

Dibab

Benutzerhandbuch

Die Verwaltungsunterlagen eines Haushalts ordnen, auf Ihren eigenen Maschinen

Version 0.9.0-alpha

2026-11-26

Übersetzung. Maßgeblich ist die englische Fassung.

`dibab.davalan.fr`

Inhaltsverzeichnis

Einführung **3**

Was Dibab tut . . . 3

Was es nicht tut . . . 3

Ein Werkzeug für den Haushalt . . . 3

Aufbau dieses Handbuchs . . . 4

Konventionen . . . 4

Installation **5**

Was Sie brauchen . . . 5

Die Modelle nach Rolle . . . 5

Die fertige Verzeichnisstruktur . . . 5

Installation aus dem Debian-Paket . . . 6

Installation unter Fedora, openSUSE Leap und Arch . . . 7

Aus den Quellen installieren . . . 7

Die Tests ausführen (optional) . . . 8

Wo die Konfiguration gesucht wird . . . 8

Konfiguration **9**

Die Schlüssel, die Sie schreiben müssen . . . 9

`destinations`: der Tresor jedes Bereichs . . . 9

`destination`, `arborescence`, `executer_sur` . . . 10

`suivi`: Zustand, Journal und Berichte . . . 10

`ollama`: die Modelle . . . 11

`extraction` und `classification` . . . 12

`categories`, `personnes`, `emetteurs_connus`, `pieces_communes` . . . 13

`nommage`: der Name eines abgelegten Dokuments . . . 14

`extensions_cibles` und `scan` . . . 14

`sources_locales` . . . 15

`ssh`, `sources_distantes`, `imap`, `oauth2`, `sources_imap` . . . 15

`mesure` und `inventaire` . . . 15

Erster Durchlauf **16**

Simulieren: `--dry-run` . . . 16

Die Zuordnung ohne die Modelle prüfen . . . 18

Ablegen: `--execute` . . . 19

Rückgängig machen: `--rollback` . . . 21

Exit-Codes . . . 21

Tresor und Verschlüsselung **23**

Was ein Tresor ist . . . 23

Die Prüfung des Einhängens . . . 23

Ein Beispiel mit gocryptfs . . . 24

Wo der Zustand liegt . . . 24

Platz auf dem Datenträger . . . 25

Quellen über ssh **26**
Was die entfernte Maschine bieten muss . . . 26
Die Maschine deklarieren . . . 26
Ein erster Durchlauf: auflisten, dann abholen . . . 27
Erneut ausführen . . . 28
Wenn die Verbindung scheitert . . . 29

IMAP-Postfächer und Google-Konten **30**
Was vom Server verlangt wird . . . 30
Ein Postfach deklarieren . . . 30
Das Passwort . . . 31
Ein erster Durchlauf: auflisten, dann ablegen . . . 31
Wenn die Verbindung scheitert . . . 33
Google-Konten mit OAuth2 . . . 34

Inventur und Rotation **36**
Lesen, was ein Tresor schon enthält: `--inventaire` . . . 36
Das Journal kurz halten: die Rotation . . . 37

Diagnose **38**
Die Zustände . . . 38
Die Prüfungen in der Reihenfolge . . . 39
Einen Fehlschlag lesen . . . 40
Eine frische Installation . . . 41
In einer anderen Sprache . . . 41

Souveränität und Grenzen **42**
Was das Werkzeug garantiert . . . 42
Achtung des Haushalts . . . 42
Die Grenzen . . . 42
Ein Problem melden . . . 43

Einführung

Dibab („wählen“ auf Bretonisch) sammelt, analysiert und ordnet die persönlichen Verwaltungsdokumente eines Haushalts: Rechnungen, Briefe, Verträge, Bescheide von Behörden. Es verschiebt jedes Dokument in eine normierte Verzeichnisstruktur, unter einem Namen, der Datum, Absender und Kategorie nennt, und es führt ein Journal über alles, was es tut, sodass sich jede Ablage rückgängig machen lässt. Dieses Handbuch beschreibt die Version 0.9.0-alpha.

Es läuft im souveränen Modus. Keine Daten verlassen Ihr lokales Netz: kein Drittanbieter-Dienst, kein entferntes Modell, keine Cloud. Die Sprachmodelle sind die von Ollama, das auf Ihrer eigenen Maschine läuft.

Was Dibab tut

- Es durchsucht lokale Verzeichnisse, entfernte Maschinen über `ssh` und Postfächer über IMAP. Eine entfernte Datei wird kopiert und per Fingerabdruck geprüft, das Original bleibt unberührt. Ein Postfach wird gelesen, ohne verändert zu werden: Nur die Anhänge werden geholt, und keine Nachricht wird als gelesen markiert.
- Es extrahiert den Text jeder Datei (die Textebene eines PDFs, sonst das Lesen der Seitenbilder durch ein Bildmodell), sondert aus, was nicht zur Verwaltung gehört, und ordnet den Rest nach Kategorie und nach Bereich, privat oder beruflich, im Konsens mehrerer Modelle ein.
- Es benennt jedes Dokument um und verschiebt es in den Tresor seines Bereichs, schreibt ein zeitgestempeltes Journal und kann eine ganze Sitzung mit `--rollback` rückgängig machen.

Was es nicht tut

- Es verschiebt keine Datei, bei der es nicht sicher ist. Der Zweifel führt in einen Ordner namens `ZU_PRÜFEN/`, nie zu einer riskanten Ablage.
- Es löscht nie ein Dokument. Es kopiert, prüft und entfernt das Original erst, wenn die Kopie als identisch feststeht.
- Es verändert weder ein Postfach noch eine entfernte Maschine.
- Es läuft nicht im Hintergrund. Sie starten es von Hand, und sein Standardmodus ist die Simulation, `--dry-run`.
- Es hängt keinen Tresor ein, verschlüsselt ihn nicht und synchronisiert ihn nicht. Das ist Ihre Sache; Kapitel 5 erklärt, wie das Werkzeug es prüft.

Ein Werkzeug für den Haushalt

Dibab ist für das Verwaltungsleben eines Haushalts gedacht, durch den Haushalt. Eine Maschine, ein Verzeichnis oder ein Mail-Konto gehört nur dann zu seinem Umfang, wenn die betroffene Person ausdrücklich oder stillschweigend zugestimmt hat; im Zweifel lassen Sie es weg. Die Software kann diese Achtung nicht gewährleisten: Der Administrator der Instanz verpflichtet sich dazu.

Das Journal enthält Dateinamen und Absender, die schon viel verraten (ein Brief einer Gesundheitsbehörde genügt). Sein Platz ist deshalb im Tresor, nicht daneben: Legen Sie die Schlüssel `suivi.*` in den Tresor, sonst wird es unverschlüsselt unter `~/.local/state/dibab/` geschrieben.

Aufbau dieses Handbuchs

1. Einführung (dieses Kapitel).
2. Installation: was zu installieren ist, wo alles liegt, und eine erste Prüfung.
3. Konfiguration: die Konfigurationsdatei, Schlüssel für Schlüssel.
4. Erster Durchlauf: simulieren, den Bericht lesen, wirklich ablegen, rückgängig machen.
5. Tresor und Verschlüsselung: wie das Werkzeug feststellt, dass ein Tresor eingehängt ist, und wo der Zustand liegt.
6. Quellen über ssh: entfernte Maschinen.
7. IMAP-Postfächer und Google-Konten.
8. Inventur und Rotation: lesen, was ein Tresor schon enthält, und das Journal kurz halten.
9. Diagnose: `--diagnose`, Zustand für Zustand.
10. Souveränität und Grenzen: was das Werkzeug garantiert und was nicht.

Konventionen

Befehle werden so gezeigt, wie man sie in einem Terminal eintippt, für den Benutzer der in Kapitel 2 beschriebenen Installation. Der Befehl heißt `dibab`. Der Text nach einem Befehl ist seine echte Ausgabe, erzeugt durch das erneute Ausführen des Befehls auf einem Übungsbereich aus erfundenen Dokumenten; die in diesen Ausgaben genannten Modelle (`essai-ocr`, `essai-tri`, `essai-classification`) sind die Platzhalter dieses Übungsbereichs, und Ihre eigenen sind die, die Sie in Ihrer Konfiguration nennen. Jedes Dokument, jede Person, jeder Absender und jede Firma in diesem Handbuch ist erfunden.

Die Sprache der Meldungen folgt Ihrer Sitzung: Dibab liest die erste nicht leere Variable unter `LC_ALL`, `LC_MESSAGES` und `LANG` und spricht entsprechend Englisch, Französisch, Deutsch, Spanisch oder vereinfachtes Chinesisch (jeder andere Wert ergibt Englisch). Die Namen der Optionen, die Zustandskennzeichen wie `OK` und `ÉCHEC`, die Bezeichner, die Journale und die Plandateien ändern sich nie mit der Sprache, sodass ein in einer Sprache geschriebenes Journal von `--rollback` in jeder anderen wiederholt wird. Die Ausgaben dieses Handbuchs sind englisch.

Installation

Was Sie brauchen

- Ein Linux-System mit Python 3.10 oder neuer. Dibab verwendet nur die Standardbibliothek von Python, sodass für Python nichts weiter zu installieren ist.
- `findmnt` (Paket `util-linux`), das feststellt, dass ein Tresor eingehängt ist.
- `pdftotext` und `pdftoppm` (Paket `poppler-utils`), die PDF-Dateien lesen.
- `ssh` (Paket `openssh-client`), nur wenn Sie entfernte Maschinen durchsuchen, mit einem Schlüssel, der eine Sitzung ohne Passwort öffnet.
- Ollama, installiert und laufend, mit mindestens einem Modell für jede Rolle (siehe unten). Dibab installiert Ollama nie und lädt nie ein Modell herunter.
- Ein Zielverzeichnis für jeden Bereich, privat und beruflich, möglichst auf einem verschlüsselten Datenträger (Kapitel 5).
- `git`, um die Quellen zu holen.

Unter Debian oder Ubuntu werden die Pakete mit einem Befehl installiert:

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

Die Modelle nach Rolle

Drei Rollen verwenden je ein Modell. Die Rolle `ocr` liest eingescannte Seiten, `tri_rapide` („Schnellsortierung“) sondert aus, was nicht zur Verwaltung gehört, und `classification` ordnet ein. Die Modelle der Rolle `classification` stimmen ab: Ein Dokument wird nur eingeordnet, wenn ihre Übereinstimmung `ollama .seuil_confiance` erreicht (Kapitel 3). Zwei optionale Rollen, `tri_decision` und `objet`, überlassen die Sortierung einem `decision`-Modell in einem Durchgang und den Gegenstand einem kleinen Modell; ohne sie sortiert `tri_rapide` (Kapitel 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
```

Das sind die Modelle der Referenzinstanz; beliebige andere genügen, solange Sie sie in `ollama.roles` nennen. Was installiert ist, zeigt `ollama list`.

Die fertige Verzeichnisstruktur

~ ist Ihr Home-Verzeichnis. Passen Sie die Pfade an Ihre Installation an.

```
~/applications/dibab/              das geklonte Repository
├── dibab.py                       der Einstiegspunkt
├── modules/                            eine Datei pro Zuständigkeit
│   └── langues/                        die Meldungswörterbücher: en, fr, de, es, zh
├── docs/                               die Handbuchseite und die übersetzten README-Dateien
```

```
├── tests/                              die Tests
└── config.exemple.json                 die Beispielkonfiguration
~/.local/bin/dibab                  ein Link auf dibab.py: der Befehl
~/.config/dibab/config.json         Ihre Konfiguration, nie versioniert
~/.local/state/dibab/               der Zustand, das Journal und die Sitzungsberichte (Standard
↪ ort)
~/Classement/Personnel/                 der Tresor des privaten Bereichs
~/Classement/Professionnel/             der Tresor des beruflichen Bereichs
```

Installation aus dem Debian-Paket

Unter Debian 12 und 13 sowie unter Ubuntu 22.04 und 24.04 installiert eine einzige Datei das Programm und seine Abhängigkeiten (`python3`, `poppler-utils`, `util-linux`, `openssh-client`). Sie enthält weder Ollama noch ein Modell noch eine Konfiguration. Laden Sie `dibab_0.9.0-alpha_all.deb` von der Download-Seite von dibab.davalan.fr herunter und führen Sie dann in dem Ordner, der die Datei enthält, Folgendes aus:

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

Schreiben Sie die Konfiguration in der Sprache Ihrer Sitzung und folgen Sie dann den Schritten 4 und 5 der Installation aus den Quellen weiter unten (der Tresor jeder Domäne, die Schlüssel der Konfiguration):

```
dibab --initialiser
```

`--initialiser` schreibt die Konfiguration in der Sprache Ihrer Sitzung: der Prüfordner, sein Präfix und der Ordner für Duplikate erhalten die Namen dieser Sprache. Es legt die Datei mit Modus `600` in einem Ordner mit Modus `700` an und überschreibt nie eine vorhandene Datei; eine spätere Sitzung in einer anderen Sprache benennt sie nicht um.

Prüfen Sie die Installation, ohne etwas zu schreiben:

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

Das Paket legt den Befehl unter `/usr/bin/dibab`, das Programm unter `/usr/lib/dibab/` sowie die README-Dateien, die Beispielkonfiguration und die Handbuchseite (`man dibab`, in fünf Sprachen) unter `/usr/share/` ab. Das Entfernen schont Ihre Daten: `sudo apt remove dibab` und `sudo apt purge dibab` löschen die Dateien des Pakets und rühren Ihre Konfiguration, Ihren Zustand, Ihr Journal und Ihre Dokumente nie an; das Purge nennt, was es zurücklässt.

Stufe: Installation und Diagnose im Container validiert unter Debian 12 und 13 sowie unter Ubuntu 22.04 und 24.04 (Installation, `--version`, `--diagnose`, die Weigerung, ohne Konfiguration zu schreiben, und die Weigerung, in einen nicht eingehängten Tresor zu schreiben, eine simulierte Ablage, Entfernen und Purge). Ein Container zeigt weder das Einhängen eines verschlüsselten Tresors noch einen Lauf mit Ihrem eigenen Ollama und Ihren eigenen Dokumenten.

Installation unter Fedora, openSUSE Leap und Arch

Unter Fedora 44, openSUSE Leap 16.0 und Arch Linux installiert ebenfalls eine einzige Datei das Programm und seine Abhängigkeiten. Laden Sie die Datei für Ihr System von der Download-Seite von dibab.davalan.fr herunter und führen Sie dann in dem Ordner, der die Datei enthält, die Zeile für Ihr System aus:

```
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
```

Die Pakete sind nicht signiert. Der Rest ist derselbe wie beim Debian-Paket: `dibab --initialiser`, dann `dibab --version` und `dibab --diagnose`. Das Entfernen des Pakets (`sudo dnf remove dibab`, `sudo zypper remove dibab`, `sudo pacman -R dibab`) rührt Ihre Konfiguration, Ihren Zustand, Ihr Journal und Ihre Dokumente nie an.

Stufe: Installation und Diagnose im Container validiert unter Fedora 44, openSUSE Leap 16.0 und Arch Linux (Installation, `--version`, `--diagnose`, die beiden Weigerungen zu schreiben, eine simulierte Ablage, Neuinstallation über eine defekte Installation, Entfernen). Ein Container zeigt weder das Einhängen eines verschlüsselten Tresors noch einen Lauf mit Ihrem eigenen Ollama und Ihren eigenen Dokumenten.

Aus den Quellen installieren

1. Holen Sie die Quellen:

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

2. Machen Sie den Befehl verfügbar. Der Einstiegspunkt ist ausführbar; ein Link in `~/.local/bin` gibt ihm den Namen `dibab` (auf den meisten Distributionen liegt dieses Verzeichnis schon im `PATH` einer Login-Shell):

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

3. Schreiben Sie die Konfiguration in der Sprache Ihrer Sitzung (Kapitel 3 erklärt jeden Schlüssel):

```
dibab --initialiser
```

`--initialiser` schreibt die Konfiguration in der Sprache Ihrer Sitzung: der Prüfordner, sein Präfix und der Ordner für Duplikate erhalten die Namen dieser Sprache. Es legt die Datei mit Modus `600` in einem Ordner mit Modus `700` an und überschreibt nie eine vorhandene Datei; eine spätere Sitzung in einer anderen Sprache benennt sie nicht um.

4. Legen Sie den Tresor jedes Bereichs an und lesen Sie den Typ seines Dateisystems (Kapitel 5 erklärt, warum):

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

```
ext4
```

Der Befehl gibt den Typ des Datenträgers aus, der das Verzeichnis enthält, zum Beispiel `ext4`. Tragen Sie diesen Typ bei einem gewöhnlichen Verzeichnis in `destinations.<domain>.systemes_fichiers_acceptes` ein, oder öffnen Sie zuerst Ihren verschlüsselten Datenträger und tragen Sie den Typ ein, den `findmnt` dafür meldet.

5. Bearbeiten Sie `~/.config/dibab/config.json`, mindestens die in Kapitel 3 als erforderlich gekennzeichneten Schlüssel, und prüfen Sie dann die Installation, ohne etwas zu schreiben:

```
dibab --version
```

```
dibab 0.9.0-alpha
```

Der Befehl gibt eine Zeile aus und endet. Er liest keine Konfiguration, berührt keine Datei und spricht mit keinem Dienst.

Die Tests ausführen (optional)

Die Quellen bringen ihre eigenen Tests mit, die `pytest` in einer virtuellen Umgebung brauchen. Sie sind für den Betrieb von Dibab nicht nötig; führen Sie sie nach einem Update aus oder wenn Sie die Installation anzweifeln:

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

Die letzte Zeile meldet die bestandenen und die übersprungenen Tests; die übersprungenen sind die Versuche gegen ein laufendes Ollama, die die Testsuite nicht verlangt.

Wo die Konfiguration gesucht wird

Dibab liest die erste gefundene Datei aus der folgenden Reihenfolge: die Option `--config <path>`, die Umgebungsvariable `DIBAB_CONFIG`, `$XDG_CONFIG_HOME/dibab/config.json` (`~/.config/dibab/config.json`, wenn die Variable leer ist), dann `config.json` neben `dibab.py`. Es gibt keine Zusammenführung: Die erste gefundene Datei ist die einzige, die gelesen wird. Eine Option oder Variable, die eine fehlende Datei benennt, beendet das Programm mit dem Exit-Code 1. Es weicht nie auf eine andere Datei aus, denn sonst würde es mit einer Konfiguration arbeiten, die Sie nicht bestimmt haben, also mit einem anderen Tresor.

Konfiguration

In Dibab ist nichts fest einprogrammiert. Jeder Pfad, jeder Modellname, jeder Schwellenwert, jede Frist und jede Benennungsregel steht in einer einzigen JSON-Datei, `config.json`. Das Verhalten des Werkzeugs zu ändern heißt, diese Datei zu ändern, nicht den Code. Die Quellen enthalten ein vollständiges, realistisches Beispiel, `config.exemple.json`; dieses Kapitel erklärt es Block für Block. Kapitel 2 sagt, wo das Programm die Datei sucht.

Pfade mit ~ oder `$HOME` werden aufgelöst. Das Beispiel enthält keinen absoluten Pfad. In diese Datei wird nie ein Geheimnis geschrieben: Ein Mail-Passwort gehört in eine Umgebungsvariable, ein Google-Token in eine eigene Datei mit dem Modus `600` (Kapitel 7).

Ein fehlender Schlüssel wird nicht stillschweigend durch eine Vermutung ersetzt. Das Programm bricht mit dem Exit-Code 1 ab und nennt die fehlenden Schlüssel; `dibab --diagnose` (Kapitel 9) ist also das Erste, was Sie nach einer Änderung ausführen.

Die Schlüssel, die Sie schreiben müssen

Öffnen Sie die Datei, die `dibab --initialiser` geschrieben hat, und ändern Sie dann mindestens diese:

Schlüssel	Was einzutragen ist
`destinations.personnel.racine`, `destinations.professionnel.racine`	die Wurzel des Tresors jedes Bereichs
`destinations.*.systemes_fichiers_acceptes`	der Dateisystemtyp, an dem sich erkennen lässt, dass der Tresor eingehängt ist (Kapitel 5)
`personnes`	die Vornamen der Haushaltsmitglieder; sie benennen die Ordner des privaten Bereichs
`sources_locales[].chemin`	die zu durchsuchenden Verzeichnisse
`ollama.roles`	für jede Rolle die bevorzugten Modelle, der Reihe nach (`config.exemple.json` lässt die drei Listen leer)
`suivi.fichier_date`, `suivi.journal`, `suivi.sessions`	optional, aber innerhalb des Tresors empfohlen (Kapitel 5)

Löschen Sie die Beispieleinträge von `sources_distantes` und `sources_imap`, die Sie nicht verwenden: Eine deklarierte Quelle ist eine Quelle, die gelesen wird.

`destinations`: der Tresor jedes Bereichs

Ein Eintrag pro Bereich, `personnel` und `professionnel`.

Schlüssel	Bedeutung
`racine`	die Wurzel des Tresors dieses Bereichs
`systemes_fichiers_acceptes`	die Dateisystemtypen, an denen sich erkennen lässt, dass der Tresor eingehängt ist; das Beispiel erwartet `fuse.gocryptfs`

Schlüssel	Bedeutung
sous_repertoire_classement	der Ordner unterhalb der Wurzel, in dem die Dokumente dieses Bereichs abgelegt werden (Administratif für den privaten Bereich, Classement für den beruflichen)
sous_repertoire_a_verifier	der Ordner für die Dokumente, bei denen Dibab nicht sicher ist (ZU_PRÜFEN/)

Der Name des Prüfordners und sein Präfix werden bei der Initialisierung in der Sprache der Sitzung geschrieben: sous_repertoire_a_verifier setzt den Ordner, nommage.prefixe_a_verifier das Präfix. Der Ordner an der Wurzel eines Tresors, der Duplikate aufnimmt, wird durch destination.repertoire_doublon gesetzt.

destination, arborescence, executer_sur

Schlüssel	Bedeutung
destination.sous_repertoire_logs	der Ordner der Fachprotokolle innerhalb eines Tresors (logs); die Inventurpläne werden dort geschrieben
destination.vide_poches_prefixe	das Präfix der „Sammelordner“ (vide-poches-), in denen abgeholte Dateien warten
destination.repertoire_doublon	der Ordner an der Wurzel eines Tresors, der ein Dokument aufnimmt, dessen Fingerabdruck schon abgelegt ist (Duplikat/)
arborescence.personnel	der Unterpfad eines privaten Dokuments unter seinem Ablageordner: {personne}/{annee} ergibt Alice/2026
arborescence.professionnel	dasselbe für ein berufliches: {annee}/{categorie} ergibt 2026/Factures
arborescence.personne_inconnue	der Ordner, der verwendet wird, wenn die Person nicht in personnes steht (Commun)
executer_sur	die Maschinennamen (hostname), auf denen --execute erlaubt ist; eine leere Liste bedeutet jede Maschine, und --execute anderswo endet mit dem Exit-Code 4

suivi: Zustand, Journal und Berichte

Schlüssel	Bedeutung
fichier_date	die Datei dernier_scan.json, die das Datum des letzten Durchlaufs jeder Quelle festhält
journal	das Aktionsjournal journal.jsonl, eine JSON-Zeile pro Aktion, von --rollback zurückgelesen; es enthält Pfade, nie den Text eines Dokuments
sessions	der Ordner der Sitzungsberichte

Schlüssel	Bedeutung
logs_systeme	für ein technisches Protokoll reserviert; heute schreibt nichts hinein
securite_jours	um wie viele Tage vor dem letzten Durchlauf ein neuer Durchlauf neu ansetzt (1), damit eine parallel hinzugefügte Datei nicht übersehen wird
premier_scan_jours	wie weit der allererste Durchlauf einer Quelle zurückreicht (90 Tage)
rotation.journal_taille_max_mo	die Größe, ab der das Journal rotiert wird (20 MB)
rotation.journal_archives_max	wie viele rotierte Journale aufbewahrt werden (5)
rotation.sessions_max	wie viele Sitzungsdateien aufbewahrt werden (100)

Die vier Pfadschlüssel (fichier_date, journal, sessions, logs_systeme) sind optional. Ohne sie liegen die Dateien unter $XDG_STATE_HOME/dibab/, also ~/.local/state/dibab/, als dernier_scan.json, journal.jsonl, sessions/ und dibab.log, mit dem Modus 600 für Dateien und 700 für Ordner. Diese Standardorte liegen unverschlüsselt auf dem Datenträger: Kapitel 5 erklärt, wie man sie in einen Tresor verlegt. Drei weitere Dateien folgen fichier_date, wo immer es liegt: rapatries.json, imap_deposes.json und dibab.verrou.

ollama: die Modelle

Schlüssel	Bedeutung
url	die Adresse von Ollama (http://localhost:11434); jeder Aufruf ist eine einfache HTTP-Anfrage an sie
roles.ocr	die geordneten Präferenzen des Modells, das eingescannte Seiten liest
roles.tri_rapide	die geordneten Präferenzen des Modells, das aussondert, was nicht zur Verwaltung gehört
roles.classification	die Modelle, die einordnen; sie laufen alle bei jedem Dokument und stimmen ab
roles.tri_decision	optional: die geordneten Präferenzen eines decision-Modells (ollama show nennt die Fähigkeit), das in einem Durchgang mit Schwellen sortiert, anstelle von tri_rapide
roles.objet	optional: das kleine Generierungsmodell, das den Gegenstand eines Dokuments angibt, wenn das Entscheidungsmodell es sortiert
seuil_confiance	zwischen 0 und 1; ein Bereich wird nur behalten, wenn der Anteil der übereinstimmenden Klassifikationsmodelle ihn erreicht (0.4 im Beispiel)
timeout_secondes	die Frist jeder Anfrage an Ollama (120)
keep_alive_entre_pages	wie lange das OCR-Modell zwischen zwei Seiten eines Dokuments geladen bleibt (2m)

Beim Start fragt Dibab Ollama nach den installierten Modellen und behält für ocr und tri_rapide die erste installierte Präferenz; ein Name ohne Tag steht für :latest. Für classification behält es jede installierte Präferenz, und sie stimmen ab. Eine Rolle ohne installiertes Modell beendet das Programm mit dem Exit-Code 3, listet die vorhandenen Modelle auf und nennt die zu korrigierende Zeile, bevor irgendetwas geschrieben wird. Die beiden optionalen Rollen halten es nie an: Wenn tri_decision deklariert, aber unbrauchbar ist (kein Modell installiert, keine Fähigkeit decision, keine Schwellen, kein Modell für objet oder ein Ollama, das /v1/systemone nicht kennt), läuft die Sortierung über tri_rapide, und der Kopf der Sitzung sagt warum.

extraction und classification

Schlüssel	Bedeutung
extraction.seuil_caracteres_alphanumeriques	unterhalb dieser Anzahl von Buchstaben und Ziffern wird ein PDF vom OCR-Modell statt aus seiner Textebene gelesen (200)
extraction.pages_ocr	wie viele Seiten die OCR liest (3)
extraction.resolution_ocr	die Auflösung der gerenderten Seiten in dpi (200)
extraction.taille_max_texte	der extrahierte Text wird auf diese Zeichenzahl gekürzt (20000)
extraction.repertoire_temporaire	wo die Seiten gerendert werden; halten Sie es im Arbeitsspeicher (/dev/shm)
extraction.timeout_outils_secondes	die Frist von pdftotext und pdftoppm (60)
extraction.consigne_ocr	die Anweisung, die mit jedem Bild an das OCR-Modell geht
classification.temperature, classification.num_ctx	die mit jeder Anfrage gesendeten Optionen (0 und 8192); ein zu kurzer Kontext schneidet den Anfang der Anweisung ab
classification.caracteres_max_prompt	wie viel vom Text an die Klassifikationsmodelle geht (12000)
classification.caracteres_max_tri	wie viel vom Text an das Sortiermodell geht (3000); der Anfang eines Dokuments sagt, worum es geht
classification.keep_alive_entre_documents	wie lange ein Modell zwischen zwei Dokumenten geladen bleibt (2m)
classification.longueur_max_champ, longueur_max_citation	die größte Länge des behaltenen Absenders, Empfängers und Gegenstands sowie eines zitierten Auszugs (80 und 120)
classification.marqueurs_mentions	Wörter, deren Vorkommen ein schwaches berufliches Signal ist (TVA, SIRET, …)

Schlüssel	Bedeutung
`classification.seuils_tri_decision`	optional: für jedes Entscheidungsmodell `non_administratif` und `administratif` (zwischen 0 und 1) und `caracteres_min`; die vom Modell gelieferte Wahrscheinlichkeit für nicht administrativ bedeutet nicht administrativ ab der ersten, administrativ unter der zweiten, unsicher dazwischen, und ein Text mit weniger als `caracteres_min` Buchstaben und Ziffern ist unsicher. Die Schwellen gelten für ein Modell und ein `caracteres_max_tri`: messen Sie sie neu, wenn sich eines davon ändert

Der Text eines Dokuments wird als Daten behandelt, nie als Anweisung: Er wird durch eine Zufallsmarke begrenzt, das Modell antwortet mit einem geschlossenen JSON, das Dibab prüft, und jede ungültige Antwort zählt als „unsicher“. Ein Modell kann nie eine Aktion auslösen.

`categories`, `personnes`, `emetteurs_connus`, `pieces_communes`

Schlüssel	Bedeutung
`categories`	die geschlossene Liste der Kategorien (`Factures`, `Aides`, `Impôts`, …); ein Modell darf nur mit einer davon antworten oder mit „keine“
`personnes`	die Haushaltsmitglieder; ein Modell darf nur mit einem von ihnen antworten oder mit „keiner“
`emetteurs_connus`	bekannte Absender je Kategorie, jeweils `{"nom": ..., "domaine": "personnel" | "professionnel" | "mixte"}`; ein bekannter Absender ist ein Bereichssignal, und eine leere Liste liefert kein solches Signal
`pieces_communes`	die Kategorien, die in den Tresor jedes Bereichs kopiert werden, mit demselben Namen und demselben Fingerabdruck (`Identité`, `Domicile`, `RIB`), nur wenn jedes Klassifikationsmodell genau diese Kategorie liefert

Die Kategorien und die bekannten Absender des Beispiels sind generisch; passen Sie sie an Ihr Land an.

`nommage`: der Name eines abgelegten Dokuments

Schlüssel	Bedeutung
`format`	die Vorlage eines abgelegten Dokuments mit `{date}`, `{emetteur}`, `{categorie}`, `{personne}`, `{objet}` (der von der Sortierung gelieferte Gegenstand), `{ext}` (`{date}_{emetteur}_{categorie}{ext}`)
`separateur`, `espaces_vers`	das Trennzeichen und der Ersatz für Leerzeichen (_ und -)
`date_format`	das Datumsformat (`%Y-%m-%d`)
`date_inconnue_prefixe`	das Präfix des Änderungsdatums, das verwendet wird, wenn im Dokument kein Datum steht (~)
`emetteur_inconnu`	der Name, der bei unbekanntem Absender verwendet wird (`Inconnu`)
`prefixe_a_verifier`	das Präfix eines Dokuments im Prüfordner (`[ZU_PRÜFEN]`)
`format_a_verifier`	die Vorlage nach dem Präfix, mit denselben Variablen und `{origine}`, dem ursprünglichen Namen; ein leerer oder unbekannter Teil entfällt samt Trennzeichen, und der ganze Name bleibt bei höchstens 180 Zeichen (`{date}_{emetteur}_{objet}_{origine}{ext}`); ohne diesen Schlüssel behält das Dokument seinen ursprünglichen Namen
`suffixe_collision`	das Suffix eines schon vergebenen Namens, ab 2 (`_{n}`)

Mit der Konfiguration, die `dibab --initialiser` schreibt, wird ein Bescheid einer öffentlichen Stelle vom 15. März 2026 zu `2026-03-15_Organisme-public_Aides.pdf`, und ein Schreiben ohne Briefkopf im Prüfordner wird zu `[ZU_PRÜFEN]courrier-de-suivi-de-dossier_courrier-sans-en-tete.pdf`.

`extensions_cibles` und `scan`

Schlüssel	Bedeutung
`extensions_cibles`	die berücksichtigten Endungen (`.pdf`, `.png`, `.jpg`, `.jpeg`), ohne Beachtung der Groß- und Kleinschreibung verglichen
`scan.recursif_par_defaut`	Unterordner durchsuchen (`true`); jede Quelle kann das überschreiben
`scan.exclusions_automatiques`	Unterordner, die für alle Quellen nie durchsucht werden (`rebut`, und der durch `destination.repertoire_doublon` benannte Ordner)
`scan.suivre_symlinks`	symbolischen Links folgen (`false`, damit der Durchlauf seinen Umfang nicht verlassen kann)
`scan.ignorer_dotfiles`	versteckte Dateien und Ordner ignorieren (`true`)

`sources_locales`

Jeder Eintrag ist ein zu durchsuchendes Verzeichnis.

Schlüssel	Bedeutung
`id`	ein eindeutiger Name; er ist der Schlüssel der Quelle in `dernier_scan.json`
`chemin`	das Verzeichnis
`date_reference`	`modification` (empfohlen) oder `creation`; das Datum, nach dem aktuelle Dateien behalten werden
`recursif`	überschreibt `scan.recursif_par_defaut` für diese Quelle
`exclusions`	Namen von Unterordnern, die nur für diese Quelle übersprungen werden

Der erste Durchlauf einer Quelle reicht `suivi.premier_scan_jours` Tage zurück; die folgenden setzen beim Datum des letzten vollständigen Durchlaufs an, abzüglich `suivi.securite_jours`.

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

Diese Blöcke werden dort erklärt, wo sie gebraucht werden: `ssh` und `sources_distantes` in Kapitel 6, `imap`, `oauth2` und `sources_imap` in Kapitel 7.

`mesure` und `inventaire`

Schlüssel	Bedeutung
`inventaire.exclusions`	die Ordnernamen, die `--inventaire` nicht liest (Kapitel 8)
`mesure.graine`, `documents_par_coffre`, `exclusions`, `taux_erreur_max`	nur vom Messwerkzeug der Quellen (`_outils/mesurer_domaines.py`) gelesen, das abgelegte Dokumente stichprobenartig prüft und die Modelle am Tresor misst; die Werte des Beispiels sind ein Ausgangspunkt

Erster Durchlauf

Ein erster Durchlauf hat vier Schritte: die Dokumente in ein Quellverzeichnis legen, mit `--dry-run` simulieren, lesen, was die Simulation vorschlägt, und dann mit `--execute` wirklich ablegen. Jede Ablage lässt sich mit `--rollback` rückgängig machen. Beginnen Sie immer mit der Simulation: Sie verschiebt nichts, schreibt in keinen Tresor und zeigt genau, wohin jedes Dokument käme.

Dieses Kapitel verwendet das Quellverzeichnis `~/Classement/Entrant` aus der Konfiguration des Übungsbereichs, das vier erfundene Dokumente enthält: einen Bescheid einer öffentlichen Stelle, eine Hosting-Rechnung, ein Kuchenrezept und einen Brief ohne jeden Briefkopf.

```
ls ~/Classement/Entrant

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

Simulieren: `--dry-run`

`--dry-run` ist der Standardmodus: `dibab` allein tut dasselbe. Er braucht Ollama, weil die ganze Klassifikation läuft, aber er verschiebt keine Datei und schreibt weder `dernier_scan.json` noch ein abgelegtes Dokument.

```
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/ZU_PRÜFEN/[ZU_PRÜFEN]courrier-de-s
       ↪ uivi-de-dossier_courrier-sans-en-tete.pdf (not moved)
```

Die Fortschrittszeilen (`model 1/1, document 1/3`) gehen auf die Fehlerausgabe, eine pro Modell und pro Dokument. Der erste Durchlauf ist langsam, da jedes Klassifikationsmodell jedes Dokument liest, und diese Zeilen zeigen Ihnen, dass das Programm nicht hängt.

Den Bericht lesen

Der Bericht hat vier Teile, in dieser Reihenfolge.

1. Der Kopf: die Version und die Uhrzeit der Sitzung, der Modus, die gelesene Konfigurationsdatei, jeder Tresor mit der Prüfung seines Einhängens, der Pfad des Sitzungsjournals, das für jede Rolle behaltene Modell und die Konsensregel.
2. Der Durchlauf: für jede Quelle die Kandidatendateien mit Größe und Datum, dann die Summen. Eine Datei ist Kandidat, wenn ihre Endung in `extensions_cibles` steht und ihr Datum aktuell genug ist.
3. Die Textextraktion: für jeden Kandidaten, wie sein Text gewonnen wurde (`text layer` oder `OCR` mit dem Modell, das ihn gelesen hat), wie viele Zeichen und Seiten es waren und wie lange es dauerte. Der Text selbst wird weder gezeigt noch geschrieben.
4. Die Klassifikation: ein Block pro Dokument. Die erste Zeile gibt die Sortierung an (Verwaltung, keine Verwaltung oder unsicher), den Bereich, den Dibab behält, und die Übereinstimmung, also wie viele der befragten Klassifikationsmodelle diesen Bereich liefern (`agreement 2/3`: zwei von drei Modellen). Die Zeile `subject:` sagt in wenigen Worten, was das Dokument ist, so wie das Sortiermodell es gelesen hat; sie ist nie ein Auszug aus dem Text. Für ein Dokument, das in den Prüfordner kommt, sagt die Zeile `reason:` nach einer geschlossenen Liste, warum. Die letzte Zeile, nach `->`, gibt das Ziel an.

Bei der Ausführung nennt eine letzte Zeile den Sitzungsbericht.

Ein Ziel liest sich `-> vault <domain>, <zone>: <path>`. Die Zone sagt, was geschehen wird:

Zone oder Text	Bedeutung
`classement`	das Dokument wird in der Ablagezone dieses Tresors unter seinem normierten Namen abgelegt
`a_verifier`	Dibab ist nicht sicher: Das Dokument kommt in den Prüfordner `ZU_PRÜFEN/`, unter dem Präfix `[ZU_PRÜFEN]`, gefolgt von Datum, Absender und Gegenstand, wenn das Dokument sie angibt, und von seinem ursprünglichen Namen; die Simulation fügt `(not moved)` hinzu, weil in der Ablagezone nichts abgelegt wird
`stays in place (not administrative)`	die Sortierung hat das Dokument ausgesondert; es wird weder verschoben noch umbenannt und bei jedem Durchlauf erneut aufgelistet
`(duplicate: would go to Duplikat/)`	der Fingerabdruck des Dokuments ist schon abgelegt; es kommt in den Ordner `Duplikat/`, statt zweimal abgelegt zu werden

Der Bereich `incertain` führt immer in den Prüfordner, ebenso ein Dokument, bei dem sich die Modelle nicht einig sind. Ein gemeinsames Stück (ein Ausweis, ein Meldebescheinigung, eine Bankverbindung) wird nur dann im Tresor jedes Bereichs vorgeschlagen, wenn jedes Klassifikationsmodell dieselbe Kategorie liefert; sonst kommt es in den Prüfordner. Im Zweifel bevorzugt Dibab eine menschliche Entscheidung vor einer riskanten Ablage.

Die Zuordnung ohne die Modelle prüfen

Um zu sehen, welches Ziel zu einem Urteil führt, ohne auf ein Modell zu warten, geben Sie `--dry-run` eine JSON-Datei, die einem Dateinamen ein Urteil zuordnet. Darauf wird die Zuordnungstabelle angewendet und das Ziel angezeigt. Die Option gibt es nur mit `--dry-run`: Das Programm lehnt sie zusammen mit `--execute` mit dem Exit-Code 2 ab.

```
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
```

Ablegen: `--execute`

Wenn die Simulation Ihnen passt, führen Sie denselben Befehl mit `--execute` aus. Jedes Dokument wird an sein Ziel kopiert, die Kopie wird an ihrem SHA256-Fingerabdruck geprüft, und erst dann wird das Original aus der Quelle entfernt. Nichts wird entfernt, bevor die Kopie als identisch feststeht.

```
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/ZU_PRÜFEN/[ZU_PRÜFEN]courrier-de-suivi-de-dossier_courrier-sans
       ↪  -en-tete.pdf

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

Bei der Ausführung endet jede Zeile mit der ausgeführten Aktion: `classe` (in der Ablagezone abgelegt), a `_verifier` (in den Prüfordner gelegt) oder `doublon` (in den Ordner `Duplikat/` gelegt). Diese Wörter sind in jeder Sprache dieselben, ebenso wie die im Journal geschriebenen Aktionen.

Die Dokumente liegen nun dort, wo der Bericht es sagte:

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

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

Drei Dateien haben `Entrant` verlassen. Das Rezept blieb: Es gehört nicht zur Verwaltung, und Dibab rührt nichts an, was es nicht ablegen sollte.

Was `--execute` schreibt

Jede Verschiebung wird in das Aktionsjournal `suivi.journal` geschrieben, eine JSON-Zeile pro Aktion:

```
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"}]}
```

Die Zeile enthält die Aktion und ihre Kennung, den Bereich, die Pfade und die Fingerabdrücke davor und danach, aber keinen Text des Dokuments. Jede Sitzung hinterlässt außerdem zwei Dateien im Ordner `suivi.sessions`: ein Sitzungsjournal (`session_<time>.jsonl`, dieselben Ereignisse wie auf dem Bildschirm) und einen Bericht (`rapport_<time>.json`) mit der Zahl jeder Aktion, den Fehlern und der Dauer, ohne irgendeinen Dokumentnamen. Das Datum des Durchlaufs jeder Quelle rückt in `dernier_scan.json` nur für die Quellen vor, die ohne Fehler gelesen wurden. Kapitel 8 erklärt, wie diese Dateien kurz gehalten werden.

Es läuft immer nur eine Ausführung. Ein zweites `--execute` oder `--rollback`, das die Sperre `dibab.verrou` belegt findet, endet mit dem Exit-Code 5 und rührt nichts an.

Rückgängig machen: `--rollback`

`--rollback` macht die letzte Ablagesitzung rückgängig. `--rollback --session <time>` macht eine bestimmte Sitzung rückgängig, deren Zeitangabe die im Kopf ausgegebene ist und in den Namen ihrer Dateien vorkommt.

```
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
```

Jedes Dokument kehrt an seinen ursprünglichen Platz zurück, die Kopie per Fingerabdruck geprüft, und sein Ziel wird entfernt. Dem Journal, das nie umgeschrieben wird, wird eine Zeile `annulation` hinzugefügt. Ein gemeinsames Stück wird zugleich aus dem Tresor jedes Bereichs entfernt. Eine bereits rückgängig gemachte Aktion wird nicht zweimal ausgeführt.

```
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
```

Lässt sich eine Aktion nicht rückgängig machen, wird dafür nichts entfernt, und der Grund wird mit dem Exit-Code 1 angezeigt. Der Grund lautet `destination modified since the filing` oder `destination not found`, wenn das Dokument von Hand geändert oder verschoben wurde (legen Sie es zurück oder lassen Sie es), oder `source occupied by another file`, wenn eine andere Datei den ursprünglichen Platz eingenommen hat (verschieben Sie sie und führen Sie den Befehl erneut aus).

Exit-Codes

Code	Bedeutung
`0`	Erfolg
`1`	Konfiguration nicht gefunden, ungültig oder ohne einen Schlüssel; mindestens eine Quelle oder ein Dokument fehlgeschlagen; Rückgängigmachen nur teilweise erfolgt; OAuth2-Einwilligung fehlgeschlagen; `--diagnose` mit mindestens einem `ÉCHEC`
`2`	Einhängen des Tresors vor einem schreibenden Modus nicht festgestellt; ungültige Befehlszeile
`3`	kein Modell für eine Rolle installiert, oder Ollama antwortet nicht

Code	Bedeutung
`4`	`--execute` auf dieser Maschine abgelehnt (Schlüssel `executer_sur`)
`5`	eine andere Ausführung hält die Sperre `dibab.verrou`

`--version` liefert `0`, oder `2` bei einer ungültigen Befehlszeile.

Tresor und Verschlüsselung

Was ein Tresor ist

Ein Tresor ist das Verzeichnis, das die Dokumente eines Bereichs aufnimmt: einer für den privaten Bereich, einer für den beruflichen (`destinations.personnel.racine` und `destinations.professionnel.racine`). Dibab legt ihn nicht an, hängt ihn nicht ein, verschlüsselt ihn nicht und synchronisiert ihn nicht. Es erwartet, dass das Verzeichnis da ist, und, weil die Dokumente sensibel sind, dass es auf einem verschlüsselten Datenträger liegt, den Sie vor dem Durchlauf öffnen.

Die Gefahr ist leicht beschrieben. Ist ein verschlüsselter Datenträger geschlossen, so ist sein Einhängepunkt nur ein leeres Verzeichnis auf dem unverschlüsselten Datenträger. Ein Programm, das dort schreibt, schreibt die Dokumente im Klartext, unterhalb der Verschlüsselung, ohne jeden Fehler. Dibab stellt deshalb vor jedem Schreiben fest, dass jeder Tresor wirklich eingehängt ist.

Die Prüfung des Einhängens

Vor `--execute`, `--rollback`, `--inventaire` oder `--autoriser-google` führt Dibab für die Wurzel jedes Tresors `findmnt -n -T <root> -o FSTYPE` aus und vergleicht den Dateisystemtyp mit der Liste `systemes_fichiers_acceptes` dieses Tresors. Es prüft auch, dass die Wurzel beschreibbar ist. Steht der Typ nicht in der Liste, so ist das Einhängen des Tresors nicht festgestellt, und das Programm endet mit dem Exit-Code 2, ohne etwas anzurühren.

Die Beispielkonfiguration erwartet `fuse.gocryptfs`, den Typ, den `findmnt` für einen eingehängten gocryptfs-Datenträger meldet. Steht stattdessen ein gewöhnliches Verzeichnis dort, so sieht ein geschlossener Tresor aus, und die Ablehnung erfolgt sofort:

```
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
```

Die Simulation hält bei einem Tresor, dessen Einhängen nicht festgestellt ist, nicht an: Sie meldet es mit `[!]` im Kopf und läuft weiter, damit Sie die Vorschläge lesen können, bevor Sie den Tresor öffnen. Sie schreibt ihr Sitzungsjournal nicht in einen Pfad, den der Tresor enthält, denn das würde das Journal im Klartext auf den Datenträger legen.

Die zu akzeptierenden Typen wählen

Lesen Sie den Typ Ihres eigenen Datenträgers, sobald er geöffnet ist, mit dem Befehl aus Kapitel 2 und tragen Sie ihn in die Liste ein:

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

Verschlüsseln Sie anders (LUKS, eCryptfs, ein Datenträger eines anderen Werkzeugs), öffnen Sie den Datenträger und tragen Sie den Typ ein, den `findmnt` dafür liefert. Auf einem unverschlüsselten Datenträger kann die Prüfung ein fehlendes Einhängen nicht mehr von einem vorhandenen unterscheiden: Sie verzichten dann auf diesen Schutz, und die Diagnose aus Kapitel 9 erinnert Sie daran, dass der Zustand im Klartext liegt.

Ein Beispiel mit gocryptfs

gocryptfs gehört nicht zu Dibab, und nichts in Dibab verlangt es. Es ist eine Möglichkeit, den Tresor jedes Bereichs aufzubauen. Legen Sie jedes verschlüsselte Verzeichnis einmal an und wählen Sie dabei eine Passphrase, die Sie außerhalb der Maschine aufbewahren; der Befehl fragt danach.

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

Öffnen Sie jeden Tresor vor einem Durchlauf. Der Befehl fragt erneut nach der Passphrase und hängt die Klartextsicht in das in der Konfiguration genannte Verzeichnis ein:

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

Schließen Sie sie nach dem Durchlauf:

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

Die verschlüsselten Verzeichnisse `~/Chiffre/...` sind das, was Sie sichern und synchronisieren. Dibab weiß nichts davon.

Wo der Zustand liegt

Neben den Dokumenten führt Dibab einen Zustand: das Datum des letzten Durchlaufs jeder Quelle, das Aktionsjournal, die Sitzungsjournale und -berichte, das Verzeichnis der abgeholten Dateien und die Sperre. Ohne einen Pfad `suivi.*` in der Konfiguration liegt das alles unter `~/.local/state/dibab/` (oder `$XDG_STATE_HOME/dibab/`), im Klartext, mit dem Modus `700` für Ordner und `600` für Dateien. Das Journal enthält Dateinamen und Absender; das verrät schon einiges. Die Diagnose meldet es als Warnung: `AVERTISSEMENT emplacement_etat`.

Um den Zustand zu verschlüsseln, legen Sie ihn in einen Tresor, mit drei Schlüsseln des Blocks `suivi`:

```
"suivi": {
  "fichier_date": "~/Classement/Personnel/logs/dernier_scan.json",
```

```
  "journal": "~/Classement/Personnel/logs/journal.jsonl",
  "sessions": "~/Classement/Personnel/logs/sessions"
}
```

Die übrigen Dateien des Zustands (`rapatries.json`, `imap_deposes.json`, `dibab.verrou`) folgen `fichier_date`. Mit dieser Einstellung meldet die Diagnose keine Warnung mehr.

Platz auf dem Datenträger

Dibab verschiebt und kopiert und löscht nie. Ein abgelegtes Dokument wird an sein Ziel kopiert, geprüft und dann aus der Quelle entfernt, der Platzbedarf auf dem Datenträger wächst also mit einer Ablage nicht. Ein Dokument, das ein Duplikat eines bereits abgelegten ist, kommt in den Ordner `Duplikat/` des Tresors, statt zweimal abgelegt zu werden; diesen Ordner zu leeren ist Ihre Entscheidung, nie die des Werkzeugs.

Quellen über ssh

Eine entfernte Maschine des Haushalts wird über `ssh` gelesen. Dibab listet die Dateien der Verzeichnisse auf, die Sie nennen, kopiert sie in einen Sammelordner des privaten Tresors, prüft jede Kopie an ihrem Fingerabdruck und legt sie dann wie lokale Dokumente ab. Die Originaldateien bleiben auf der entfernten Maschine unberührt: Dibab entfernt oder verändert dort nie etwas.

Was die entfernte Maschine bieten muss

- Ein Linux-System mit GNU `find` (die Auflistung verwendet `-printf`), `sha256sum` und `cat`.
- Einen `ssh`-Alias, der eine Sitzung ohne Passwort öffnet. Alles zur Verbindung, die Adresse, der Benutzer und der Schlüssel, steht unter dem Alias in der `~/.ssh/config` der lokalen Maschine; Dibab weiß davon nichts. Zum Beispiel mit einer erfundenen Maschine:

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

Prüfen Sie den Alias vor dem ersten Durchlauf von Hand. Der Befehl darf nichts ausgeben und muss `0` zurückgeben:

```
ssh machine1 true
```

Die Maschine deklarieren

Zwei Blöcke von `config.json` sind beteiligt: `ssh`, der die Fristen angibt, und `sources_distantes`, der die Maschinen auflistet.

```
"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": []
  }
]
```

Schlüssel	Bedeutung
`ssh.delai_connexion_secondes`	die Frist zum Öffnen der Verbindung (`10`)

Schlüssel	Bedeutung
`ssh.delai_commande_secondes`	die Frist einer Auflistung oder eines Fingerabdrucks (`120`)
`ssh.delai_copie_secondes`	die Frist zum Kopieren einer Datei (`900`)
`id`	ein eindeutiger Name; er ist der Schlüssel der Maschine in `dernier_scan.json`
`alias_ssh`	der Alias aus `~/.ssh/config`
`vide_poches_destination`	der Name des Sammelordners, ohne Trennzeichen, der unter der Wurzel des privaten Tresors angelegt wird; die abgeholten Dateien kommen dort an
`repertoires`	die auf der entfernten Maschine zu lesenden Verzeichnisse; `~` ist das Home-Verzeichnis des entfernten Benutzers
`date_reference`, `recursif`, `exclusions`	wie bei einer lokalen Quelle (Kapitel 3)

Die drei Fristen von `ssh` sind erforderlich, sobald eine entfernte Quelle deklariert ist.

Ein erster Durchlauf: auflisten, dann abholen

Die Konfiguration dieses Kapitels deklariert die Maschine und sonst nichts. Die Simulation listet die Dateien auf, ohne etwas zu kopieren:

```
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)
```

Der Block `Remote source` zeigt jede Datei mit dem Wort `remote`. Im Sammelordner wird nichts geschrieben. Bei einem ersten Durchlauf werden nur die Dateien aufgelistet, die in den letzten `suivi.premier_scan_jours` Tagen (standardmäßig 90) geändert wurden, und nur jene, deren Endung in `extensions_cibles` steht.

Die Ausführung holt die Dateien ab und legt sie dann ab:

```
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
```

Jede Datei wird nach `vide-poches-machine1` kopiert, zurückgelesen und per SHA256 mit dem auf der entfernten Maschine berechneten Fingerabdruck verglichen. Eine Kopie, die nicht übereinstimmt, wird nicht behalten. Der Durchlauf behandelt den Sammelordner dann wie eine lokale Quelle, ohne jeden Datumsfilter, denn ein Sammelordner leert sich mit dem Fortschreiten der Ablage: Was darin liegt, wartet. Das Verzeichnis `rapatries.json` neben `dernier_scan.json` merkt sich die abgeholten Dateien per Fingerabdruck.

Erneut ausführen

Eine zweite Ausführung holt nichts Neues ab. Die entfernte Datei wird erneut aufgelistet, aber ihr Fingerabdruck steht im Verzeichnis: Sie wird also weder kopiert noch ein zweites Mal abgelegt:

```
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
```

Das Datum der Maschine in `dernier_scan.json` rückt erst nach einer vollständigen Auflistung vor, in der jede Kopie geprüft wurde. Eine fehlgeschlagene Datei wird bei der nächsten Ausführung erneut abgeholt, ohne Duplikat. Eine Datei, die sowohl auf der entfernten Maschine als auch in einem lokalen Durchlauf vorkommt, kommt in den Ordner `Duplikat/`: Sie wird einmal abgelegt.

Wenn die Verbindung scheitert

Eine Ablehnung beendet die betroffene Maschine, und nur diese; die anderen Quellen laufen weiter. Dibab versucht es nicht in einer Schleife erneut, und Sie sollten das auch nicht tun.

Meldung	Wahrscheinliche Ursache
`Could not resolve hostname` oder `Permission denied`	die Maschine ist aus, oder der Alias oder der Schlüssel ist falsch; korrigieren Sie ihn und führen Sie dann einmal erneut aus
eine überschrittene Frist	erhöhen Sie `ssh.delai_commande_secondes` (Auflistung) oder `ssh.delai_copie_secondes` (Kopie)
`remote directory partly readable or absent`	korrigieren Sie `repertoires`
kein Kandidat, obwohl die Maschine Dateien hat	die Dateien sind älter als `suivi.premier_scan_jours`, oder ihre Endung steht nicht in `extensions_cibles`

IMAP-Postfächer und Google-Konten

Dibab liest die Anhänge eines Postfachs über IMAP, mit einem Passwort oder, bei einem Google-Konto, mit einem OAuth2-Token. Es liest, ohne zu verändern: Das Postfach wird schreibgeschützt geöffnet, keine Nachricht wird als gelesen markiert, kein Flag wird gesetzt, und auf dem Server wird nichts kopiert, verschoben oder gelöscht.

Was vom Server verlangt wird

Das Programm sendet dem Server nur `EXAMINE` (das schreibgeschützte Öffnen eines Ordners), `UID SEARCH ... SINCE` (die seit einem Datum empfangenen Nachrichten) und `UID FETCH` für das Empfangsdatum, die Struktur der Nachricht und jeweils einen Anhang mit `BODY.PEEK`, der Form, die die Nachricht nicht als gelesen markiert. Der Textkörper einer Nachricht wird nie angefordert. Die Verbindung ist immer TLS, und das Zertifikat wird gegen den Zertifikatsspeicher des Systems geprüft: Eine Quelle mit `"ssl": false` wird abgelehnt.

Ein Anhang ist ein Teil der Nachricht, der einen Namen trägt und dessen Disposition `attachment` ist, oder `inline` ohne `Content-ID` (die Bilder einer Signatur tragen eine und bleiben in der Nachricht). Seine Endung muss in `extensions_cibles` stehen. Sein Name sind Daten, die der Absender liefert: Er wird wie der Name einer abgeholten Datei dekodiert und bereinigt (kein Trennzeichen, kein Steuerzeichen, nie versteckt, begrenzte Länge) und gelangt weder in das Sitzungsjournal noch in den Bericht.

Ein Postfach deklarieren

```
"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"
  }
]
```

Schlüssel	Bedeutung
`imap.delai_secondes`	die Frist jedes Netzwerkaustauschs, Verbindung eingeschlossen (`60`)

Schlüssel	Bedeutung
`imap.taille_max_piece_mo`	ein Anhang, der nach der vom Server angekündigten Größe größer ist, wird nicht abgeholt (`25`)
`id`	ein eindeutiger Name; er ist der Schlüssel der Quelle in `dernier_scan.json`
`nom`	eine Bezeichnung
`serveur`, `port`, `utilisateur`	das Konto
`ssl`	muss `true` sein
`authentification`	`login` (Passwort in einer Umgebungsvariable) oder `oauth2` (Google-Konto); jeder andere Wert beendet die Quelle
`mot_de_passe_env`	der Name der Umgebungsvariable, die das Passwort enthält (nur `login`)
`dossiers`	die zu lesenden Ordner, jeweils schreibgeschützt geöffnet
`vide_poches_destination`	der Sammelordner, ein einzelner Name, unter der Wurzel des privaten Tresors
`date_reference`	`reception`: das vom Server aufgezeichnete Empfangsdatum, nie der Header `Date`, den der Absender schreibt

Die beiden Blöcke `imap` und `sources_imap` sind nur zusammen erforderlich; ein fehlender Schlüssel beendet das Programm mit dem Exit-Code 1 und nennt ihn.

Das Passwort

Das Passwort steht nie in `config.json`. Dibab liest es aus der Umgebungsvariable, die `mot_de_passe_env` nennt, und schreibt, protokolliert oder zeigt es nie. Definieren Sie sie in dem Terminal, das Dibab startet, ohne sie irgendwo zu notieren:

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

Eine Quelle, deren Variable nicht definiert ist, wird mit einer Warnung übersprungen, ohne jede Verbindung. Das geschieht auch mit dem Beispieleintrag aus `config.exemple.json`, solange Sie ihn deklariert lassen.

Ein erster Durchlauf: auflisten, dann ablegen

Die Konfiguration dieses Kapitels deklariert das Postfach und sonst nichts. Die Simulation listet die Anhänge auf, die abgeholt würden, mit dem Wort `imap`, und lädt keinen herunter:

```
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)
```

Die Ausführung lädt jeden Anhang herunter, prüft seinen Fingerabdruck, legt ihn im Sammelordner ab und legt ihn wie ein lokales Dokument ab:

```
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
```

Die Datei `imap_deposes.json` neben `dernier_scan.json` merkt sich, was abgelegt wurde (Ordner, `UIDVALIDITY`, `UID`, Teil und Fingerabdruck). Ein Anhang, dessen Fingerabdruck schon darin oder im Aktionsjournal steht, wird nicht erneut abgelegt:

```
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
```

Das Datum der Quelle in `dernier_scan.json` rückt erst nach einem vollständigen Lesen vor, bei dem jede Ablage gelungen ist.

Wenn die Verbindung scheitert

Eine abgelehnte Anmeldung oder eine fehlgeschlagene Verbindung beendet die Quelle, ohne zweiten Versuch. Die anderen Quellen laufen weiter.

Meldung	Wahrscheinliche Ursache
`environment variable ... not defined, source skipped`	exportieren Sie das Passwort in dem Terminal, das Dibab startet

Meldung	Wahrscheinliche Ursache
`authentication refused by the server`	das Passwort oder der Benutzer ist falsch; korrigieren Sie es und führen Sie dann einmal erneut aus, nicht in einer Schleife
`connection impossible`	der Server, der Port oder das Netz
`folder ... not found`	korrigieren Sie `dossiers`
`attachment(s) above imap.taille_max_piece_mo`	erhöhen Sie diesen Wert

Google-Konten mit OAuth2

Ein Google-Konto wird statt mit einem Passwort mit einem OAuth2-Token gelesen. Die Schritte in der Google-Konsole sind Ihre Sache, mit dem zu lesenden Google-Konto, und die Bezeichnungen der Konsole ändern sich mit der Zeit: Die Begriffe sind die ihrer OAuth2-Dokumentation.

1. Legen Sie in der Google-Cloud-Konsole ein Projekt an, aktivieren Sie die Gmail-API und konfigurieren Sie den OAuth-Zustimmungsbildschirm (Typ „Extern“, Ihre Adresse als Kontakt). Veröffentlichen Sie die Anwendung dann, damit ihr Status „In Produktion“ lautet. Beim Status „Test“ läuft die Autorisierung nach sieben Tagen ab.
2. Erstellen Sie eine OAuth-Anmeldedaten vom Typ „Desktop-App“ und laden Sie ihre JSON-Datei herunter.
3. Bewahren Sie diese Datei im Tresor auf, in einem Ordner mit dem Modus `700`, mit dem Modus `600`. Dibab verändert sie nie. Sie ist die `client_fichier`.
4. Deklarieren Sie die Quelle mit `"authentification": "oauth2"`, `client_fichier` und `token_fichier` anstelle von `mot_de_passe_env` und fügen Sie den Block `oauth2` hinzu:

```
"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"
  }
]
```

Schlüssel von `oauth2`	Bedeutung
`url_autorisation`, `url_jeton`	die Endpunkte von Google; sie müssen `https` sein, außer auf dem lokalen Loopback
`portee`	der OAuth-Scope (`https://mail.google.com/`)
`marge_renouvellement_secondes`	das Zugriffstoken wird erneuert, wenn ihm weniger als dieser Wert bleibt (`300`)
`delai_secondes`	die Frist jedes Austauschs mit dem Autorisierungsserver (`30`)
`delai_consentement_secondes`	wie lange `--autoriser-google` auf Ihre Antwort wartet (`300`)
`chemins_interdits`	Ordner, in denen ein Token abgelehnt wird (`~/Documents`)

5. Stimmen Sie einmal zu, in einem Terminal der Maschine, mit der `id` der Quelle:

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

Dibab gibt eine Adresse aus, die mit `https://accounts.google.com/` beginnt. Öffnen Sie sie in einem Browser. Akzeptieren Sie die Warnung „nicht verifizierte App“, die erscheint, weil die Anwendung Ihre eigene ist, und dann die Zustimmung. Das Programm empfängt die Antwort auf dem lokalen Loopback, prüft sie, tauscht sie gegen ein Token ein und schreibt die Token-Datei mit dem Modus `600` in einen Ordner mit dem Modus `700`. Die Token-Datei muss außerhalb jedes Git-Repositorys und außerhalb von `chemins_interdits` liegen; eine für die Gruppe oder andere lesbare Token-Datei beendet die Quelle. Wird die Zustimmung verweigert, kommt keine Antwort rechtzeitig, oder liefert Google kein Erneuerungstoken, wird nichts geschrieben; führen Sie den Befehl erneut aus.

Von da an läuft der Durchlauf wie bei einer Quelle mit Passwort, ohne jede Umgebungsvariable: Das Zugriffstoken erneuert sich selbst, und das Token erscheint weder im Journal noch auf dem Bildschirm. Eine Token-Datei, die nicht existiert, ist kein Fehler: Die Quelle wird übersprungen, mit dem auszuführenden Befehl. Widerruft Google die Autorisierung (Sie haben den Zugriff der Anwendung entfernt, das Passwort des Google-Kontos geändert, oder sechs Monate sind ohne Durchlauf vergangen), beendet Dibab die Quelle mit einer ausdrücklichen Meldung, und Sie führen `--autoriser-google` erneut aus.

Der Scope `https://mail.google.com/` ist der einzige, den Google für IMAP dokumentiert, und er erlaubt Lesen, Schreiben und Löschen. Dass das Postfach unverändert bleibt, ist durch das gewährleistet, was Dibab sendet, nämlich nur die am Anfang dieses Kapitels aufgelisteten Befehle, nicht durch den Scope. Kapitel 10 kommt darauf zurück.

Inventur und Rotation

Lesen, was ein Tresor schon enthält: `--inventaire`

Haben Sie bereits Dokumente in einem Tresor, von Hand oder von einem anderen Werkzeug abgelegt, so liest `--inventaire` sie und schreibt einen Migrationsplan, ohne etwas zu verschieben, umzubenennen oder zu löschen. Es ist schreibgeschützt: Vorher und nachher ist jeder Tresor identisch. Es braucht die Prüfung des Einhängens aus Kapitel 5, da jeder Plan in seinem Tresor geschrieben wird.

Damit es etwas zu lesen gibt, legen Sie die vier Dokumente des Übungsbereichs erneut ab; das Rückgängigmachen aus Kapitel 4 hat sie in ihre Quelle zurückgelegt:

```
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/ZU_PRÜFEN/[ZU_PRÜFEN]courrier-de-suivi-de-dossier_courrier-sans⌋
       ↪  -en-tete.pdf

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

Lesen Sie dann den Tresor jedes Bereichs:

```
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 ZU_PRÜFEN (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 ZU_PRÜFEN (not listed)
      plan written: ~/Classement/Professionnel/logs/inventaire_2026-11-23T12:09:03.json
```

Für jeden Tresor nennt der Bericht die Zahl der gelesenen Dokumente, wie viele schon in der Ablagezone liegen, wie viele außerhalb liegen und abgelegt werden sollten, wie viele Duplikate eines anderen Dokuments desselben Tresors sind und wie viele auch im anderen Tresor vorhanden sind. Der Plan ist eine

JSON-Datei, `logs/inventaire_<time>.json`, die in dem Tresor geschrieben wird, den sie beschreibt:

```
ls ~/Classement/Personnel/logs

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

Der Plan jedes Tresors nennt nur die Dateien dieses Tresors. Vom anderen Tresor behält er nur eine einzige Tatsache, „dieser Fingerabdruck ist auch dort“, sodass der Plan eines Tresors nie den Pfad einer Datei des anderen enthält.

Jedes Dokument des Plans erhält einen Vorschlag:

Vorschlag	Bedeutung
nichts zu tun	das Dokument liegt in der Ablagezone
in einen Sammelordner legen, dann den Dry-Run lesen	das Dokument liegt außerhalb der Ablagezone: Legen Sie es in einer Quelle ab und lassen Sie den nächsten Durchlauf es einordnen
identisch mit einem anderen Dokument: nach Prüfung in `Duplikat/` abzulegen	der Fingerabdruck ist schon vorhanden

Die in `inventaire.exclusions` aufgeführten Ordner (`logs`, `Archive`, `rebut`, und der durch `destination.repertoire_doublon` benannte Ordner, im Beispiel), der Prüfordner, die Sammelordner und die versteckten Ordner werden nicht gelesen.

Das Journal kurz halten: die Rotation

Zwei Dateien wachsen mit der Nutzung: das Aktionsjournal `suivi.journal` und der Ordner der Sitzungen `suivi.sessions`. Die Rotation wendet die Grenzen von `suivi.rotation` an, wenn `--execute` oder `--rollback` startet:

Schlüssel	Wirkung
`journal_taille_max_mo`	überschreitet das Journal diese Größe, wird es in `journal.1.jsonl` umbenannt, die älteren Archive rücken um eine Nummer weiter, und ein neues Journal beginnt
`journal_archives_max`	die Zahl der aufbewahrten Archive; das älteste wird entfernt, wenn ein neues sie überschreiten würde
`sessions_max`	die Zahl der aufbewahrten Sitzungsjournale und -berichte; die ältesten werden entfernt

Das ist die einzige Löschung, die Dibab vornimmt, und sie betrifft seine eigenen Journale, nie ein Dokument. Ein Archiv, das noch eine nicht rückgängig gemachte Aktion enthält, wird nie entfernt: Die Rotation wartet lieber, als zu verlieren, was `--rollback` lesen können muss. Machen Sie eine Ablage, an der Sie zweifeln, rückgängig, bevor ihr Journal alt wird.

Die Dateien des Zustands werden mit dem Modus `600` geschrieben und ihr Ordner mit dem Modus `700`. Wo sie aufbewahrt werden, ist die in Kapitel 5 beschriebene Wahl.

Diagnose

`--diagnose` prüft die Installation, die Konfiguration, Ollama und den Tresor jedes Bereichs, eine Zeile pro Prüfung, und schreibt nichts: Keine Datei wird angelegt, nichts wird verschoben, keine Sperre wird gesetzt. Es öffnet nur eine einzige Verbindung, zu Ollama, um die Liste der Modelle zu lesen. Es öffnet keine zu einer entfernten Maschine oder einem Postfach, sodass man es jederzeit gefahrlos ausführen kann. Es spielt jede Prüfung bis zum Ende durch, was auch immer die vorherigen ergeben haben.

```
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
```

Jede Zeile liest sich `<STATE> <identifier> : <message>`, dann schließt eine Zusammenfassung den Bericht ab. Die Meldung steht in der Sprache Ihrer Sitzung; die Zustände und die Bezeichner ändern sich nie mit der Sprache, damit ein Skript sie vergleichen kann.

Die Zustände

Zustand	Bedeutung
`OK`	die Prüfung ist bestanden
`AVERTISSEMENT`	ein Mangel, der keinen Modus verhindert, oder ein Mangel einer Komponente, die diese Instanz nicht nutzt
`ÉCHEC`	die Prüfung ist an etwas gescheitert, das die vorgefundene Installation oder Konfiguration verlangt
`NON_CONFIGURÉ`	die Konfiguration, die die Prüfung liest, existiert noch nicht
`NON_VÉRIFIÉ`	die Prüfung hängt von einer anderen ab, die nicht bestanden wurde, oder von einer fehlenden Konfiguration

Der Exit-Code ist `0`, wenn es keinen `ÉCHEC` gibt, und `1`, wenn es mindestens einen gibt. Eine Warnung, eine fehlende Konfiguration und ein ungeprüfter Punkt ändern ihn nicht.

Die Prüfungen in der Reihenfolge

Bezeichner	Was geprüft wird	Wenn es nicht OK ist
python	der Interpreter ist 3.10 oder neuer	darunter ÉCHEC
outil_pdftotext, outil_pdftoppm, outil_findmnt	das Werkzeug ist im PATH ausführbar	ÉCHEC, wenn es fehlt
outil_ssh	ssh ist im PATH ausführbar	ÉCHEC, wenn die Konfiguration sources_distantes deklariert, sonst AVERTISSEMENT
configuration	eine Konfiguration wird gefunden (Kapitel 2) und ist gültig: lesbares JSON, die Schlüssel, die jeder Modus verlangt	keine gefunden: NON_CONFIGURÉ, und die Meldung listet die vier versuchten Orte auf; gefunden, aber ungültig: ÉCHEC, mit den fehlenden Schlüsseln
emplacement_etat	der Zustand und das Journal liegen unter der Wurzel eines deklarierten Tresors	außerhalb jedes Tresors: AVERTISSEMENT (der Zustand liegt im Klartext)
ollama	GET <ollama.url>/api/tags antwortet innerhalb von ollama.timeout_secondes	keine Antwort: ÉCHEC
modele_ocr, modele_tri_rapide, modele_classification	die Rolle hat unter ihren Präferenzen mindestens ein installiertes Modell	keines installiert oder leere Liste: ÉCHEC, und die Meldung nennt ollama.roles.<role>
modele_tri_decision	die optionale Sortierung per Entscheidung (Kapitel 3): erstes installiertes Modell von ollama.roles.tri_decision, Fähigkeit decision, gültige Schwellen in classification.seuils_tri_decision, ein installiertes Modell für ollama.roles.objet	Rolle fehlt oder ist leer: NON_CONFIGURÉ; deklariert, aber unbrauchbar: AVERTISSEMENT, die Meldung nennt den zu korrigierenden Schlüssel, und die Sortierung läuft über tri_rapide
destination_personnel, destination_professionnel	die Prüfung des Einhängens aus Kapitel 5: der Dateisystemtyp der Wurzel ist akzeptiert und die Wurzel ist beschreibbar	nicht festgestellt: ÉCHEC, und die Meldung nennt den Grund

Eine Quelle mit Passwort, deren Umgebungsvariable nicht definiert ist, wird in der Prüfung configuration mit einer Warnung gemeldet, da das Programm eine solche Quelle zur Laufzeit überspringt. Keine Prüfung schreibt ein Geheimnis, einen Dokumentnamen oder den Inhalt einer Umgebungsvariable.

Einen Fehlschlag lesen

Eine Konfiguration des Übungsbereichs deklariert für jeden Bereich einen Tresor, der `fuse.gocryptfs` sein muss, während die Verzeichnisse gewöhnliche sind, so wie ein geschlossener Tresor aussieht:

```
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
```

Die beiden Prüfungen `destination_*` schlagen fehl und der Code ist `1`, während Ollama, die Modelle und die Werkzeuge in Ordnung sind. Die Meldung sagt, was zu tun ist: den Tresor öffnen oder den richtigen Typ in `systemes_fichiers_acceptes` eintragen.

Weitere häufige Fehlschläge:

Zeile	Was zu tun ist
`ÉCHEC ollama`	der Dienst antwortet nicht: prüfen Sie `ollama.url`, dann ob Ollama läuft (`systemctl status ollama`)
`ÉCHEC modele_<role>`	korrigieren Sie `ollama.roles.<role>` anhand von `ollama list`
`AVERTISSEMENT modele_tri_decision`	installieren Sie das genannte Entscheidungs- oder Gegenstandsmodell, oder schreiben Sie seine Schwellen; bis dahin läuft die Sortierung über `tri_rapide`
`ÉCHEC destination_<domain>`	öffnen Sie den genannten Tresor oder korrigieren Sie die akzeptierten Typen
`ÉCHEC configuration`	ergänzen Sie die genannten Schlüssel, indem Sie den Block aus `config.exemple.json` kopieren

Eine frische Installation

Gleich nach der Installation, ohne Konfiguration und ohne Ollama, ist die Diagnose kein Fehlschlag: Eine fehlende Konfiguration ist der Normalzustand einer frischen Installation und heißt `NON_CONFIGURÉ`, nicht mit einer Panne verwechselt. In einem leeren Home-Verzeichnis:

```
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
```

Die Werkzeuge sind `OK`, die Konfiguration ist `NON_CONFIGURÉ`, und die acht Prüfungen, die von ihr abhängen, sind `NON_VÉRIFIÉ`. Der Exit-Code ist `0`. Sobald eine Konfiguration existiert, ist alles, was sie deklariert und die Maschine nicht hat, ein `ÉCHEC`: Eine Diagnose, die bei gestopptem Ollama `0` lieferte, würde genau die Panne verbergen, die Sie suchen.

In einer anderen Sprache

Die Sprache folgt `LC_ALL`, dann `LC_MESSAGES`, dann `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)
```

Souveränität und Grenzen

Was das Werkzeug garantiert

- **Kein Dokument verlässt Ihr Netz**. Die Aufrufe der Sprachmodelle sind einfache HTTP-Anfragen an `ollama.url` (standardmäßig `http://localhost:11434`). Kein Dokument und kein Dateiname geht irgendwohin sonst, und es gibt keine Telemetrie.
- **Die einzigen externen Dienste, die der Code erreichen kann, sind die, die Sie deklarieren**: Ihre Maschinen über `ssh`, Ihre IMAP-Postfächer und, wenn Sie OAuth2 aktivieren, die Autorisierungsendpunkte von Google.
- **In `config.json` wird kein Geheimnis gespeichert**. Ein Mail-Passwort lebt in einer Umgebungsvariable, ein Google-Token in einer Datei mit dem Modus `600` außerhalb jedes Repositorys. Ein Token wird nie in ein Journal geschrieben, auf dem Bildschirm gezeigt oder in eine Fehlermeldung gesetzt.
- **Eine Ablage verliert nie ein Dokument**. Es wird kopiert, die Kopie wird per SHA256 geprüft, und erst dann wird das Original entfernt. Jede Ablage wird im Aktionsjournal festgehalten und lässt sich mit `--rollback` rückgängig machen.
- **Ein Postfach und eine entfernte Maschine bleiben unberührt**. Die gesendeten IMAP-Befehle sind die in Kapitel 7 aufgelisteten; eine entfernte Maschine wird nur gelesen, und dort wird nichts entfernt.
- **Der Text eines Dokuments sind Daten, nie eine Anweisung.** Er wird durch eine Zufallsmarke begrenzt, und das Modell antwortet mit einem geschlossenen JSON, das Dibab prüft. Eine Antwort, die nicht zum Schema passt, zählt als „unsicher“ und führt in den Prüfordner. Ein Modell kann nie eine Aktion auslösen.
- **Der Zweifel führt zu einer menschlichen Entscheidung**. Ein unsicherer Bereich, uneinige Modelle oder ein gemeinsames Stück ohne einstimmige Kategorie legen das Dokument in `ZU_PRÜFEN/`, nicht an einen riskanten Ort.
- **Das Journal enthält Pfade und Kategorien, nie Text**. Es enthält allerdings Dateinamen und Absender, die schon viel verraten; bewahren Sie es in einem Tresor auf (Kapitel 5).

Der Code steht unter der GNU Affero General Public License, Version 3. Er verwendet nur die Standardbibliothek von Python, es gibt also kein Fremdpaket zu prüfen.

Achtung des Haushalts

Eine Maschine, ein Verzeichnis oder ein Mail-Konto gehört nur dann zum Umfang des Werkzeugs, wenn die betroffene Person ausdrücklich oder stillschweigend zugestimmt hat. Die Software kann das nicht prüfen: Der Administrator der Instanz verpflichtet sich dazu. Richten Sie Dibab nicht auf das Postfach oder die Dateien einer Person, die nicht weiß, dass sie gelesen werden.

Die Grenzen

- **Entfernte Quellen wurden nur gegen falsche Server erprobt**. Die `ssh`-Quelle und die IMAP-Quelle, mit Passwort oder mit OAuth2, sind durch Tests und durch die Wiederholungen dieses Handbuchs abgedeckt, die ein falsches `ssh`, einen falschen IMAP-Server und einen falschen Autorisierungsserver verwenden. Ihr Einsatz auf einer echten Maschine des Haushalts und einem echten Postfach, einschließlich der Zustimmung bei Google, steht noch aus: Solange dieser Versuch nicht stattgefunden hat, betrachten Sie diese Quellen als unter realen Bedingungen nicht validiert und beginnen Sie mit `--dry-run`.

- **Die Qualität der Ablage ist die Qualität Ihrer Modelle**. Auf der Referenzinstanz hat ein erster Versuch mit 15 echten Dokumenten 2 abgelegt, 3 als nicht zur Verwaltung gehörend ausgesondert und aus Vorsicht 10 in `ZU_PRÜFEN/` gelegt. Messen Sie, bevor Sie vertrauen; das Messwerkzeug der Quellen, `_outils/mesurer_domaines.py`, zieht eine Stichprobe abgelegter Dokumente und vergleicht das Urteil der Modelle mit dem Tresor, in dem jedes liegt.
- **Nur Linux**. Die Prüfung des Einhängens stützt sich auf `findmnt`, das zu `util-linux` gehört.
- **Dibab hängt nichts ein, verschlüsselt nichts und synchronisiert nichts**, und es verwaltet weder den Speicherplatz noch die Sicherungen: Das ist Ihre Sache.
- **Der Google-Scope ist weiter gefasst als die Nutzung**. Der Scope `https://mail.google.com/` ist der einzige, den Google für IMAP dokumentiert. Er erlaubt Lesen, Schreiben und Löschen. Dass das Postfach unverändert bleibt, beruht auf den Befehlen, die Dibab sendet und die in Kapitel 7 aufgelistet sind, nicht auf dem Scope.
- **Die Voreinstellungen sind Beispiele**. Die Kategorien, die bekannten Absender und die Benennungsvorlage von `config.exemple.json` sind generisch; passen Sie sie an Ihr Land an.
- **Die Schritte in der Google-Konsole hängen von der Oberfläche von Google ab**, die sich ändert. Die Begriffe bleiben die ihrer OAuth2-Dokumentation.
- **Das Projekt ist jung**. Die Version ist 0.9.0-alpha; es gibt keine kontinuierliche Integration. Die Tests der Quellen werden von Hand ausgeführt.
- **Ollama gehört nicht zu Dibab**. Es wird von ihm nie installiert und nie aktualisiert; ein Modell, das upstream umbenannt oder entfernt wird, ist eine Zeile, die in `ollama.roles` zu korrigieren ist.

Ein Problem melden

Eine Schwachstelle wird privat gemeldet, gemäß der Datei `SECURITY.md` der Quellen, nie in einem öffentlichen Issue. Für einen Beitrag lesen Sie `CONTRIBUTING.md` am selben Ort.