Intelligence artificielle

Procédure de mise en production

Cette procédure de mise en production s'applique à tous les outils liés à l'IA

1. Phase de test.

Tester les modifications qui seront mises, en amont, au préalable.

2. Présenter les changements (liste des documents), pour validation, au responsable des études.

La mise en production ne peut pas commencer sans l'accord du responsable des études.

3. Définir la date pour la MEP et s'assurer que plusieurs personnes compétentes sont disponibles à la date définie.

Procéder à la mise en production en matinée idéalement. Le vendredi est à éviter.

4. Prévenir les usagers que l'application sera indisponible (en cas de gros changements ou d'indisponibilité sur plusieurs minutes ou plus)

- Personnes ayant suivi l'atelier d'initiation sur l'IA :

https://onlyoffice.vincennes.fr/Products/Files/DocEditor.aspx?fileid=39384

- Immich : prévenir la DirCom

5. Se préparer à un retour en arrière en cas de problème durant la mise en production

- Faire une copie des fichiers, avant modification, pour pouvoir facilement rétablir la version précédente

- Pour l'extension Assistant IA, déployer la version précédente : https://addons.mozilla.org/fr/developers/addon/331f39fefe294d7d8498/versions

- Éventuellement faire un snapshot

6. Procéder aux modifications nécessaires et tester à la fin si tout est fonctionnel

En cas de soucis, il faut s'assurer que l'application soit fonctionnelle :

- tenter de résoudre le problème

- contacter les autres personnes qui pourraient aider à résoudre le problème

- revenir en arrière en restituant la version précédente (voir le point numéro 5)

Extension Firefox "Assistant IA"

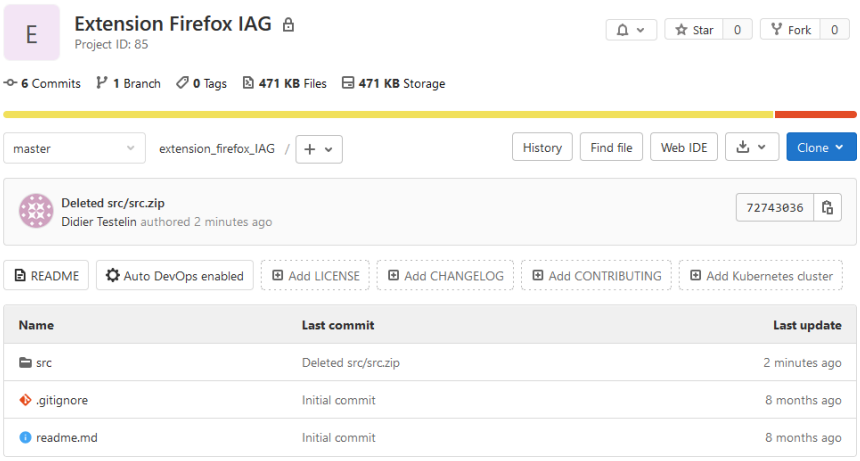
Mettre à jour les prompts par défaut de l'extension

L'extension Firefox 'Assistant IA" propose des prompts par défaut : Corriger, Résumer...

Pour modifier l'un des prompts, il est nécessaire de modifier le code javascript correspondant.

1. Aller sur le projet du gitlab :

https://gitlab.vincennes.fr/application-interne/extension_firefox_IAG



2. Télécharger l'ensemble des fichier en cliquant sur  puis "zip".

Modifier le code se trouvant dans /src/content.js

```
if (message.id == "corriger-le-texte") {
  demande_copilot = "corriger la phrase (orthographe et grammaire, répondre strictement à la demande, ne pas proposer de source) : '";
}
if (message.id == "resumer-le-texte") {
  demande_copilot = "résumer le texte suivant : '";
}

if (message.id == "expliquer-le-texte") {
  demande_copilot = "expliquer le texte suivant en apportant des sources : '";
```

```
}

if (message.id == "reformuler") {
  demande_copilot = "reformuler un texte avec les éléments suivants : '";
}

if (message.id == "traduire") {
  demande_copilot = "traduire en français : '";
}
```

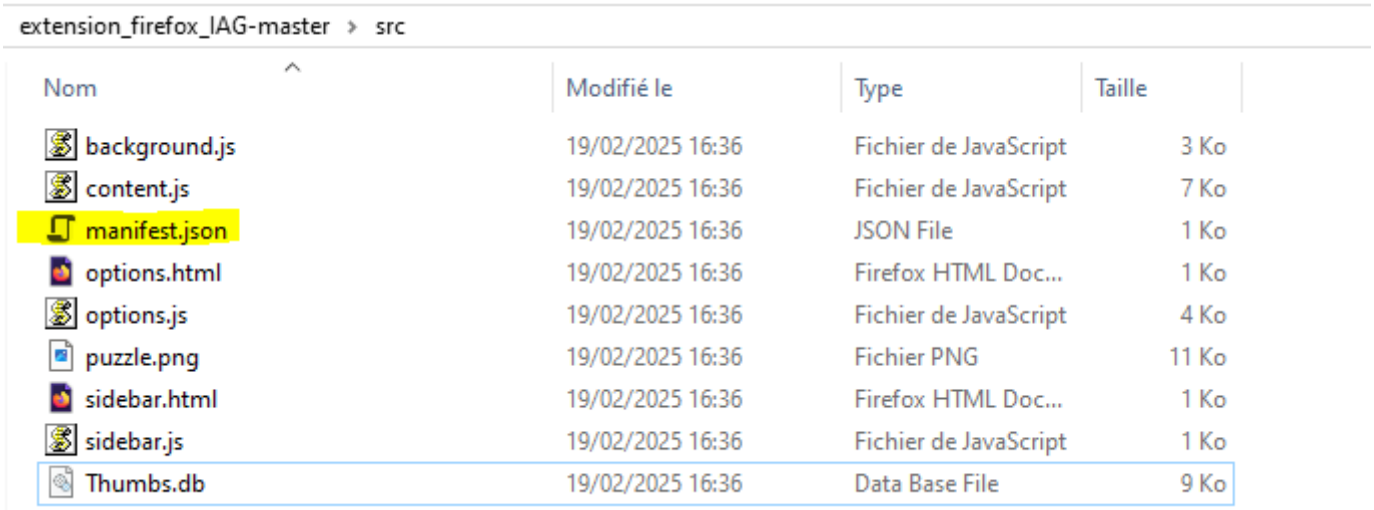
3. Tester le code

- Désactiver l'extension "Assistant IA" si elle est déjà installée dans Firefox.

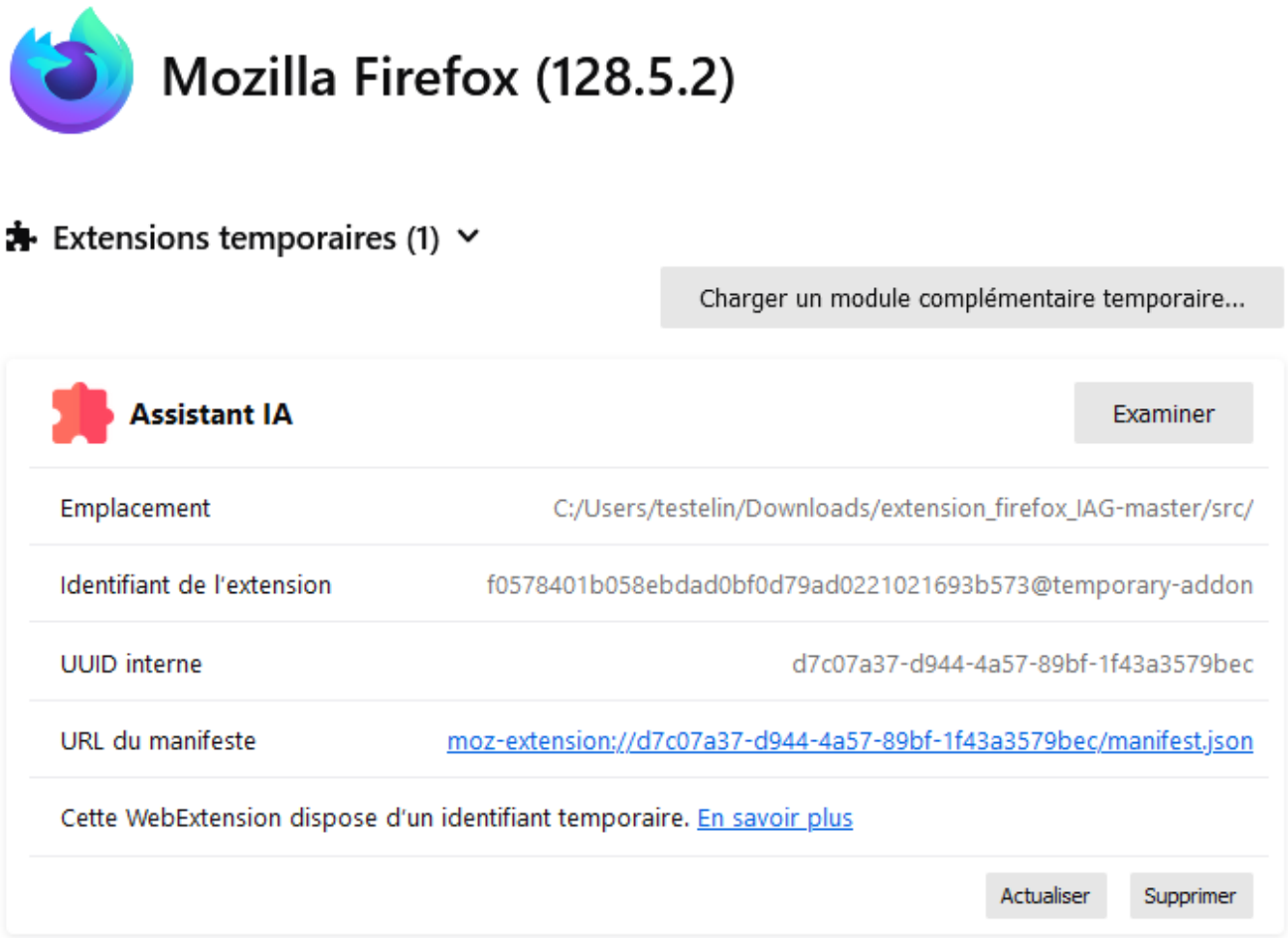
- Ouvrir un onglet Firefox et copier/coller about:debugging#/runtime/this-firefox

- Cliquer sur "Charger un module complémentaire temporaire"

- Dans le répertoire de fichiers de l'extension, sélectionner manifest.json



- L'extension se charge et l'écran devient



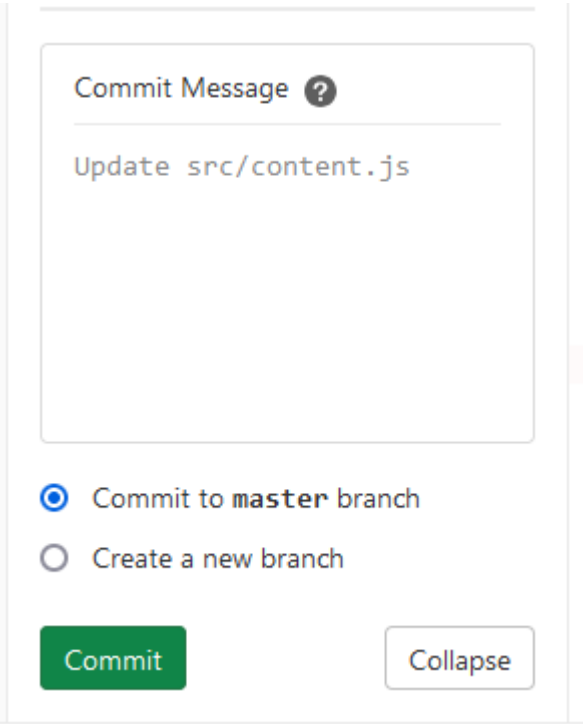
- A chaque changement du fichier content.js => cliquer sur "Actualiser" pour avoir la dernière version.

- Supprimer l'extension dès que le rendu est satisfaisant.

2. Retourner sur https://gitlab.vincennes.fr/application-interne/extension_firefox_IAG et Cliquer sur le bouton Web IDE puis sélectionner le fichier /src/content.js

3. Modifier le code en fonction des tests précédents

4. Cliquer sur le bouton "Commit" puis sélectionner "Commit sur master branch"



5. Cliquer à nouveau sur "Commit"

Le gitlab est mis à jour avec la dernière version du document.

6. Faire signer l'extension par Mozilla

https://formagent.vincennes.fr/books/intelligence-artificielle/page/signer-une-extension-firefox

Signer une extension Firefox

Une extension développée pour Firefox peut être testée avec le mode développeur comme précisé sur le tutoriel :

https://formagent.vincennes.fr/books/intelligence-artificielle/page/mettre-a-jour-les-prompts-par-defaut-de-lextension

Pour la déployer sur l'ensemble des postes, il est nécessaire de la faire reconnaître par Mozilla, c'est à dire la faire signer.

1. Aller sur https://addons.mozilla.org/fr/developers/

2. S'identifier sur le portail en utilisant le compte referentlogiciels@vincennes.fr disponible sur psono.

3. Préparer le package

Modifier le fichier manifest.json

```
{
  "manifest_version": 2,
  "name": "Assistant IA",
  "version": "2.1",
  "description": "Assistant pour les agents de la Ville de Vincennes",
  "permissions": [
    "contextMenus",
    "storage"
  ],
  "background": {
    "scripts": [
      "background.js"
    ]
  },
  "icons": {
    "48": "puzzle.png"
  },
  "options_ui": {
```

```
    "page": "options.html",
    "open_in_tab": false,
    "browser_style": true
  },
  "sidebar_action": {
    "default_title": "Assistant",
    "default_panel": "sidebar.html"
  },
  "browser_action": {
  "default_icon": {
    "48": "puzzle.png"
  },
  "default_title": "Ouvrir le sidebar"
  },

  "content_scripts": [
    {
      "matches": [
        "https://dify.vincennes.fr/chat/lfDdUckNIH9hGtQ4"
      ],
      "js": [
        "content.js"
      ]
    }
  ]
}
```

> Le numéro de version doit être modifié. Le cas échéant, Mozilla indiquera que la version est déjà présente.

Préparer les fichiers

Zipper les fichiers dans un fichier extension.zip. Le nom importe peu.

L'importance est que l'ensemble des fichiers soient à la racine du zip, et non dans un répertoire.

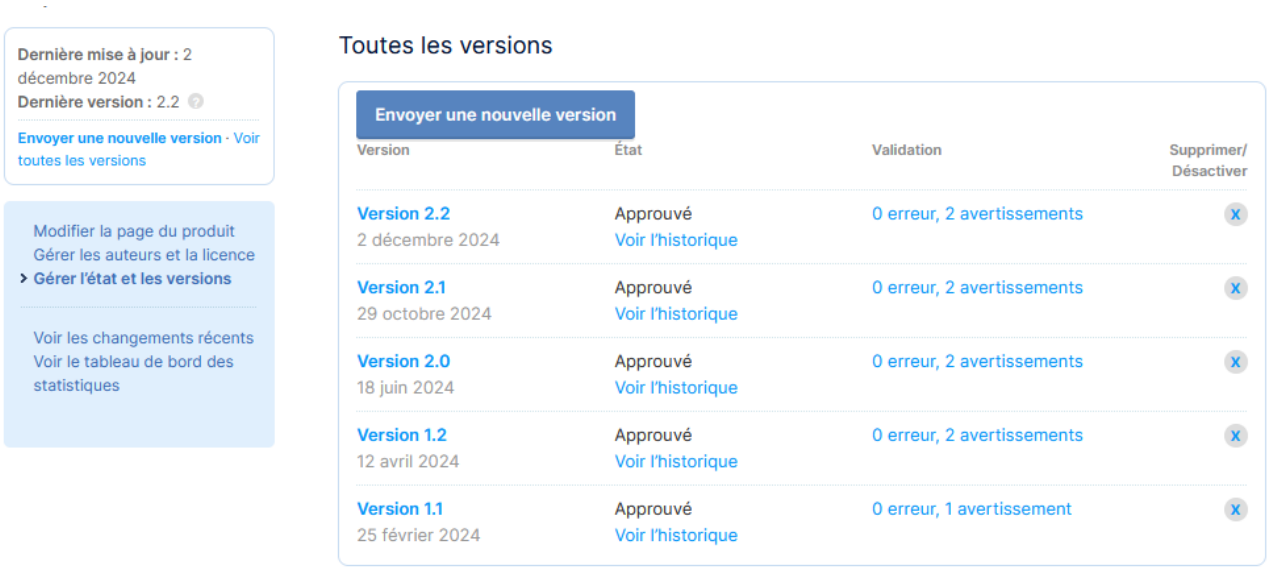
4. Déposer le package

Aller sur la page https://addons.mozilla.org/fr/developers/addon/submit/upload-listed

Déposer le package et suivre les instructions.

Corriger les erreurs éventuelles signalées par Mozilla.

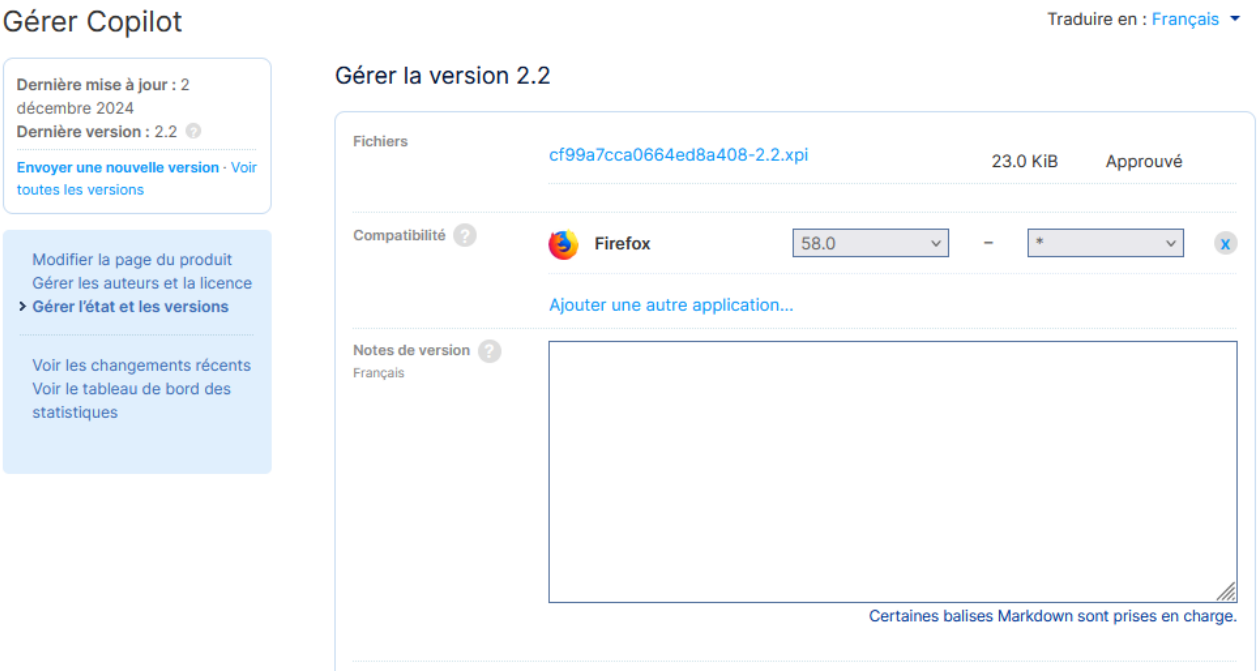
5. Se rendre sur la page d'état du module



Attendre que le module passe en état "Approuvé".

6. Télécharger le .xpi

Cliquer sur le numéro de version et télécharger le .xpi (clic droit puis enregistrer la cible du lien sous...).



7. Tester l'extension sur Firefox

8. Demander au service exploitation le déploiement du .xpi

Librechat

Procédure de mise en production sur Librechat

1. Phase de test sur PreLibrechat.

Tester les modifications qui seront mises sur Librechat (SP26) au préalable.

- Modifications sur le fichier de configuration de prelibrechat (librechat.yaml)

- Modifications dans la stack (docker-compose) de prelibrechat sur SD11

2. Présenter les changements (liste des documents), pour validation, au responsable des études.

La mise en production ne peux pas commencer sans l'accord du responsable des études.

3. Définir la date pour la MEP et s'assurer que plusieurs personnes compétentes sont disponibles à la date définie.

Procéder à la mise en production en mâtiné idéalement.

4. Prévenir les usagers de Librechat que l'application sera indisponible

5. Se préparer à un retour en arrière en cas de problème durant la mise en production

- Faire une copie des fichiers, avant modification, pour pouvoir facilement rétablir la version précédente

(- Éventuellement faire un snapshot)

6. Procéder aux modifications nécessaires et tester à la fin si tout est fonctionnel

En cas de soucis, il faut s'assurer que l'application soit fonctionnelle :

- tenter de résoudre le probleme

- contacter les autres personnes qui pourraient aider à résoudre le probleme

- revenir en arrière en restituant la version précédente (voir le point numéro 5)

Mettre à jour Librechat

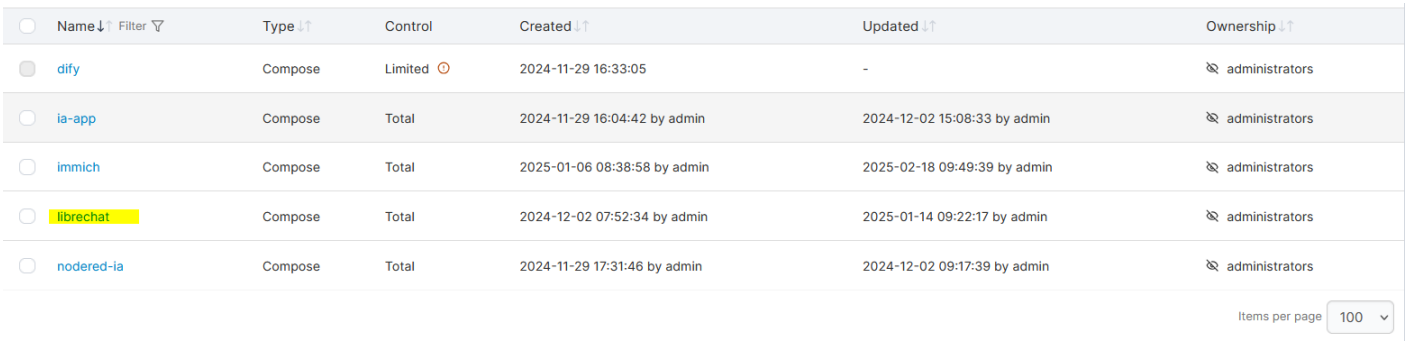
Librechat repose sur une stack Docker.

Le changelog se trouve sur https://github.com/danny-avila/LibreChat/releases

Attention : ne prendre que les versions finales. Eviter les beta ou release candidate (rc).

1. Aller sur portainer et choisir le serveur SD11 (pour la préproduction) ou SP26 (pour la production)

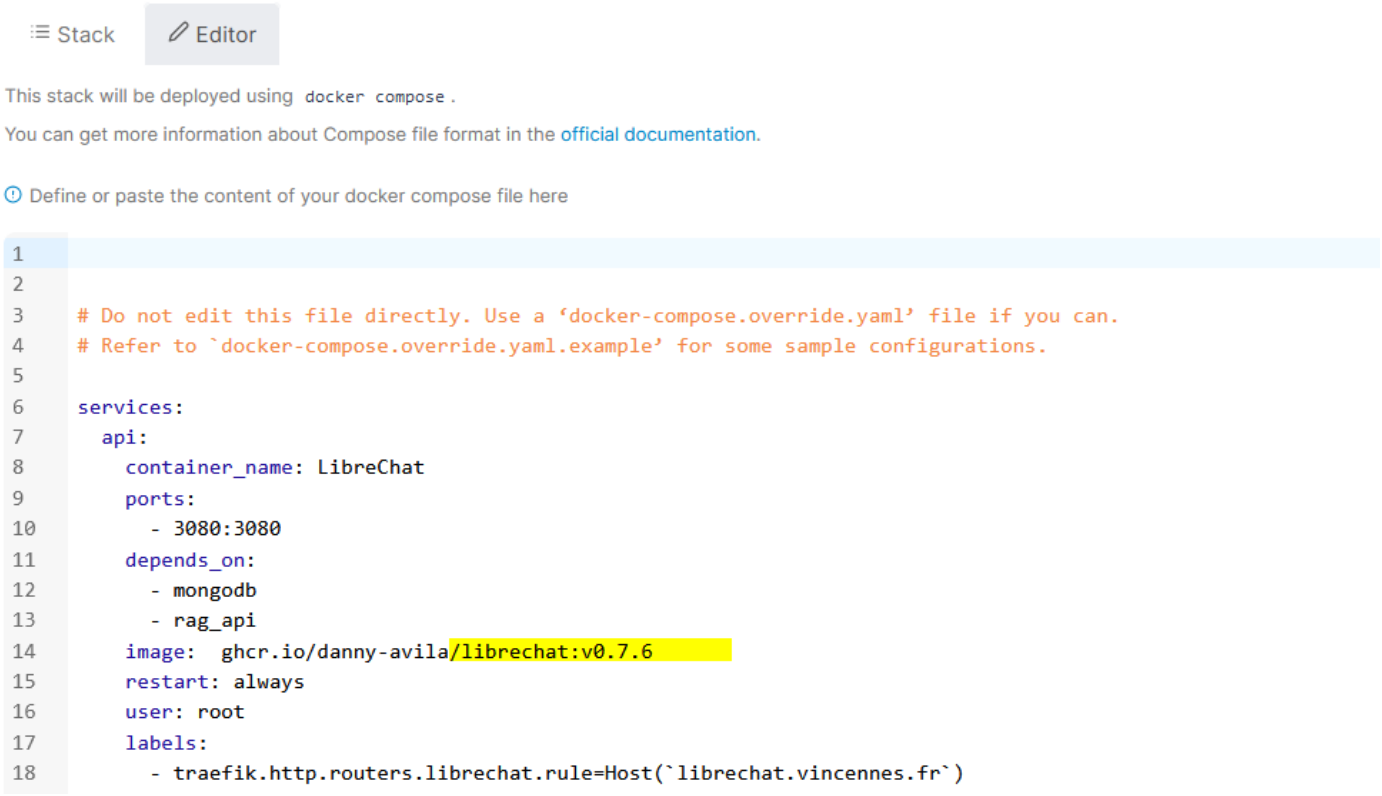
2. Lister les stacks et choisir librechat



Name	Type	Control	Created	Updated	Ownership
dify	Compose	Limited	2024-11-29 16:33:05	-	administrators
ia-app	Compose	Total	2024-11-29 16:04:42 by admin	2024-12-02 15:08:33 by admin	administrators
immich	Compose	Total	2025-01-06 08:38:58 by admin	2025-02-18 09:49:39 by admin	administrators
librechat	Compose	Total	2024-12-02 07:52:34 by admin	2025-01-14 09:22:17 by admin	administrators
nodered-ia	Compose	Total	2024-11-29 17:31:46 by admin	2024-12-02 09:17:39 by admin	administrators

Items per page 100

3. Cliquer sur l'onglet Editor et modifier le numéro de version



4. Cliquer sur déployer

5. Vérifier le bon fonctionnement et corriger les éventuelles erreurs

Les erreurs sont essentiellement dues à des évolutions du fichier de configuration ou des autres services précisés dans le docker-compose.

Mettre à jour les modèles dans Librechat

La configuration de Librechat se fait à deux niveaux :

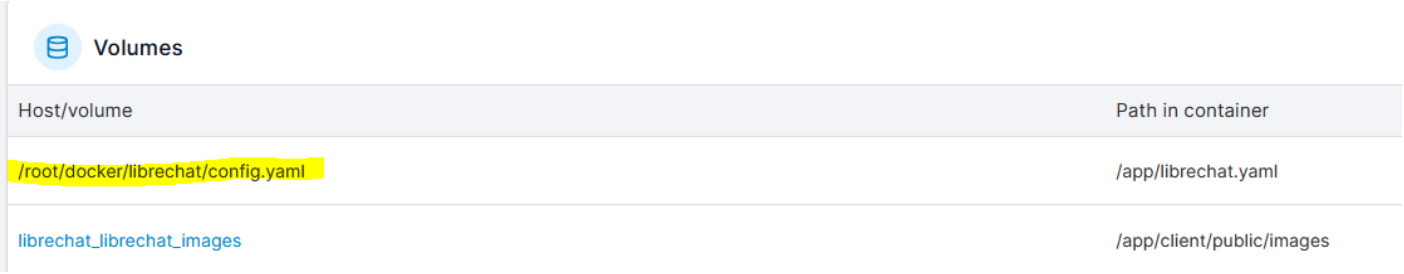
- Dans le fichier de configuration de Librechat

- Dans la stack (docker-compose) de Librechat

Fichier de configuration de Librechat

1. Identifier la localisation du fichier

Aller sur le container Librechat puis sur la partie Volumes



Le fichier peut être édité

- soit dans le container (via /app/librechat/yaml)

- soit en SSH sur le serveur (via /root/docker/librechat/config.yaml dans le cas présent)

2. Editer le fichier suivant la méthode choisie

Ci-dessous un exemple de configuration pour les modèles Ollama :

```
    - name: "Ollama"
      apiKey: "ollama"
      # use 'host.docker.internal' instead of localhost if running LibreChat in a docker
container
      baseURL: "http://sp26.vincennes.fr:11434/v1/"
```

```
      models:
        default: [
         "llama3.2-vision:11b","llama3.1:8b"
          ]
        # fetching list of models is supported but the `name` field must start
        # with `ollama` (case-insensitive), as it does in this example.
        fetch: false
      titleConvo: true
      titleModel: "current_model"
      summarize: false
      summaryModel: "current_model"
      forcePrompt: false
      modelDisplayLabel: "Ollama"
```

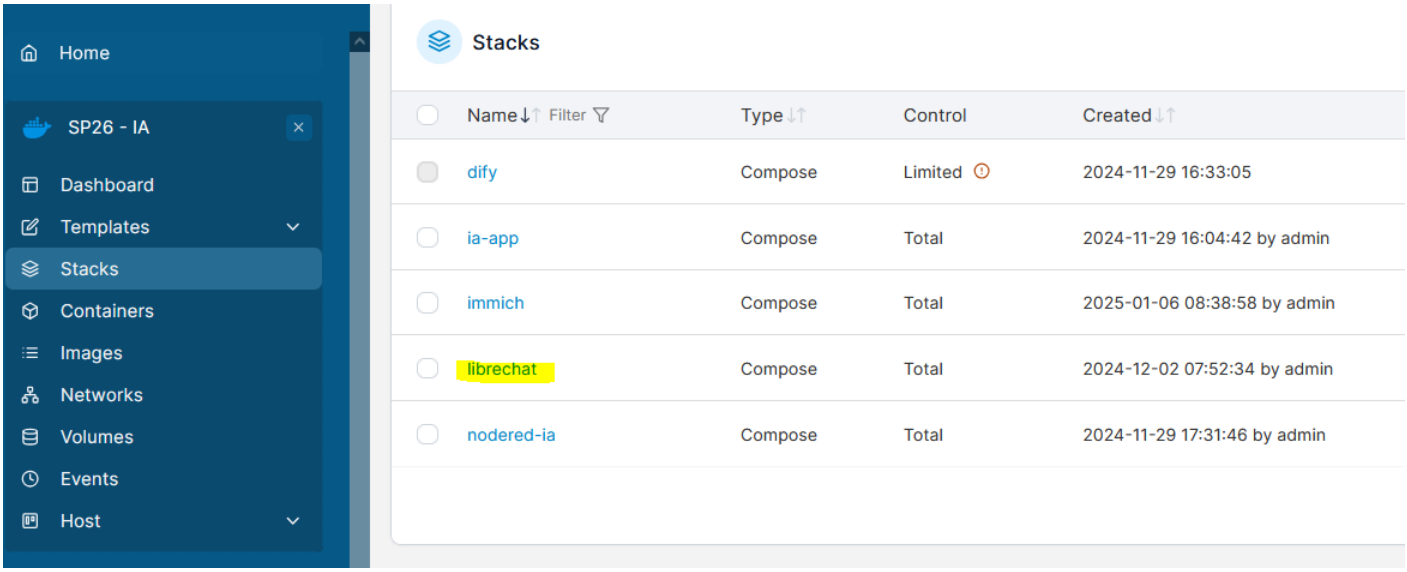
La documentation de Librechat précise la signification des différents champs :

https://www.librechat.ai/docs/configuration/librechat_yaml/object_structure/custom_endpoint

> Le numéro de version du fichier config doit être changé

Fichier docker-compose de Librechat

1. Dans stacks, sélectionner Librechat



2. Cliquer sur Editor

3. Modifier la variable d'environnement ENDPOINTS

```
33        - DEBUG_LOGGING=true
34        - DEBUG_CONSOLE=true
35        - ENDPOINTS=Ollama
36
37        - PROXY=
```

> Elle doit correspondre au champ "name" de config.yaml

4. Embeddings

Dans le cas où le modèle d'embedding du RAG doit être modifié, les options se trouvent sur la partie "rag_api".

```
165   rag_api:
166     container_name: rag_api
167     image: ghcr.io/danny-avila/librechat-rag-api-dev:latest
168     environment:
169       - DB_HOST=vectordb
170       - RAG_PORT=${RAG_PORT:-8000}
171       - EMBEDDINGS_PROVIDER=ollama
172       - OLLAMA_BASE_URL=http://sp26.vincennes.fr:11434/
173       - EMBEDDINGS_MODEL=bge-m3
174     restart: always
175     depends_on:
176       - vectordb
```

Les explications pour les différents champs sur dans la documentation Librechat :

https://www.librechat.ai/docs/configuration/rag_api

5. Cliquer sur Deploy pour mettre à jour Librechat

Tests

Vérifier le bon fonctionnement.

Les logs du container Librechat indiqueront les problèmes éventuels.

Immich

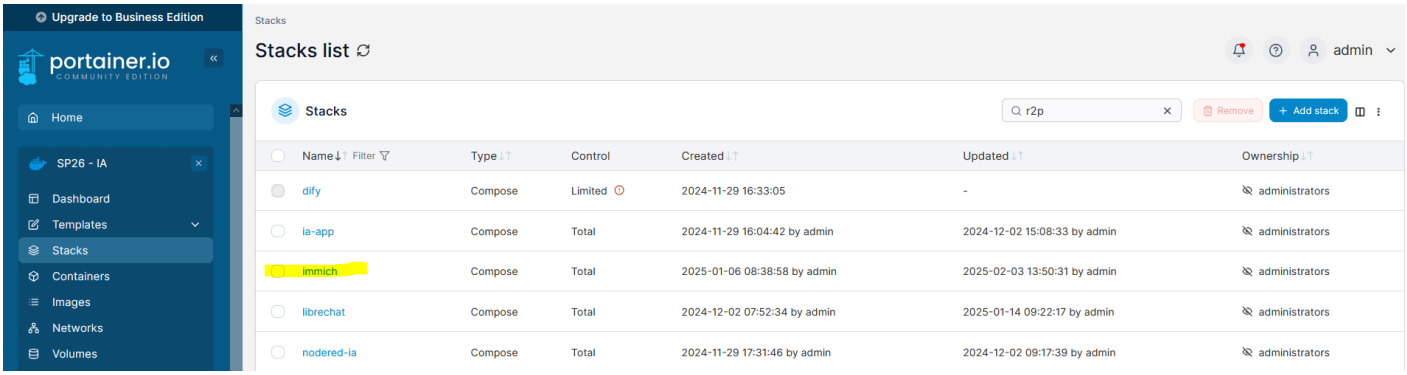
Mettre à jour Immich

> Penser à faire un snapshot de la VM et prévenir les utilisateurs.

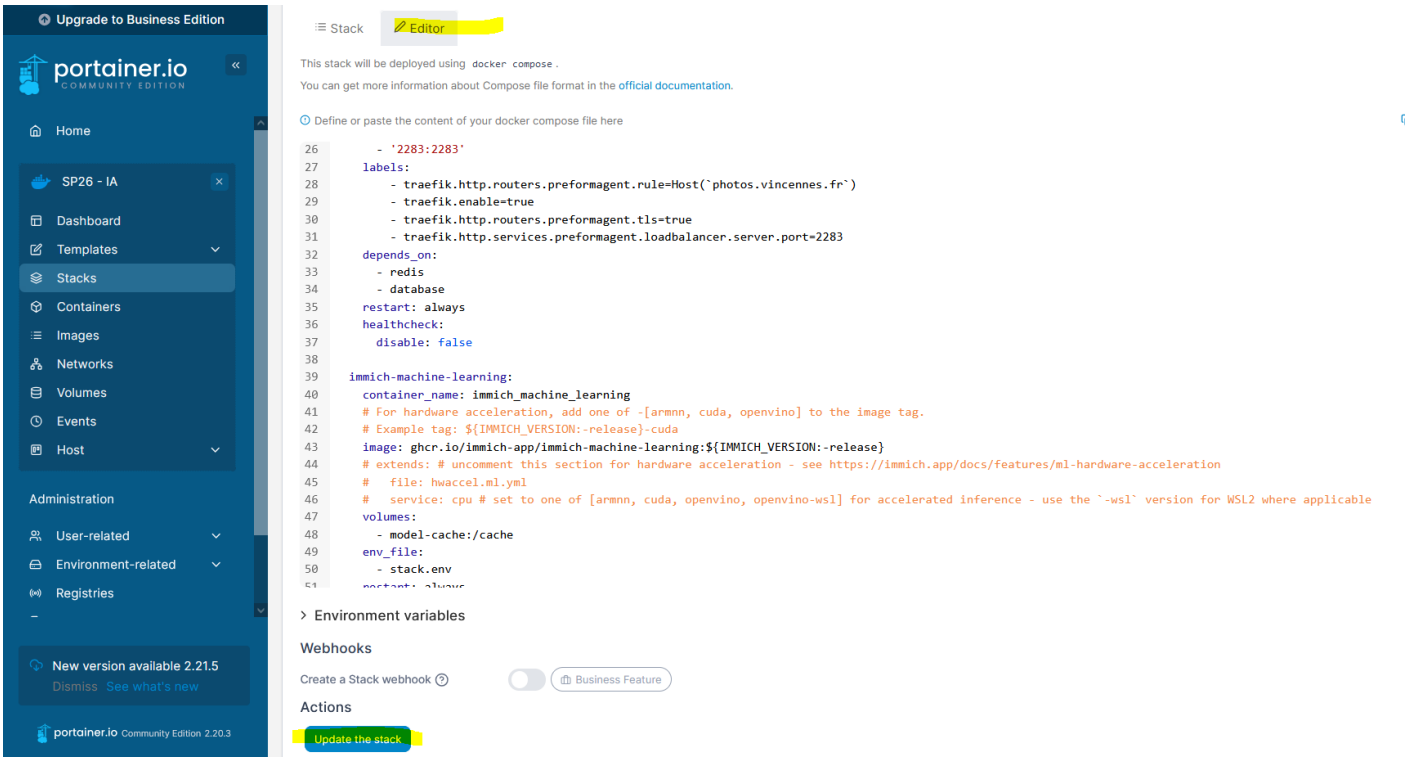
Pour mettre à jour l'application, il faut se rendre sur Portainer : https://st30.vincennes.fr:9000

1. Sélectionner le serveur SP26

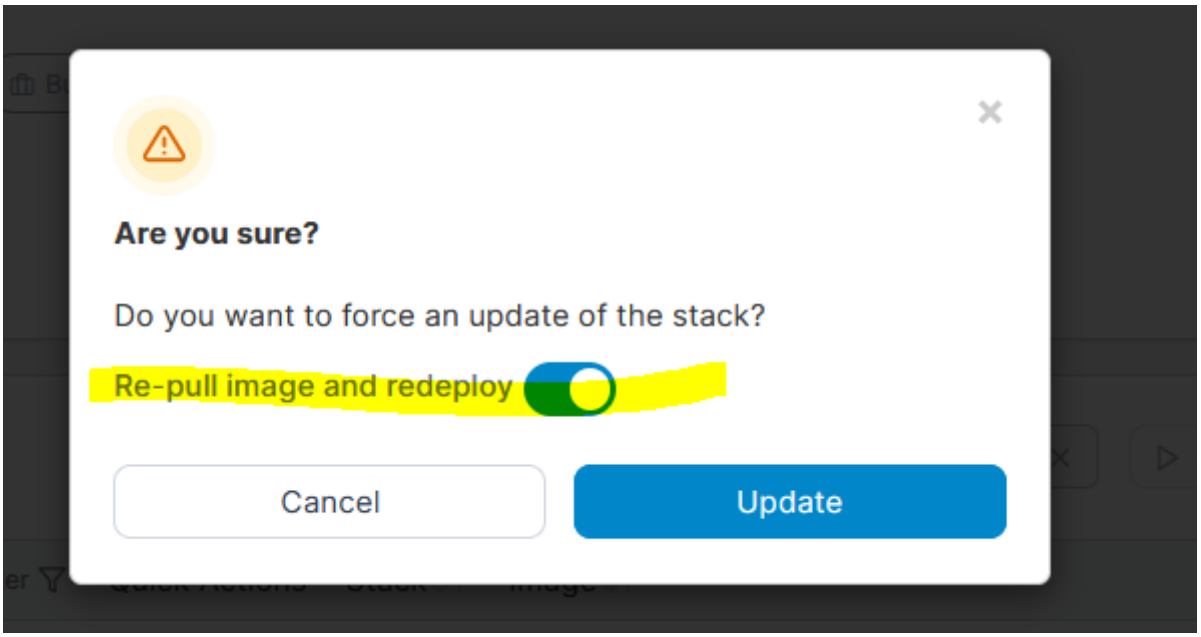
2. Sélectionner "Stacks" puis Immich :



3. Cliquer sur l'onglet "Editor" puis "Update the stack" :



4. Activer "Re-pull image and redeploy", cliquer sur "Update" et attendre quelques minutes :

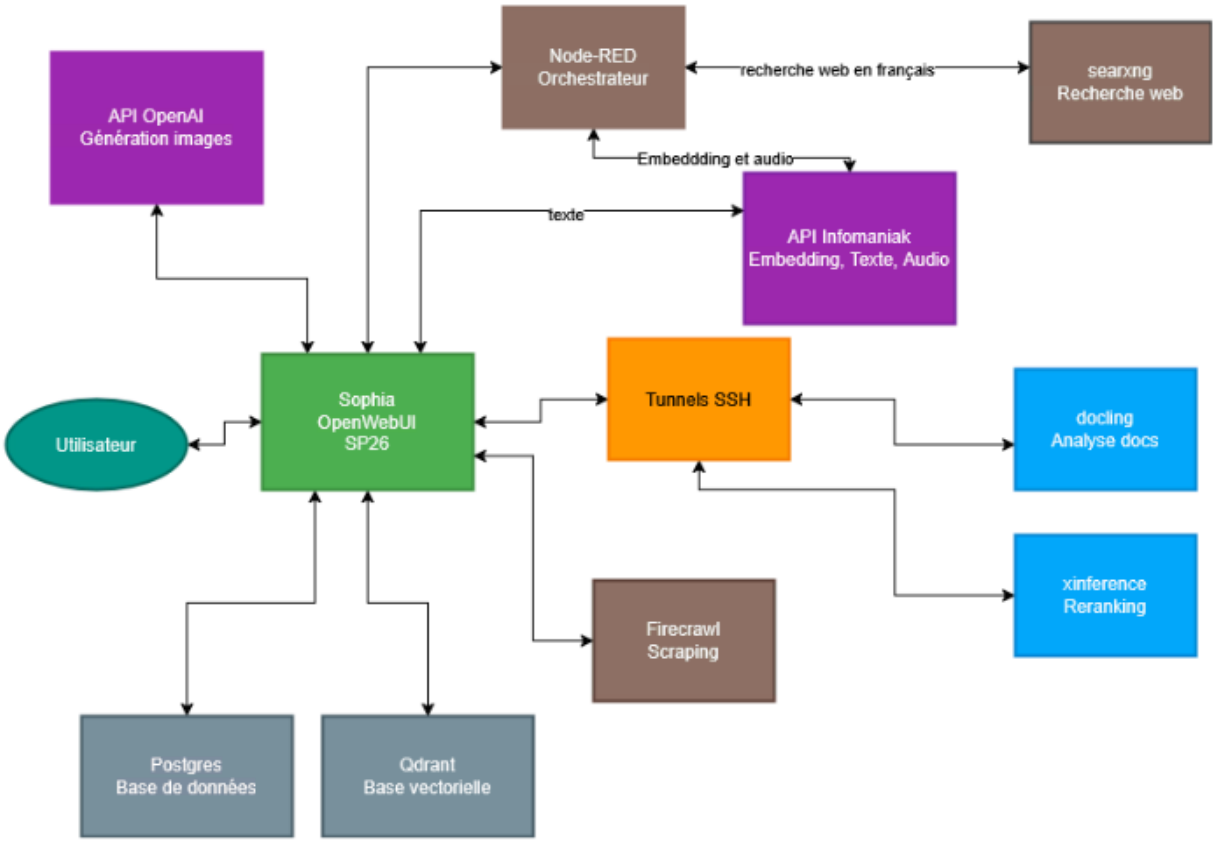


Sophia

Architecture générale

Schéma d'architecture générale de Sophia

profile



Pour modifier l'image : https://onlyoffice.vincennes.fr/Products/Files/DocEditor.aspx?fileid=40912

L'architecture Today at 19:45

L'architecture est basée sur la documentation d'openwebui : https://docs.openwebui.com/.

Tableau récapitulatif des conteneurs - visualisation par Portainer

Conteneur / Service	Stack / Origine	Rôle	Port hôte (local)	Port conteneur	URL d’accès depuis Sophia
`openwebui`	Sophia	Interface principale	3002	8080	`https://sophia.vincennes.fr`
`qdrant`	Sophia	Base vectorielle (RAG)	6333, 6334	6333, 6334	Interne (via API locale)
`postgres`	Sophia	Base de données d’application	5437	5432	Interne
`sophia-nodered-openwebui-1`	Sophia	Automatisation, intégration des services	1886	1880	`http://sp26.vincennes.fr:1886`
`infomaniak-xinference-1`	Infomaniak (tunnel SSH)	Tunnel vers xinference (reranking)	9995	9998	`http://sp26.vincennes.fr:9995`
`infomaniak-docling-1`	Infomaniak (tunnel SSH)	Tunnel vers docling (analyse de documents)	5002	5001	`http://sp26.vincennes.fr:5002`
`infomaniak-portainer_agent-1`	Infomaniak (tunnel SSH)	Agent Portainer pour supervision distante	9002	9001	Interne (gestion Docker)
`searxng`	ia-app	Moteur de recherche web	8056	8080	Via Node-RED (recherche)
`firecrawl-api-1`	firecrawl	Analyse de pages web (scraping intelligent)	3003	3002	`http://sp26.vincennes.fr:3003`
`docling` (VM Infomaniak)	-	Analyse de documents (PDF, etc.)	5001	5001	Accès distant (via tunnel SSH depuis SP26:5002)
`xinference` (VM Infomaniak)	-	Modèle de reranking (`bge-reranker-v2-m3`)	9997	9997	Accès distant (via tunnel SSH depuis SP26:9995)
`portainer_agent` (VM Infomaniak)	-	Agent Docker pour supervision	9001	9001	Accès distant (via tunnel SSH depuis SP26:9002)

Réglages dans l’interface administrateur - visualisation par Openwebui

1. Menu Connexions

- **URL** : `https://sophia.vincennes.fr/admin/settings/connections`
- **Fonction** : Gestion des connexions aux modèles IA hébergés sur Infomaniak.
- **Détails** :
  - Affiche l'état de disponibilité des endpoints IA (texte, audio, image).
  - Aucune action de configuration possible directement : en cas de problème, contact avec le support requis.
  - Vérification de la connectivité aux API distantes.

2. Menu Documents

- **URL** : `https://sophia.vincennes.fr/admin/settings/documents`
- **Fonction** : Configuration du traitement des documents (analyse, embedding, reranking).
- **Paramètres** :
  - **Analyse de documents** :
    → URL : `http://sp26.vincennes.fr:5002/`
    → Connecté au service **docling** via tunnel SSH.
  - **Embedding** :
    → URL : `http://sp26.vincennes.fr:1886/`
    → Passerelle via **Node-RED**, qui relaye vers le modèle `bge_multilingual_gemma2` sur Infomaniak.
    → Utilisation de Node-RED pour limiter et réguler les appels API.
  - **Reranking (RAG)** :
    → URL : `http://sp26.vincennes.fr:9995/v1/rerank`
    → Connecté au modèle `bge-reranker-v2-m3` via **xinference** sur Infomaniak (tunnel SSH).

3. Menu Recherche web

- **URL** : `https://sophia.vincennes.fr/admin/settings/web`
- **Fonction** : Configuration de la recherche web et de l'analyse des résultats.
- **Paramètres** :
  - **Recherche** :
    → URL : `http://sp26.vincennes.fr:1886/search?q=<query>`
    → Passerelle via **Node-RED**, qui interroge **searxng**.
    → Node-RED assure la traduction de la requête en français si nécessaire.
  - **Analyse des pages** :
    → URL : `http://sp26.vincennes.fr:3003/`
    → Envoi des URLs récupérées vers **Firecrawl** pour extraction de contenu structuré.

4. Menu Interface utilisateur

- **Fonction** : Génération automatique de métadonnées pour améliorer l'expérience utilisateur.
- **Utilisation** :
  - Appels aux **API Infomaniak** pour :
    - Génération de **titres** pertinents.
    - Création de **tags** et de **mots-clés** pour la recherche.
  - Ces fonctionnalités enrichissent les réponses et les contenus générés.

5. Menu Audio

- **URL** : `https://sophia.vincennes.fr/admin/settings/audio`
- **Fonction** : Transcription de fichiers audio en texte.
- **Paramètres** :
  - **Transcription** :
    → URL : `http://sp26.vincennes.fr:1886/`
    → Utilisation de **Node-RED** pour gérer l'appel à l'API Infomaniak.
    → Le processus est asynchrone : Node-RED surveille la tâche et renvoie le texte une fois disponible.
  - Permet de traiter des fichiers audio longs via une file d'attente.

6. Menu Images

- **Fonction** : Génération d'images à partir de descriptions textuelles.
- **Paramètres** :
  - Appel direct aux **endpoints OpenAI** (ex: DALL·E).
  - Aucun traitement local ou intermédiaire : la requête est envoyée directement à OpenAI.
  - Les images générées sont intégrées dans les réponses de l'interface.

Sophia

Vérification du bon fonctionnement

Voici une **checklist de vérification du bon fonctionnement** du système, suivie d'un **tableau récapitulatif des problèmes possibles en cas d'indisponibilité**.

? Checklist de vérification du bon fonctionnement

Élément à vérifier	Procédure de vérification	Statut (✓/✗)
1. Containers dans Portainer	Accéder à l'interface Portainer et s'assurer que tous les conteneurs sont en état **"Running"** : - `openwebui`, `qdrant`, `postgres`, `sophia-nodered-openwebui-1` - `infomaniak-xinference-1`, `infomaniak-docling-1`, `infomaniak-portainer_agent-1` - `searxng`, `firecrawl-api-1 (et les containers de la stack firecrawl)`	
2. Interface Sophia (OpenWebUI)	Accéder à : `https://sophia.vincennes.fr` → Vérifier que la page charge correctement et que l'interface est interactive.	
3. Interface Node-RED	Accéder à : `http://sp26.vincennes.fr:1886` → Vérifier que l'éditeur s'ouvre et que les flux sont actifs.	
4. Firecrawl (API et message par défaut)	Accéder à : `http://sp26.vincennes.fr:3003` → Vérifier que la réponse "SCRAPERS-JS: Hello, world! K8s! "s'affiche.	

Élément à vérifier	Procédure de vérification	Statut (✓/✗)
5. Interface Docling	Accéder à : `http://sp26.vincennes.fr:5002/ui` → Vérifier que l'interface utilisateur de Docling est accessible.	
6. XInference et modèle de reranking	1. Accéder à l'interface XInference via le tunnel : `http://sp26.vincennes.fr:9995` 2. Vérifier que le modèle `bge-reranker-v2-m3` est **chargé et en cours d'exécution** dans la section `/reranker`.	
7. SearXNG	Accéder à : `http://sp26.vincennes.fr:8056` → Vérifier que l'interface de recherche s'affiche. → Effectuer une recherche test (ex: "test").	
8. Endpoint OpenAI	Depuis l'interface Sophia (Menu Images) : → Lancer une génération d'image. → Vérifier que la requête atteint OpenAI et retourne une image.	
9. Endpoints Infomaniak (via Node-RED)	Vérifier dans les menus Sophia : - **Connexions** : état des endpoints texte/audio/image - **Documents**, **Audio**, **Recherche** : que les appels passent via `http://sp26.vincennes.fr:1886` → Vérifier les logs Node-RED en cas d'erreur.	
10. Logs des conteneurs de tunnel SSH	Vérifier les logs des conteneurs suivants dans Portainer : - `infomaniak-xinference-1` - `infomaniak-docling-1` - `infomaniak-portainer_agent-1` → Rechercher des erreurs de connexion SSH, timeout ou refus. → S'assurer que la VM Infomaniak est joignable (ping ou test de connexion réseau).	

?? **Tableau des problèmes possibles en cas d'indisponibilité**

Service indisponible	Symptôme observé	Cause probable	Impact fonctionnel
Conteneur non démarré dans Portainer	Interface inaccessible, erreur 502/503	Conteneur crashé, mauvaise configuration, dépendance manquante	Blocage total du service associé
Sophia (OpenWebUI)	Page blanche, erreur de chargement	Problème réseau, conteneur down, certificat SSL expiré	Interface principale inutilisable
Node-RED	Éditeur inaccessible ou flux inactifs	Conteneur down, erreur dans un flux critique	Toutes les intégrations (recherche, audio, embedding) bloquées
Firecrawl	API ne répond pas ou timeout	Conteneur down, VM surchargée, erreur de scraping	Impossible d'extraire du contenu web pour les recherches
Docling	`/ui` inaccessible ou erreur 500	Tunnel SSH rompu, service Docling arrêté sur Infomaniak	Analyse des documents PDF/HTML impossible
XInference / bge-reranker-v2-m3	Modèle non chargé ou erreur 404/500	Modèle non démarré, mémoire insuffisante, tunnel SSH défaillant	Détérioration de la qualité des réponses (RAG)
SearXNG	Page de recherche inaccessible	Conteneur down ou configuration réseau incorrecte	Recherche web impossible via Sophia
Endpoint OpenAI	Génération d'image échoue	Clé API invalide, réseau bloqué, quota dépassé	Fonctionnalité d'image inactive
Endpoints Infomaniak	Erreurs dans les menus Sophia (Connexions, Audio, etc.)	Tunnel SSH rompu, VM éteinte, problème d'authentification	Toutes les fonctions IA distantes (transcription, embedding) inactives
Logs des tunnels SSH	Messages répétés : `Connection refused`, `Broken pipe`, `Failed to connect`	VM Infomaniak éteinte, firewall, clé SSH expirée, réseau instable	Perte de contrôle et de données vers les services distants

---

> □□ **Recommandation** : Cette checklist peut être utilisée quotidiennement ou après chaque déploiement/mise à jour. En cas d'indisponibilité, consulter en priorité les **logs des conteneurs de tunnel SSH** et vérifier la **connectivité réseau vers la VM Infomaniak**.

Sophia

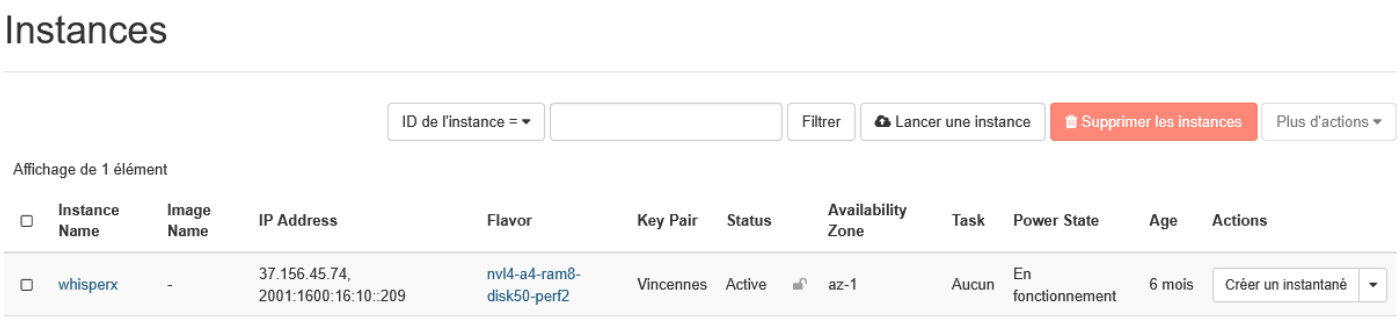
Redémarrage de la VM GPU sur Infomaniak

Aller sur https://api.pub2.infomaniak.cloud/horizon/project/

Les identifiants sont dans psono "Infomaniak openstack / Horizon"

Puis sur la page https://api.pub2.infomaniak.cloud/horizon/project/instances/

Une seule instance est configurée : "whisperx".



Dans le cas où la VM présentera un problème de blocage persistant, elle peut être redémarrée via l'option disponible en bout de ligne : "Redémarrer matériellement l'instance".

Le redémarrage prend 2 à 3 minutes.

Une fois relancée, les containers réapparaissent fonctionnels sur Portainer / Infomaniak.

Les accès à docling et xinference doivent être vérifiés (voir points 5 et 6 de https://formagent.vincennes.fr/books/intelligence-artificielle/page/verification-du-bon-fonctionnement).