer. Si Google révoque l'autorisation (vous avez retiré l'accès de l'application, vous avez changé le mot de passe du compte Google, ou six mois ont passé sans scan), Dibab arrête la source avec un message explicite, et vous relancez `--autoriser-google`.

La portée `https://mail.google.com/` est la seule que Google documente pour IMAP, et elle permet de lire, d'écrire et de supprimer. Que la boîte reste non modifiée est garanti par ce que Dibab envoie, qui se réduit aux commandes listées au début de ce chapitre, non par la portée. Le chapitre 10 y revient.

Inventaire et rotation

Lire ce qu'un coffre contient déjà : `--inventaire`

Si vous avez déjà des documents dans un coffre, classés à la main ou par un autre outil, `--inventaire` les lit et écrit un plan de migration, sans rien déplacer, renommer ni supprimer. Il est en lecture seule : avant et après, chaque coffre est identique. Il demande la vérification du montage du chapitre 5, puisque chaque plan est écrit dans son coffre.

Pour avoir quelque chose à lire, classez de nouveau les quatre documents de l'aire d'essai ; l'annulation du chapitre 4 les a remis dans leur source :

```
dibab --execute | tail -n 9

model 1/1, document 1/3
model 1/1, document 2/3
model 1/1, document 3/3
  - notification-organisme.pdf: classe
      -> ~/Classement/Personnel/Administratif/Alice/2026/2026-03-15_Organisme-public_Aides.pdf
  - facture-hebergement.pdf: classe
      -> ~/Classement/Professionnel/Classement/2026/Factures/2026-03-15_Hebergeur-Exemple_Factu
       ↪  res.pdf
  - recette-gateau.pdf: not administrative, stays in place
  - courrier-sans-en-tete.pdf: a_verifier
      -> ~/Classement/Personnel/À_VÉRIFIER/[À_VÉRIFIER]courrier-de-suivi-de-dossier_courrier-sa
       ↪  ns-en-tete.pdf

Session report: ~/.local/state/dibab/sessions/rapport_2026-11-23T12:09:03.json
```

Lisez ensuite le coffre de chaque domaine :

```
dibab --inventaire

Dibab 0.9.0-alpha — session 2026-11-23T12:09:03
  Mode               : inventaire
  Configuration      : ~/.config/dibab/config.json
  Vault personnel     : ~/Classement/Personnel — mounted
  Vault professionnel : ~/Classement/Professionnel — mounted
  Session journal    : ~/.local/state/dibab/sessions/session_2026-11-23T12:09:03.jsonl

Inventory of what exists (read-only, nothing is moved)
  Vault personnel     : 3 listed document(s), 3 in the filed zone, 0 to file, 0 duplicate(s), 0
   ↪  present in the other vault, 1 in À_VÉRIFIER (not listed)
      plan written: ~/Classement/Personnel/logs/inventaire_2026-11-23T12:09:03.json
  Vault professionnel : 1 listed document(s), 1 in the filed zone, 0 to file, 0 duplicate(s), 0
   ↪  present in the other vault, 0 in À_VÉRIFIER (not listed)
      plan written: ~/Classement/Professionnel/logs/inventaire_2026-11-23T12:09:03.json
```

Pour chaque coffre, le rapport donne le nombre de documents lus, combien sont déjà dans la zone de classement, combien sont en dehors et devraient être classés, combien sont des doublons d'un autre document du même coffre, et combien sont aussi présents dans l'autre coffre. Le plan est un fichier JSON, `logs/`

`inventaire_<time>.json`, écrit dans le coffre qu'il décrit :

```
ls ~/Classement/Personnel/logs

inventaire_2026-11-23T12:09:03.json
```

Le plan de chaque coffre ne nomme que les fichiers de ce coffre. De l'autre coffre, il ne garde qu'un seul fait, « cette empreinte s'y trouve aussi », de sorte que le plan d'un coffre ne contient jamais le chemin d'un fichier de l'autre.

Chaque document du plan reçoit une proposition :

Proposition	Signification
rien à faire	le document est dans la zone de classement
le mettre dans un vide-poches, puis lire le dry-run	le document est hors de la zone de classement : déposez-le dans une source et laissez le passage suivant le classer
identique à un autre document : à classer dans `doublon/` après vérification	l'empreinte est déjà présente

Les dossiers listés dans `inventaire.exclusions` (`logs`, `Archive`, `rebut`, et le dossier nommé par `destination.repertoire_doublon`, dans l'exemple), le dossier à vérifier, les vide-poches et les dossiers cachés ne sont pas lus.

Garder le journal court : la rotation

Deux fichiers grossissent avec l'usage : le journal des actions, `suivi.journal`, et le dossier des sessions, `suivi.sessions`. La rotation applique les limites de `suivi.rotation` au démarrage de `--execute` ou de `--rollback` :

Clé	Effet
`journal_taille_max_mo`	quand le journal dépasse cette taille, il est renommé `journal.1.jsonl`, les archives plus anciennes montent d'un numéro et un nouveau journal commence
`journal_archives_max`	le nombre d'archives conservées ; la plus ancienne est retirée quand une nouvelle dépasserait ce nombre
`sessions_max`	le nombre de journaux et de rapports de session conservés ; les plus anciens sont retirés

C'est la seule suppression que Dibab effectue, et elle concerne ses propres journaux, jamais un document. Une archive qui contient encore une action non annulée n'est jamais retirée : la rotation attend plutôt que de perdre ce que `--rollback` doit pouvoir lire. Annulez un classement dont vous doutez avant que son journal ne vieillisse.

Les fichiers de l'état sont écrits avec le mode `600` et leur dossier avec le mode `700`. L'endroit où ils sont gardés est le choix décrit au chapitre 5.

Diagnostic

`--diagnose` vérifie l'installation, la configuration, Ollama et le coffre de chaque domaine, une ligne par contrôle, et n'écrit rien : aucun fichier n'est créé, rien n'est déplacé, aucun verrou n'est pris. Il n'ouvre qu'une seule connexion, vers Ollama, pour lire la liste des modèles. Il n'en ouvre aucune vers une machine distante ni vers une boîte mail : on peut donc le lancer à tout moment. Il joue chaque contrôle jusqu'au bout, quel que soit le résultat des précédents.

```
dibab --diagnose

OK python : 3.13.5
OK outil_pdftotext : /usr/bin/pdftotext
OK outil_pdftoppm : /usr/bin/pdftoppm
OK outil_findmnt : /usr/bin/findmnt
OK outil_ssh : /usr/bin/ssh
OK configuration : ~/.config/dibab/config.json
AVERTISSEMENT emplacement_etat : the state and the journal are in clear text, outside any vault:
 ↪  ~/.local/state/dibab
OK ollama : http://127.0.0.1:18434 answers, 3 model(s) installed
OK modele_ocr : essai-ocr:latest
OK modele_tri_rapide : essai-tri:latest
OK modele_classification : essai-classification:latest, 1 model(s) in the consensus
NON_CONFIGURÉ modele_tri_decision : ollama.roles.tri_decision absent or empty: the triage goes
 ↪  through ollama.roles.tri_rapide
OK destination_personnel : ~/Classement/Personnel
OK destination_professionnel : ~/Classement/Professionnel
Diagnostic: 12 OK, 1 warning(s), 0 failure(s), 1 not configured, 0 not checked
```

Chaque ligne se lit `<STATE> <identifier> : <message>`, puis un récapitulatif clôt le rapport. Le message est dans la langue de votre session ; les états et les identifiants ne changent jamais avec la langue, de sorte qu'un script puisse les comparer.

Les états

État	Signification
`OK`	le contrôle a réussi
`AVERTISSEMENT`	un défaut qui n'empêche aucun mode, ou un défaut d'un composant que cette instance n'utilise pas
`ÉCHEC`	le contrôle a échoué sur une chose que l'installation ou la configuration trouvée exige
`NON_CONFIGURÉ`	la configuration que le contrôle lit n'existe pas encore
`NON_VÉRIFIÉ`	le contrôle dépend d'un autre qui n'a pas réussi, ou d'une configuration absente

Le code de sortie est `0` quand il n'y a aucun `ÉCHEC` et `1` quand il y en a au moins un. Un avertissement, une configuration absente et un point non vérifié ne le changent pas.

Les contrôles, dans l'ordre

Identifiant	Ce qui est vérifié	Si ce n'est pas OK
python	l'interpréteur est en 3.10 ou plus récent	ÉCHEC en dessous
outil_pdftotext, outil_pdftoppm, outil_findmnt	l'outil est exécutable dans le PATH	ÉCHEC s'il est absent
outil_ssh	ssh est exécutable dans le PATH	ÉCHEC si la configuration déclare sources_distantes, AVERTISSEMENT sinon
configuration	une configuration est trouvée (chapitre 2) et valide : JSON lisible, les clés que chaque mode exige	aucune trouvée : NON_CONFIGURÉ, et le message liste les quatre emplacements essayés ; trouvée mais invalide : ÉCHEC, avec les clés manquantes
emplacement_etat	l'état et le journal sont sous la racine d'un coffre déclaré	hors de tout coffre : AVERTISSEMENT (l'état est en clair)
ollama	GET <ollama.url>/api/tags répond dans ollama.timeout_secondes	pas de réponse : ÉCHEC
modele_ocr, modele_tri_rapide, modele_classification	le rôle a au moins un modèle installé parmi ses préférences	aucun installé ou liste vide : ÉCHEC, et le message nomme ollama.roles.<role>
modele_tri_decision	le tri par décision, facultatif (chapitre 3) : premier modèle installé de ollama.roles.tri_decision, capacité decision, seuils valides dans classification.seuils_tri_decision, un modèle installé pour ollama.roles.objet	rôle absent ou vide : NON_CONFIGURÉ ; déclaré mais inutilisable : AVERTISSEMENT, le message nomme la clé à corriger, et le tri passe par tri_rapide
destination_personnel, destination_professionnel	la vérification du montage du chapitre 5 : le type de système de fichiers de la racine est accepté et la racine est accessible en écriture	non établie : ÉCHEC, et le message donne la raison

Une source à mot de passe dont la variable d'environnement n'est pas définie est signalée par un avertissement dans le contrôle configuration, puisque le programme ignore une telle source à l'exécution. Aucun contrôle n'écrit un secret, un nom de document ou le contenu d'une variable d'environnement.

Lire un échec

Une configuration de l'aire d'essai déclare pour chaque domaine un coffre qui doit être `fuse.gocryptfs`, alors que les répertoires sont ordinaires, ce qui est l'aspect d'un coffre fermé :

```
dibab --diagnose --config ~/.config/dibab/config-ferme.json; echo "exit code: $?"

OK python : 3.13.5
OK outil_pdftotext : /usr/bin/pdftotext
OK outil_pdftoppm : /usr/bin/pdftoppm
OK outil_findmnt : /usr/bin/findmnt
OK outil_ssh : /usr/bin/ssh
OK configuration : ~/.config/dibab/config-ferme.json
AVERTISSEMENT emplacement_etat : the state and the journal are in clear text, outside any vault:
 ↪  ~/.local/state/dibab
OK ollama : http://127.0.0.1:18434 answers, 3 model(s) installed
OK modele_ocr : essai-ocr:latest
OK modele_tri_rapide : essai-tri:latest
OK modele_classification : essai-classification:latest, 1 model(s) in the consensus
NON_CONFIGURÉ modele_tri_decision : ollama.roles.tri_decision absent or empty: the triage goes
 ↪  through ollama.roles.tri_rapide
ÉCHEC destination_personnel : vault not mounted (type ext4, expected fuse.gocryptfs) (~/Classem
 ↪  ent/Personnel)
ÉCHEC destination_professionnel : vault not mounted (type ext4, expected fuse.gocryptfs) (~/Cla
 ↪  ssement/Professionnel)
Diagnostic: 10 OK, 1 warning(s), 2 failure(s), 1 not configured, 0 not checked
exit code: 1
```

Les deux contrôles `destination_*` sont en échec et le code est `1`, alors qu'Ollama, les modèles et les outils vont bien. Le message dit quoi faire : ouvrir le coffre, ou écrire le bon type dans `systemes_fichiers_acceptes`.

Autres échecs fréquents :

Ligne	Que faire
`ÉCHEC ollama`	le service ne répond pas : vérifiez `ollama.url`, puis qu'Ollama tourne (`systemctl status ollama`)
`ÉCHEC modele_<role>`	corrigez `ollama.roles.<role>` d'après `ollama list`
`AVERTISSEMENT modele_tri_decision`	installez le modèle decision ou le modèle d'objet nommé, ou écrivez ses seuils ; d'ici là, le tri passe par `tri_rapide`
`ÉCHEC destination_<domain>`	ouvrez le coffre nommé, ou corrigez les types acceptés
`ÉCHEC configuration`	ajoutez les clés nommées, en copiant le bloc de `config.exemple.json`

Une installation toute neuve

Juste après l'installation, sans configuration et sans Ollama, le diagnostic n'est pas un échec : une configuration absente est l'état normal d'une installation neuve et se nomme `NON_CONFIGURÉ`, sans être confondue avec une panne. Dans un répertoire personnel vide :

```
HOME="$(mktemp -d)" dibab --diagnose; echo "exit code: $?"

OK python : 3.13.5
OK outil_pdftotext : /usr/bin/pdftotext
OK outil_pdftoppm : /usr/bin/pdftoppm
OK outil_findmnt : /usr/bin/findmnt
OK outil_ssh : /usr/bin/ssh
NON_CONFIGURÉ configuration : no configuration found (the --config option: not given; the DIBAB
↪  _CONFIG variable: not given; the XDG location: ~/.config/dibab/config.json; the script
↪  folder: ~/applications/dibab/config.json)
NON_VÉRIFIÉ emplacement_etat : depends on the configuration
NON_VÉRIFIÉ ollama : depends on the configuration
NON_VÉRIFIÉ modele_ocr : depends on the configuration
NON_VÉRIFIÉ modele_tri_rapide : depends on the configuration
NON_VÉRIFIÉ modele_classification : depends on the configuration
NON_VÉRIFIÉ modele_tri_decision : depends on the configuration
NON_VÉRIFIÉ destination_personnel : depends on the configuration
NON_VÉRIFIÉ destination_professionnel : depends on the configuration
Diagnostic: 5 OK, 0 warning(s), 0 failure(s), 1 not configured, 8 not checked
exit code: 0
```

Les outils sont `OK`, la configuration est `NON_CONFIGURÉ` et les huit contrôles qui en dépendent sont `NON_VÉRIFIÉ`. Le code de sortie est `0`. Dès qu'une configuration existe, tout ce qu'elle déclare et que la machine n'a pas est un `ÉCHEC` : un diagnostic qui rendrait `0` avec Ollama arrêté cacherait la panne même que vous êtes venu chercher.

Dans une autre langue

La langue suit `LC_ALL`, puis `LC_MESSAGES`, puis `LANG` :

```
LC_ALL=fr_FR.UTF-8 dibab --diagnose | tail -n 1

Diagnostic : 12 OK, 1 avertissement(s), 0 échec(s), 1 non configuré(s), 0 non vérifié(s)
```

Souveraineté et limites

Ce que l'outil garantit

- **Aucun document ne quitte votre réseau**. Les appels aux modèles de langage sont de simples requêtes HTTP vers `ollama.url` (par défaut `http://localhost:11434`). Aucun document et aucun nom de fichier ne va ailleurs, et il n'y a aucune télémétrie.
- **Les seuls services extérieurs que le code peut atteindre sont ceux que vous déclarez** : vos machines par `ssh`, vos boîtes IMAP et, si vous activez OAuth2, les points d'accès d'autorisation de Google.
- **Aucun secret n'est stocké dans `config.json`**. Un mot de passe de messagerie vit dans une variable d'environnement, un jeton Google dans un fichier en mode `600` hors de tout dépôt. Un jeton n'est jamais écrit dans un journal, affiché à l'écran ni mis dans un message d'erreur.
- **Un classement ne perd jamais un document**. Il est copié, la copie est vérifiée par SHA256, et seulement alors l'original est retiré. Chaque classement est écrit dans le journal des actions et peut être annulé avec `--rollback`.
- **Une boîte mail et une machine distante restent intactes**. Les commandes IMAP envoyées sont celles listées au chapitre 7 ; une machine distante n'est que lue, et rien n'y est retiré.
- **Le texte d'un document est une donnée, jamais une instruction**. Il est délimité par un marqueur aléatoire et le modèle répond par un JSON fermé que Dibab valide. Une réponse qui ne respecte pas le schéma compte comme « incertaine » et mène au dossier à vérifier. Un modèle ne peut jamais déclencher une action.
- **Le doute mène à une décision humaine**. Un domaine incertain, des modèles en désaccord, ou une pièce commune sans catégorie unanime mettent le document dans `À_VÉRIFIER/`, pas dans un endroit hasardeux.
- **Le journal contient des chemins et des catégories, jamais de texte**. Il contient bien des noms de fichiers et des émetteurs, qui révèlent déjà beaucoup ; gardez-le dans un coffre (chapitre 5).

Le code est publié sous la licence GNU Affero General Public License, version 3. Il n'utilise que la bibliothèque standard de Python : il n'y a donc aucun paquet tiers à auditer.

Le respect du foyer

Une machine, un répertoire ou un compte mail n'entre dans le périmètre de l'outil qu'avec l'accord, explicite ou implicite, de la personne concernée. Le logiciel ne peut pas le vérifier : c'est l'administrateur de l'instance qui s'y engage. Ne dirigez pas Dibab vers la boîte mail ou les fichiers d'une personne qui ignore qu'ils sont lus.

Les limites

- **Les sources distantes n'ont été essayées que contre de faux serveurs**. La source `ssh` et la source IMAP, avec un mot de passe ou avec OAuth2, sont couvertes par des tests et par les rejeux de ce manuel, qui emploient un faux `ssh`, un faux serveur IMAP et un faux serveur d'autorisation. Leur emploi sur une vraie machine du foyer et une vraie boîte mail, consentement avec Google compris, reste à faire : tant que cet essai n'a pas eu lieu, considérez ces sources comme non validées en conditions réelles et commencez par `--dry-run`.

- **La qualité du classement est celle de vos modèles**. Sur l'instance de référence, un premier essai sur 15 documents réels en a classé 2, mis 3 de côté comme non administratifs et placé 10 dans `À_VÉRIFIER/`, par prudence. Mesurez avant de faire confiance ; l'outil de mesure des sources, `_outils/mesurer_domaines.py`, échantillonne des documents classés et compare le verdict des modèles avec le coffre où se trouve chacun.
- **Linux seulement**. La vérification du montage s'appuie sur `findmnt`, qui appartient à `util-linux`.
- **Dibab ne monte, ne chiffre et ne synchronise rien**, et il ne gère ni la place sur le disque ni les sauvegardes : cela vous revient.
- **La portée Google est plus large que l'usage**. La portée `https://mail.google.com/` est la seule que Google documente pour IMAP. Elle permet de lire, d'écrire et de supprimer. Que la boîte reste non modifiée repose sur les commandes que Dibab envoie, listées au chapitre 7, non sur la portée.
- **Les valeurs par défaut sont des exemples**. Les catégories, les émetteurs connus et le modèle de nommage de `config.exemple.json` sont génériques ; adaptez-les à votre pays.
- **Les étapes dans la console Google dépendent de l'interface de Google**, qui change. Les notions restent celles de sa documentation OAuth2.
- **Le projet est jeune**. La version est 0.9.0-alpha ; il n'y a pas d'intégration continue. Les tests des sources se lancent à la main.
- **Ollama ne fait pas partie de Dibab**. Il n'est jamais installé ni mis à jour par lui ; un modèle renommé ou retiré en amont est une ligne à corriger dans `ollama.roles`.

Signaler un problème

Une vulnérabilité se signale en privé, selon le fichier `SECURITY.md` des sources, jamais dans un ticket public. Pour une contribution, lisez `CONTRIBUTING.md` au même endroit.