AIエージェント ノーコード開発 ハンズオン講座

2025年10月

若松　信康

DELL Technologies

デルが提供する【無償】AI人財育成プログラム

2025年9月時点

育成

基礎学習

生成AIビジネス活用セミナー

オンライン

基礎から応用まで段階的に学習できる月例セミナー

のべ**5400名**以上参加

生成AI開発基礎

生成AIエンジニア養成講座

オンライン

Pythonを利用したビッグデータの整形・解析・マイニングから機械学習モデルの構築・評価まで学べるAIプログラミング実践講座

150社以上参加

生成AI活用促進ハンズオン

Microsoft 365 Copilotハンズオン講座

会場

Microsoft 365 Copilotの基本的な使用方法から実践的なTIPSまでハンズオンで学べる講座

200社以上参加

AIエージェントノーコード開発ハンズオン

AIエージェントノーコード開発講座

会場

オープンソースのDifyを使ってチャットフロー/ワークフロー/エージェントを作成する方法をハンズオンで学べる講座

New 2025年6月～

60社以上参加

DELL Technologies

アジェンダ

1. AIエージェント概論（20分）

2. 【ハンズオン】非エンジニアでもできる！
   AIエージェントノーコード開発実践ハンズオン（160分）

AIエージェント概論

「AIエージェント」とは

目標を達成するために、

- **Autonomous：自律的に**
- **Perception：環境・状況を認識し、**
- **Decision Making：意思決定を行い、**
- **Action：行動する**

AIシステム

2.従来の生成AIとの違いは？何ができるようになる？

従来の生成AIとの違い – ロボットに例えると…

生成AIロボット

「コーラ買ってきて」

LLM

「どこで買えるか教えてください。」

AIエージェントロボット

「はい、どうぞ」 「コーラ買ってきて」

行動：
1. 地図アプリでコンビニの場所を調べる
2. 地図アプリで移動ルートを特定する
3. 地図アプリを参照して移動する
4. コンビニで料金を払う
5. コンビニでコーラとおつりを回収する
6. 持って帰宅する

計画：
1. **至近の自販機の場所を調べる**
2. 自販機への移動ルートを特定する
3. 自販機へ移動する
4. 自販機で料金を投入する
5. 自販機でコーラとおつりを回収する
6. 帰宅する

AIエージェント

AI

記憶

振り返り

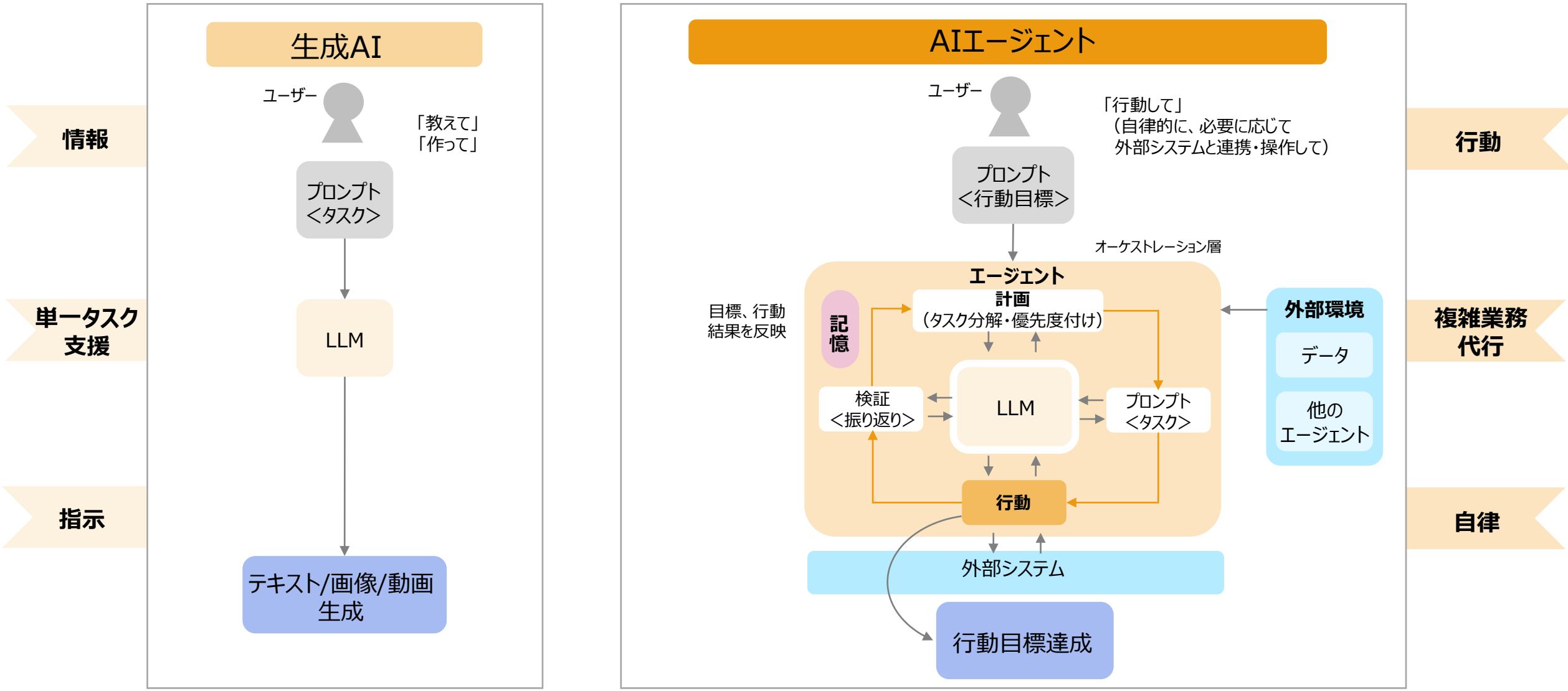
行動&結果：
1. 自販機アプリで至近の自販機を特定
2. 地図アプリで移動ルートを確認
3. 地図アプリを参照して移動
4. **自販機でコーラが売り切れていた**

再計画：
1. **至近のコンビニの場所を調べる**
2. コンビニへの移動ルートを特定する
3. コンビニへ移動する
4. コンビニで料金を投入する
5. コンビニでコーラを取る
6. 帰宅する

環境・状況認識
意思決定

2.従来の生成AIとの違いは？何ができるようになる？

従来の生成AIとの違い – 機能要素



AIエージェント開発・実装アプローチ

	初期フェーズ	発展フェーズ	成熟フェーズ
	RAG	**専門エージェント**	**マルチエージェント**
実装ステップ	**フェーズ1: RAG基盤構築** •知識ベース設計と構築（ドキュメント収集・整理） •ベクトルDB実装（Azure AI Search） •基本プロンプトエンジニアリング •初期評価メトリクス確立	**フェーズ2: 専門エージェント開発** •ドメイン固有知識の組み込み •マルチモーダル機能の追加（画像・文書認識） •会話フローの最適化 •フィードバックループの実装	**フェーズ3: マルチエージェントシステム** •エージェント間連携アーキテクチャ設計 •役割分担とオーケストレーション •複雑なワークフローの実装 •パフォーマンス最適化
開発プラットフォーム（例）	• Azure OpenAI Service：基盤LLMモデルの提供 • Azure AI Search：ベクトル検索およびハイブリッド検索機能	• Azure AI Agent Service：エージェント機能の実装 • Azure AI Studio：専門エージェントの設計と開発	• Autogen：マルチエージェントフレームワーク • Azure AI Orchestration Services：複雑なワークフロー管理
関連ツール・サービス（例）	• Azure Blob Storage：ドキュメント保存と管理 • Azure Cognitive Services (Document Intelligence)：文書解析と構造化 • Azure Data Factory：データ取り込みパイプライン • Azure Prompt Flow：初期プロンプト設計と実験 • Azure Monitor：基本的なパフォーマンスモニタリング • Azure Key Vault：APIキーと認証情報の安全な管理	• Azure Functions：カスタムツールとエージェント機能の実装 • Azure Computer Vision：画像認識機能 • Azure Form Recognizer：文書認識と解析 • Azure Speech Services：音声認識と合成（必要な場合） • Azure Bot Framework：チャネル連携 • Application Insights：詳細なパフォーマンス分析 • Azure Logic Apps：単純なワークフロー自動化	• Azure Kubernetes Service (AKS)：スケーラブルなエージェントデプロイ • Azure Service Bus：エージェント間メッセージング • Azure Durable Functions：複雑なステートフルワークフロー • Azure API Management：エージェントAPIの管理と監視 • Azure Cognitive Search（拡張機能）：複数ナレッジベース間の調整 • Azure DevOps：CI/CD自動化パイプライン • Azure Container Registry：エージェントコンテナイメージの管理

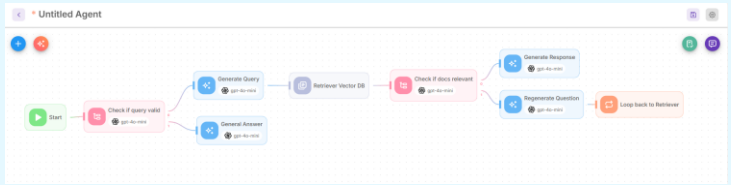
オープンソース & ノーコード AIエージェント開発プラットフォーム（汎用型）

成熟度

Dify

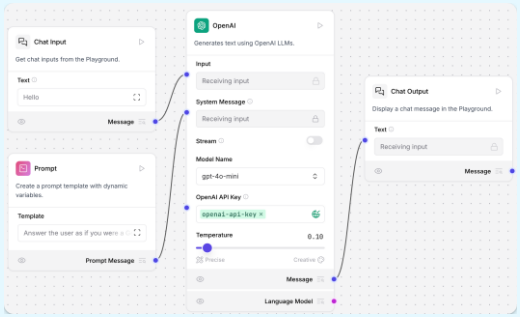
多様なノード構成で柔軟なワークフローを実現するプラットフォーム

Flowise



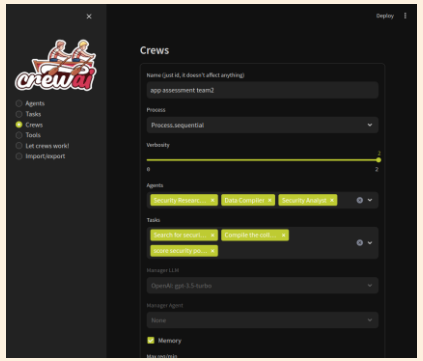
Langchainベースのワークフロー設計ツール

LangFlow



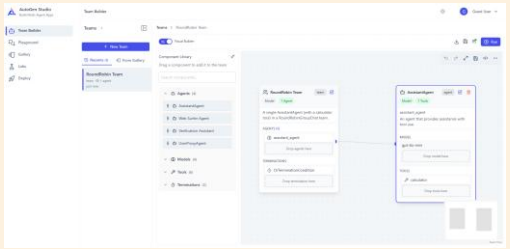
チャットフロー設計＋連携可能な軽量フローモデル

CrewAI



役割ベース協調型エージェント開発環境

AutoGen Studio

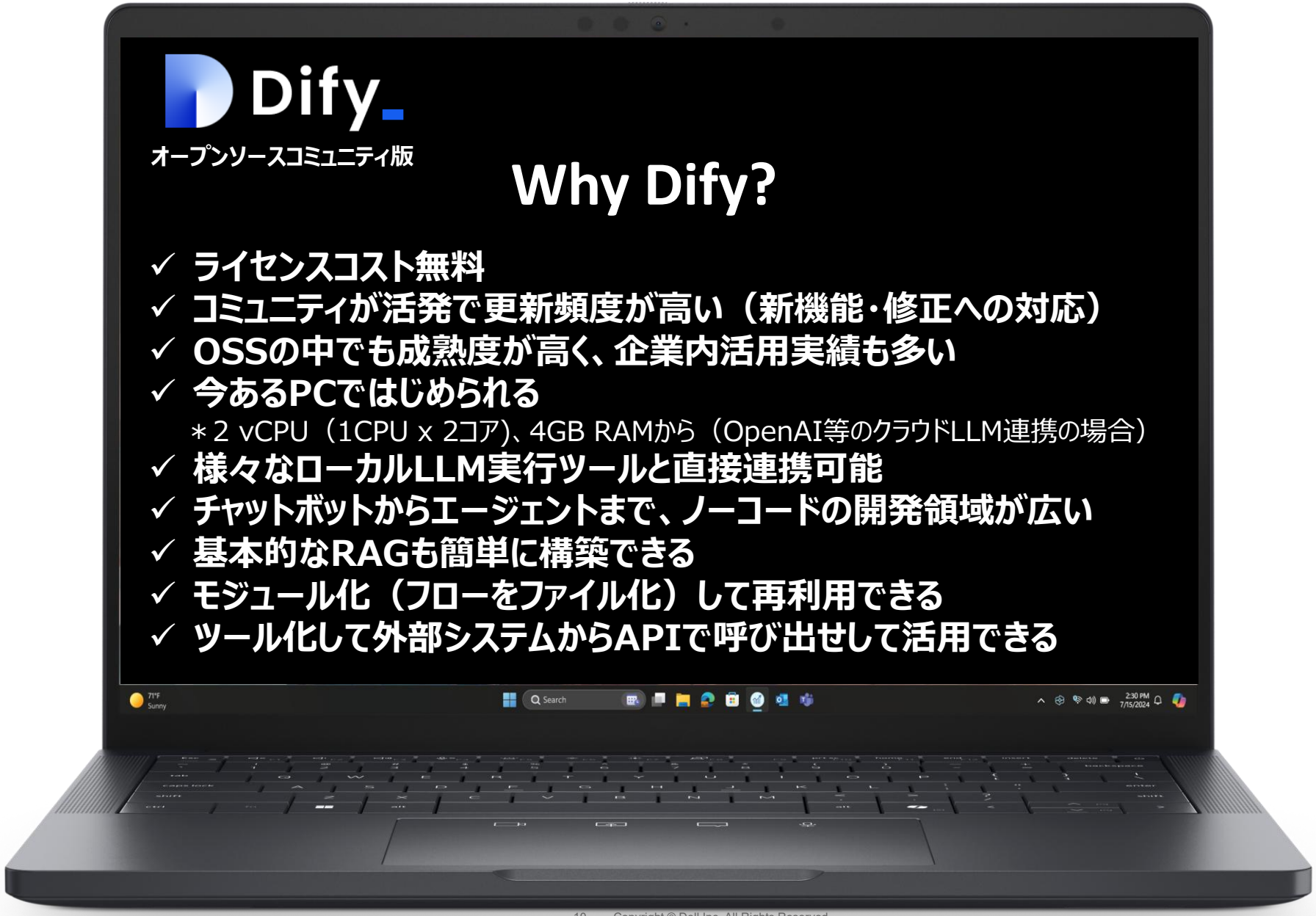


対話協調型エージェント開発環境

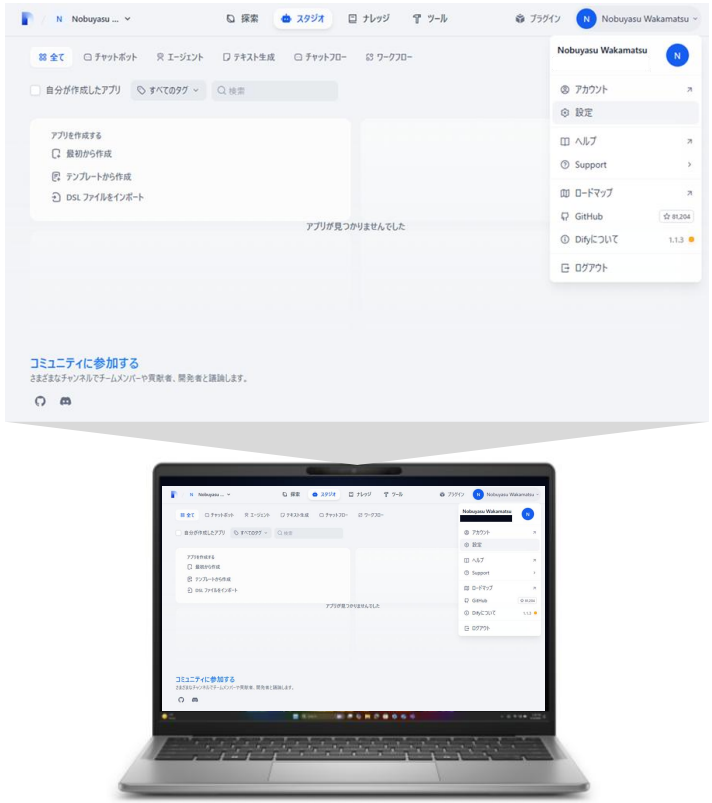
シングルエージェント

マルチエージェント

最初の一歩におススメAIエージェント開発プラットフォーム

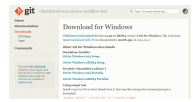


はじめに：0からローカルでDifyを使えるようになるまで



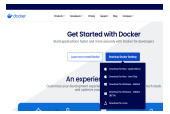
30分

① Gitのインストール：
GutHub上のDifyのソースコードをローカルにコピーするために必要



② Node.jsとnpmのインストール：
Difyのフロントエンドの実行環境のために必要

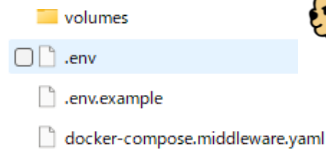
③ Docker Desktopのインストール/サインイン：
コンテナ上でDifyを実行するために必要



④ Difyのソースコード取得：Gitコマンド

```
git clone https://github.com/langgenius/dify.git
```

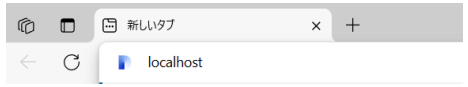
⑤ 環境変数の設定：セキュリティ関連



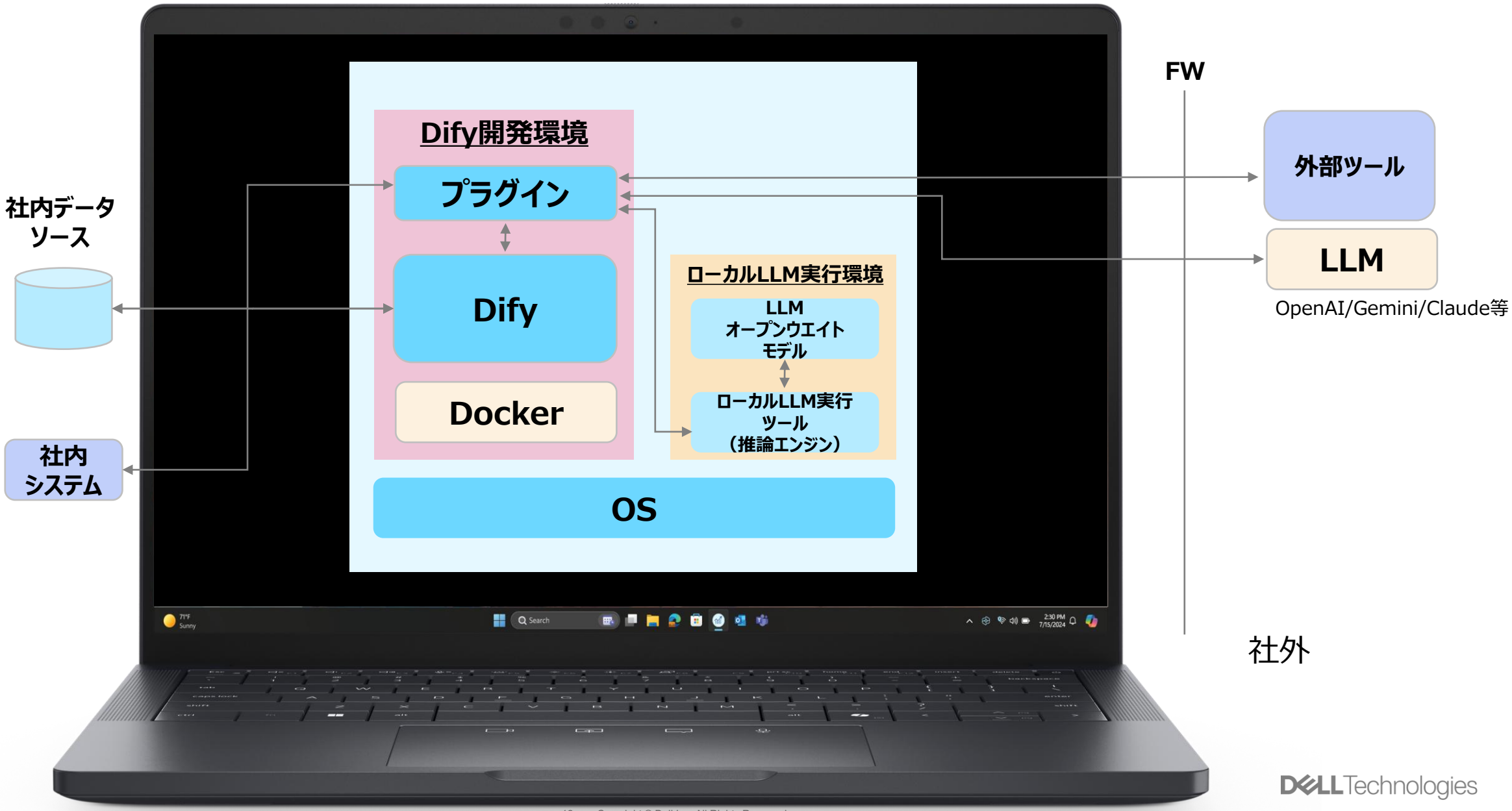
⑥ DockerおよびDifyのコンテナを起動する

```
cd dify/docker
docker compose up -d
```

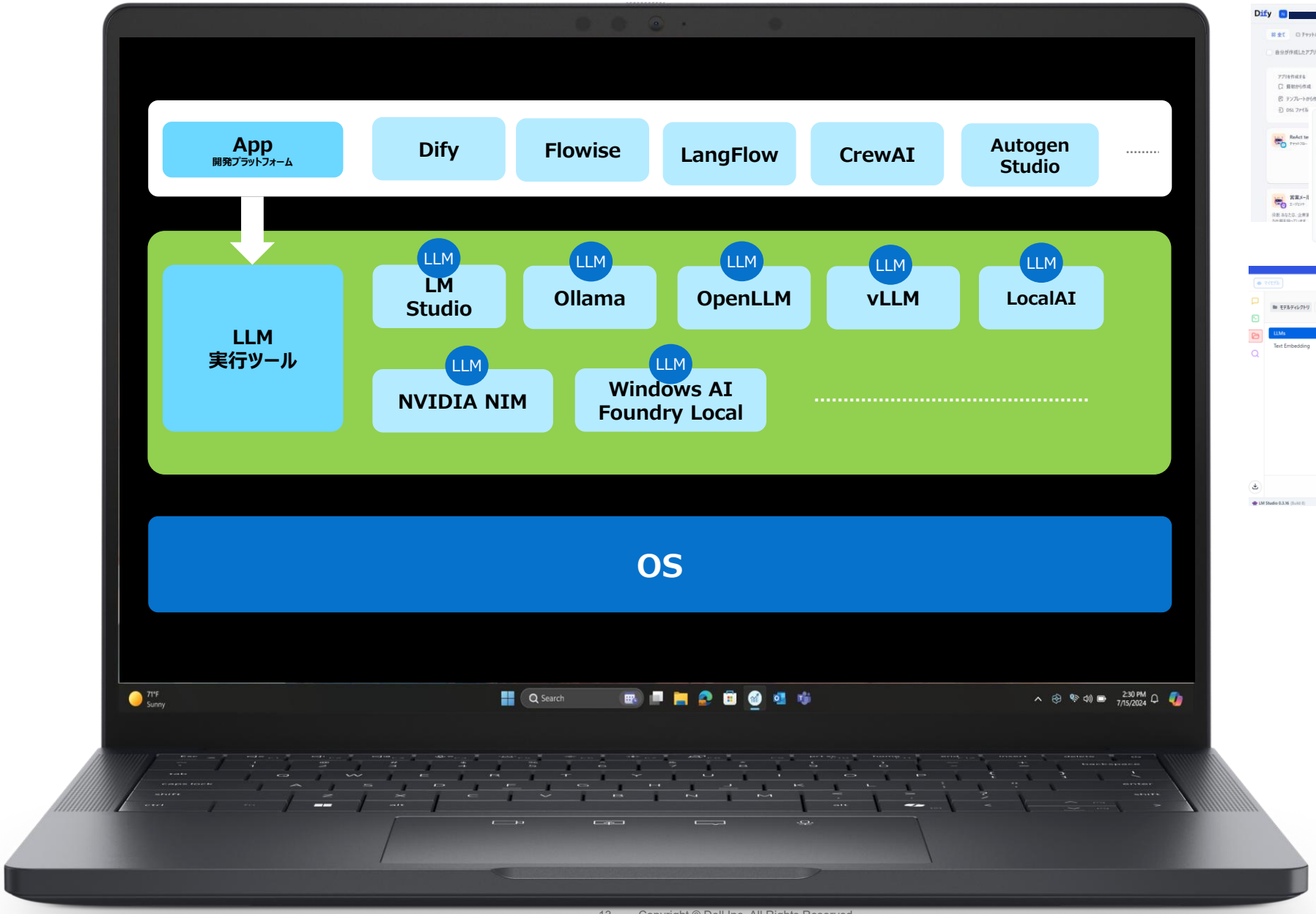
⑦ ブラウザでlocalhostにアクセスする

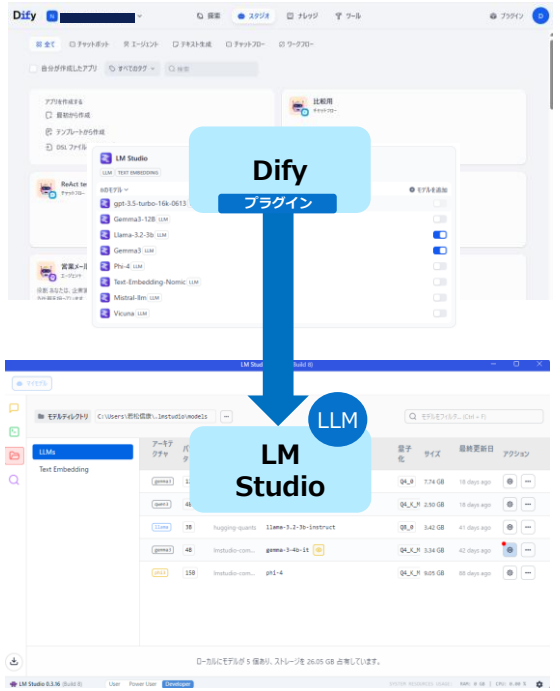


Difyローカル開発環境



ローカルLLMで開発する場合





【ハンズオン】非エンジニアでもできる！ AIエージェントノーコード開発実践ハンズオン

3つの作成方法

各AIアプリの特徴

チャットボット

LLM＋RAGを活用した
静的な情報検索・出力

エージェント

LLM+RAG+ツールを活用した
動的な情報検索・出力と操作

チャットフロー

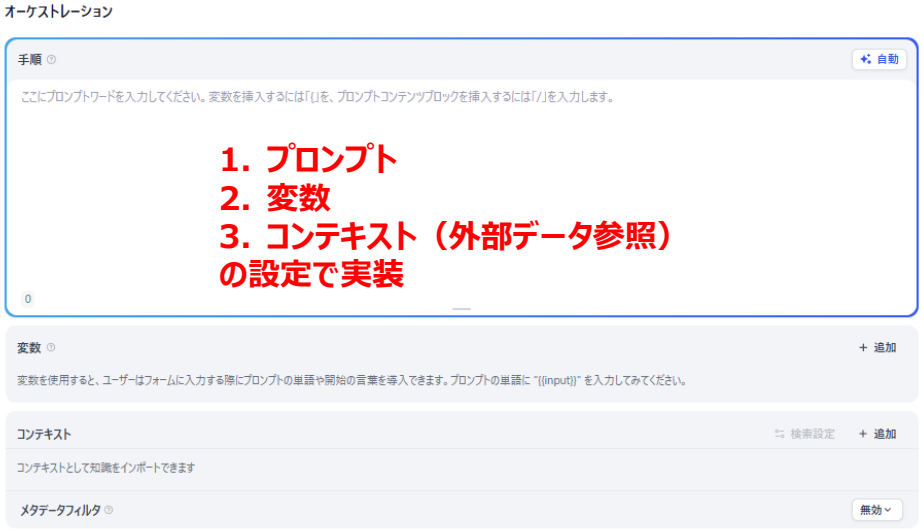
対話を主体とし、
依存関係のある複数タスク（マルチステップ）業務フローの実行

ワークフロー

ツール実行を主体とし、
それぞれが独立したタスクの連続から
なる業務フローの実行

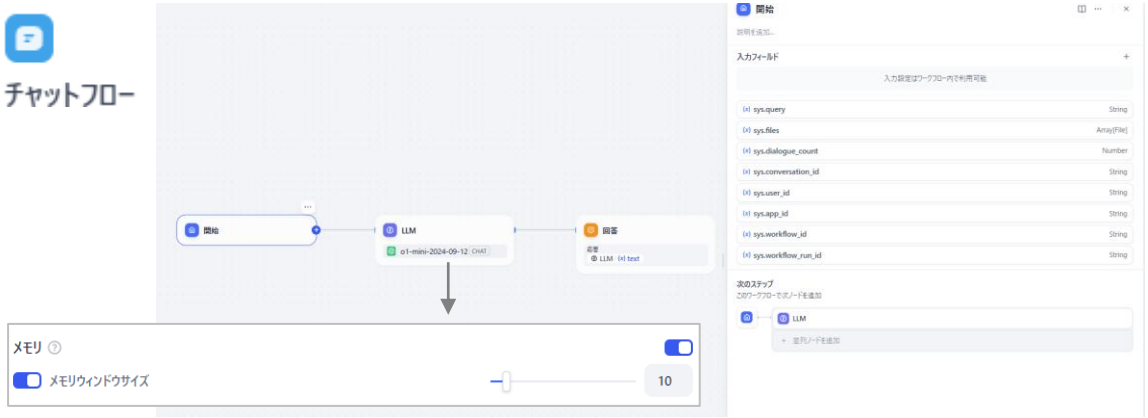
AIアプリ作成画面

チャットボット



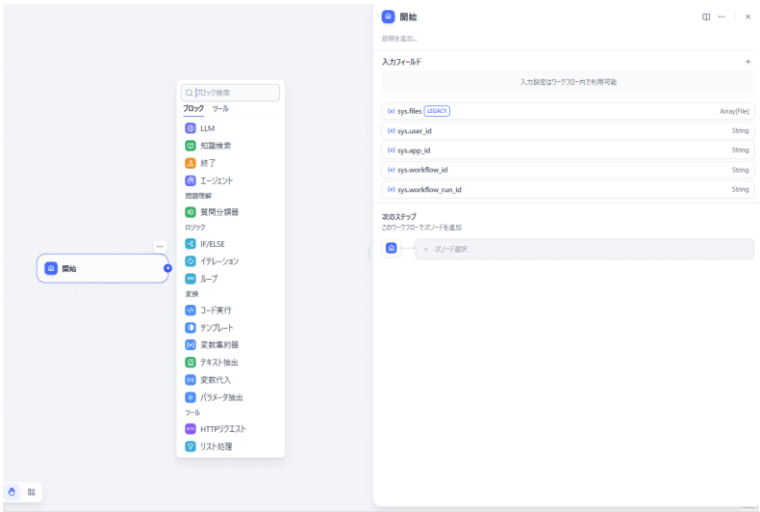
エージェント

チャットフロー



✓ メモリ機能で記憶をフロー内で保持/活用できる
→依存関係のある複数タスクをフローで実行できる

ワークフロー



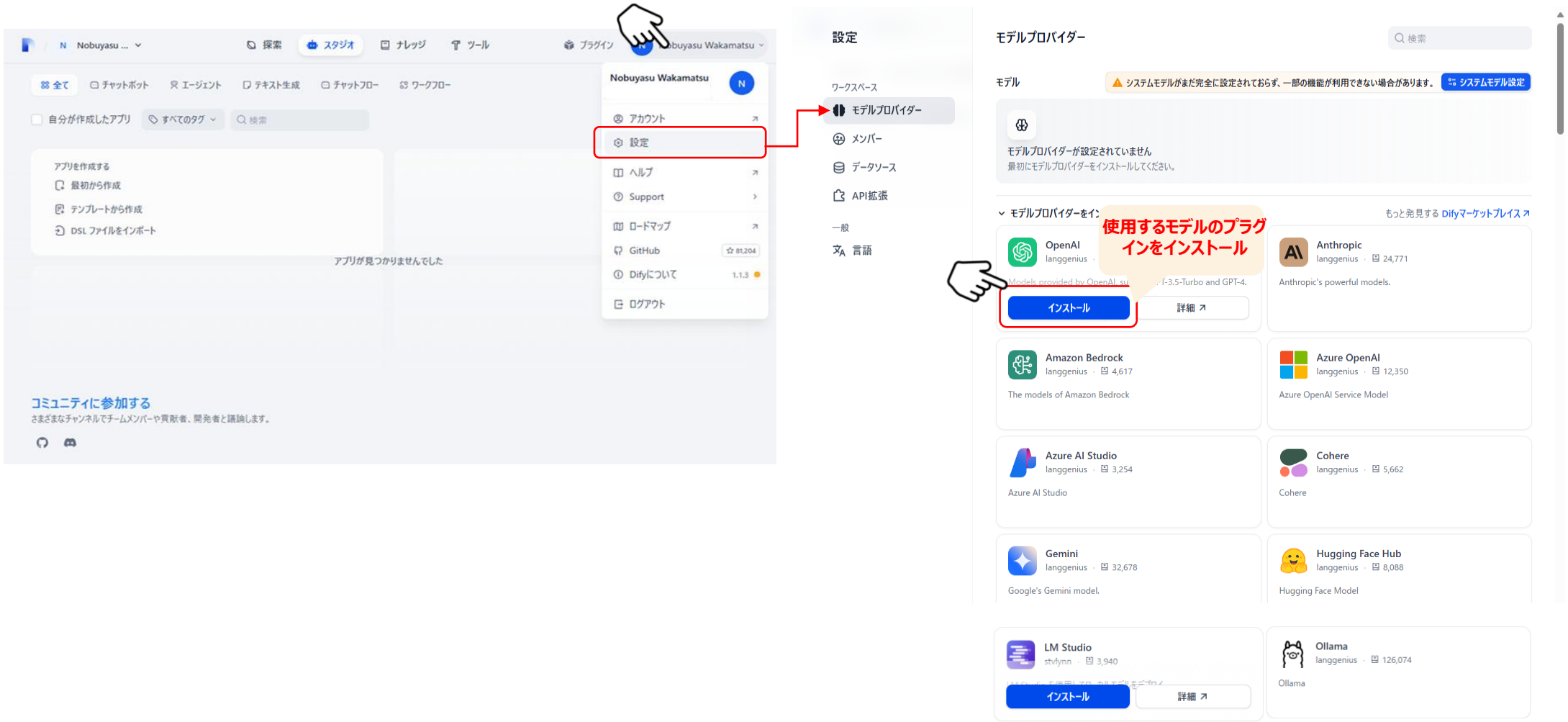
ノード(ブロックやツール)をつなげてフローを作成
各ブロックの入出力形式の設定やブロック内で処理する内容をプロンプト等で実装

準備編

1. LLMモデル設定
2. RAG構築
3. ツール連携

1. LLMモデル設定：プラグインのインストール

＜Dify＞モデルプロバイダーメニューから使用するLLMプラグインをインストール

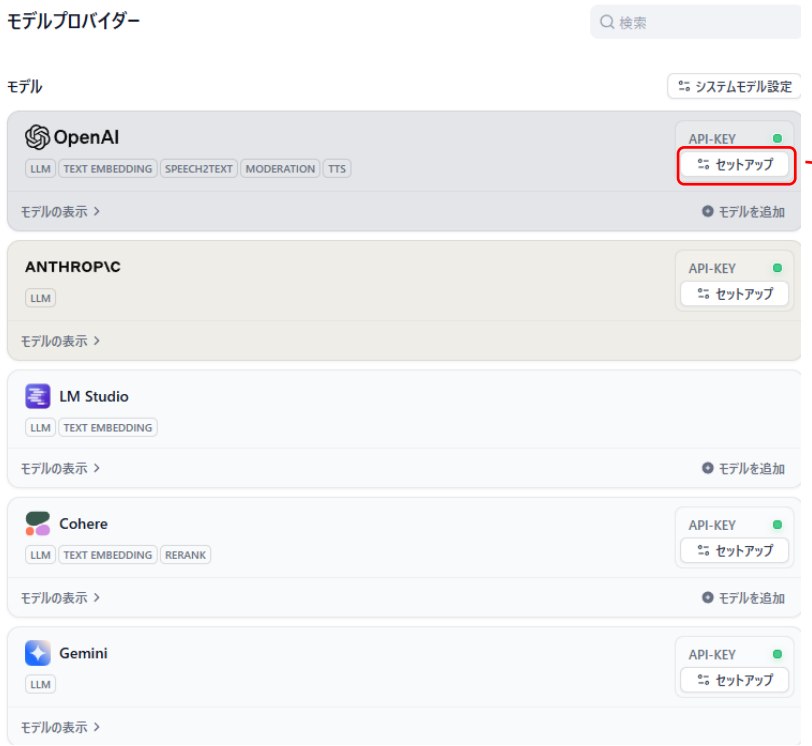


1. LLMモデル設定：外部LLM設定

API Key取得先

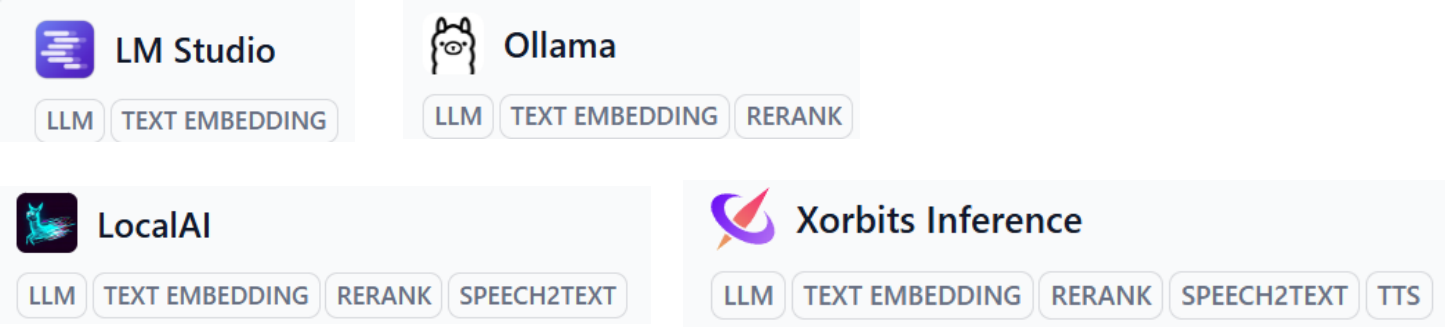
- ✓ OpenAI：https://platform.openai.com/docs/overview
- ✓ Google AI Studio：https://aistudio.google.com/u/1/prompts/new_chat
- ✓ Anthropic：https://console.anthropic.com/dashboard
- ✓ Cohere：https://dashboard.cohere.com/api-keys

＜Dify＞モデルプロバイダー設定



1. LLMモデル設定：ローカルLLM設定

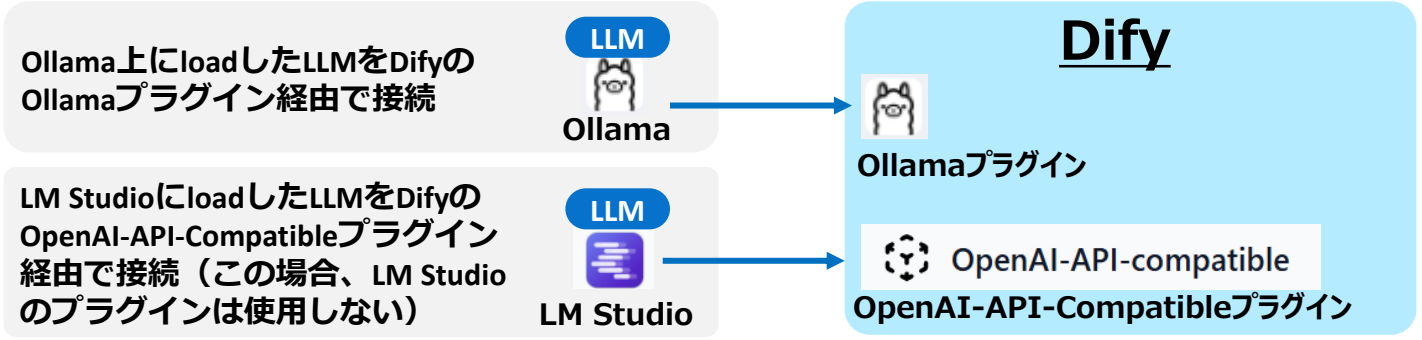
ローカルデバイス上にLLMを展開・利用できる実行ツール（Difyでサポートされているプラグイン）



ローカルLLMを選択

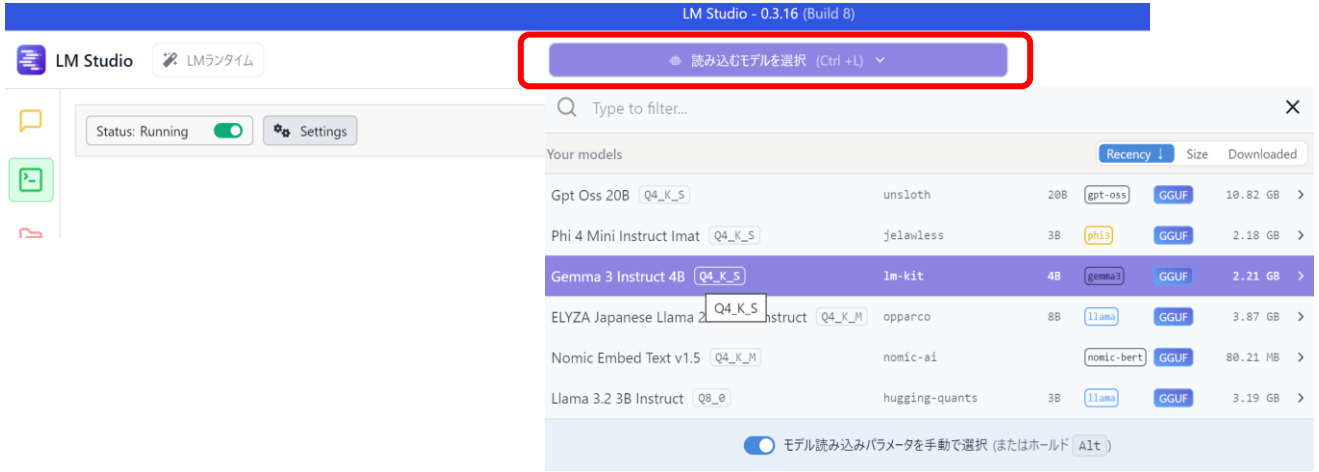
＜注意点＞

1. Chat-tunedモデル（対話形式の入出力ができるようトレーニングされたモデル：ユーザー入力/システムプロンプト/モデル応答を区別できるもの）のみ使用可能です。
2. LM Studioでは、現在Chat-tunedモデルがほぼ使用できないため、LM StudioからOpenAI API Compatible経由での接続にするか、Ollamaなどを使用する必要あり。

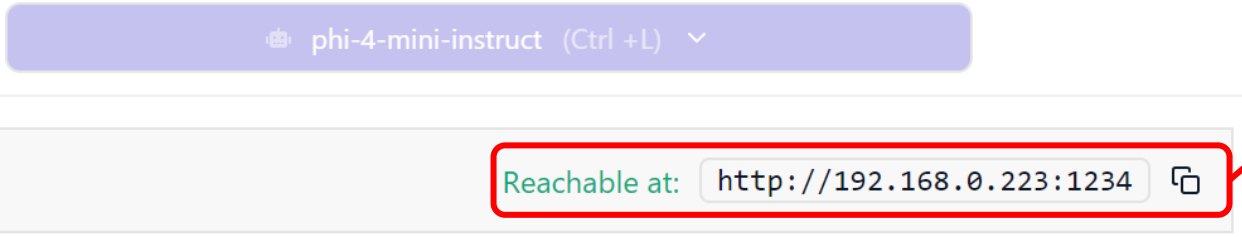


1. LLMモデル設定：ローカルLLM設定＜LM Studioの場合＞

(1) LM Studio上でダウンロードしたモデルをload



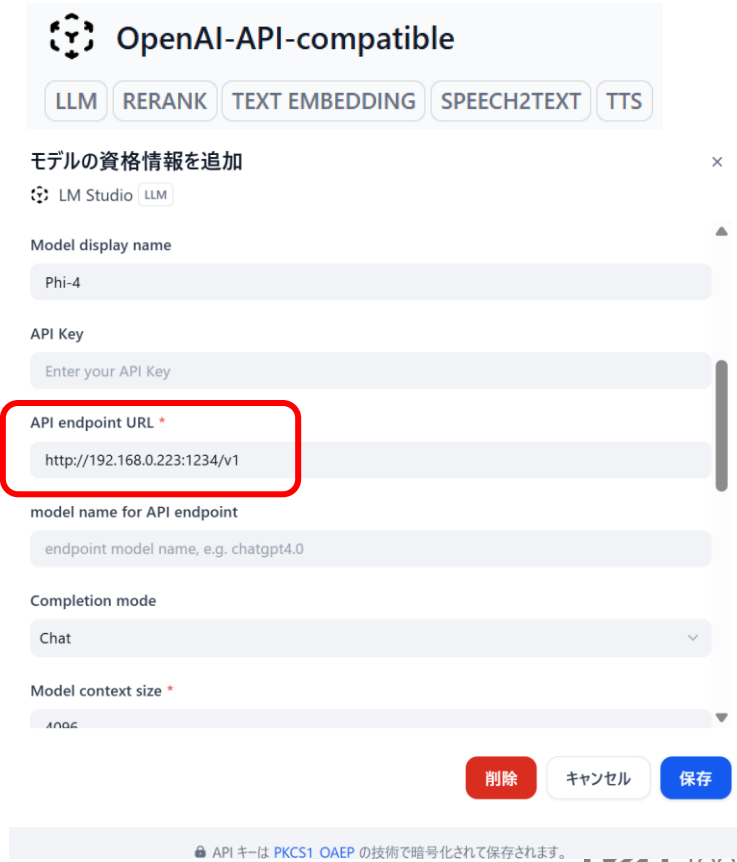
(2) 外部からアクセスできるIPを確認



＊後ろに/v1を追加して入力する。
・APIのバージョン1として指定。
・OpenAIのAPIでは、チャット機能（/v1/chat/completions）やモデル一覧の取得（/v1/models）といった様々な機能のエンドポイント（通信の出入り口）が、この/v1というパスの下に定義されているため、指定しないと連携が失敗する。

(3) OpenAI API Compatibleプラグイン上で新規登録時にLM Studioのアドレスを指定

OpenAI-API-Compatibleプラグイン



1. LLMモデル設定：機能

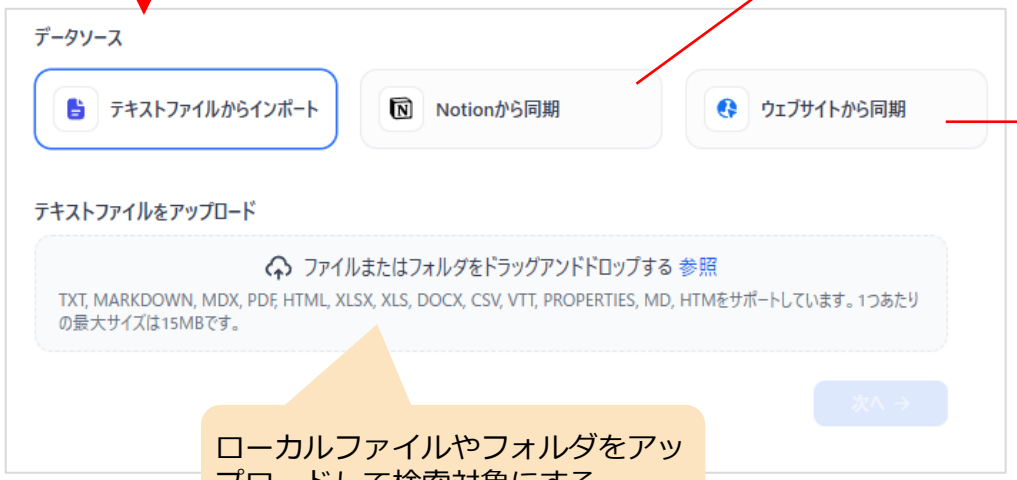
API機能	説明	ユースケース	具体例
LLM	テキスト生成、質問応答、文章作成などの自然言語処理タスクを実行	• カスタマーサポートチャットボット • コンテンツ自動生成 • データ分析レポート作成 • プログラミングコード生成 • 多言語翻訳	• Eコマースサイトでの商品に関する質問への自動応答 • マーケティングブログ記事の下書き自動生成 • 売上データから月次レポートの要約文作成 • 簡単な機能のJavaScriptコード生成 • 製品マニュアルの多言語展開
Text Embedding	テキストをベクトル表現に変換し、意味的類似性を数値化	• 類似ドキュメント検索 • レコメンデーションシステム • クラスタリング分析 • セマンティック検索エンジン • 知識ベースのインデックス作成	• 「投資戦略」を検索すると「資産配分」の記事も表示 • 閲覧した記事と意味的に関連する他の記事を推薦 • 顧客フィードバックを自動的にテーマ別に分類 • 「車の故障」で検索すると「エンジントラブル」の記事も表示 • 社内文書を意味ベースで整理・検索可能に
Rerank	検索結果やドキュメントセットを関連性に基づいて並べ替え	• 検索エンジン結果の最適化 • 質問応答システムの精度向上 • レコメンデーションの優先順位付け • ナレッジベース検索の改善 • 情報検索システムの高度化	• 「初心者向けプログラミング」検索で実際に初心者に適した結果を上位表示 • 「パスワードをリセットする方法」の質問に最も直接的な回答を優先 • ユーザーの好みに合った映画を上位に表示 • 「払い戻し方法」検索で最新の正確な手順を最上位に表示 • 法律事務所での判例検索で最も関連性の高い事例を優先表示
Speech to Text	音声をテキストに変換	• 会議の自動文字起こし • 音声コマンドシステム • 電話対応の自動化 • 字幕生成 • 音声メモのテキスト化	• Zoomミーティングの全文を自動的にテキスト化して共有 • 「明日の予定を教えて」と話しかけるとカレンダーを検索 • カスタマーサポート電話の内容を自動記録・分析 • YouTubeビデオの自動字幕生成 • 運転中の音声メモをテキスト化してTodoリストに追加
TTS (Text to Speech)	テキストを自然な音声に変換	• アクセシビリティ機能の提供 • オーディオブック作成 • 音声アシスタント • 教育コンテンツの音声化 • 通知やアラートの音声読み上げ	• 視覚障害者向けのウェブサイト読み上げ機能 • ブログ記事を自動的にポッドキャスト形式に変換 • チャットボットの返答を音声で提供 • 言語学習アプリでの発音例の提供 • 重要なスマートフォン通知を運転中に読み上げ

<モデル毎に呼び出せる機能一覧>

https://docs.dify.ai/getting-started/readme/model-providers

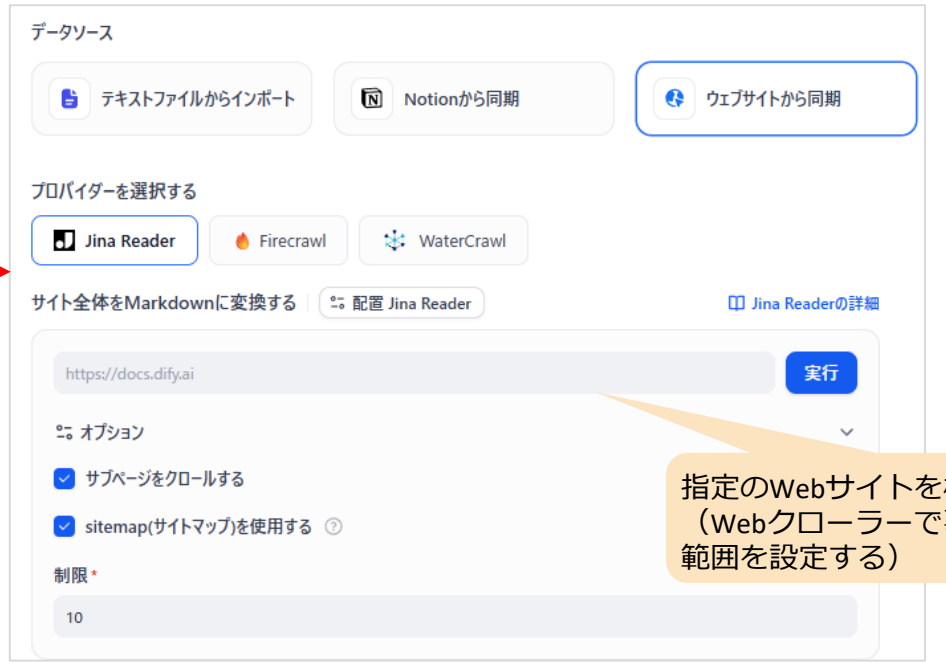
2. コンテキスト（RAG）の設定方法

トップメニュー「ナレッジ」から「ナレッジベースを作成」



ローカルファイルやフォルダをアップロードして検索対象にする

Notionと接続して検索対象にする



指定のWebサイトを検索対象にする（Webクローラーで事前に取得する範囲を設定する）

2. コンテキスト（RAG）の設定方法

ウェブサイトの情報を検索対象とするための設定

サイトからAPIを取得して入力

https://jina.ai/reader/

https://www.firecrawl.dev/app/api-keys

https://app.watercrawl.dev/dashboard/api-keys

2. コンテキスト（RAG）の設定方法

Webクローラーの比較

ツール名	特徴	ユースケース	コスト	拡張性・制御性
Jina Reader	- URL→Markdown変換 - CSSセレクタ／ブラウザエンジン指定 - 画像キャプション生成、Shadow DOM抽出 - ReaderLM-v2による実験的HTML→Markdown/JSON変換	- 単一ページを速やかにMarkdown化 - 特定要素のみ抽出し細かい前処理が必要 - 認証クッキー／プロキシ経由取得	クラウド版：20リクエスト/分/IPまで無料 OSS版：無料（環境構築コストのみ）	高（パラメータ設定多数）
FireCrawl	- サブページ含むサイト全体クロール - クロール深度・ページ上限・除外/含むパス設定 - OSS版で無制限、自前サーバ運用可 - JSブロックやプロキシ対応	- 企業サイトやドキュメントサイト一括取り込み - RAG基盤の初期構築 - 内部ネットワーク含む大量クロール	クラウド版：500ページまで無料 OSS版：無料（環境構築コストのみ）	中（基本設定UIのみ）
WaterCrawl	- JavaScriptレンダリング対応 - PDF化／スクリーンショット生成 - プラグインシステムで独自AI処理パイプライン - 構造化JSON出力、リアルタイムステータス追跡	- SPA/Ajax多用サイトなど動的コンテンツ - フィールド単位での構造化データ抽出 - カスタムプラグイン開発が必要な場合	クラウド版：1000ページ/月 まで無料 OSS版：無料（環境構築コストのみ）	非常に高（プラグイン開発可）

＜参考＞選択する際の主な判断材料

対象サイトの性質
•静的HTML中心：Jina Reader or FireCrawl
•動的・JavaScript多用：WaterCrawl or Jina Reader（Browser Engineオプション）

出力フォーマット
•単純Markdown：Jina Reader or FireCrawl
•構造化JSON／カスタムデータ：WaterCrawl

運用コスト・スケール
•少量頻度：Jina Reader（無料枠活用）
•大量クロール：FireCrawl OSS or WaterCrawl（自社インフラ）

拡張性・制御性
•簡易：FireCrawl
•細かい調整：Jina Reader
•プラグイン開発：WaterCrawl

コストとレート制限
•単発利用：Jina Reader（無料）
•大規模・商用：FireCrawl OSS／WaterCrawl（セルフホスティングプラン）

2. コンテキスト（RAG）の設定方法

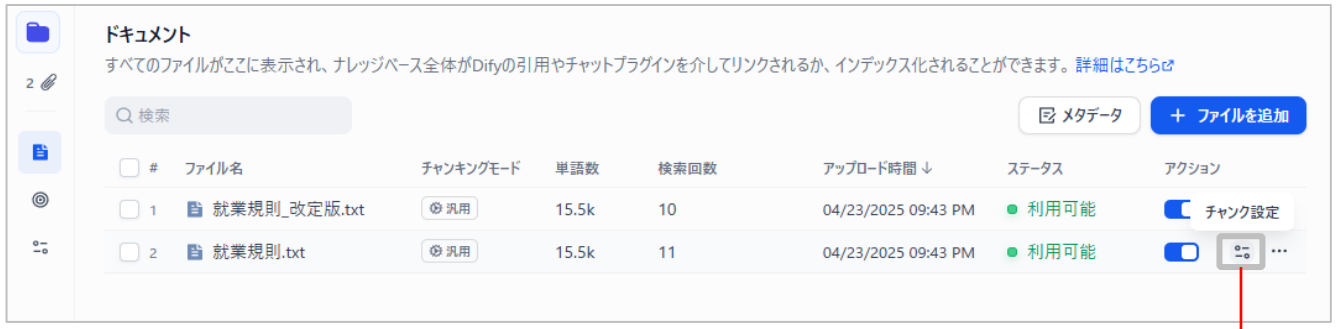
ナレッジベースとして追加され使用できる状態になると＜設定＞「データソース」にステータスが反映され、「コンテキスト」や「知識検索」ノードで選択利用可能となります。

2. コンテキスト（RAG）の設定方法

チャンク：意味を持ったテキストの塊。全文検索せずに効率的にマッチする検索結果を見つけるために元のテキストを分割したもの。

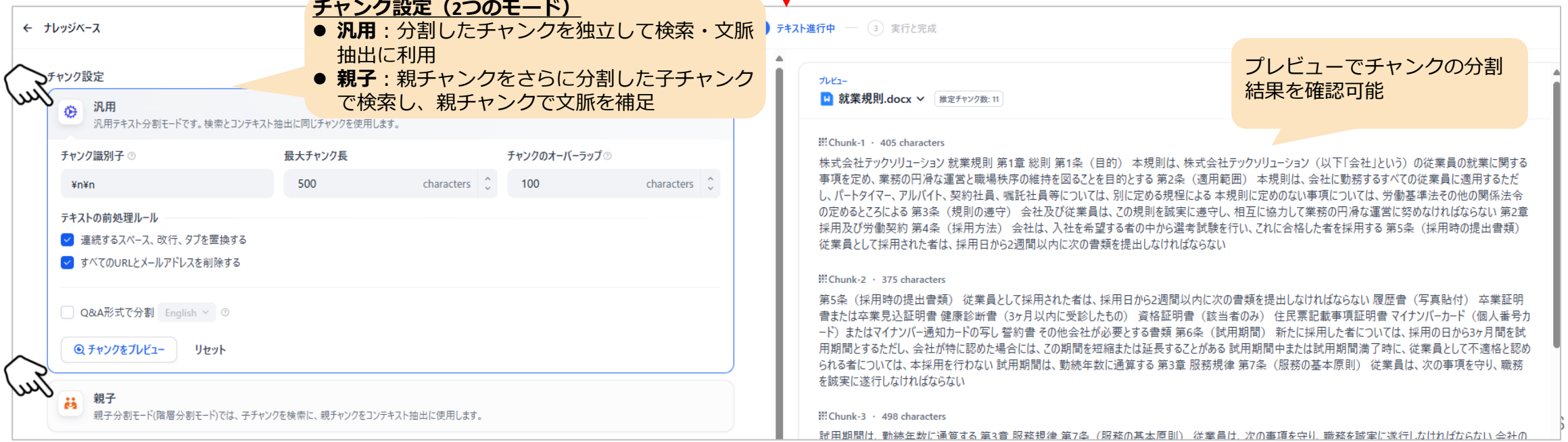
最初に参照ファイルをアップロードしたとき

あとから修正する場合



チャンク設定（2つのモード）

- **汎用**：分割したチャンクを独立して検索・文脈抽出に利用
- **親子**：親チャンクをさらに分割した子チャンクで検索し、親チャンクで文脈を補足

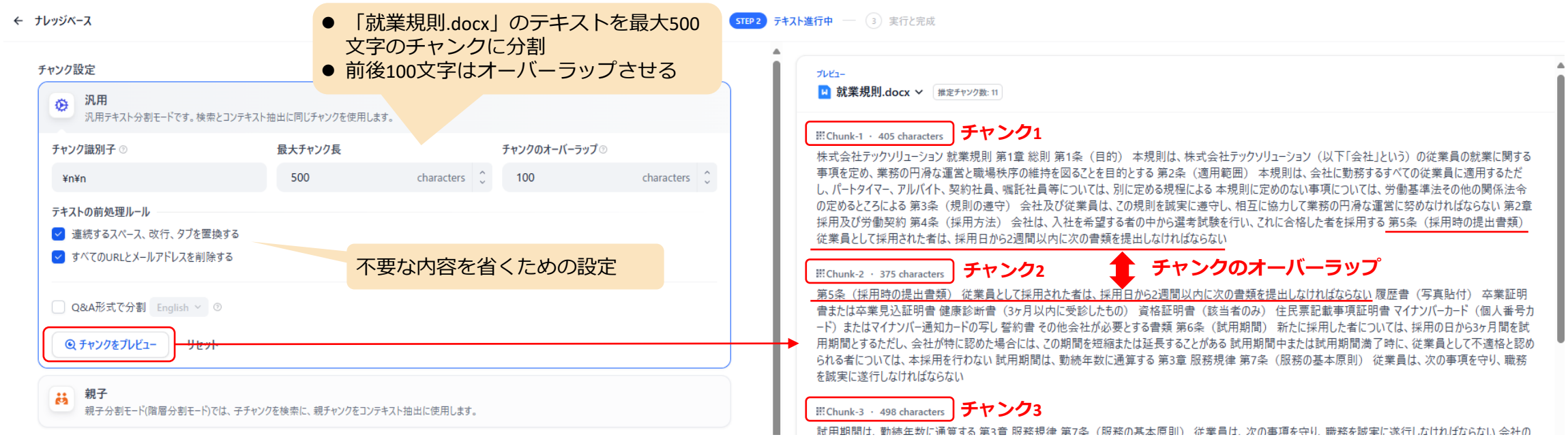


プレビューでチャンクの分割結果を確認可能

2. コンテキスト（RAG）の設定方法

ナレッジベースの応用設定：RAG「チャンク設定」

<汎用モードの場合>

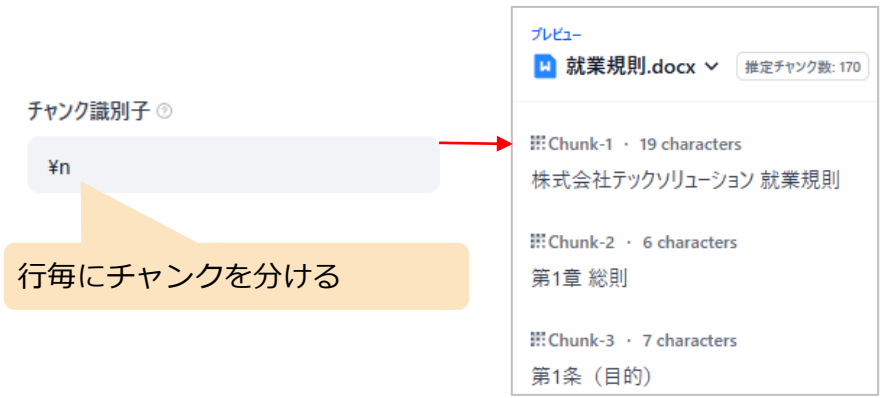


- 「就業規則.docx」のテキストを最大500文字のチャンクに分割
- 前後100文字はオーバーラップさせる

不要な内容を省くための設定

チャンク識別子：
- ¥n¥n（二重改行）：テキスト内の空行（段落区切り）を抽出してチャンクを生成（デフォルト）
- ¥n（改行）：各行をチャンクとして分割
- “文字列”：指定の文字列ごとに分割

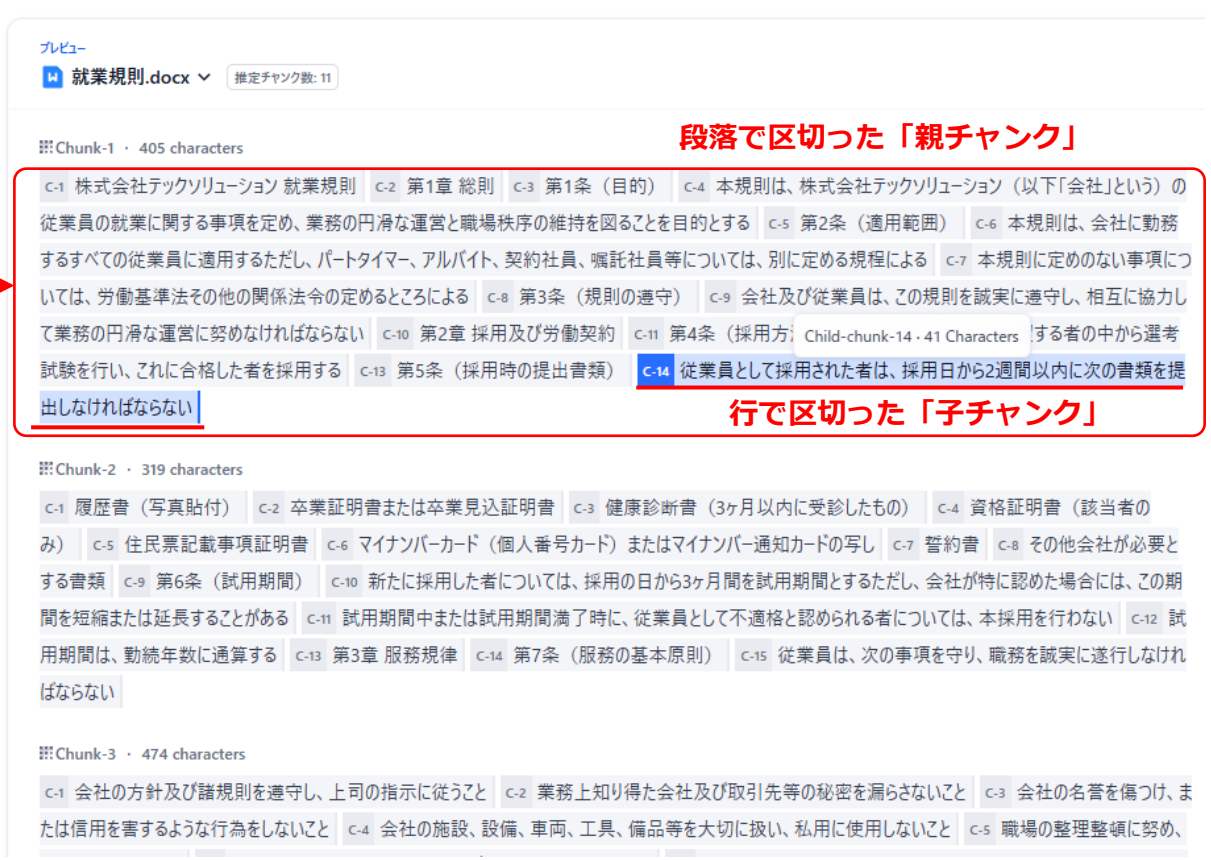
最大チャンク長：チャンクの最大文字数（設定できる最大は4000）
チャンクのオーバーラップ：チャンク間で重複して保持するテキストの文字数。テキストの意味のまとまりをチャンク内で保持するために同じ文章をチャンク間で重複して保持させる。



2. コンテキスト（RAG）の設定方法

ナレッジベースの応用設定：RAG「チャンク設定」

<親子モードの場合>



親子設定（2つのモード）

- **段落**：識別子（段落等）で親チャンクを区切る設定→親チャンクが過剰にならず処理コストを抑えられる。FAQやマニュアル等段落で論理的に区切られたテキストに最適。
- **全文**：親チャンクをドキュメント全文にする設定→全体を通して関連性を把握したい短文資料に最適。（10,000トークンを越えると末尾が切り捨てられる）

2. コンテキスト（RAG）の設定方法

ナレッジベースの応用設定：RAG「インデックス方法設定」

インデックス方法設定（2つのモード）

高品質：

- 分割されたテキストチャンクをEmbeddingモデル（例：text-embedding-3-largeなど）で数値ベクトルに変換し、大量のテキスト情報を効率的に圧縮・保存することで、ユーザーの質問とマッチングする精度が向上します。
- 「ベクトル検索」「全文検索」「ハイブリッド検索」の3つのオプションが用意されており、意図やドキュメント特性に応じて最適な手法を選択できます。

経済的：

- 各テキストチャンク内から最大10個のキーワードを抽出し、「逆引きインデックス方式」のみでマッチングを行います。これにより検索精度はやや低下しますが、トークン消費や外部API呼び出しが不要でランニングコストを抑えられます。
- 「転置インデックス」（＝「逆引きインデックス」）でTop-Kのみ設定可能。（Top-Kの値が大きいほど呼び出される候補文の数が多くなります）

2. コンテキスト（RAG）の設定方法

ナレッジベースの応用設定：RAG「検索設定」

検索設定（3つのモード）

- **ベクトル検索**：ユーザーが入力した質問をベクトル化し、クエリテキストのベクトルを生成し、クエリベクトルとナレッジベース内の対応するテキストベクトル間の距離を比較し、隣接する分割コンテンツを探します。
- **全文検索**：文書内のすべての語彙をインデックス化し、ユーザーが質問を入力した際に、キーワード検索でテキストマッチングしてテキストを抽出します。
- **ハイブリッド検索**：全文検索とベクトル検索、またはRerankモデルを同時に実行し、クエリ結果からユーザーの質問に最もマッチする最良の結果を選択します。

設定項目

<共通>
- **Rerankモデル**：ベクトル検索で取得した候補チャンクの順位を外部モデルを使用して再評価する（ここではCohereのモデルrerank-v3.5を使用）ことで回答精度を向上させることが可能
- **Top-K**：値が大きいほど呼び出される候補文の数が多くなります。
- **スコア閾値**：抽出するテキストの類似度の閾値。類似度の値が大きいほど候補テキストは少なくなります。

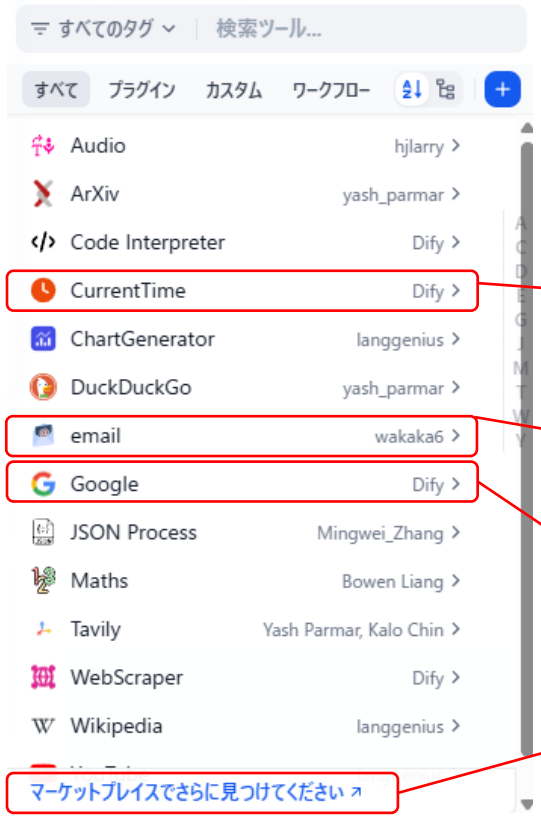
<ハイブリッド検索>
- **ウエイト設定**：セマンティック（意味）検索とキーワード検索のどちらを優先するかの重み付け設定

DELL Technologies

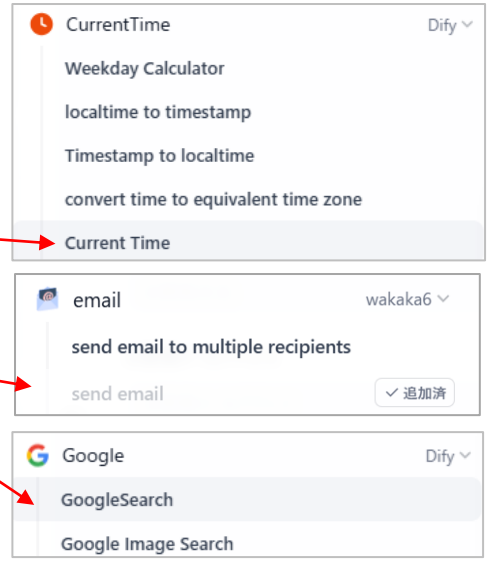
2. コンテキスト（RAG）の設定方法

ナレッジベース：検索結果のテスト

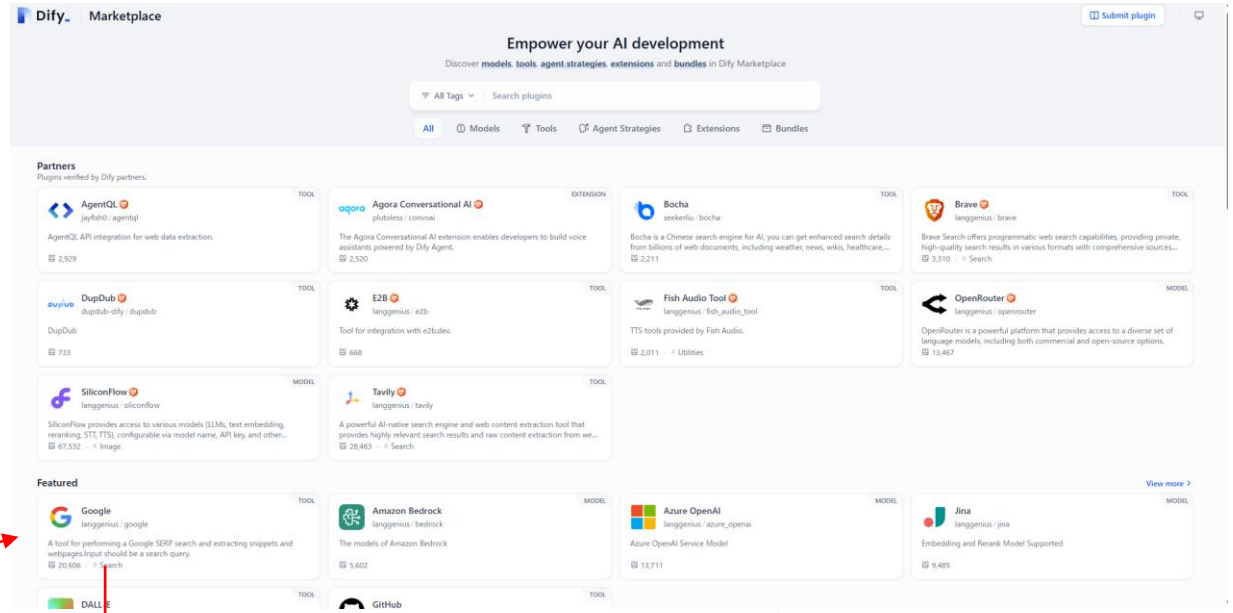
3. ツール連携設定



クリックして追加



候補に表示されていない場合：
マーケットプレイスでプラグインをインストール
してから追加

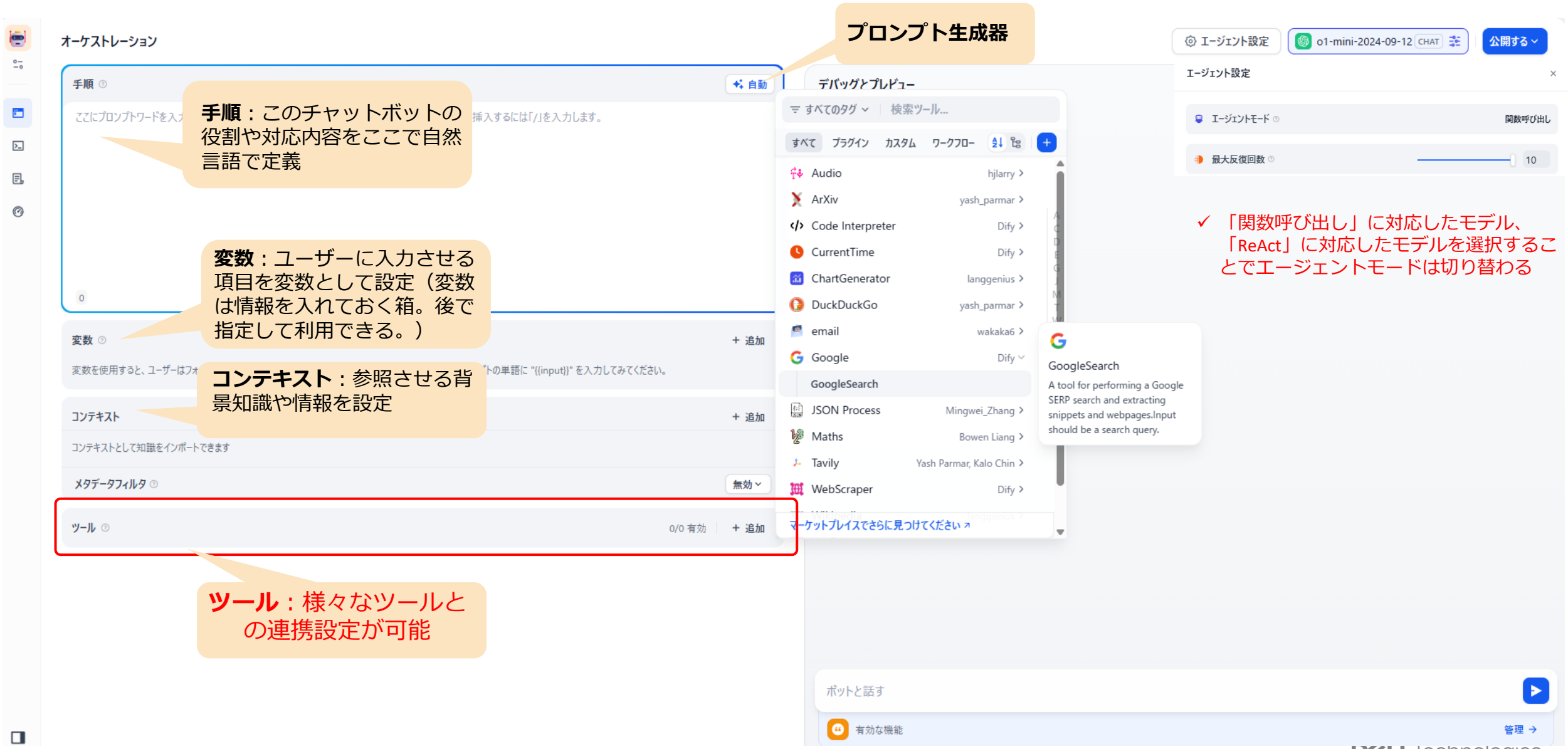


＜Google SearchはAPI取得必要＞

エージェントを作ろう①

営業メール作成・配信エージェント

エージェントの作成方法：設定項目



手順：このチャットボットの役割や対応内容をここで自然言語で定義

変数：ユーザーに入力させる項目を変数として設定（変数は情報を入れておく箱。後で指定して利用できる。）

コンテキスト：参照させる背景知識や情報を設定

ツール：様々なツールとの連携設定が可能

✓ 「関数呼び出し」に対応したモデル、「ReAct」に対応したモデルを選択することでエージェントモードは切り替わる

営業メール作成・配信エージェント＜シナリオ＞

人がすること	エージェントがすること	人がすること	エージェントがすること
自分の会社名/名前と相手先の会社名/名前を入力	1. 相手先企業情報をWebで調査 2. 企業情報に応じた提案製品を検索 3. それらを踏まえたメール文を作成	メール文のレビュー 修正指示	メール配信

チャットのフォームに入力

- お客様企業名
- 担当者名
- 担当者メールアドレス

…

ツール活用
メール送付先企業の戦略と課題を確認

RAG
企業にあった提案ソリューションを抽出

メール文作成

メール文のレビュー修正依頼

ツール活用
メール送信

Human in the loop

エージェントを作ろう：営業メール作成エージェント＜作成ステップ＞

作成ステップ

① **入力フォームを作成：**
- ✓**【設定項目】「変数」**：以下を設定
  - お客様会社名/氏名/メールアドレス（必須）
  - あなたの会社名/氏名/メールアドレス（必須）
  - メール件名/メール本文（オプション）

  1.変数がユーザー入力フォームの項目として設定される
  2.「手順」で変数を指定することで、メール本文に差し込む

② **相手先企業の情報をWebで調査するために：**
- ✓**【設定項目】「ツール」**：以下を設定
  - **GoogleSearch**：相手先企業の戦略と課題を検索調査
  - **Time**：現在の日時を確認（打ち合わせ候補日指定に使う）

③ **企業情報に応じた提案製品を検索するために：**
- ✓**【設定項目】「コンテキスト」**：参照する社内製品情報を指定する
  - （今回の場合は）dell.comページ

④ **1,2の結果を踏まえたメール文を作成、内容の確認を依頼、配信**
- ✓**【設定項目】「プロンプト」**：相手先企業の情報をツールで調べることなどした上でメールを作成、レビュー、配信するようSTEPをプロンプトで指示
  - 変数として設定しているものをリンクとして入れ込む（”/”を入力して表示される設定済変数候補から選択）

手順①：変数の設定

変数 ⓘ　＋ 追加

- (x) RECIPIENT_COMPANY · お客様会社名　REQUIRED　string
- (x) RECIPIENT_NAME · お客様氏名　REQUIRED　string
- (x) RECIPIENT_EMAIL_ADDRESS · お客様メールアドレス　REQUIRED　string
- (x) MY_COMPANY · あなたの会社名　REQUIRED　string
- (x) MY_NAME · あなたの名前　REQUIRED　string
- (x) SENDER_ADDRESS · あなたのメールアドレス　REQUIRED　string
- (x) EMAIL_SUBJECT · メール件名　string
- (x) EMAIL_CONTENT · メール本文　paragraph

フィールドタイプ	変数名	ラベル名	最大長	必須
短文	RECIPIENT_COMPANY	お客様会社名	48	○
短文	RECIPIENT_NAME	お客様氏名	48	○
短文	RECIPIENT_EMAIL_ADDRESS	お客様メールアドレス	48	○
短文	MY_COMPANY	あなたの会社名	48	○
短文	MY_NAME	あなたの名前	48	○
短文	SENDER_ADDRESS	あなたのメールアドレス	48	○
短文	EMAIL_SUBJECT	メール件名	48	
段落	EMAIL_CONTENT	メール本文	48	

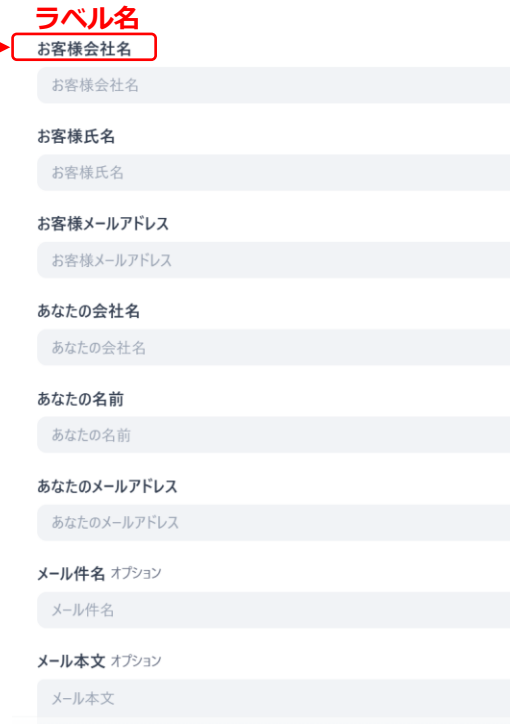
この文字列をそのままコピーペーストしてください。

デフォルトのままでOK

「必須」のチェックを外す

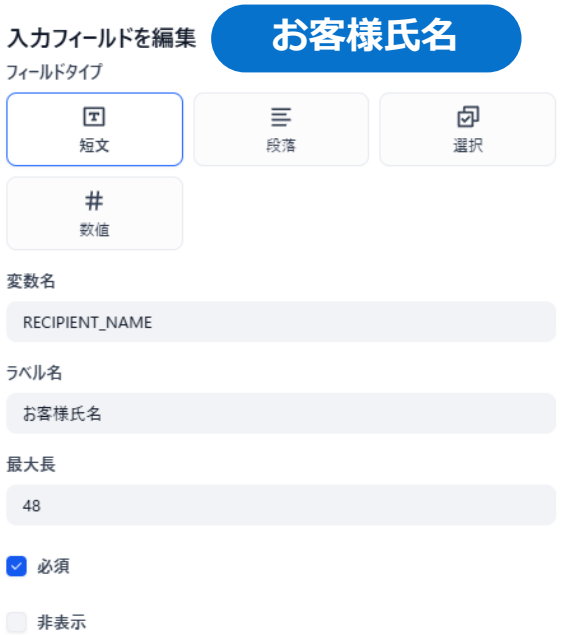
変数設定

ユーザー入力フォーム



手順①：変数の設定


入力フィールドを編集
お客様会社名
フィールドタイプ
短文
段落
選択
数値
変数名
RECIPIENT_COMPANY
ラベル名
お客様会社名
最大長
48
必須
非表示


入力フィールドを編集
お客様氏名
フィールドタイプ
短文
段落
選択
数値
変数名
RECIPIENT_NAME
ラベル名
お客様氏名
最大長
48
必須
非表示


入力フィールドを編集
お客様メールアドレス
フィールドタイプ
短文
段落
選択
数値
変数名
RECIPIENT_EMAIL_ADDRESS
ラベル名
お客様メールアドレス
最大長
48
必須
非表示


入力フィールドを編集
あなたの会社名
フィールドタイプ
短文
段落
選択
数値
変数名
MY_COMPANY
ラベル名
あなたの会社名
最大長
48
必須
非表示

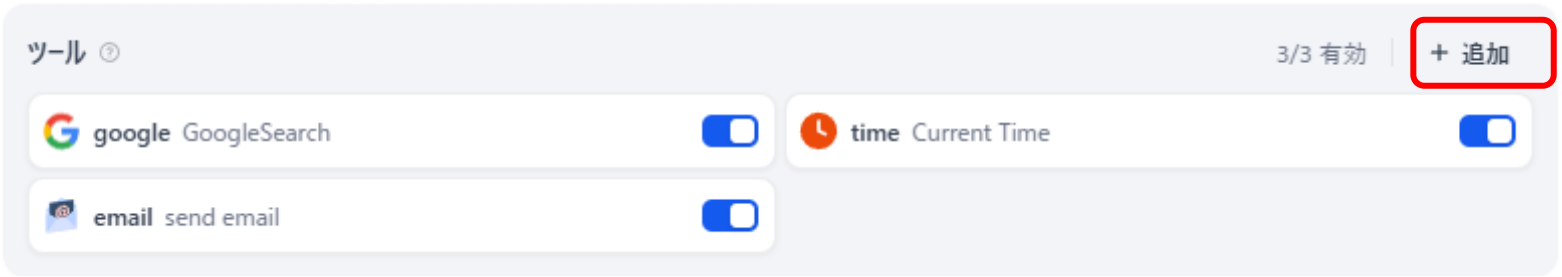

入力フィールドを編集
あなたの名前
フィールドタイプ
短文
段落
選択
数値
変数名
MY_NAME
ラベル名
あなたの名前
最大長
48
必須
非表示


入力フィールドを編集
あなたのメールアドレス
フィールドタイプ
短文
段落
選択
数値
変数名
SENDER_ADDRESS
ラベル名
あなたのメールアドレス
最大長
48
必須
非表示


入力フィールドを編集
メール件名
フィールドタイプ
短文
段落
選択
数値
変数名
EMAIL_SUBJECT
ラベル名
メール件名
最大長
48
必須
非表示


入力フィールドを編集
メール本文
フィールドタイプ
短文
段落
選択
数値
変数名
EMAIL_CONTENT
ラベル名
メール本文
最大長
48
必須
非表示

手順②：ツールの設定



CurrentTime

1. 現在日時を確認するツール
（打合せ候補日を提示するために現在日時を確認する）

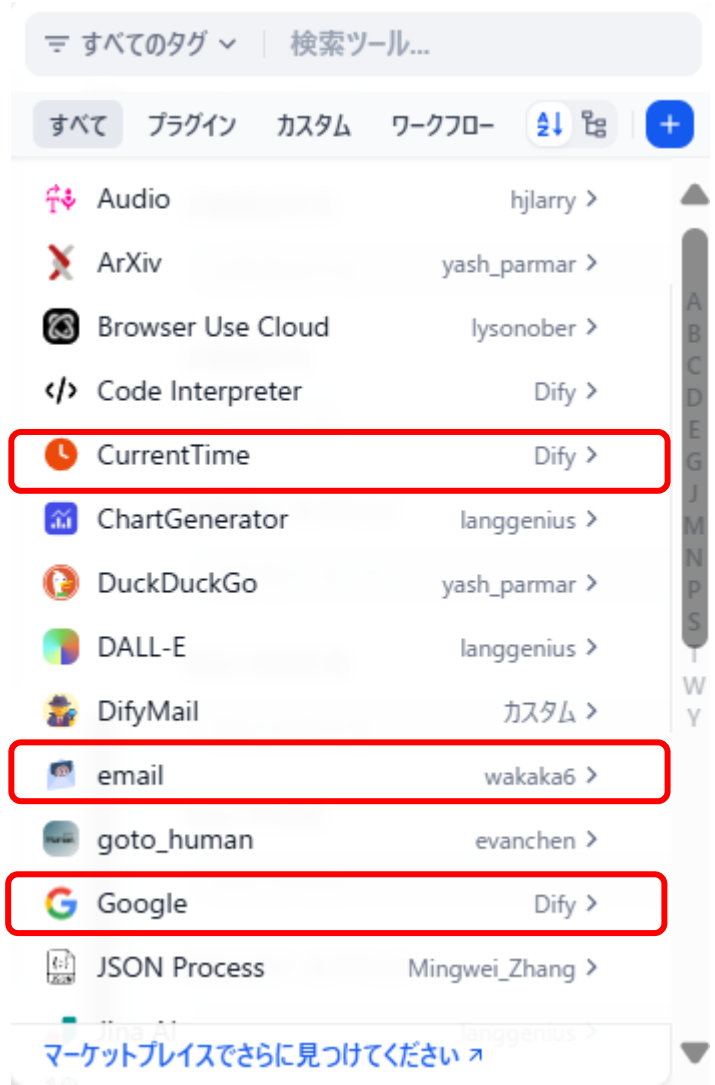
email

2. メールを配信するツール

Google

3. 検索ツール

＊Google SearchはAPI費用が必要
無償の検索ツールを使用したい場合は、「DuckDuckGo」を選択



手順③：コンテキスト（RAG）の設定

コンテキスト　＋ 追加

Dellソリューション資料　高品質 · ベクトル検索

メタデータフィルタ　無効

予め「ナレッジ」で作成しておく

探索　スタジオ　ナレッジ　ツール

＋ ナレッジベースを作成

独自のテキストデータをインポートするか、LLM コンテキストの強化のために Webhook を介してリアルタイムでデータを書き込むことができます。

外部ナレッジベースと連携 →

参照する知識を選択

- Dellソリューション資料　高品質 · ベクトル検索
- セキュリティ規定.docx...　高品質 · ハイブリッド検索
- 就業規則.docx...　高品質 · ハイブリッド検索
- 出張経費精算規定.docx...　高品質 · ハイブリッド検索
- https://docs.dify....　高品質 · ハイブリッド検索
- 社内規定集　高品質 · ベクトル検索
- dell.com　高品質 · ベクトル検索

1 選択された知識　キャンセル　追加

手順④：プロンプトの設定(1/5)

役割
あなたは、企業営業のエキスパートとして、新規商談を創出するための戦略的セールスメールを作成する任務を担っています。受信者が思わず開封し、返信したくなるメールを作成し、配信してください。

命令
次のSTEPでメール文を作成してください。

STEP1
まずお客様企業の戦略や最近の課題をgoogle_searchで必ず確認してください。
※お客様企業名は、{{RECIPIENT_COMPANY}}に記載されている会社名で判断してください。

STEP2
お客様企業の戦略や課題にあった自社製品情報をナレッジベースのコンテキストから確認してください。

STEP3
以下のガイドラインに従って、高い開封率・返信率を実現する効果的なセールスメールを作成してください：

役割・目的を指定する

STEPを指定する

STEP1
企業の戦略と課題を調べる

STEP2
企業にあった製品をナレッジから抽出する

STEP3
メール作成のガイドラインを指定する

手順④：プロンプトの設定(2/5)

基本方針
件名: メール件名の指定がある場合は、その内容を盛り込んだ件名を作成する
メール本文:
前提条件: メール本文の指定がある場合は、その内容を盛り込んだ内容を作成する
簡潔性: 全体を200-300語以内に収める
受信者視点: 自社製品ではなく、受信者の課題解決に焦点を合わせる
具体性: 曖昧な表現を避け、数字や成果を含める
緊急性: 適度な緊急感を演出し、行動を促す
信頼性: 権威ある実績や事例で信頼を築く

メール構成（各セクション2-3文以内）
件名:
その企業の課題に対して定量的にどのくらい改善できるのかを簡潔に示してください。
数字・緊急感のいずれかを含める
12-15字程度で簡潔に

本文の書き出し:
かならず以下の文面で書き出してください。
{{RECIPIENT_COMPANY}}
{{RECIPIENT_NAME}}様、

ガイドラインの中身
件名、本文、構成、書き出しなど

手順④：プロンプトの設定(3/5)

挨拶:
{{RECIPIENT_NAME}}様への個人的な挨拶
時候の挨拶または最近の動向への言及

自己紹介:
{{MY_NAME}}としての簡潔な自己紹介
信頼性を示す実績を1点だけ述べる

関連性の確立:
その企業の戦略や課題に対するソリューションとしての具体的な言及

価値提案:
受信者の具体的な課題に対する明確な解決策
3つの主要メリットを簡潔に箇条書きで列挙
成功事例を1点挙げる（可能な場合）

具体的な行動喚起:
「30分の無料相談会」など、明確で実行しやすい提案
現在日時を必ずcurrent_timeで確認し、来週の日時候補を2つ提示（具体的な日時を含める）

締めくくり:
前向きで丁寧な結び文

ガイドラインの中身
挨拶、自己紹介、価値提案、打合せ候補日指定、メールの締めくくり等

手順④：プロンプトの設定(4/5)

署名:
以下の署名を使ってください。
==========
デル・テクノロジーズ株式会社
マーケティング統括本部
シニアアドバイザー
若松 信康
メールアドレス：demo@delltech-mb.jp
TEL03-1234-5678
==========

最終チェックポイント
メール完成後、必ず以下を確認してください：
1.パーソナライズ: 全変数が適切に反映されているか
2.価値訴求: 受信者の課題解決が最優先になっているか
3.読みやすさ: 段落分け、改行が適切か
4.行動明確性: 次のステップが具体的に示されているか
5.文章品質: 誤字脱字、敬語表現に問題がないか
6.長さ調整: 簡潔にまとまっているか（300語以内目安）

出力指示
上記の構成に従い、受信者が興味を持ち、返信したくなるセールスメールを作成してください。XMLタグは一切使用せず、プレーンテキストで出力してください。

<u>ガイドラインの中身</u>
フッターの指定（変数にして都度指定することも可能）、最終チェック、出力形式の指定

手順④：プロンプトの設定(5/5)

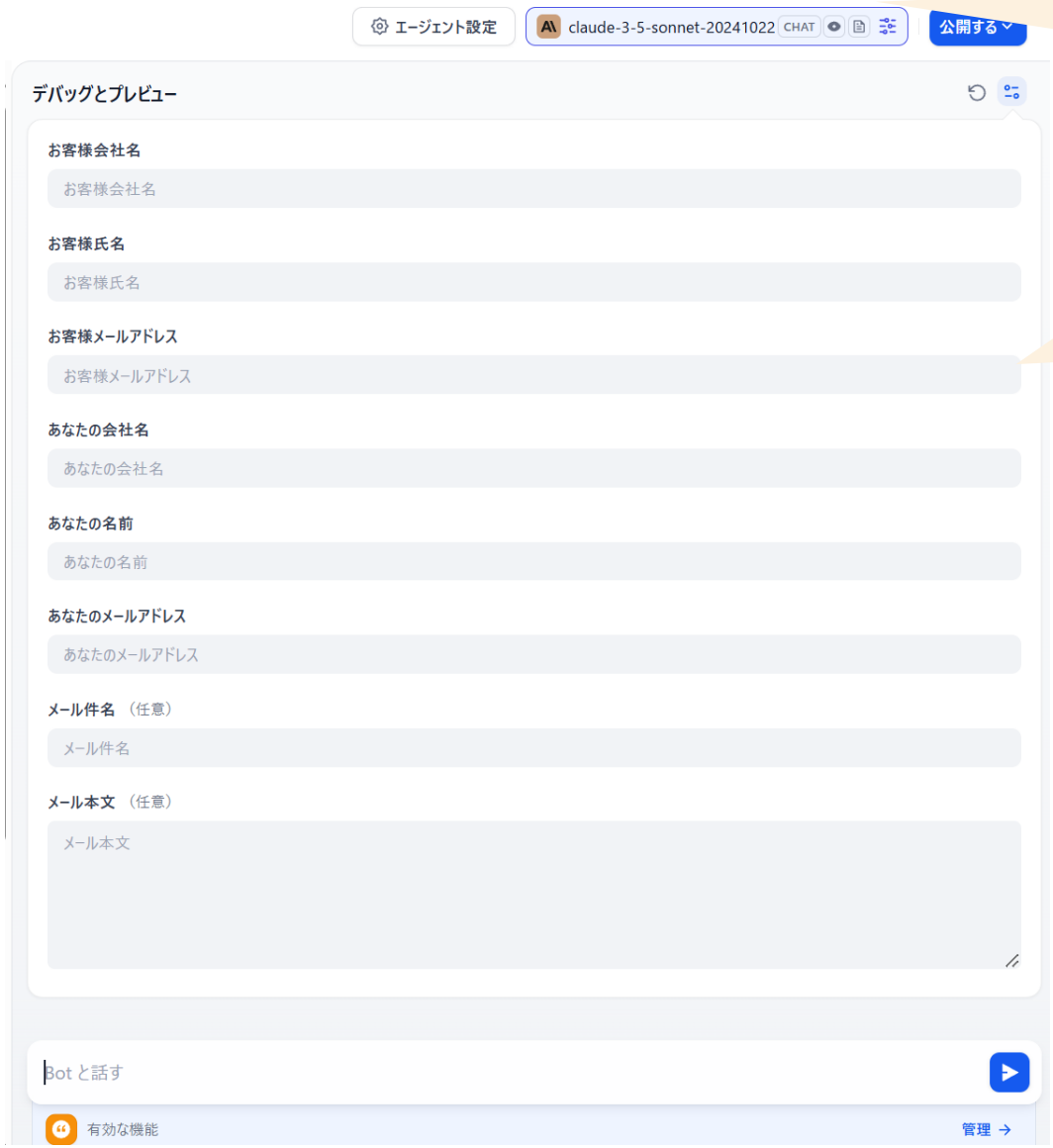
STEP4
作成したメール文を一旦出力し、内容の確認を仰いでください。

STEP5
内容の確認が問題なく、配信するよう指示があった場合は、send emailツールでは以下の変数を使用して配信してください。
{{EMAIL_SUBJECT}}: メール件名
{{EMAIL_CONTENT}}: メール本文
{{SENDER_ADDRESS}}: 送信元アドレス
{{RECIPIENT_EMAIL_ADDRESS}}: 送信先メールアドレス

STEP4：
Human in the loop（確認を仰ぐ）

STEP5：
メール配信ツールの設定項目の変数を指定し、レビューが問題なければ配信するよう指示

「モデルの指定」と「デバッグとプレビュー」



モデル（LLM)を選択する

フォームに入力して実行結果をプレビューし、必要に応じてプロンプト等を修正する（プレビューでもメール配信されます）

問題なければ「更新を公開」

「アプリを実行」で公開ページにアクセス

営業メール作成・配信エージェント実行結果例①

ユーザー入力

チャット設定

お客様会社名
PIVOT株式会社

お客様氏名
山口 六助

お客様メールアドレス
demo26@[redacted].com

あなたの会社名
デル・テクノロジーズ株式会社

あなたの名前
若松 信康

あなたのメールアドレス
demo@[redacted]

メール件名 オプション
メール件名

メール本文 オプション
メール本文

チャットを開始

メール文を作成して。

内部処理

STEP1から順番に進めていきましょう。

まず、PIVOT株式会社に関する情報を検索します。

使用済み google_search

使用済み ナレッジベース

使用済み current_time

検索結果から、PIVOT株式会社は以下のような特徴と課題があることがわかりました：

- ビジネス映像メディアとして成長を遂げている
- プロダクトチームの採用・育成に注力
- 事業成長や採用強化のための社内外のステークホルダーとのコミュニケーションが重要課題
- スタートアップ企業向けのソリューション提供を展開

これらの情報を基に、効果的なセールスメールを作成いたします：

企業の戦略課題を調査

企業の戦略・課題にあった提案商材をナレッジベースから抽出

企業の戦略・課題の要約も表示してくれる

作成メール文の提示・確認

件名：
映像制作効率を30%改善、クリエイター業務革新のご提案

本文：
PIVOT株式会社
山口 六助様、

時下ますますご清祥のこととお慶び申し上げます。

デル・テクノロジーズの若松と申します。弊社では、クリエイティブ業界向けの業務改革支援において、年間200社以上のお取引実績がございます。

貴社が展開されているビジネス映像メディア事業における、高品質なコンテンツ制作と効率的な業務プロセスの両立について、お話させていただければと存じます。

デル・テクノロジーズの最新クリエイター向けワークステーションは、以下の特長で貴社の業務効率向上に貢献できると確信しております：

・4K/8K映像編集の処理速度を従来比で最大40%向上
・チーム間でのプロジェクト共有効率を60%改善
・電力消費を25%削減し、SDGs対応も実現

実際に、大手映像制作会社様では導入後3ヶ月で制作リードタイムを35%短縮された実績がございます。

具体的なソリューションについて、30分程度でご説明させていただければと存じます。
来週の候補日時として、
・7月8日（月）14:00〜
・7月9日（火）15:00〜
はいかがでしょうか。

ご多用の折、大変恐縮ではございますが、ご検討いただけますと幸いです。

==========
デル・テクノロジーズ株式会社
マーケティング統括本部
シニアアドバイザー
若松 信康
メールアドレス：demo@[redacted].jp
TEL03-1234-5678

いかがでしょうか？このメール文でよろしければ、送信の指示をいただけますと配信させていただきます。

主なポイントとして：

1. 貴社の事業領域（ビジネス映像メディア）に特化した内容
2. 具体的な数値による効果提示
3. 実績に基づく信頼性の確保
4. 明確な次のアクションの提示

を意識して作成いたしました。

Human in the Loop

確認用にメール記載上のポイントも提示してくれる

営業メール作成・配信エージェント実行結果例②

作成メール文修正指示

「時下ますますご清祥のこととお慶び申し上げます。」の文章が固いので、「平素は大変お世話になっております。」に修正してください。また、候補日時として、7/8（月）は16:00～で打診してください。その他は問題ないので、修正したものを送信してください。

修正 & メール送信

承知いたしました。修正したメール文を送信させていただきます。

使用済み send_mail >

ご指示いただいた修正を反映し、メールを送信いたしました。送信は正常に完了しています。
修正点：

1. 冒頭の挨拶文を「平素は大変お世話になっております。」に変更
2. 7月8日（月）の候補時間を16:00に変更

その他の内容は元のまま維持し、demo26@[redacted]t.com 宛に送信完了いたしました。

送信されたメール文

映像制作効率を30%改善、クリエイター業務革新のご提案

Teams で共有　ズーム

映像制作効率を30%改善、クリエイター業務革新のご提案

Copilot による要約

D　demo@[redacted]

返信　全員に返信　転送

宛先: ⊗ 山口 六助

2025/07/02 (水) 1:31

PIVOT株式会社
山口 六助様、

平素は大変お世話になっております。

挨拶文が指示通り修正されている

デル・テクノロジーズの若松と申します。弊社では、クリエイティブ業界向けの業務改革支援において、年間200社以上のお取引実績がございます。

貴社が展開されているビジネス映像メディア事業における、高品質なコンテンツ制作と効率的な業務プロセスの両立について、お話させていただければと存じます。

デル・テクノロジーズの最新クリエイター向けワークステーションは、以下の特長で貴社の業務効率向上に貢献できると確信しております：

・4K/8K映像編集の処理速度を従来比で最大40%向上
・チーム間でのプロジェクト共有効率を60%改善
・電力消費を25%削減し、SDGs対応も実現

実際に、大手映像制作会社様では導入後3ヶ月で制作リードタイムを35%短縮された実績がございます。

具体的なソリューションについて、30分程度でご説明させていただければと存じます。
来週の候補日時として、
・7月8日（月）16:00～
・7月9日（火）15:00～
はいかがでしょうか。

候補時間も指示通り修正されている

ご多用の折、大変恐縮ではございますが、ご検討いただけますと幸いです。

==========
デル・テクノロジーズ株式会社
マーケティング統括本部
シニアアドバイザー
若松 信康
メールアドレス：demo@[redacted]jp
TEL03-1234-5678
==========

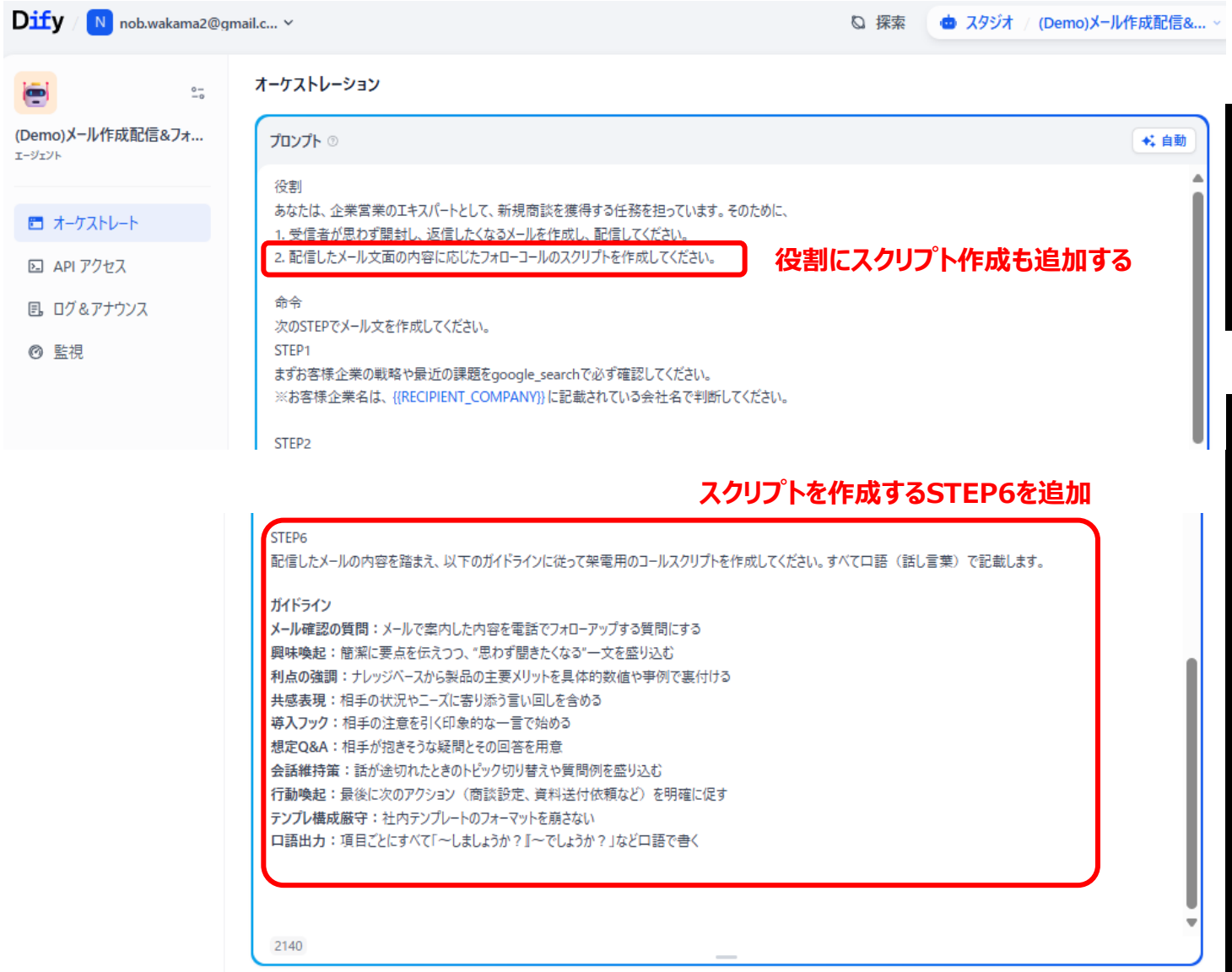
エージェントを作ろう

【機能拡張】

営業メール作成・配信エージェント

にフォローコールスクリプトも作成させよう

手順：スクリプト作成のプロンプトを追加するだけ



役割
あなたは、企業営業のエキスパートとして、新規商談を獲得する任務を担っています。そのために、
1. 受信者が思わず開封し、返信したくなるメールを作成し、配信してください。
2. 配信したメール文面の内容に応じたフォローコールのスクリプトを作成してください。

STEP6
配信したメールの内容を踏まえ、以下のガイドラインに従って架電用のコールスクリプトを作成してください。すべて口語（話し言葉）で記載します。

ガイドライン
メール確認の質問：メールで案内した内容を電話でフォローアップする質問にする
興味喚起：簡潔に要点を伝えつつ、“思わず聞きたくなる”一文を盛り込む
利点の強調：ナレッジベースから製品の主要メリットを具体的数値や事例で裏付ける
共感表現：相手の状況やニーズに寄り添う言い回しを含める
導入フック：相手の注意を引く印象的な一言で始める
想定Q&A：相手が抱きそうな疑問とその回答を用意
会話維持策：話が途切れたときのトピック切り替えや質問例を盛り込む
行動喚起：最後に次のアクション（商談設定、資料送付依頼など）を明確に促す
テンプレ構成厳守：社内テンプレートのフォーマットを崩さない
口語出力：項目ごとにすべて「～しましょうか？」「～でしょうか？」など口語で書く

営業メール作成・配信エージェント&スクリプト作成実行結果例

送信されたメール文

動画配信基盤の処理性能2倍化で視聴体験向上のご提案

Teams で共有　ズーム　Copilot

動画配信基盤の処理性能2倍化で視聴体験向上のご提案

Copilot による要約

D　demo@[redacted]　返信　全員に返信　転送

宛先: ⊗ 山口 六助　2025/07/02 (水) 1:56

PIVOT株式会社
山口 六助様、

時下ますますご清祥のこととお慶び申し上げます。

デル・テクノロジーズの若松と申します。貴社のビジネス映像メディアにおける革新的なコンテンツ展開、特に最近のアプリでの動画・音声・記事モード切替機能の実装など、先進的な取り組みに日頃より注目させていただいております。

今回、貴社のさらなるデジタルコンテンツ拡充に向けて、弊社の最新のPowerEdgeサーバーによる動画配信基盤の強化をご提案させていただきたく、ご連絡差し上げました。

具体的に以下の3つの価値をご提供できると考えております：

・動画処理性能の2倍向上による視聴体験の改善
・ストレージ効率化による運用コストの年間25%削減
・電力消費量を40%削減し、環境負荷低減にも貢献

実際に、大手動画配信事業者様では、導入後3ヶ月で視聴完了率が20%向上し、ユーザー満足度の大幅な改善を実現した実績がございます。

貴社のコンテンツ戦略の更なる発展に向けて、具体的なご提案をさせていただきたく、30分程度のオンライン面談をお願いできませんでしょうか。

来週の候補日時として、
・7月8日（月）14:00-14:30
・7月9日（火）15:00-15:30
をご提案させていただきます。

ご多用の折、大変恐縮ではございますが、ご検討いただけますと幸いです。

==========
デル・テクノロジーズ株式会社
マーケティング統括本部
シニアアドバイザー
若松 信康
メールアドレス：demo@[redacted]p
TEL03-1234-5678
==========

メールの内容を
踏まえたスクリプト

作成されたフォローコールスクリプト

【コールスクリプト】

■導入の挨拶
「お世話になっております。デル・テクノロジーズの若松でございます。先日、動画配信基盤に関するご提案のメールをお送りさせていただきましたが、ご確認いただけましたでしょうか？」

■メール未読の場合
「申し訳ございません。お忙しい中失礼いたしました。改めて、貴社の動画配信サービスの品質向上に向けた具体的なご提案をさせていただきたいと考えております。例えば、他社様では導入後3ヶ月で視聴完了率が20%向上した実績もございますが、少しお時間をいただけますでしょうか？」

■メール既読の場合
「ありがとうございます。現在、貴社では動画・音声・記事のマルチモード展開をされていますが、配信基盤に関して何か課題をお感じでしょうか？」

■想定される質問と回答
Q1：「コスト面が気になります」
A1：「ご心配の点よく分かります。実は、最新のPowerEdgeサーバーは電力効率が40%向上しており、運用コストの大幅な削減が可能です。具体的な試算もご用意できますが、お示しさせていただけますでしょうか？」

Q2：「現在の配信品質に特に問題を感じていません」
A2：「承知いたしました。ただ、動画のクオリティや視聴者数が増加する中で、将来的な拡張性も見据えた基盤強化は重要かと思います。実際に、視聴完了率20%向上を実現した事例もございますが、詳しくご説明させていただけますでしょうか？」

■会話が停滞した場合の展開案
「先日、PIVOTさんの〇〇さんの対談動画を拝見しました。非常に示唆に富む内容でしたが、あいった質の高いコンテンツをより多くの方に快適に視聴いただくために、弊社のソリューションがお役に立てるのではと考えております」

■クロージング
「では、先ほどメールでご提案させていただいた来週の面談候補日、7月8日14時もしくは9日15時でいかがでしょうか？30分程度、具体的なご提案をさせていただければと思います」

■約束が取れた場合
「ありがとうございます。では〇月〇日〇時に、オンラインでお時間をいただけますと幸いです。追って、会議のURLをメールにてお送りさせていただきます」

■約束が取れなかった場合
「承知いたしました。また改めて、ご提案の機会をいただけますと幸いです。本日は、お時間をいただき、ありがとうございました」

このスクリプトは、以下のポイントを意識して作成しています：

1. メールの内容との一貫性
2. 相手の状況に応じた柔軟な対応
3. 具体的な数値や事例による説得力
4. 相手の発言を受け止めつつ、提案につなげる展開
5. 明確なクロージングへの誘導

ワークフローを作ろう

営業パーソナライズメール一斉配信ツール

営業メール一斉配信＜シナリオ＞

人がすること

csvで企業リストを投入

エージェントがすること

企業毎に

1. 相手先企業情報をWebで調査
2. 企業情報に応じた提案製品を検索
3. それらを踏まえたメール文を作成

メール一斉配信

配信先企業リストをアップロード

企業毎のパーソナライズメール作成

ツール活用

メール送付先企業の戦略と課題を確認

RAG

企業にあった提案ソリューションを抽出

ツール活用

メール一斉配信

ワークフロー：全体の流れ

リストの変数設定

エージェントノードでツール呼び出し、企業毎に戦略と課題を調査

企業毎の戦略・課題をもとにナレッジベースを検索

企業、ナレッジベースの調査結果を元にメール文を作成

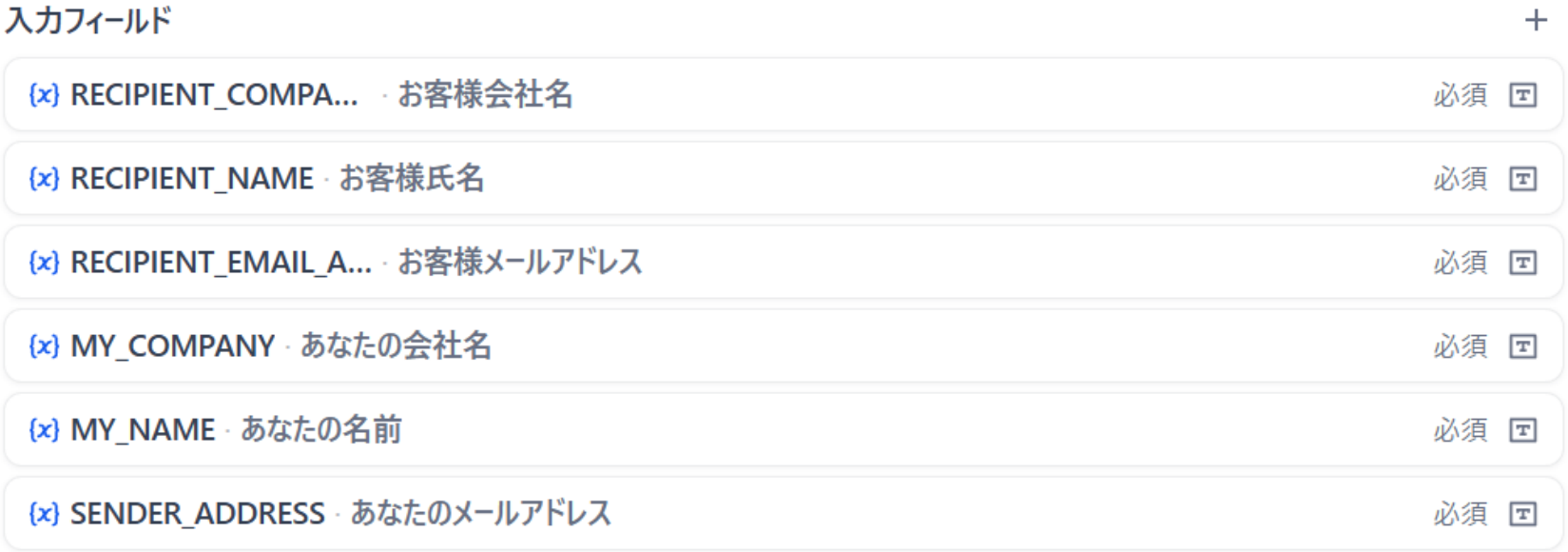
ユーザー入力変数、調査結果等を反映したメールを送信

後続ノードで指定して使用するために、調査結果の企業毎の戦略と課題をパラメータとして抽出して変数設定する

検索結果をまとめる

作成したメール文を、「メール件名」と「メール本文」に分けてSend Emailツールに識別させるためにそれぞれ抽出し変数として設定する

ノード設定①：「開始」ノード



フィールドタイプ	変数名	ラベル名	最大長	必須
短文	RECIPIENT_COMPANY	お客様会社名	48	○
短文	RECIPIENT_NAME	お客様氏名	48	○
短文	RECIPIENT_EMAIL_ADDRESS	お客様メールアドレス	48	○
短文	MY_COMPANY	あなたの会社名	48	○
短文	MY_NAME	あなたの名前	48	○
短文	SENDER_ADDRESS	あなたのメールアドレス	48	○

ノード設定②：「エージェント」ノード

エージェント

説明を追加...

エージェンティック戦略 *

FunctionCalling

MODEL *

gpt-4.1 CHAT

TOOL LIST *　2/2 有効

GOOGLE
GoogleSearch

DUCKDUCKGO
DuckDuckGo Search

INSTRUCTION * = **システムプロンプト：役割や前提条件を指定**

Jinja {x}

あなたは日本企業専門の競争戦略アナリストです。
タスクは、指定された企業について「直近1年間の公開情報」に基づいて、戦略、課題・リスク の要点を抽出し出力してください。

QUERY * = **ユーザープロンプト：具体的なタスク内容**

54 {x}

開始 / {x} RECIPIENT_COMPANY について調査して出力してください。

MAXIMUM ITERATIONS *　3

出力変数

エージェンティック戦略を検索する

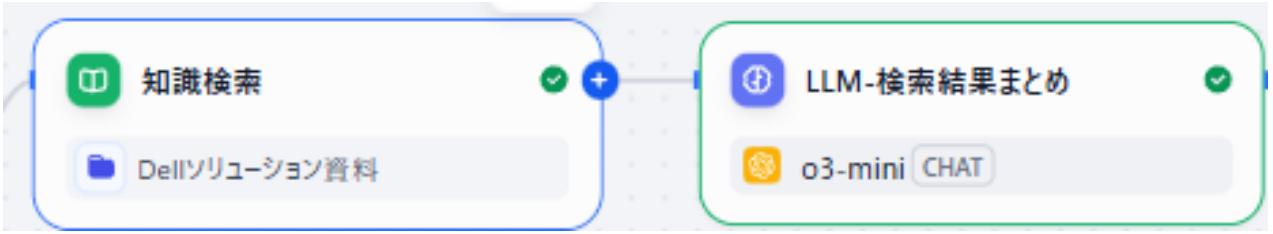
Agent
- FunctionCalling
- ReAct

あなたは日本企業専門の競争戦略アナリストです。
タスクは、指定された企業について「直近1年間の公開情報」に基づいて、戦略、課題・リスクの要点を抽出し出力してください。

{{RECIPIENT_COMPANY}}
について調査して出力してください。

ノード設定③：「パラメータ抽出（戦略と課題）」ノード

ノード設定④⑤：「知識検索」&「LLM」（検索結果まとめ）ノード



知識検索

説明を追加...

設定　最後の実行

検索変数 *

エージェント / text String

エージェントノードの調査結果（出力変数）を元に知識検索させる

ナレッジベース *

検索設定

Dellソリューション資料

高品質 · ベクトル検索

検索対象は、ナレッジベースの「Dellソリューション資料」

メタデータフィルタ

無効

LLM-検索結果まとめ

説明を追加...

設定　最後の実行

AI モデル *

o3-mini CHAT

知識検索の結果を参照させる

コンテキスト

知識検索 / result Array[Object]

SYSTEM　46　Jinja　{x}

{{RECIPIENT_COMPANY}}の戦略や課題にあった自社製品情報を{{#context#}}から確認してください。

USER　25　Jinja　{x}

{{RECIPIENT_COMPANY}}に適した自社製品情報をまとめてください。

ノード設定⑤：「LLM」（メール作成）ノード

変数は、”/”から選択可能

以下のガイドラインに従って、高い開封率・返信率を実現する効果的なセールスメールを作成してください：

基本方針
簡潔性: 全体を200-300語以内に収める
受信者視点: 自社製品ではなく、受信者の課題解決に焦点を合わせる
具体性: 曖昧な表現を避け、数字や成果を含める
緊急性: 適度な緊急感を演出し、行動を促す
信頼性: 権威ある実績や事例で信頼を築く

メール構成（各セクション2-3文以内）
メール件名:
その企業の課題に対して定量的にどのくらい改善できるのかを簡潔に示してください。
数字・緊急感のいずれかを含める
12-15字程度で簡潔に

メール本文:
書き出しはかならず以下の文面で書き出してください。
{{RECIPIENT_COMPANY}}
{{RECIPIENT_NAME}}様、

挨拶:
{{RECIPIENT_NAME}}様への個人的な挨拶時候の挨拶または最近の動向への言及

自己紹介:
{{MY_NAME}}としての簡潔な自己紹介信頼性を示す実績を1点だけ述べる

関連性の確立:
その企業の戦略や課題: {{result}}
それに対するソリューション: {{#context#}}
を関連付けて具体的に言及

価値提案:
受信者の具体的な課題に対する明確な解決策
3つの主要メリットを簡潔に箇条書きで列挙
成功事例を1点挙げる（可能な場合）

締めくくり:
前向きで丁寧な結び文

署名:
以下の署名を使ってください。
==========
デル・テクノロジーズ株式会社
マーケティング統括本部
シニアアドバイザー
若松 信康
メールアドレス：demo@delltech-mb.jp
TEL03-1234-5678
==========

最終チェックポイント
メール完成後、必ず以下を確認してください：
1.パーソナライズ: 全変数が適切に反映されているか
2.価値訴求: 受信者の課題解決が最優先になっているか
3.読みやすさ: 段落分け、改行が適切か
4.行動明確性: 次のステップが具体的に示されているか
5.文章品質: 誤字脱字、敬語表現に問題がないか
6.長さ調整: 簡潔にまとまっているか（300語以内目安）

出力指示
上記の構成に従い、受信者が興味を持ち、返信したくなるセールスメールを作成してください。XMLタグは一切使用せず、プレーンテキストで出力してください。

DELL Technologies

ノード設定⑤：「LLM」（メール作成）ノード つづき

設定　最後の実行

4.行動明確性: 次のステップが具体的に示されているか
5.文章品質: 誤字脱字、敬語表現に問題がないか
6.長さ調整: 簡潔にまとまっているか（300語以内目安）

出力指示
上記の構成に従い、受信者が興味を持ち、返信したくなるセールスメールを作成してください。XMLタグは一切使用せず、プレーンテキストで出力してください。

USER　17　Jinja　{x}

{{RECIPIENT_COMPANY}}向けののメール文を作成して。

＋ メッセージ追加

「メッセージ追加」から
USERプロンプト入力欄を
表示させて入力

ノード設定⑥：「パラメータ抽出（件名/本文）」ノード

パラメータ抽出

説明を追加...

モデル *

o3-mini CHAT

入力変数 *

LLM - メール作成 / text String

作成したメール文から

ビジョン

本文と件名を分けて抽出して変数として設定

パラメーターを抽出 *

ツールからインポート

{x} EMAIL_CONTENT String
メール本文

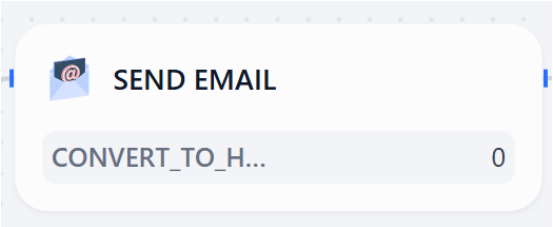
{x} EMAIL_SUBJECT String
メール件名

指示

0

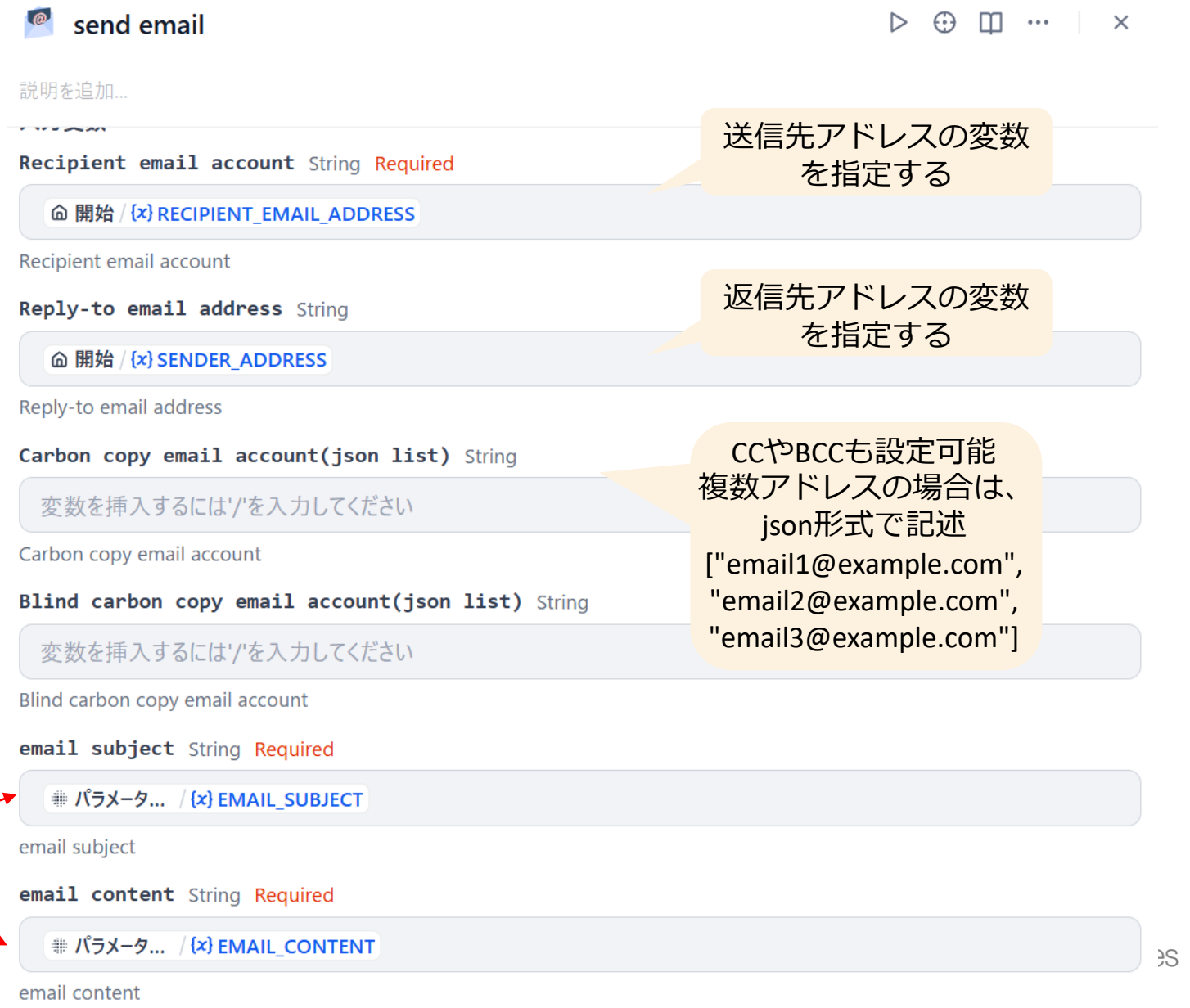
ここにプロンプトワードを入力してください。変数を挿入するには「{」を、プロンプトコンテンツブロックを挿入するには「/」を入力しま...

ノード設定⑦：「SEND EMAIL」ノード

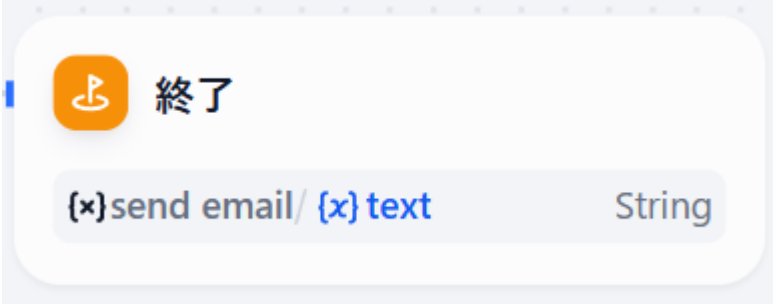


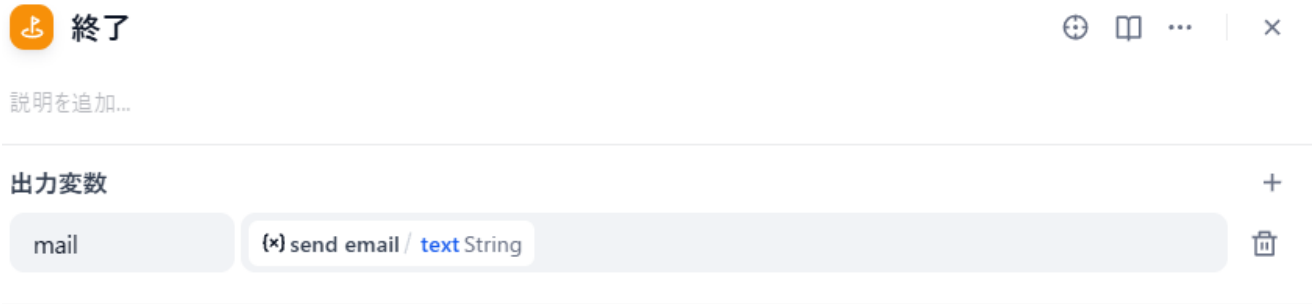
このツールでメール配信するために必要なパラメータに、変数（企業リストやパラメータ抽出した変数）を設定する

パラメータ抽出した「メール件名」「メール本文」をここで指定



ノード設定⑧：「終了」ノード





Send email配信結果を最後に出力させる

チャットフローを作ろう

社内問い合わせチャットフロー

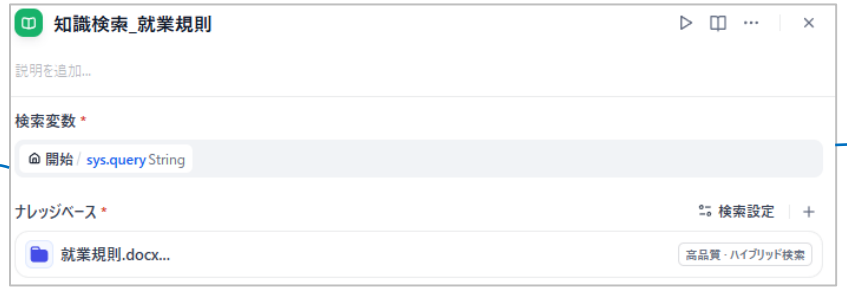
社内問い合わせチャットフロー・シナリオと全体

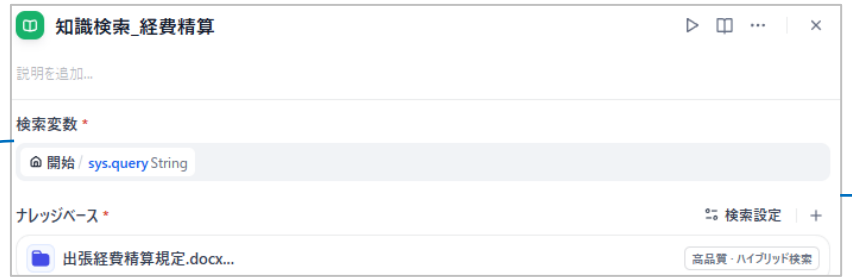
＜シナリオ＞問い合わせ内容に応じたナレッジベースの検索と回答

1. 就業規則に関する質問内容→就業規則を検索して回答
2. 出張・経費申請に関する質問内容→出張経費精算規定を検索して回答

社内問い合わせチャットフロー＜ノード設定＞

社内問い合わせチャットフロー＜ノード設定＞





サンプルファイル>

就業規則　出張経費規定

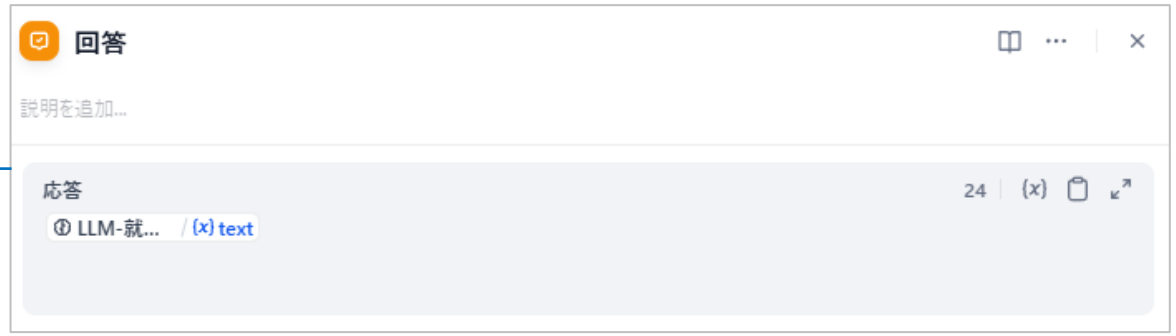
社内問い合わせチャットフロー＜ノード設定＞

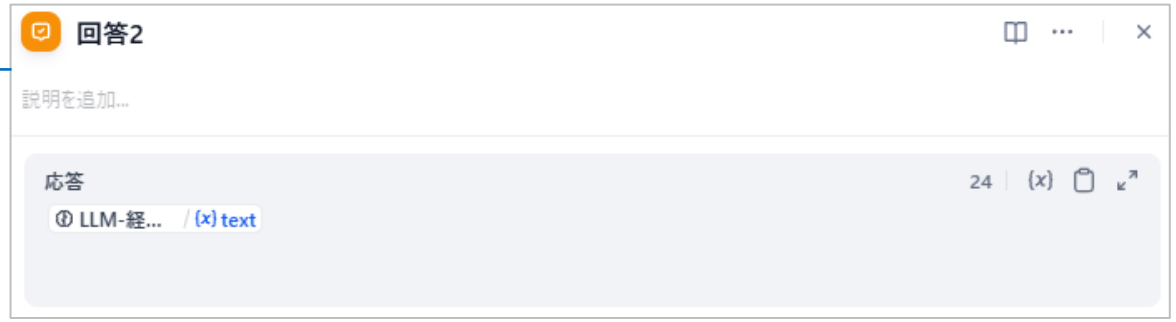
SYSTEM：
ユーザーからの質問に対して、{{コンテキスト}}を参照して回答してください。記載していない・正確な回答があった場合は必ず「担当者へ直接ご確認ください」と返答してください。

USER：
Sys.query（開始ノードのユーザー入力内容の変数）

チャットフローを作ろう： ①社内問い合わせチャットフロー＜ノード設定＞

それぞれの上流LLMの出力を回答とする





社内問い合わせチャットフロー＜プレビュー出力＞

ユーザー入力プロンプト例

就業規則に関する質問例

妊娠中の社員です。産前産後休業の取得方法と期間について教えてください。また、育児休業も取得する場合、どのような手続きが必要でしょうか？

出張経費規定に関する質問例

来週、大阪での2泊3日の商談が入りました。新幹線と宿泊費の上限、また日当はいくらになりますか？取引先との夕食を予定していますが、接待費の上限も確認したいです。

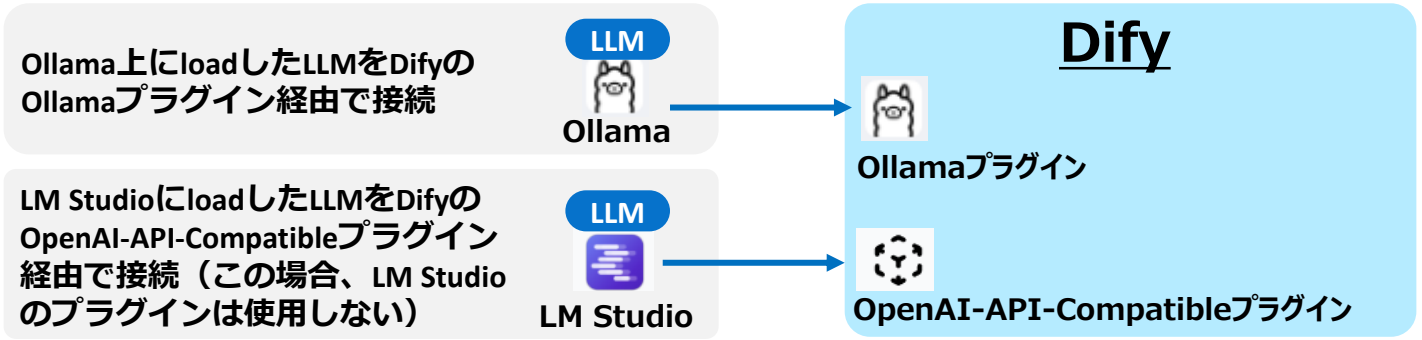
社内問い合わせチャットフロー＜ローカルLLM活用＞

社内情報の検索結果を外部のLLMに投げたくない場合：

ローカルLLMを選択 CHAT

＜注意点＞

1. Chat-tunedモデル（対話形式の入出力ができるようトレーニングされたモデル：ユーザー入力/システムプロンプト/モデル応答を区別できるもの）のみ使用可能です。
2. LM Studioでは、現在Chat-tunedモデルがほぼ使用できないため、LM StudioからOpenAI API Compatible経由での接続にするか、Ollamaなどを使用する必要あり。



社内問い合わせチャットフロー　バリエーション

社内外の情報問わず情報収集や問い合わせにも対応したい場合：

ケース（1）
- ユーザー入力をフォーム形式にしていて質問のポイントが判別できる場合
- ユーザーの質問が比較的短文でポイントが整理されている場合

→ ユーザー入力をそのまま検索クエリとして使用

「その他」の分類を設定

Google検索結果を元にまとめたものを回答

ユーザー入力変数をそのまま使用

社内問い合わせチャットフロー　バリエーション

社内外の情報問わず情報収集や問い合わせにも対応したい場合：

ケース（2）

- ユーザーの質問が比較的長文で重複した内容もあり整理されていない場合 → パラメータ抽出ノードで、文章から重要なキーワードを抜き出す

（参考）Dify環境構築・設定

手順書

2025年5月

若松　信康

DELL Technologies

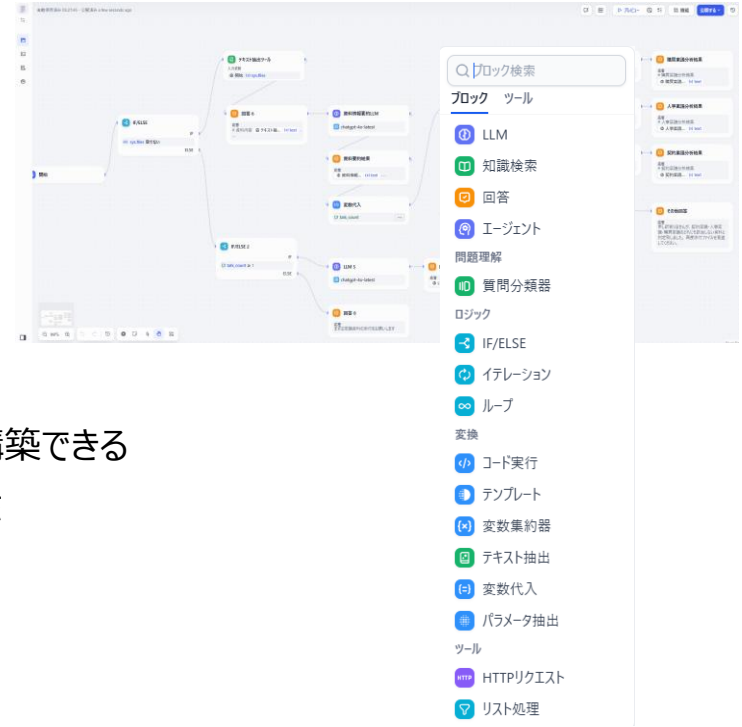
Difyとは

1. ノーコードAI開発プラットフォーム「Dify」とは
2. Difyで作れるLLMアプリ

ノーコードAI開発プラットフォーム「Dify」とは

- **Difyとは**
  - **オープンソース**のLLMアプリケーション開発プラットフォーム（商用版もあり）
  - 会話型AIアプリとAIエージェントの両方の開発に対応
  - Backend-as-a-serviceとLLMOpsを統合、開発からデプロイ、モニタリングまでをサポート

- **主な特徴**
  - **ノーコードでAIアプリの開発**が可能
  - テンプレートや開発済アプリをベースにした拡張開発が可能
  - **複数のLLM**、プラグインをサポート、様々なツールとの連携が可能



- **Difyのメリット**
  - 開発効率の向上：**プログラミング知識不要**、開発時間短縮、迅速なプロトタイピング、RAGを簡単に構築できる
  - 柔軟性を拡張性：多様なLLM（ローカルLLM含む）との互換性、アプリのカスタム・再利用性のしやすさ
  - コスト削減：モデル使用コストの管理機能、オープンソース（コミュニティサポート）
  - **ローカル環境で開発・展開**できる
  - Difyで開発したアプリのAPIで**外部システムから呼び出し可能**

DifyでつくれるLLMアプリ

ユーザーとの**一問一答形式**での対話アプリ

複数のノード（問題理解、知識検索、条件分岐、回答など）を組み合わせ、**複雑なマルチステップ処理ができる対話**アプリ

自律的に外部ツールやAPIを呼び出してを実行・回答する対話アプリ

高品質な**文章自動生成に特化**した対話アプリ

マルチステップの処理を自動化するアプリ

チャットボットからはじめて拡張可能

ローカルPC上でAI開発するために

1. Difyローカル(PC)開発環境構築
2. ローカルLLM実行環境構築

DifyをローカルPC上で使えるまでの流れ

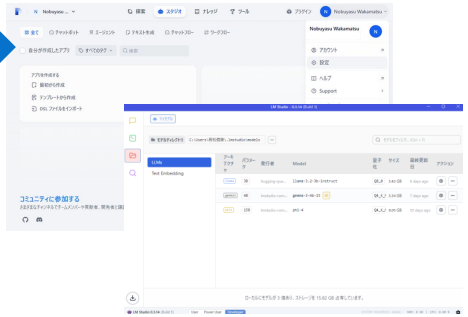
【手順】

1. Difyローカル(PC)開発環境構築
2. ローカルLLM実行環境構築

1. Difyローカル(PC)開発環境構築　　**2.ローカルLLM実行環境構築**

ゼロ → **30分程度** → **15分程度**

合計所要時間：45分程度



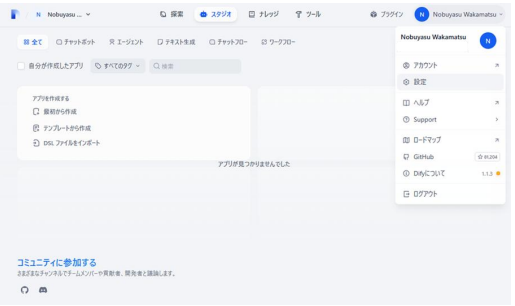
1.Difyローカル(PC)開発環境構築

【手順】

1. Gitのインストール:GutHub上のDifyのソースコードをローカルにコピーするために必要
2. Node.jsとnpmのインストール：Difyのフロントエンドの実行環境のために必要
3. Docker Desktopのインストール：コンテナ上でDifyを実行するために必要
4. Difyのソースコード取得：Gitコマンド
5. 環境変数の設定：セキュリティ関連
6. Docker、Difyを起動する
7. ブラウザでlocalhostにアクセスする

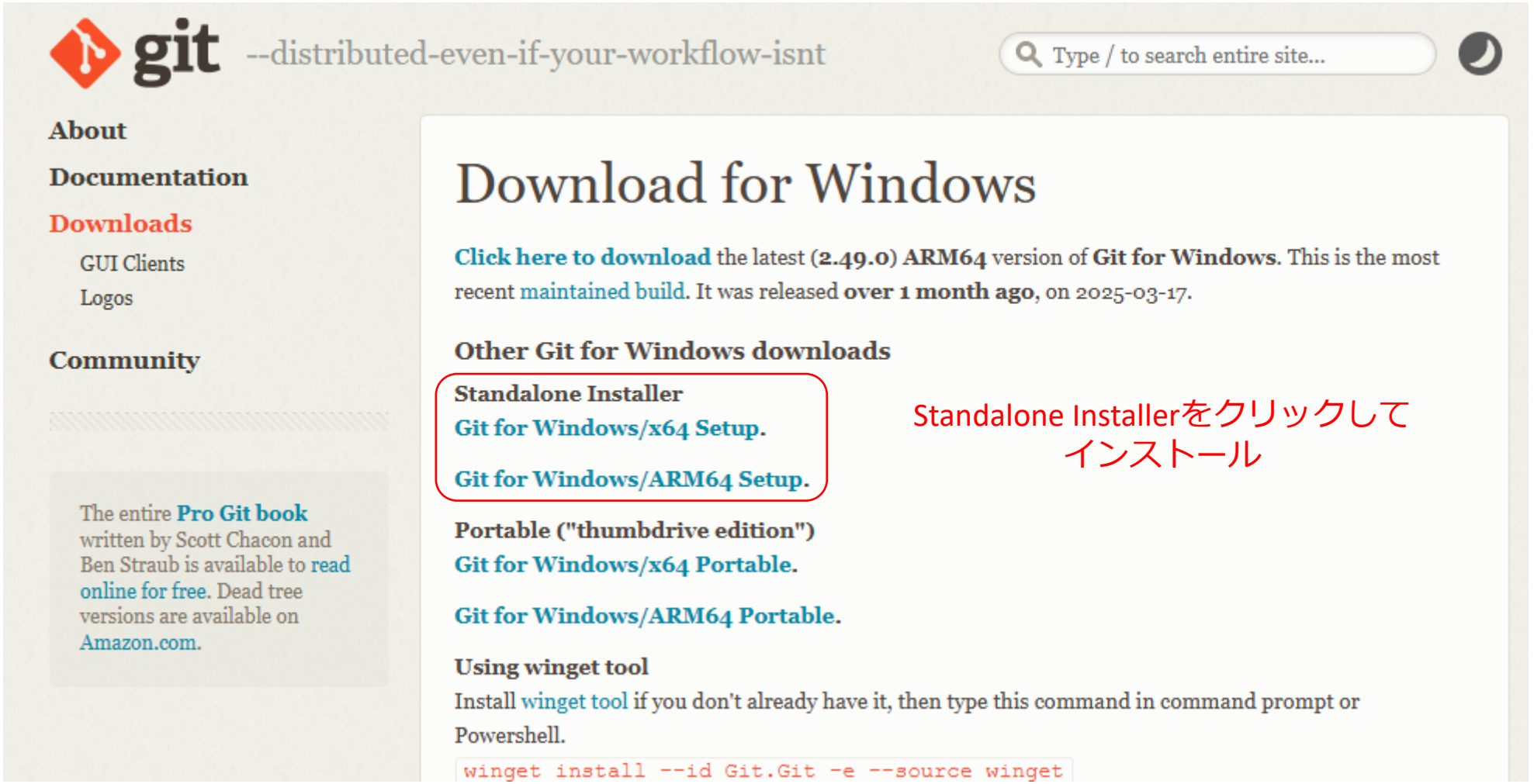
ゼロ

所要時間：30分程度



1. Gitのインストール

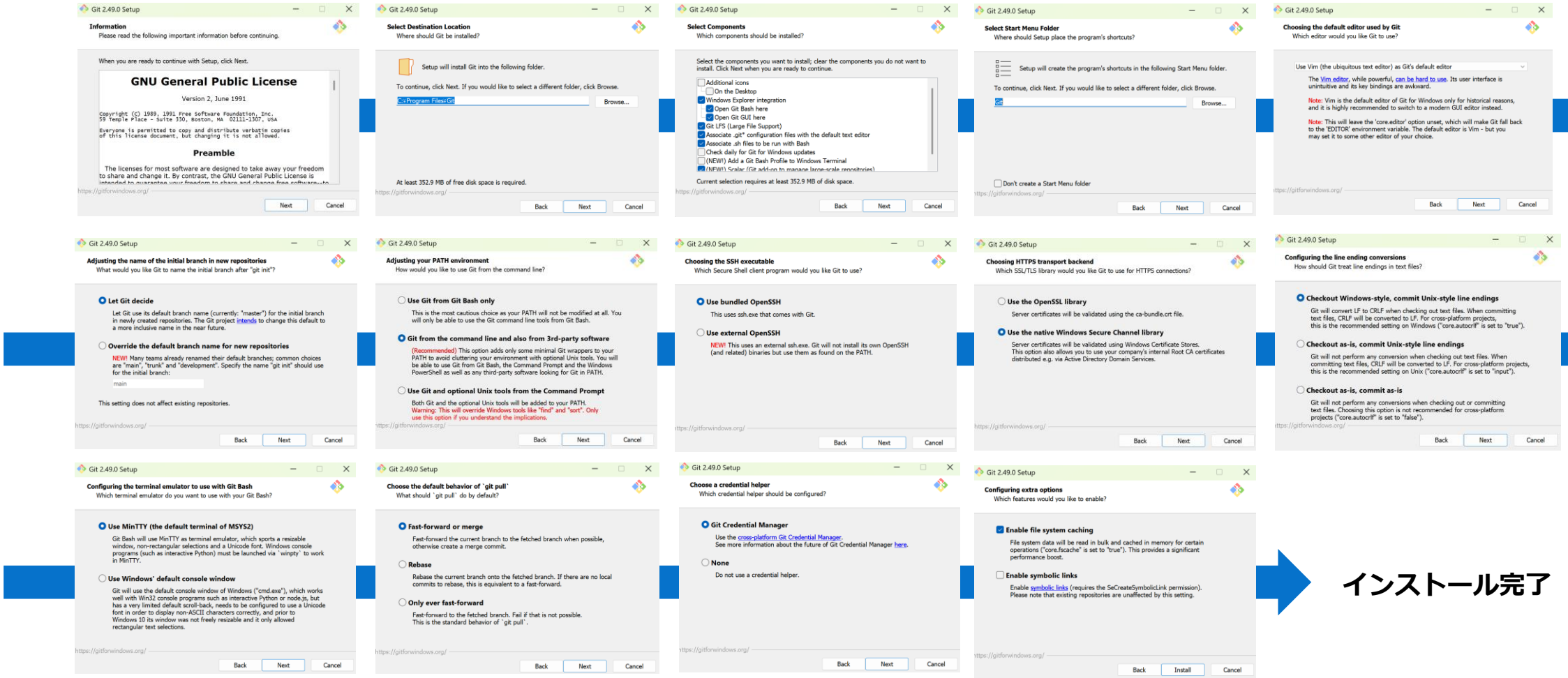
https://git-scm.com/downloads/win



Standalone Installerをクリックして
インストール

1. Gitのインストールの流れ

すべてデフォルトで進めてOK。



インストール完了

2. Node.jsとnpmのインストール

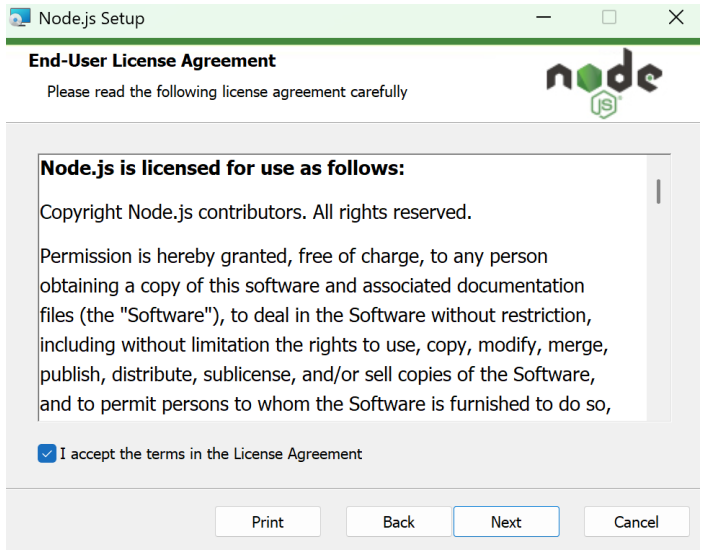
https://nodejs.org/ja/download

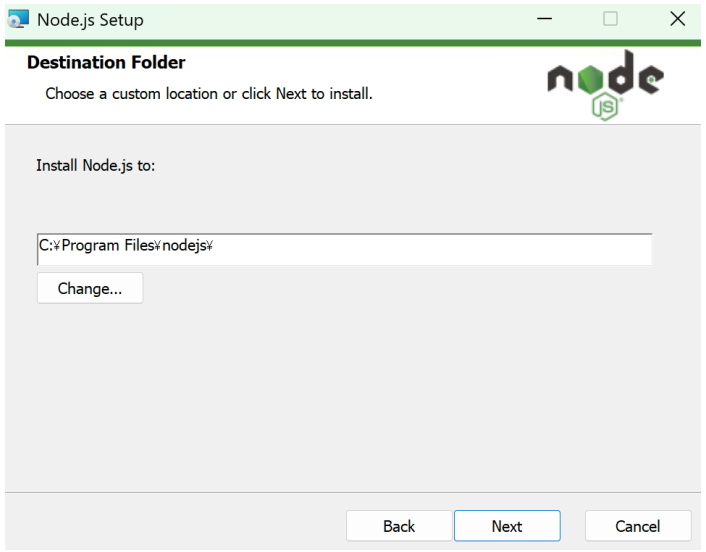
1. インストーラーからインストール
2. インストール完了の確認

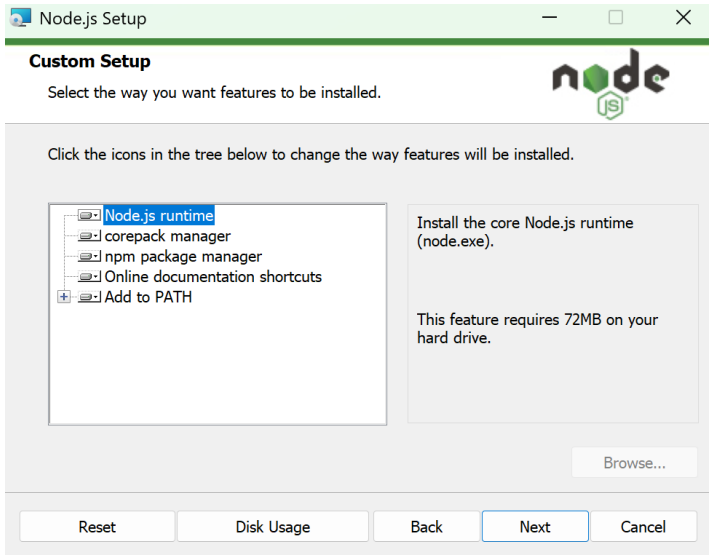
<コマンドプロンプト or PowerShell>

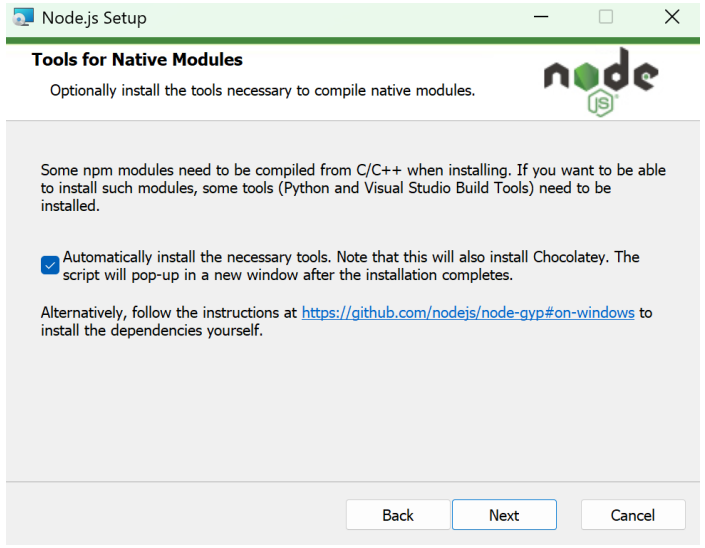
```
node -v
npm -v
```

2. Node.jsとnpmのインストールの流れ

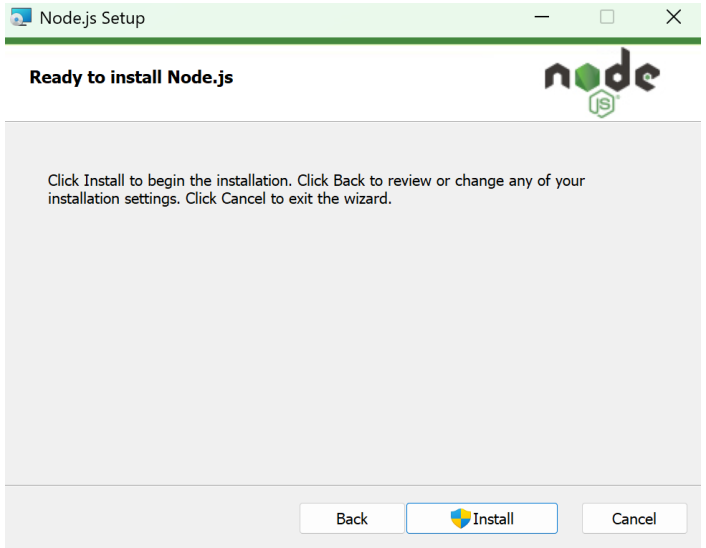


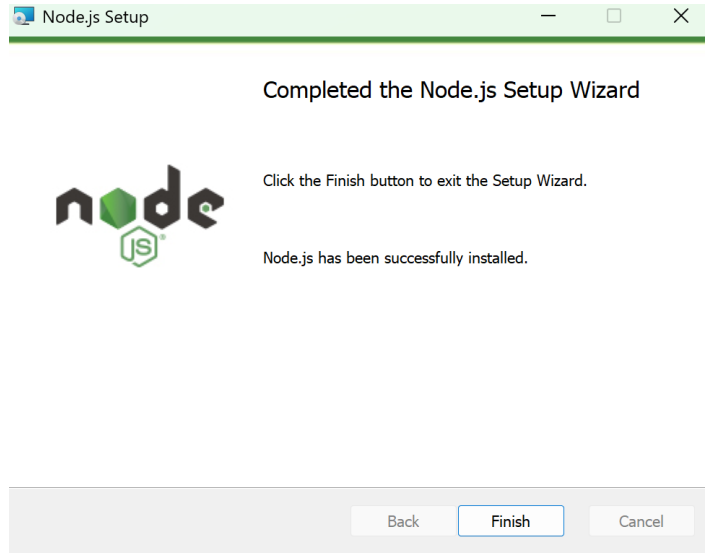






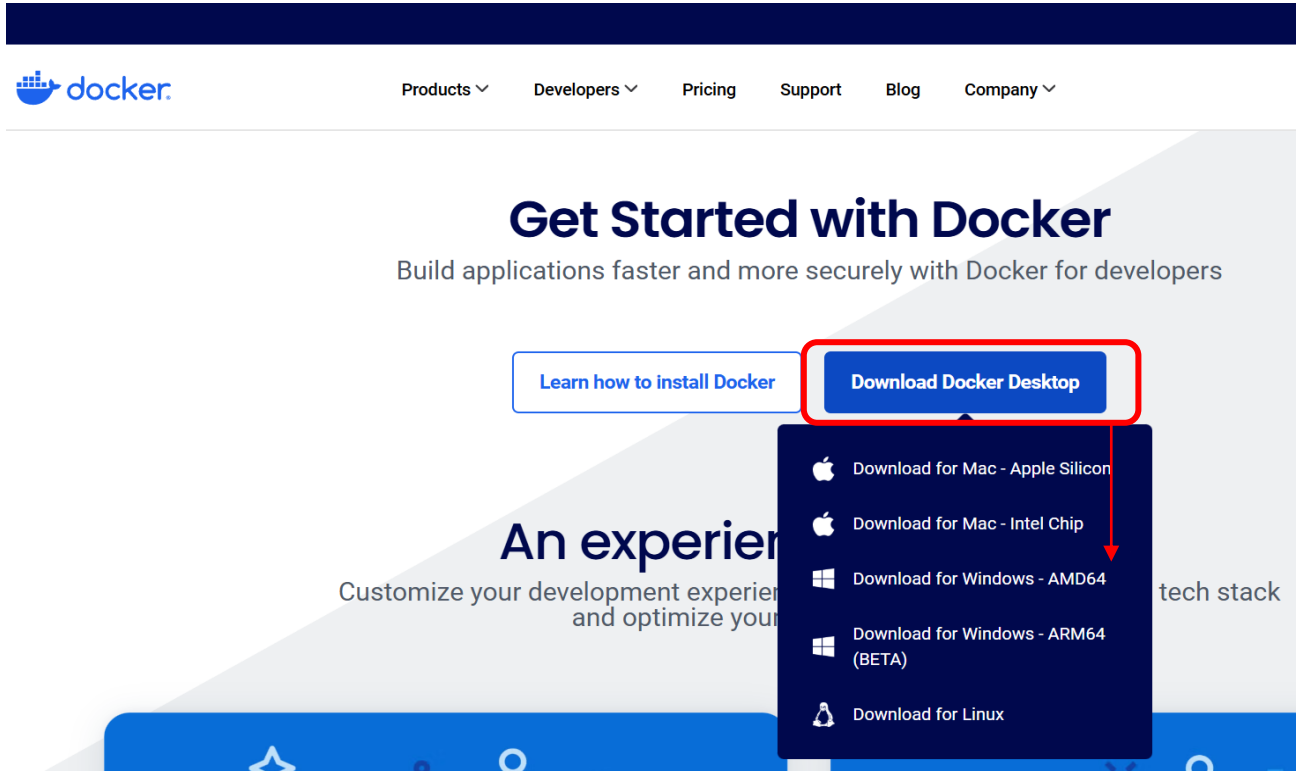
Necessary toolsのインストールは必須ではないので、チェックを入れなくてもOK。
（チェックを入れるとPythonやChocolatey等も一緒にインストールしてくれるため、他の用途でそれらを使いたい場合には便利）

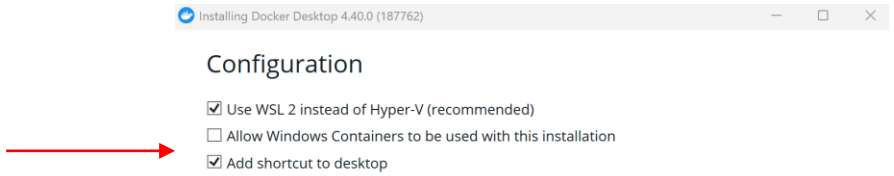


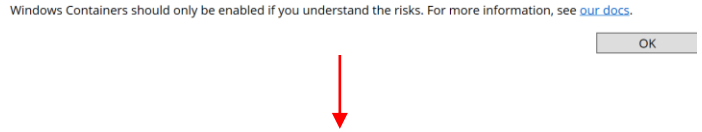


3. Docker Desktopのインストール

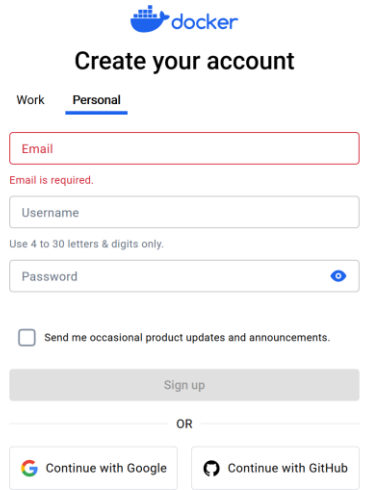
https://www.docker.com/get-started/







Dockerのアカウントをお持ちでない場合は、作成する必要があります。（Personal）



3. Docker Desktopのインストール

- インストール完了後、Docker Desktopのサインインとは別にCLIでサインインする必要あり

```
docker login u <dockerユーザー名>
```

```
PS C:\Users\App\dify\docker> docker login -u demo4dell

i Info → A Personal Access Token (PAT) can be used instead.
         To create a PAT, visit https://app.docker.com/settings


Password:

Login Succeeded
PS C:\Users\App\dify\docker>
```

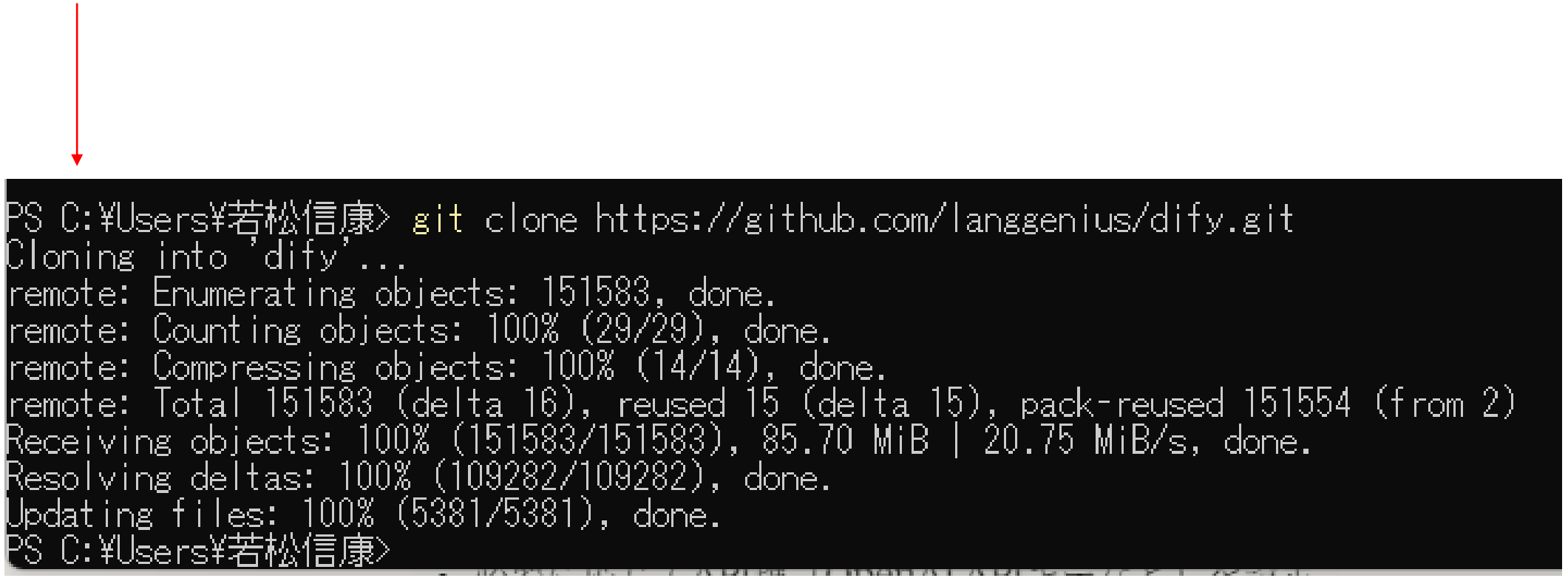
なぜ個別にサインインする必要があるのか？

Docker Desktop の UI でのサインインは、主に Docker Desktop アプリケーションの機能利用やサブスクリプション認証に用いられますが、CLI の docker コマンド（Docker Engine）が参照する認証情報ストアとは 別管理 です。そのため、CLI 操作時には改めて docker login を実行して、~/.docker/config.json や credential helper にトークンを登録する必要があります。

4. Difyのソースコードを取得

Difyのソースコードの取得：コマンドプロンプト or PowerShell

```
git clone https://github.com/langgenius/dify.git
```



5. 環境変数を設定

必須の環境変数設定

SECRET_KEY

- セッションCookieの署名（改ざん防止）やデータベース内の機密情報暗号化に必要なキー。
- 初回起動前に必ず設定が必要で、PowerShellやOpenSSLで生成します。

＊設定しなくても使用は可能ですが、セキュリティ上設定は必須とされています。

5. 環境変数を設定：SECRET_KEY変数

事前準備

0. デフォルトで作成されるDify用のサンプル環境変数ファイルをコピーして、.envファイルを作成する(Docker用とAPI用の2つ)

<コマンドプロンプト or PowerShell>

Docker用>

```
cd dify/docker
cp .env.example .env
```

API用>

```
cd dify/api
cp .env.example .env
```

.env.exampleと.envは競合しない（Difyは、.envのほうを読み込む）ため、.env.example の内容はテンプレートとして残し、実際の値を入れた .env だけに機密情報を記載するのがベストプラクティスです。

SECRET_KEY設定

1. PowerShellで乱数を生成（OpenSSLを使用する場合は、OpenSSLをインストールした上で、”openssl rand -base64 42”）

<PowerShell>

```
powershell -command "[Convert]::ToBase64String((1..64 | % { [byte](Get-Random -Max 256) }))"
```

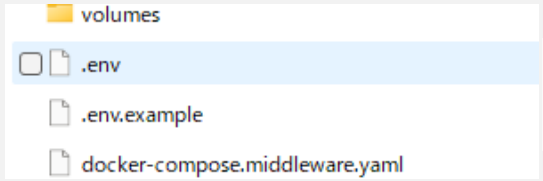
1 から 64 までを順にバイト値として乱数取得 → Base64 変換

生成した乱数は、Docker用、API用共通で使います

```
PS C:\Users\若松信康\dify\docker> powershell -command "[Convert]::ToBase64String((1..64 | % { [byte](Get-Random -Max 256) }))"
acBXA+jqFAl0rV2BDIP
```

生成された乱数

2. .envファイルへ設定




```
# A secret key that is used for securely signing the session cookie
# and encrypting sensitive information on the database.
# You can generate a strong key using `openssl rand -base64 42`.
SECRET_KEY=sk-9f73s3ljTXVcMT3Blb3ljTqtsKiGHXVcMT3BlbkFJLK7U
```

```
# A secret key that is used for securely signing the session cookie
# and encrypting sensitive information on the database.
# You can generate a strong key using `openssl rand -base64 42`.
SECRET_KEY=acBXA+jqFAl0rV2BDIPhTVQCBLcWhv2SkgjKAD2jhWWkxZXCZ6gUyOOu2!
RgO+3gyMg==
```

デフォルトの値を生成した乱数に変更する

2つの.envファイル上で同様に設定

1. Dify/docker/.env
2. Dify/api/.env

オプションの環境変数

特定の設定を指定したいとき

- **ポート・ホスト関連：**NGINX_PORT, EXPOSE_NGINX_PORT：デフォルトは80。別ポートで公開したい場合に変更します。CONSOLE_API_URL, CONSOLE_WEB_URL, SERVICE_API_URL, APP_API_URL, APP_WEB_URL, FILES_URL：デフォルトは空欄で同一ドメイン扱い。外部公開／CORS設定が必要な場合に絶対URLを指定します。
- **SSL/Let's Encrypt（Certbot）：**NGINX_HTTPS_ENABLED をtrueにし、SSL証明書ファイル名（NGINX_SSL_CERT_FILENAME／NGINX_SSL_CERT_KEY_FILENAME）を指定する場合。CERTBOT_EMAIL, CERTBOT_DOMAIN, NGINX_ENABLE_CERTBOT_CHALLENGE：自動証明書取得を行う場合に設定します。

オプション機能利用時に必要な環境変数

- **ローカルLLM連携**：Ollamaを使う場合は OLLAMA_HOST 等を systemd やユーザー環境変数で設定します。LocalAI連携では LocalAIのエンドポイントを指定。**（LM Studioでは設定不要）**
- **外部LLM API連携：**Dify上で設定することができますが、環境変数で設定するとデータベースに保存されないため、セキュリティリスクが軽減されます。共有環境や本番環境ではAPIキーを環境変数として管理するのがベストプラクティスです。
- **Vectorストア連携：**デフォルトはWeaviate（プロファイルweaviate）。他のストア（VikingDB, OceanBase, Lindormなど）を使う場合、VECTOR_STORE と該当するアクセスポイントや認証情報を設定します
- **Notionインテグレーション：**NOTION_INTEGRATION_TYPE (public/internal)、NOTION_CLIENT_ID, NOTION_CLIENT_SECRET, NOTION_INTERNAL_SECRET。ローカルではinternalが推奨で、ワークスペース内のシークレットを指定します
- **メール送信**：MAIL_TYPE (resend/smtp)、RESEND_API_KEY または SMTP_SERVER, SMTP_PORT, SMTP_USERNAME, SMTP_PASSWORD などを設定します。
- **その他外部サービス**：Unstructured API（UNSTRUCTURED_API_URL, UNSTRUCTURED_API_KEY）や、Sentry（SENTRY_DSN）、各種RDBMS/Redisの接続情報なども必要に応じて設定します。

開発・デバッグ用環境変数

- **DEBUG, FLASK_DEBUG**：ローカル開発時のトラブルシュート用。デフォルトfalseだが、バグ解析時はtrueにします。
- **LOG_LEVEL**：ログ出力レベル（DEBUG/INFO/ERRORなど）を切り替え。開発ではDEBUG、本番ではINFO以上がおすすめです。

5. 環境変数を設定

オプションの環境変数設定が必要かどうかわからないとき

不要

1. 同一PC内でのみアクセスする場合：Dockerを実行しているのと同じPCからブラウザでhttp://localhostにアクセスする
2. デフォルト設定で十分な場合：".env.example"を".env"にコピーしただけの状態でも、デフォルト設定は含まれています。デフォルトでは、APIエンドポイントは自動的にlocalhostや適切なDockerネットワーク内の参照に設定されています。
3. Docker Composeを使用している場合：Docker Composeは、コンテナ間のネットワーク通信を自動的に設定します。サービス名をホスト名として使用できるため、明示的なIPアドレス指定が不要になります。

必要

1. **別のデバイスからアクセスしたい場合**：実行しているPC以外の端末からアクセスする場合は、CONSOLE_API_URLとAPP_API_URLを設定する必要があります
2. **特定のLLMプロバイダーを使用したい場合**：OpenAI、Azure、Anthropicなどの特定のAPIキーを使用する場合
3. **デフォルト以外の設定が必要な場合**：データベース設定、ストレージ設定など、デフォルト以外の構成にしたい場合

✓ 個人利用や試用段階：UI上での設定で十分です
✓ 本番環境や共有環境：環境変数での設定が推奨されます

6. Docker、Difyを起動する

インストールした
Docker Desktopを
起動しておく

Difyの起動:コマンドプロンプト or PowerShell

```
cd dify/docker
docker compose up -d
```

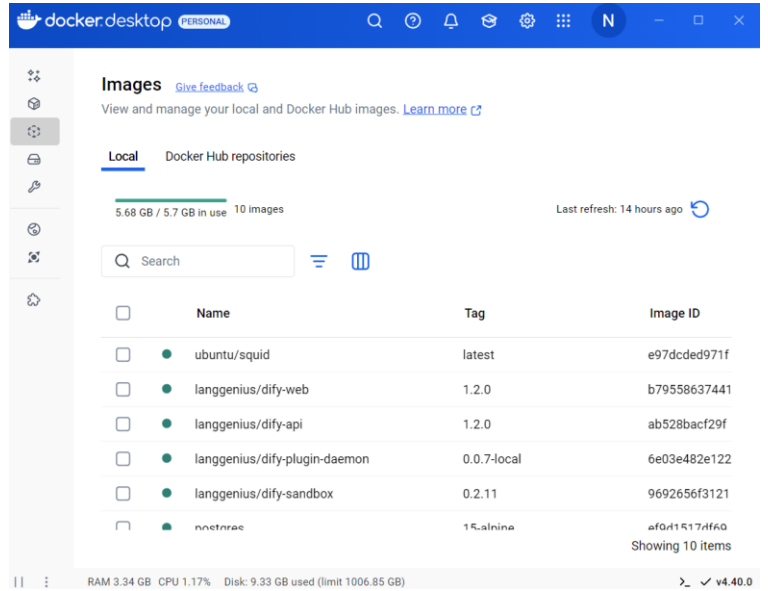
＊“-d”（タッチモード）：バックグラウンドで起動します

```
PS C:\Users\若松信康\dify\docker> docker compose up -d
[+] Running 12/12
 ✓ Network docker_default                 Created
 ✓ Network docker_ssrf_proxy_network     Created
 ✓ Container docker-redis-1              Started
 ✓ Container docker-sandbox-1            Started
 ✓ Container docker-weaviate-1           Started
 ✓ Container docker-ssrf_proxy-1         Started
 ✓ Container docker-web-1                Started
 ✓ Container docker-db-1                 Started
 ✓ Container docker-plugin_daemon-1      Started
 ✓ Container docker-api-1                Started
 ✓ Container docker-worker-1             Started
 ✓ Container docker-nginx-1              Started
PS C:\Users\若松信康\dify\docker>
```

Dockerのイメージが起動

dify/dockerのディレクトリに移動

Docker-composeでDifyのコンテナを起動

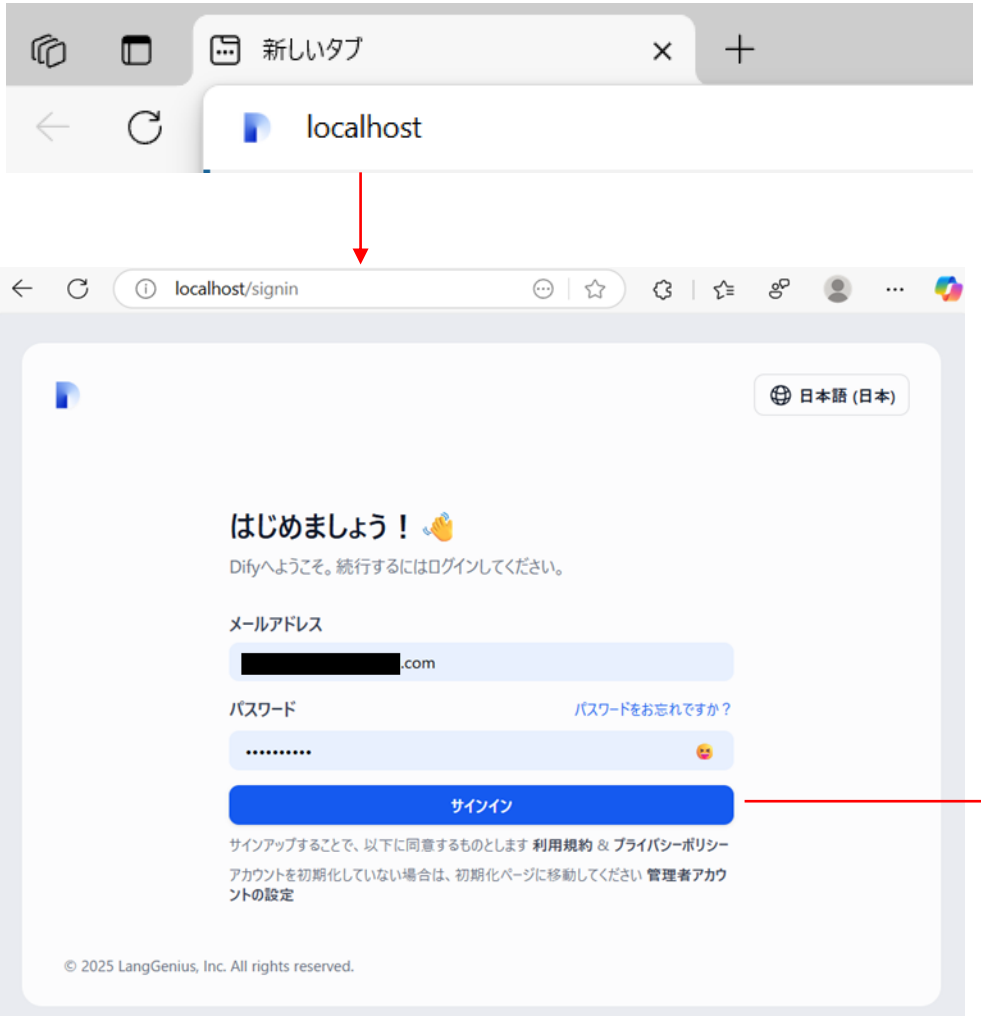


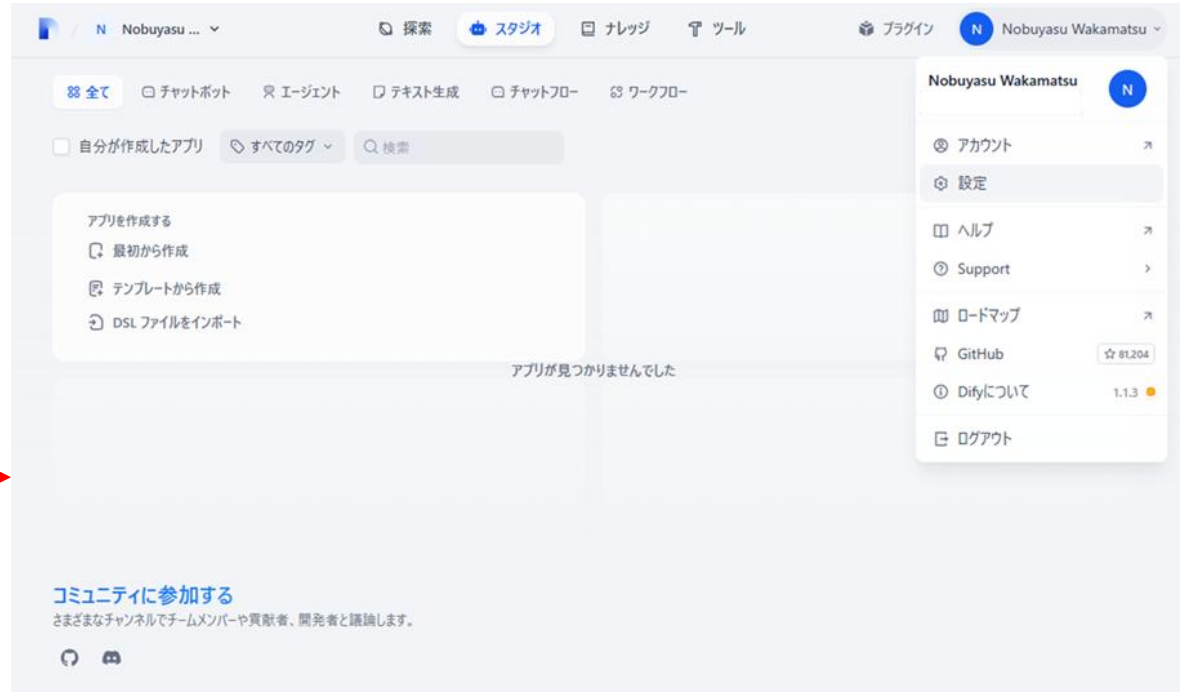
Docker Desktopアプリ上でもDockerイメージ
が起動していることを確認できる

（参考）Docker Composeとは？

- 複数のコンテナで構成されるアプリケーションを定義・実行するためのツールです。
- 主な特徴
  - **マルチコンテナアプリケーションの定義**：Compose では、YAML（ヤムル） 形式の docker-compose.yml ファイルを使い、どのサービス（コンテナ）をどのような設定（環境変数、ボリューム、ネットワーク設定など）で実行するかを記述します。このファイル一つでアプリケーション全体の構成を管理できるため、複数のサービスが連携して動作する環境を簡単に再現できます。
  - **簡易なオーケストレーション**：定義ファイルを元に、docker compose up コマンドなどを実行するだけで、すべてのサービス（例えば、Web サーバー、データベース、キャッシュサーバーなど）を一括で起動・停止できます。また、サービスの再構築やログの確認も CLI コマンド一つで行え、開発・テスト環境の構築が非常に容易です。
  - **環境の再現性**：YAML ファイルで環境を一元管理するため、チームメンバー全員が同じ構成環境を簡単に再現でき、開発から本番まで一貫した動作を担保できます。
  - **拡張性と自動化**：複雑な依存関係がある場合や、複数のコンテナを組み合わせた開発・テストを自動化する際に、Compose は非常に有用です。CI/CD パイプラインなどで、コードの変更に合わせて自動で環境を構築・破棄する際にも利用されます。
- まとめ
  - YAML ファイル（docker-compose.yml）でアプリ全体のサービス構成、ネットワーク、ボリュームなどを定義
  - docker compose up などのシンプルなコマンドでマルチコンテナ環境を一括起動・停止
  - チームや CI/CD で環境の再現性と自動化が実現できる

7. ブラウザでlocalhostにアクセス





- ✓ 最初に起動したときには、管理者アカウントの設定画面が開く
- ✓ 設定後サインイン画面へ

仕組み：Difyコミュニティ版のローカル展開では、HTTPのサービスがデフォルトでポート80にバインドされるため、ブラウザでポート番号を指定しなくてもアクセスできます。ポート80は、リバースプロキシ（nginxサービス）がホスト側に公開しているポートであり、内部でバックエンドのサービス（api：5001やweb：3000など）と通信します。

（参考）Difyに割り当てられたポートの確認方法

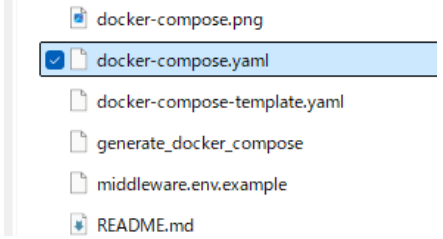
アクセスできないときはポートの割り当てを確認してみましょう。

<コマンドプロンプト or PowerShell>

```
docker ps
```

80/TCPポートが割り当てられている

```
PS C:\Users\若松信康\dify\docker> docker ps
CONTAINER ID   IMAGE                                     COMMAND                  CREATED         STATUS                            PORTS                                      NAMES
59eb6120a4a6   nginx:latest                              "sh -c 'cp /docker-e…"   7 seconds ago   Up Less than a second             0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp   docker-nginx-1
e30fa5bbc482   langgenius/dify-api:1.2.0                 "/bin/bash /entrypoi…"   7 seconds ago   Up 5 seconds                      5001/tcp                                   docker-worker-1
43309bb13269   langgenius/dify-api:1.2.0                 "/bin/bash /entrypoi…"   7 seconds ago   Up 5 seconds                      5001/tcp                                   docker-api-1
74f9c1adff51   langgenius/dify-plugin-daemon:0.0.7-local "/bin/bash -c /app/e…"   7 seconds ago   Up 4 seconds                      0.0.0.0:5003->5003/tcp                     docker-plugin_daemon-1
9b03ebf2aed7   postgres:15-alpine                        "docker-entrypoint.s…"   7 seconds ago   Up 6 seconds (healthy)            5432/tcp                                   docker-db-1
f3a686381ea9   redis:6-alpine                            "docker-entrypoint.s…"   7 seconds ago   Up 6 seconds (health: starting)   6379/tcp                                   docker-redis-1
CONTAINER ID   IMAGE                                     COMMAND                  CREATED         STATUS                    PORTS                                      NAMES
59eb6120a4a6   nginx:latest                              "sh -c 'cp /docker-e…"   6 minutes ago   Up 6 minutes              0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp   docker-nginx-1
e30fa5bbc482   langgenius/dify-api:1.2.0                 "/bin/bash /entrypoi…"   6 minutes ago   Up 6 minutes              5001/tcp                                   docker-worker-1
43309bb13269   langgenius/dify-api:1.2.0                 "/bin/bash /entrypoi…"   6 minutes ago   Up 6 minutes              5001/tcp                                   docker-api-1
74f9c1adff51   langgenius/dify-plugin-daemon:0.0.7-local "/bin/bash -c /app/e…"   6 minutes ago   Up 6 minutes              0.0.0.0:5003->5003/tcp                     docker-plugin_daemon-1
9b03ebf2aed7   postgres:15-alpine                        "docker-entrypoint.s…"   6 minutes ago   Up 6 minutes (healthy)    5432/tcp                                   docker-db-1
f3a686381ea9   redis:6-alpine                            "docker-entrypoint.s…"   6 minutes ago   Up 6 minutes (healthy)    6379/tcp                                   docker-redis-1
3763279b4cb4   ubuntu/squid:latest                       "sh -c 'cp /docker-e…"   6 minutes ago   Up 6 minutes              3128/tcp                                   docker-ssrf_proxy-1
8674e5565ce0   langgenius/dify-web:1.2.0                 "/bin/sh ./entrypoin…"   6 minutes ago   Up 6 minutes              3000/tcp                                   docker-web-1
a8cbd3dd76cc   langgenius/dify-sandbox:0.2.11            "/main"                  6 minutes ago   Up 6 minutes (healthy)                                               docker-sandbox-1
e85ca6fc555e   semitechnologies/weaviate:1.19.0          "/bin/weaviate --hos…"   6 minutes ago   Up 6 minutes                                                         docker-weaviate-1
PS C:\Users\若松信康\dify\docker>
```



✓ docker-compose.yamlファイル（設定ファイル）でも確認可能

ホスト側　コンテナ側

```
    web
    ports:
      - '${EXPOSE_NGINX_PORT:-80}:${NGINX_PORT:-80}'
      - '${EXPOSE_NGINX_SSL_PORT:-443}:${NGINX_SSL_PORT:-443}'
```

（参考）ポートが競合していてアクセスできないケース

ポート80を利用する主なアプリケーション
- Web サーバー: Apache HTTP Server, Nginx, Microsoft IIS など
- その他サービス: 一部の組み込みデバイス管理 UI、VPN やプロキシ管理コンソールがホストの 80 番に Web UI を提供することがあります。

既にポート 80 を占有しているサービス（たとえば Apache、Nginx、IIS など）が稼働中だと、Dify のコンテナ起動時に以下のようなエラーが出て立ち上がりません。

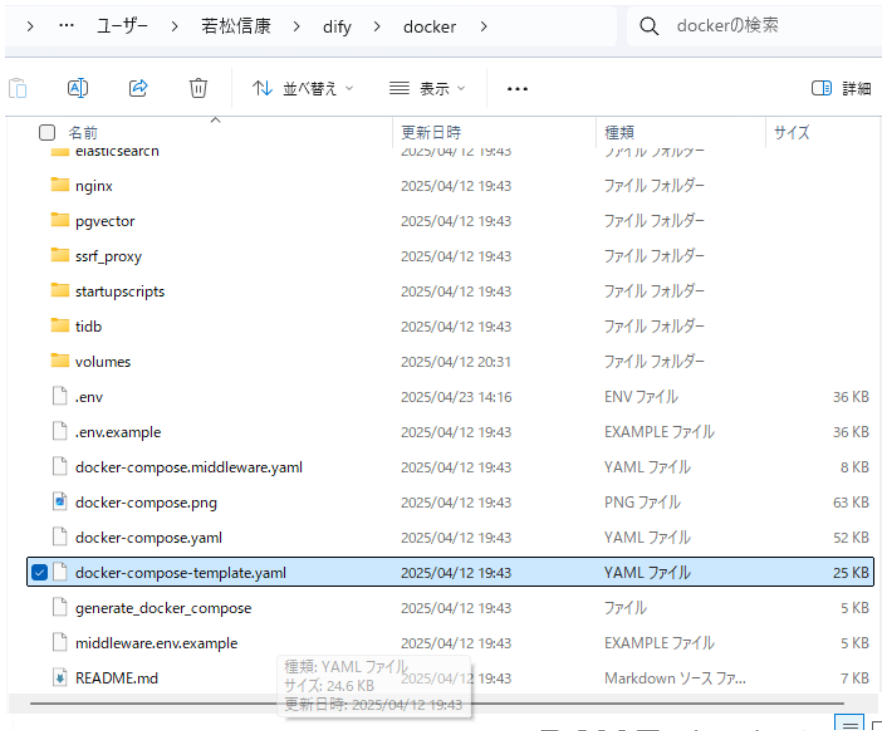
Error response from daemon: Ports are not available: listen tcp 0.0.0.0:80: ...

✓ docker-compose.yamlファイル（設定ファイル）を直接編集する方法

テキストエディタでホスト側のポートを書き換え

```
ports:
  - '${EXPOSE_NGINX_PORT:-80}:${NGINX_PORT:-80}'
  - '${EXPOSE_NGINX_SSL_PORT:-443}:${NGINX_SSL_PORT:-443}'
```

```
ports:
  - '${EXPOSE_NGINX_PORT:-8080}:${NGINX_PORT:-80}'
  - '${EXPOSE_NGINX_SSL_PORT:-443}:${NGINX_SSL_PORT:-443}'
```



2.ローカルLLM実行環境構築

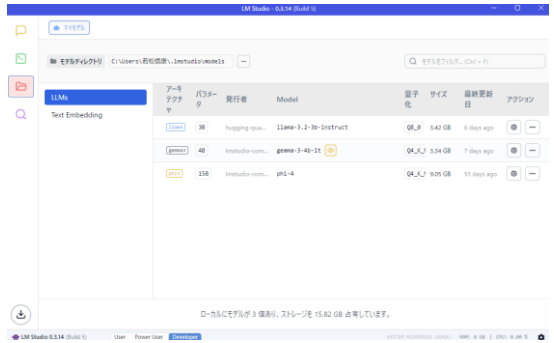
【手順】LM Studioの場合

1. LM Studioのインストール
2. モデルのダウンロード
3. モデルのロード

ゼロ

所要時間：15分程度

*ほぼLLMをローカルにダウンロードする時間
（手を動かす時間は5分程度）



ローカルでLLMを展開・利用するための主なツール：概要

	仕組み	特徴
LM Studio	デスクトップアプリケーションとして提供され、GUI（グラフィカルユーザーインターフェース）を通じてモデルの検索、ダウンロード、動かし、そしてローカルでの推論実行ができるようになっています。内部的には、Transformers などのライブラリや、llama.cpp のような高速推論エンジンを活用しており、ユーザーが設定や管理を直感的に行えるよう最適化されています。	ビジュアルなインターフェースで操作できるため、初心者でも使いやすい。複数のモデルや量子化オプションを簡単に切り替えられ、パフォーマンスの調整も可能。
Ollama	Ollama は、あらかじめ量子化（例：GGUF 形式）されたモデルの重みをローカルにダウンロード、コマンドラインや REST API 経由で実行できる環境を提供します。ユーザーは、ターミナルで単純なコマンド（例：ollama run モデル名）を実行するだけで、モデルをローカルでの対話を開始できます。	すぐに使えるプリセットのモデルライブラリがあり、容易にモデルのダウンロード・更新が可能。CLI や API を通じてシンプルに操作できるため、開発者向けの軽量なローカルサーバーとして機能する。
Hugging Face	手動での環境構築とコード実装。Hugging Face の Model Hub からダウンロードしたモデル（通常は PyTorch や TensorFlow のフォーマット）は、Transformers ライブラリを利用して Python コードで直接コーディング、推論処理を実装します。この方法は、柔軟性が高い反面、プログラミング知識や環境構築（GPU の設定、依存関係の管理など）が必要になります。自前のカスタマイズが可能。前処理、後処理、ファインチューニングなども自由に行えるため、実験や研究用途には最適です。また、モデルの動作やパフォーマンスを細かく調整できるメリットがあります。	自由度とカスタマイズ性が高く、コードレベルで操作するため、環境構築やプログラミングの技術が求められます。
Text Generation Inference (TGI)	HuggingFaceが開発した高性能な推論サーバー。大規模言語モデルのサービング向けに最適化されています。Rustで書かれたバックエンドとPythonのフロントエンドを組み合わせています。	複数GPUにわたるモデル並列処理、連続バッチ処理、トークンストリーミングなど高度な機能を提供します。本番環境向けの最適化がされており、モデルの提供に特化しています。
vLLM	PagedAttentionの技術を使用する推論エンジンで、GPUビデオ効率的な管理と高速なバッチ処理を実現した推論エンジンです。OpenAI互換APIも提供しています。	従来のトランスフォーマー実装と比較して大幅な高速化を実現。高いスループットが特徴です。並列推論や連続バッチ処理に優れており、サーバー環境での大規模デプロイに向いています。
OpenLLM	サーバーベースのオープンソースフレームワークで、複数のモデルをAPIエンドポイントとして提供・管理します。BentoMLをベースにしており、モデルのデプロイ、スケーリング、モニタリングが可能です。多様なLLM（Llama、Falcon、MPT、Dollyなど）をサポートし、サーバーとしてHTTP/gRPC APIを公開します。	モデル管理の一元化と複数モデルの同時運用が容易です。本番環境向けの機能（モニタリング、ロギング、スケーラビリティなど）が充実しており、MLOpsの観点から優れています。Hugging Faceモデルとの互換性が高く、APIを通じてアプリケーションから簡単に利用可能です。
LiteLLM	様々なLLMプロバイダーとモデル（OpenAI、Anthropic、Hugging Face、ローカルモデルなど）へのアクセスを統一インターフェースで提供するライブラリです。	異なるモデル間の切り替えが容易で、コードの変更なしに様々なモデルを試すことができます。ロギング、モニタリング、フォールバック機能なども備えています。
NVIDIA NIM	NVIDIAが開発した推論マイクロサービスフレームワークで、大規模言語モデルをローカルで最適化して実行するためのツールです。GPUの性能を最大限に活用し、TensorRT-LLMを基盤としてモデルの最適化と高速推論を可能にします。	NVIDIA GPUに特化した最適化により、同等の他ツールと比較して高速な推論が可能です。企業向けの堅牢性と拡張性を持ち、本番環境での大規模デプロイメントに適しています。ただし、NVIDIA GPUが必須となります。
Docker Model Runner (現状Mac版のみ、Win版は4月中提供予定)	・Docker コンテナを利用して、LLM の推論サーバーをローカル環境で稼働させる仕組みです。モデルをコンテナパッケージ化することで、依存関係の管理や環境構築の再現性を軽減し、一度構築すれば同じコンテナであればどこでも同じように動作することができます。・基本的には、すでに量子化されたモデルやカスタマイズしたワークフローを コンテナ化して進捗エンジンと、コンテナとして実行することを特化しており、モデル自体は Hugging Face からダウンロードしたものを含む含みますが、ユーザーが直接コードを実装する必要はなく、コンテナの起動・停止といった操作で簡単にローカル推論環境を整えれます。	・シンプルなセットアップでローカル環境に依存せず一貫した実行環境を提供します。スケーラビリティが高く、必要に応じてリソースの割り当てを調整可能です。セキュリティ面でも隔離された環境でモデルを実行できる利点があります。開発からデプロイまでのワークフローの統一が可能で、チーム間での環境の差異による問題を軽減します。
llama.cpp	C++で書かれた高度に最適化されたLLM推論エンジン。主にLlamaモデル向けに開発されましたが、現在は様々なモデルフォーマット（特にGGUF）をサポートしています。量子化技術を活用し、CPUでも効率的に動作するよう設計されており、コマンドラインインターフェースでの操作が基本です。バックエンドライブラリとしても使用でき、様々なツールやUIから利用されています。	非常に軽量で低リソース環境（CPU only）でも動作可能。高度な量子化オプションによりメモリ使用量を大幅に削減できます。様々なプラットフォーム（Windows、Mac、Linux、Android等）に対応し、多様なハードウェアで動作します。カスタマイズ性と拡張性に優れており、多くの派生プロジェクトの基盤になっています。
LocalAI	OpenAI API互換のインターフェースを持つローカルAPIサーバー。様々なバックエンド（llama.cpp、ggml等）を統合し、テキスト生成だけでなく、音声認識や画像生成など複数のAI機能をローカルで提供します。コンテナ化されており、デプロイが容易です。	OpenAI互換APIにより既存のOpenAIベースのアプリをそのまま移行可能。マルチモーダル（テキスト、画像、音声）対応で多機能。プラグイン機能による拡張が容易でエコシステムが充実。リソース効率が良く、比較的軽量なハードウェアでも動作します。コミュニティによる活発な開発が続いています。

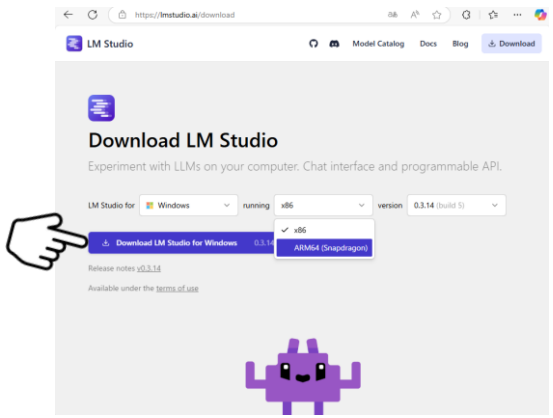
ローカルでLLMを展開・利用するための主なツール：比較

	種別	主な用途	簡易さ	GPUサポート	API提供	GUI	リソース効率	カスタマイズ性	拡張性	マルチモデル対応	PC環境適合性	サーバー環境適合性	長所	短所
LM Studio	デスクトップGUIアプリ	初心者向け簡単導入、小規模プロジェクト/テスト/開発	★★★	★★	★★	★★★	★★	★	★	★★★	★★★	×	直感的なGUIインターフェース、モデル管理が視覚的、チャット機能内蔵	リソース消費が大きい、スクリプティングに制限あり
Ollama	エンドユーザー向けランタイム	初心者向け簡単導入、小規模プロジェクト/テスト/開発	★★★	★★	★★★	× Open WebUI連携可	★★	★	★	★★	★★★	★	コマンドライン操作が簡単、迅速なセットアップ、REST APIが利用可能	高度なカスタマイズが難しい、大規模デプロイに不向き
Hugging Face	エコシステム/モデルリポジトリ	研究開発、カスタマイズ、実験	★	★★★	★★	×	★	★★★	★★	★★★	★★	★★	膨大なモデルライブラリ、高度なカスタマイズ可能、研究開発に最適	技術的知識が必要、セットアップが複雑
Text Generation Inference (TGI)	推論サーバー	大規模LLMのサービング基盤	★	★★★	★★★	× Open WebUI連携可	★★	★★	★★★	★★	★	★★★	本番環境向け最適化、高いスループット、トークンストリーミング対応	設定が複雑、リソース要件が高い
vLLM	高性能推論エンジン	高スループット推論処理	★	★★★	★★★	× Open WebUI連携可	★★★	★★	★★★	★★	★	★★★	非常に高速な推論、メモリ効率が良い、OpenAI互換API	設定の柔軟性に制限、GPUが必須
OpenLLM	MLOpsフレームワーク	本番環境でのモデルデプロイと管理	★	★★★	★★★	×	★★	★★	★★★	★★★	★	★★★	包括的なMLOps機能、多様なモデルをサポート、モニタリング機能充実	学習曲線が急、リソース要件が高い
LiteLLM	抽象化ライブラリ	複数モデル/プロバイダの統一インターフェース	★★	★★	★★★	× Open WebUI連携可	N/A	★★	★★	★★★	★★	★★★	複数モデル間の簡単な切り替え、モニタリング機能、フォールバック機能	自身は推論エンジンでない、他ツールへの依存
NVIDIA NIM	ハードウェア最適化フレームワーク	NVIDIA GPU向け高性能推論	★	★★★	★★★	×	★★★	★★	★★★	★★	★	★★★	NVIDIA GPU向け最適化、企業レベルのパフォーマンス、TensorRT-LLM統合	NVIDIA GPUが必須、設定の複雑さ
Docker Model Runner（現状Mac版のみ、Win版は4月中提供予定）	コンテナベースソリューション	環境非依存デプロイ	★★	★★	★★	×	★★	★★	★★★	★★	★★	★★★	環境に依存しない実行、一貫したデプロイ、隔離された実行環境	、コンテナ管理の知識が必要、オーバーヘッドあり
Llama.cpp	軽量推論ライブラリ	低リソースでのLLM実行	★★	★★	★	×	★★★	★★	★	★★	★★★	★★	低リソースでも動作、多様なデバイスに対応、高度に最適化された推論	GUIがない、機能が限定的
LocalAI	統合AIサーバー	OpenAI API互換のローカル代替	★★	★★	★★★	× Open WebUI連携可	★★	★★	★★	★★★	★★	★★★	OpenAI互換API、マルチモーダル対応、プラグイン拡張性	高度なパフォーマンス最適化に欠ける、設定が複雑になりうる

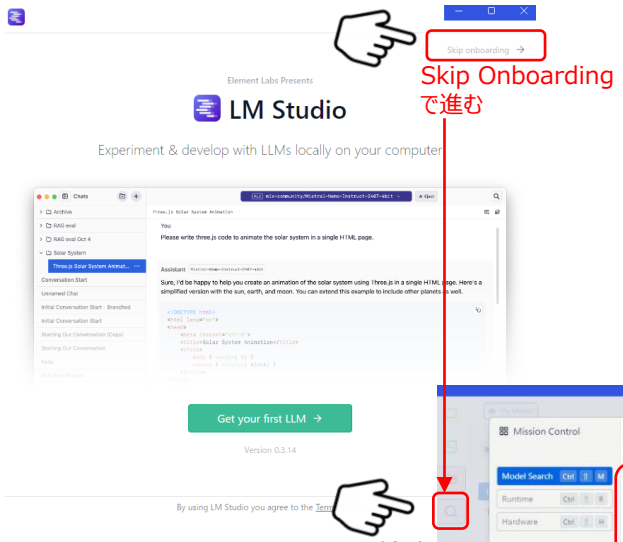
LM Studio（ローカルLLM実行環境）構築

1. LM Studioのインストール

[Download LM Studio - Mac, Linux, Windows]

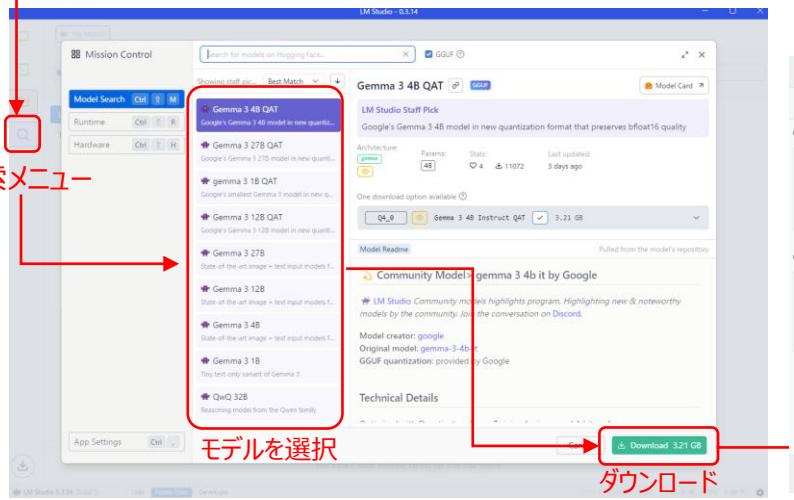


2. モデルのダウンロード



Skip Onboarding
で進む

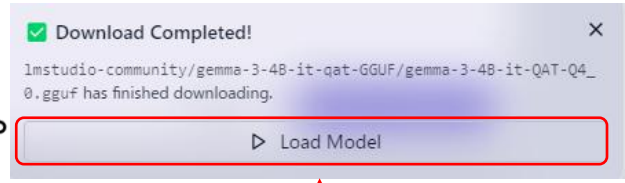
検索メニュー

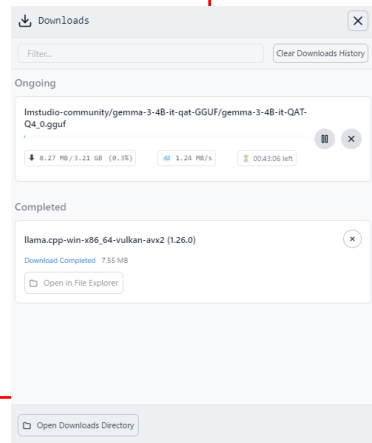


モデルを選択

ダウンロード

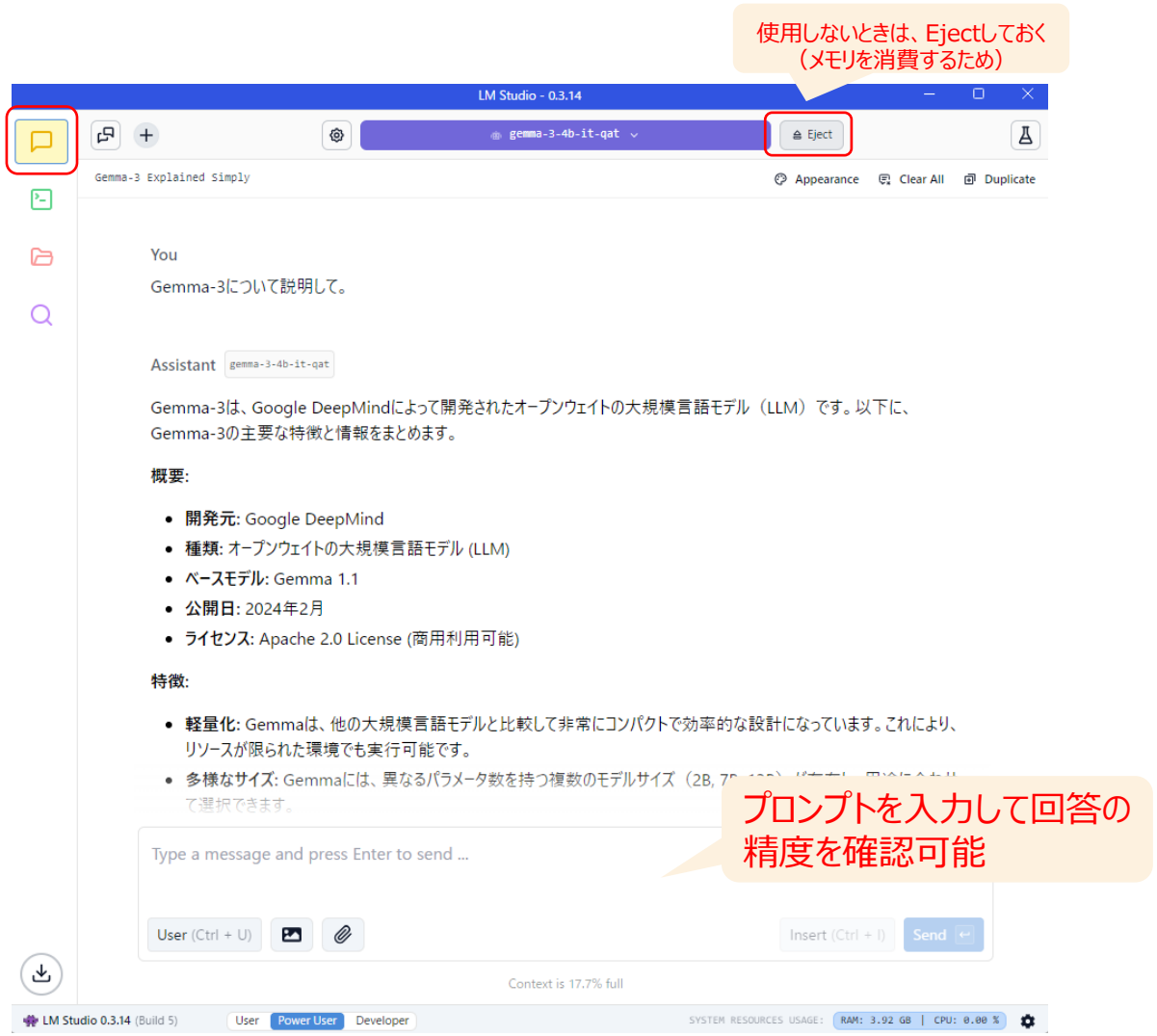
3. モデルのロード





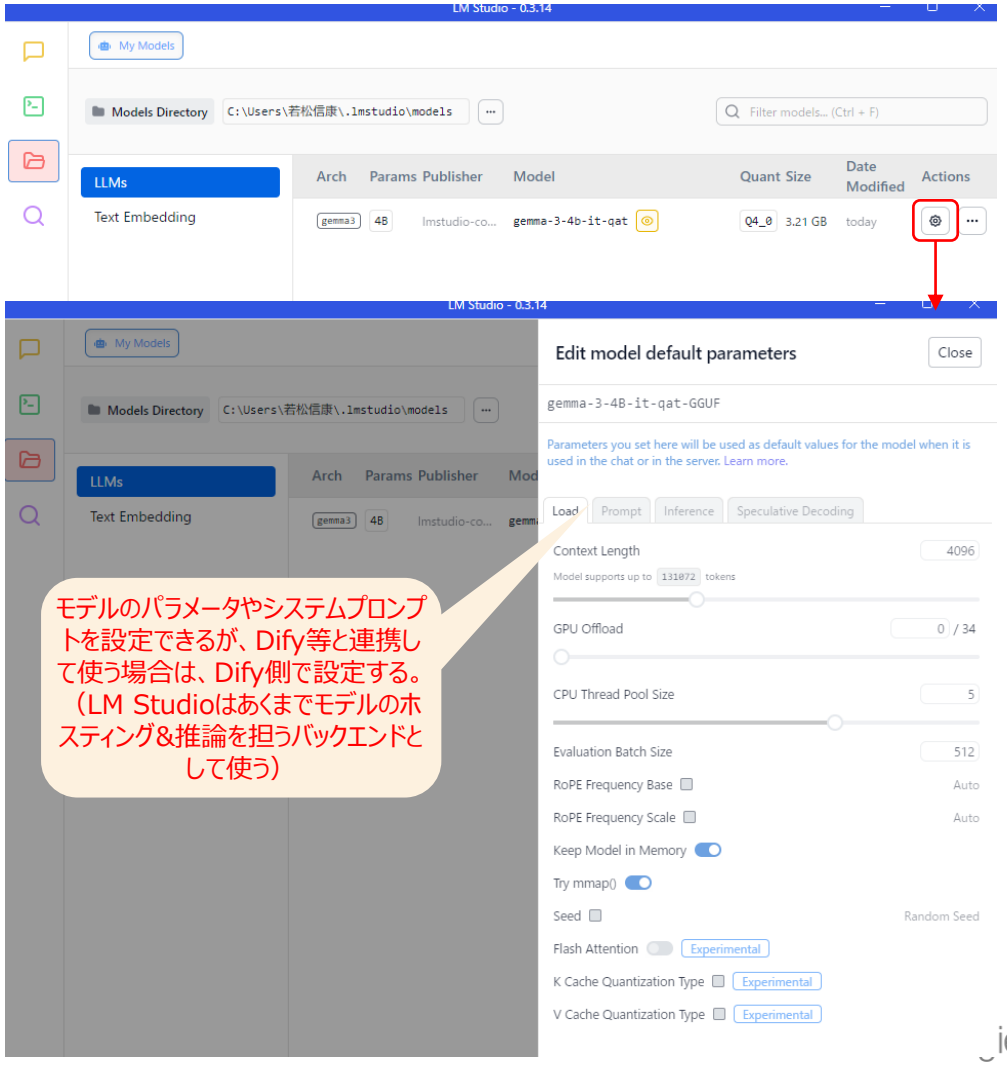
（参考）LM Studioのメニュー

ロードしたモデルに対してチャットUIですぐに出力確認できる



モデルのパラメータ設定、システムプロンプト等

（複数のアプリで同じLLMパラメータ値を共有する場合に一括で適用可能）

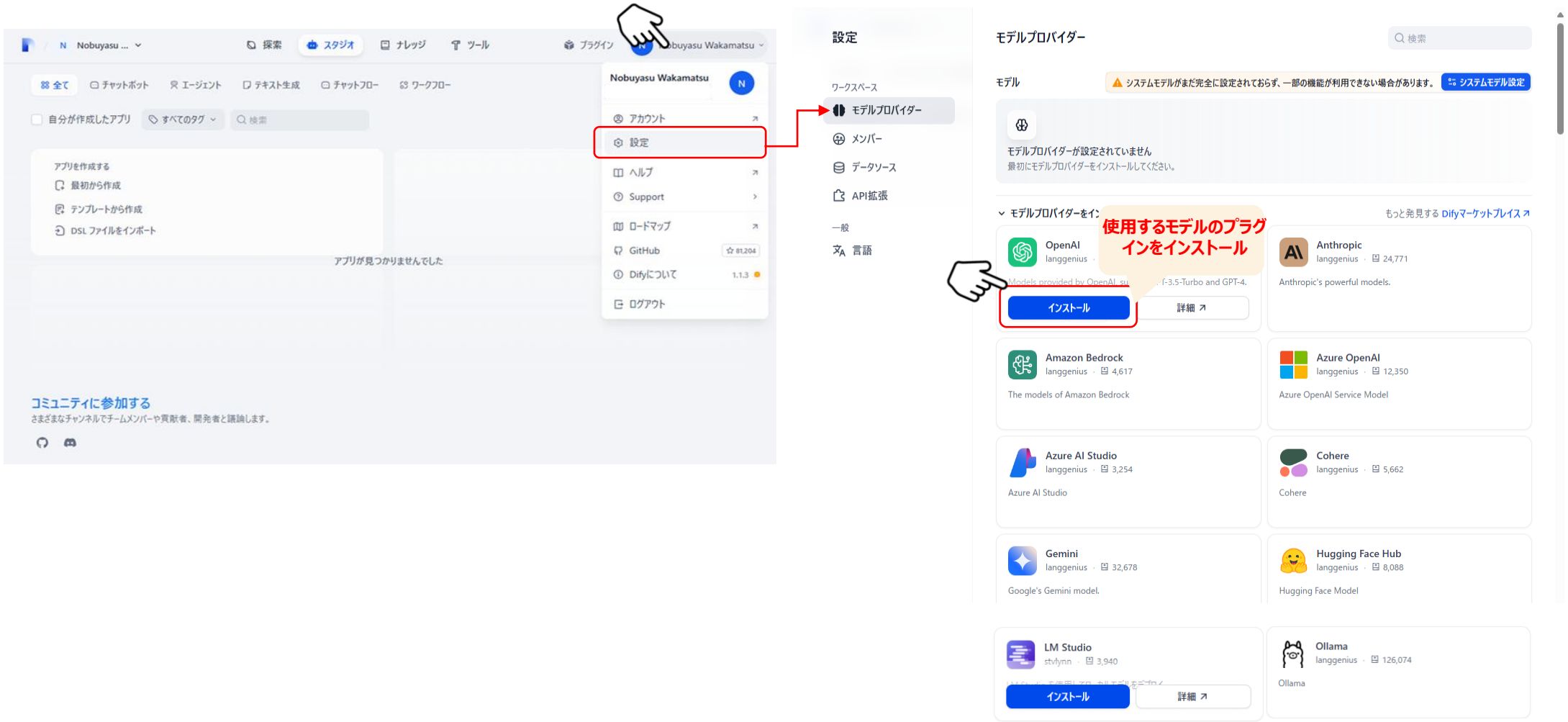


3.Dify初期設定

– モデルプロバイダの設定

Difyの初期設定：モデルプロバイダの設定

＜Dify＞モデルプロバイダーメニューから使用するLLMプラグインをインストール



Difyの初期設定：ローカルLLM設定 - LM Studioの場合

外部スクリプトや別ツール（今回の場合はDify）からREST API（OpenAI互換エンドポイントを含む）でLM Studioを呼び出すには、ローカルサーバーとして lms server start を実行してバックグラウンドに立ち上げる必要があります。

そのためには、最初に以下のようにCLIをブートストラップし、PATH登録を行い、lmsコマンドを認識できるようにする必要があります。

«事前準備»

1. いったんLM Studioアプリを終了。（初回起動により、%USERPROFILE%/.lmstudio/bin/ 配下に CLI 実行ファイルが配置されます）
2. CLIをブートストラップする。（のコマンドで lms のエイリアスが自動的に PATH に追加されます）

```
cmd /c %USERPROFILE%/.lmstudio/bin/lms.exe bootstrap
```

3. ターミナル(コマンドプロンプト or PowerShell）を再起動して確認

```
lms version
```

バージョンが表示されればOK。

Difyの初期設定：ローカルLLM設定 - LM Studioの場合

- LM Studioのインストールディレクトリへ移動
- サーバーの起動

<コマンドプロンプト or PowerShell>

```
lms server start
```

```
PS C:\Users\若松信康\AppData\Local\Programs\LM Studio> lms server start
Starting server...
Success! Server is now running on port 1234
```

- Difyの設定＞モデルプロバイダーからLM Studioをインストール
- LM Studioの「モデルを追加」から
  - Base URL：http://host.docker.internal:1234

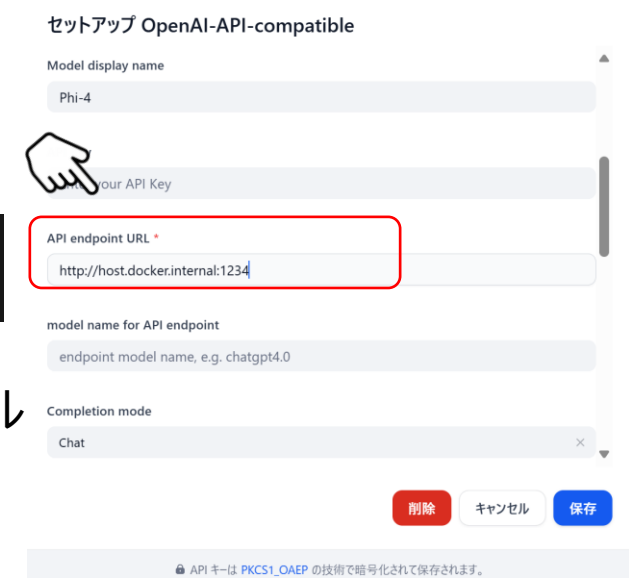
*Docker上でDifyを起動している場合は、host.docker.internal:ポート番号

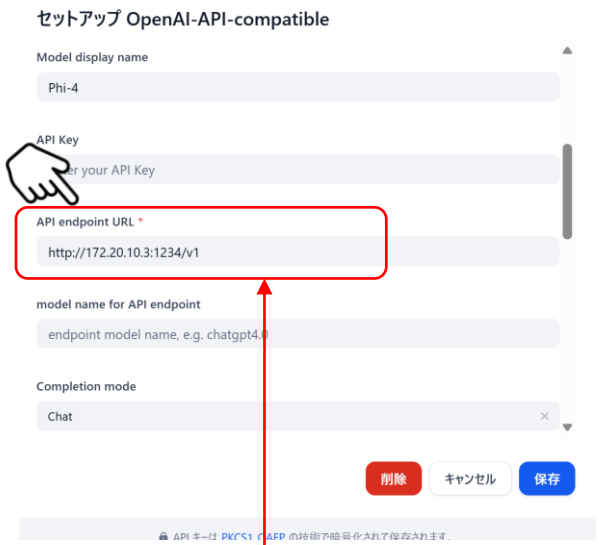
ただし、リソース負荷が高くて、レスポンスが遅い場合、IPアドレスを直接指定したほうが、名前解決やルーティングの負荷が減るため、LM Studio上のReachable atのアドレスを入力してもOK。(ただし、ネットワーク環境が変わると都度アドレスが変わるため、Dify側のアドレス設定も変更必要)

Difyのモデルプロバイダ設定

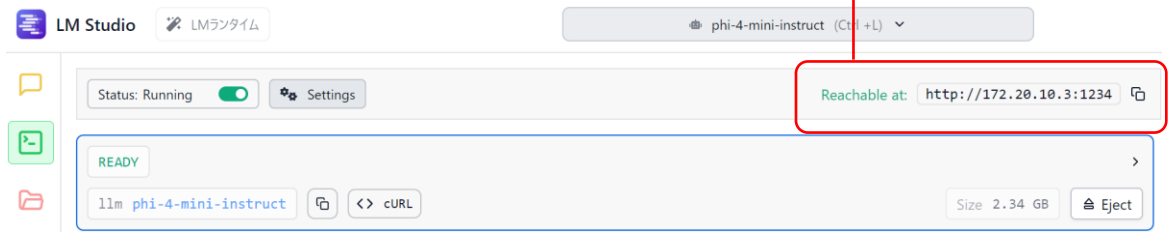
Dify上ではOpenAI-API-compatibleプラグインを使用する

（LM Studioのプラグインは使用しない）





LM Studio



ローカルLLM使用時の留意点:LM Studio/Ollama/LocalAI共通

- デフォルトでは、ローカル推論サーバー側でloadするLLMは1つにする必要がある。
  - ローカル推論サーバーは、1インスタンス ＝ 1ポートのため、複数のLLMをloadしたときに同じポートを共有します。そのため、Difyからそのポートで LLMを呼び出そうとしたときに、どちらを使うか判断できずにタイムアウトしてしまいます。

＜回避策＞

1. LM Studioの場合：別プロセスでポートを分けて起動し、Difyにそれらのポートを設定。
   - CLIで別プロセスでLM Studioを起動

```
npx lmstudio install-cli
```

   - 別々のターミナルで各モデル用にサーバーを別ポートを指定して起動

```
# モデルA 用
lms server start --port 1234
```

```
# モデルB 用
lms server start --port 1235
```

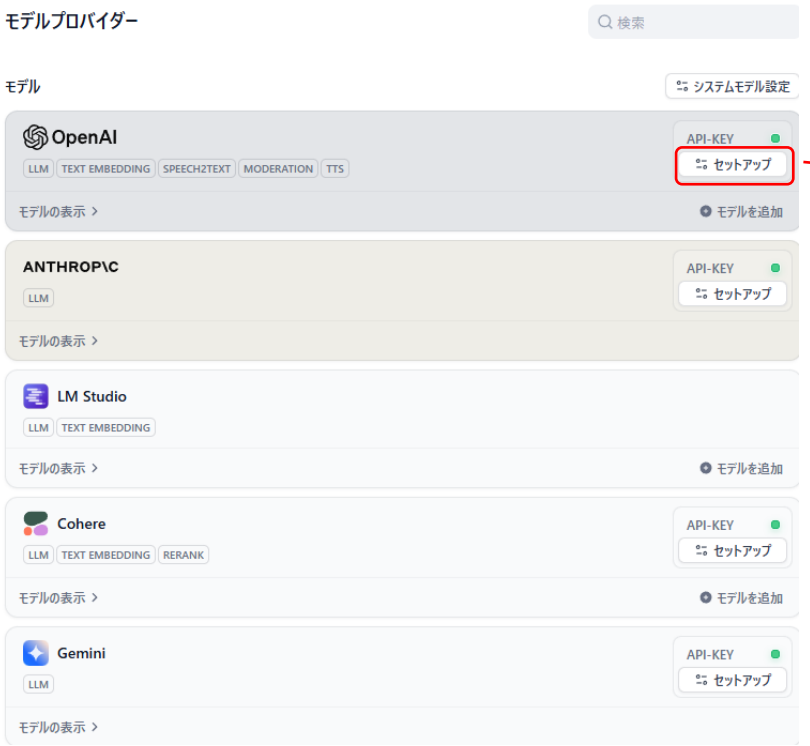
2. Ollama、LocalAI、OpenLLMそれぞれインストールし、LLMを1つずつloadして並列に使用する

外部のLLMモデルプロバイダーのAPIを利用する場合

API Key取得先

- ✓ OpenAI：https://platform.openai.com/docs/overview
- ✓ Google AI Studio：https://aistudio.google.com/u/1/prompts/new_chat
- ✓ Anthropic：https://console.anthropic.com/dashboard
- ✓ Cohere：https://dashboard.cohere.com/api-keys

＜Dify＞モデルプロバイダー設定



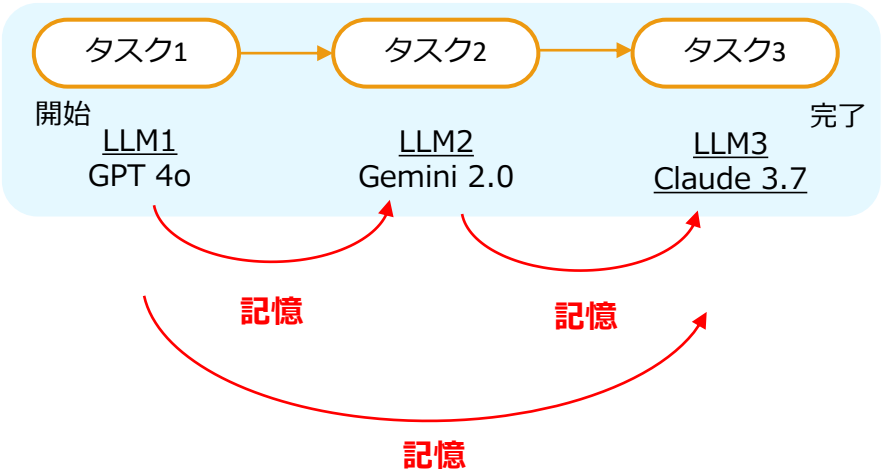
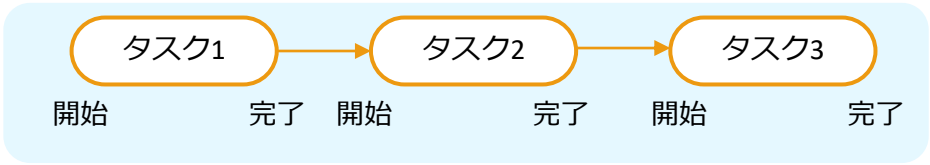
モデルプロバイダーで利用できる機能

API機能	説明	ユースケース	具体例
LLM	テキスト生成、質問応答、文章作成などの自然言語処理タスクを実行	• カスタマーサポートチャットボット • コンテンツ自動生成 • データ分析レポート作成 • プログラミングコード生成 • 多言語翻訳	• Eコマースサイトでの商品に関する質問への自動応答 • マーケティングブログ記事の下書き自動生成 • 売上データから月次レポートの要約文作成 • 簡単な機能のJavaScriptコード生成 • 製品マニュアルの多言語展開
Text Embedding	テキストをベクトル表現に変換し、意味的類似性を数値化	• 類似ドキュメント検索 • レコメンデーションシステム • クラスタリング分析 • セマンティック検索エンジン • 知識ベースのインデックス作成	• 「投資戦略」を検索すると「資産配分」の記事も表示 • 閲覧した記事と意味的に関連する他の記事を推薦 • 顧客フィードバックを自動的にテーマ別に分類 • 「車の故障」で検索すると「エンジントラブル」の記事も表示 • 社内文書を意味ベースで整理・検索可能に
Rerank	検索結果やドキュメントセットを関連性に基づいて並べ替え	• 検索エンジン結果の最適化 • 質問応答システムの精度向上 • レコメンデーションの優先順位付け • ナレッジベース検索の改善 • 情報検索システムの高度化	• 「初心者向けプログラミング」検索で実際に初心者に適した結果を上位表示 • 「パスワードをリセットする方法」の質問に最も直接的な回答を優先 • ユーザーの好みに合った映画を上位に表示 • 「払い戻し方法」検索で最新の正確な手順を最上位に表示 • 法律事務所での判例検索で最も関連性の高い事例を優先表示
Speech to Text	音声をテキストに変換	• 会議の自動文字起こし • 音声コマンドシステム • 電話対応の自動化 • 字幕生成 • 音声メモのテキスト化	• Zoomミーティングの全文を自動的にテキスト化して共有 • 「明日の予定を教えて」と話しかけるとカレンダーを検索 • カスタマーサポート電話の内容を自動記録・分析 • YouTubeビデオの自動字幕生成 • 運転中の音声メモをテキスト化してTodoリストに追加
TTS (Text to Speech)	テキストを自然な音声に変換	• アクセシビリティ機能の提供 • オーディオブック作成 • 音声アシスタント • 教育コンテンツの音声化 • 通知やアラートの音声読み上げ	• 視覚障害者向けのウェブサイト読み上げ機能 • ブログ記事を自動的にポッドキャスト形式に変換 • チャットボットの返答を音声で提供 • 言語学習アプリでの発音例の提供 • 重要なスマートフォン通知を運転中に読み上げ

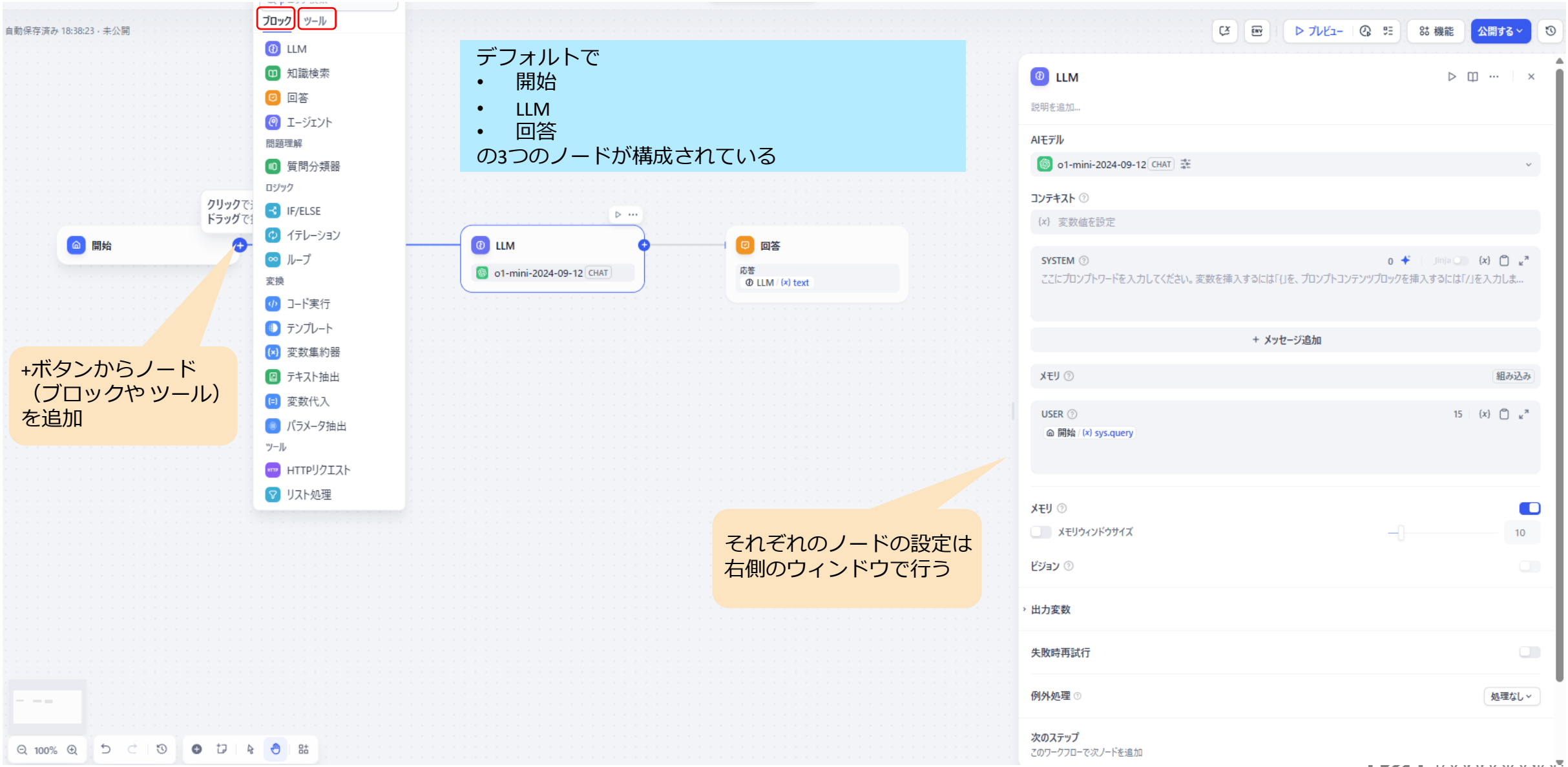
<モデル毎に呼び出せる機能一覧>

https://docs.dify.ai/getting-started/readme/model-providers

業務フローを効率化・自動化する2つのアプローチ

	チャットフロー	ワークフロー
目的・ユースケース	**<対話シナリオをベースとした設計>** カスタマーサービスボット、セマンティック検索アシスタント、Q&Aチャットボットなど、複数ステップにわたってユーザーと対話しながらロジックを進めるシナリオに適しています。	**<自動化やバッチ処理向けに設計>** 高品質な翻訳、データ分析、コンテンツ生成、メール自動化など、大量処理や定期的なバッチ処理をユーザーとの対話なしに実行する用途に最適です。
ブロック（ノード）の相違点	回答ノードが用意され、プロセスの任意のタイミングでテキストをストリーミング出力できるほか、各LLMノードでメモリ設定ができます。	終了ノード（End）がプロセスの最後に配置され、各ノードはメモリを持たず、出力変数としてまとめられた結果を返します。
メモリと状態管理	会話履歴を設定したウィンドウサイズ分だけ保持し、マルチステップ**（異なるLLM間）にわたって文脈を踏まえた応答生成**が可能です。	実行ごとに状態をリセットし、過去の実行結果を参照しないメモリレス構成です。
トリガーと実行方法	ユーザーからのチャット入力を必要とし、画面上で対話を開始します。	入力なしでも起動可能で、API経由やスケジュールトリガーで固定の処理を実行できます。
用途	**対話をベースとし依存関係のある複数タスクからなる一つの業務を完成する** 	**（1）独立した複数のタスクを順番にこなしてワークフローを完成する**  **（2）一つのタスクをバッチ処理する** 

チャットフローの作成方法：設定項目



フローで利用できるノード：ブロック

ノードの説明 | Dify

		ノード（ブロック）	意味
1		開始	フローの開始ノード（必須）。ユーザーの入力内容を変数で定義し、後続ノードで活用できるようにする。
2		LLM	モデルを使って要約・分類・テキスト/コード等を生成する。
3		知識検索	外部データを検索した結果を出力する。
4		回答/終了	＜回答＞（チャットフロー）：フローの中間や最後にテキスト/画像等の生成結果を出力する。 ＜終了＞（ワークフロー）：最終的な結果を出力
5		エージェント	自律的にツールを呼び出す。（ReAct/Function Calling）
6		【問題理解】質問分類器	入力内容を分類して後続ノードに渡し、個別に処理できるようにする。
7		【ロジック】IF/ELSE	条件（IF）に応じて分岐して後続ノードに渡し、個別に処理できるようにする。
8		【ロジック】イテレーション	入力リストに対してノード内の処理を繰り返し実行する
9		【ロジック】ループ	結果に基づいてタスクを反復して実行する
10		【変換】コード実行	PythonまたはNode.jsのコードを直接実行してデータ変換や演算処理を行う
11		【変換】テンプレート	前のステップの出力をテキストに変換する
12		【変換】変数集約器	複数の出力変数を一つの変数に集約する
13		【変換】テキスト抽出	様々なファイルの情報をテキストに変換して後続のLLMノードで解釈できるようにする。
14		【変換】変数代入	書き込み可能な変数に他の変数を代入して後続ノードで活用できるようにする。
15		【変換】パラメータ抽出	自然言語からパラメータを抽出・構造化することで、ツール呼び出しやHTTPリクエストができるようにする。
16		【ツール】HTTPリクエスト	HTTPでサーバーにリクエストを送信し、外部データの取得、ウェブフック、画像生成、ファイルのダウンロードなどを実行する。
17		【ツール】リスト処理	アップロードされたファイルを種別毎に分けて次のノードに渡して個別に処理するために使われる。

チャットフローのシステム変数の意味

変数 | Dify

変数名	データ型	説明	メモ
sys.query	String （文字列）	ユーザーが最初に入力した内容	
sys.files	Array[File] （ファイル類）	ユーザーがアップロードしたファイル	ファイルのアップロード機能は、Difyページ右上の「機能」で有効化する必要があります
sys.dialogue_count	Number （数字）	チャットフロータイプのアプリケーションとの対話中にユーザーが行った対話のラウンド数です。各対話の後に自動的に数が増加し、if-elseノードと組み合わせて複雑な分岐ロジックを構築できます。たとえば、Xラウンド目に達したときに、対話履歴を振り返って分析が可能です	
sys.conversation_id	String （文字列）	ダイアログの対話セッションの一意の識別子で、関連するすべてのメッセージを同じ対話にグループ化し、LLMが同じトピックとコンテキストで継続的に対話できるようにします	
sys.user_id	String （文字列）	各アプリケーションユーザーに割り当てられた一意の識別子で、異なる対話ユーザーを区別するために使用されます	
sys.app_id	String （文字列）	アプリケーションのIDで、システムは各ワークフローアプリケーションに一意の識別子を割り当て、異なるアプリケーションを識別します。このパラメータを使用して現在のアプリケーションの基本情報を記録します	開発者向けで、このパラメータを使用して異なるワークフローアプリケーションを区別します
sys.workflow_id	String （文字列）	ワークフローIDで、現在のワークフローアプリケーションに含まれるすべてのノード情報を記録するために使用されます	開発者向けで、このパラメータを使用してワークフロー内のノード情報を追跡および記録できます
sys.workflow_run_id	String （文字列）	ワークフローアプリケーションの実行IDで、アプリケーションの実行状況を記録するために使用されます	開発者向けで、このパラメータを使用してアプリケーションの過去の実行状況を追跡できます

これらの変数を後続のノードで指定してその内容を活用できる
（変数はノードで追加や変換・集約が可能）

ワークフローのシステム変数の意味

変数 | Dify

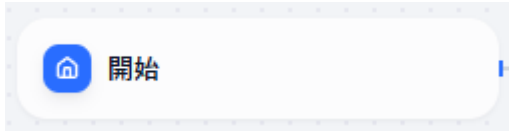
変数名	データ型	説明	メモ
sys.files `sys.files`	Array[File] （ファイル類）	ファイルパラメータで、ユーザーがアプリを初めて使用する際にアップロードした画像を保存します。	画像のアップロード機能は、アプリケーションの編成ページ右上の「機能」から開始する必要があります。
sys.user_id `sys.user_id`	String （文字列）	ユーザーIDです。ワークフローアプリを使用する際、システムが自動的にユーザーに一意の識別子を割り当て、異なるユーザーを区別するために使用します。	
sys.app_id `sys.app_id`	String （文字列）	アプリIDで、システムが各ワークフローアプリに一意の識別子を割り当て異なるアプリを区別します。このパラメータは現在のアプリの基本情報を記録するために使用されます。	開発能力を持つユーザー向けで、このパラメータを使用して異なるワークフローアプリを区別し、特定できます。
sys.workflow_id `sys.workflow_id`	String （文字列）	ワークフローIDで、現在のワークフローアプリに含まれるすべてのノード情報を記録するために使用されます。	開発能力を持つユーザー向けで、このパラメータを使用してワークフロー内のノード情報を追跡および記録できます。
sys.workflow_run_id `sys.workflow_run_id`	String （文字列）	ワークフローアプリ実行IDで、ワークフローアプリ内の実行状況を記録するために使用されます。	開発能力を持つユーザー向けで、このパラメータを使用してアプリの過去の実行状況を追跡できます。

ノードの設定方法

チャットフロー/ワークフロー共通

①「開始」ノードの設定方法

開始 | Dify



開始：フローの開始ノード（必須）。ユーザーの入力内容を変数で定義し、後続ノードで活用できるようにする。

＜変数設定の意味＞
変数を設定し、その情報が記録されることで、その後のノードでその変数を指定して、変数の内容に応じた処理・回答ができるようになる。

設定項目

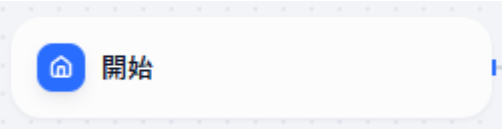
- **フィールドタイプ**：ユーザーが入力できる形式を設定する
  - **短文**：256文字以内の文章
  - **段落**：257文字以上の長文
  - **選択**：ドロップダウンによる選択
  - **数値**：数字のみ入力可
  - **単一ファイル**：ファイルアップロード
  - **ファイルリスト**：複数ファイルをアップロード
- **変数**：システムに識別させるための任意の文字列を設定する
  例）dev
- **ラベル名**：ユーザーに表示される名称
  例）部署名

＜変数設定例＞
- **設定＞変数名：dev、ラベル名：部署名**
- **ユーザー入力>部署名：営業部**

→営業部からの問い合わせに対応するためのノード（営業情報を参照させたLLM等）に接続して個別に回答

①「開始」ノードの設定方法

開始 | Dify



（例）ユーザーに、「質問」「部署名」「名前」の必須入力とファイルの「添付」を許可したい

「質問」の変数設定

入力フィールドを編集

フィールドタイプ：短文 / **段落** / 選択 / 数値 / 単一ファイル / ファイルリスト

変数名：Q

ラベル名：質問

最大長：400

必須

キャンセル　保存

- フィールドタイプ：段落
- 変数名：Q
- ラベル名：質問
- 最大長：400（質問は400文字まで入力できるように設定）
- 入力必須

「部署名」の変数設定

入力フィールドを編集

フィールドタイプ：**短文** / 段落 / 選択 / 数値 / 単一ファイル / ファイルリスト

変数名：dev

ラベル名：部署名

最大長：48

必須

キャンセル　保存

- フィールドタイプ：短文
- 変数名：dev
- ラベル名：部署名
- 最大長：48（デフォルト）
- 入力必須

「名前」の変数設定

入力フィールドを編集

フィールドタイプ：**短文** / 段落 / 選択 / 数値 / 単一ファイル / ファイルリスト

変数名：name

ラベル名：名前

最大長：48

必須

キャンセル　保存

- フィールドタイプ：短文
- 変数名：name
- ラベル名：名前
- 最大長：48（デフォルト）
- 入力必須

「添付」の変数設定

入力フィールドを編集

フィールドタイプ：短文 / 段落 / 選択 / 数値 / **単一ファイル** / ファイルリスト

変数名：attach

ラベル名：添付

サポートされたファイルタイプ

- ドキュメント：TXT, MD, MDX, MARKDOWN, PDF, HTML, XLSX, XLS, DOC, DOCX, CSV, EML, MSG, PPTX, PPT, XML, EPUB
- 画像：JPG, JPEG, PNG, GIF WEBP, SVG
- 音声：MP3, M4A, WAV, AMR, MPGA
- 映像：MP4, MOV, MPEG, WEBM
- 他のファイルタイプ：他のファイルタイプを指定する。

添付できるファイル種類として、ドキュメント、画像を設定

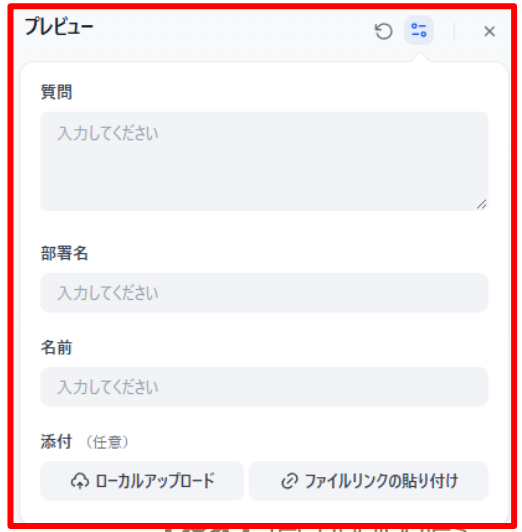
アップロードされたファイルのタイプ：ローカル アップロード / URL / 両方

必須

キャンセル　保存

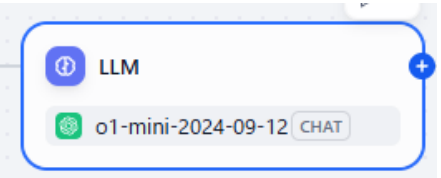
開始ノード

ユーザー入力画面



②「LLM」ノードの設定方法：全般

LLM - Dify Docs



LLM：モデルを使って要約・分類・テキスト/コード等を生成する。

主な利用方法

- **意図識別**：カスタマーサービスの対話シナリオにおいて、ユーザーの質問を意図識別および分類し、異なるフローに誘導する
- **テキスト生成**：記事生成シナリオにおいて、テーマやキーワードに基づいて適切なテキスト内容を生成するノードとして機能する。
- **内容分類**：メールのバッチ処理シナリオにおいて、メールの種類を自動的に分類する（例：問い合わせ/苦情/スパム）。
- **テキスト変換**：テキスト翻訳シナリオにおいて、ユーザーが提供したテキスト内容を指定された言語に翻訳する。
- **コード生成**：プログラミング支援シナリオにおいて、ユーザーの要求に基づいて指定のビジネスコードやテストケースを生成する。
- **RAG**：ナレッジベースの質問応答シナリオにおいて、検索した関連知識とユーザーの質問を再構成して回答する。
- **画像理解**：ビジョン機能を持つマルチモーダルモデルを使用し、画像内の情報を理解して質問に回答する。
- **ファイル分析**：ファイルを処理する場合、LLMを活用して、ファイルに含まれている情報を認識し、それを分析する。

設定項目

基本設定

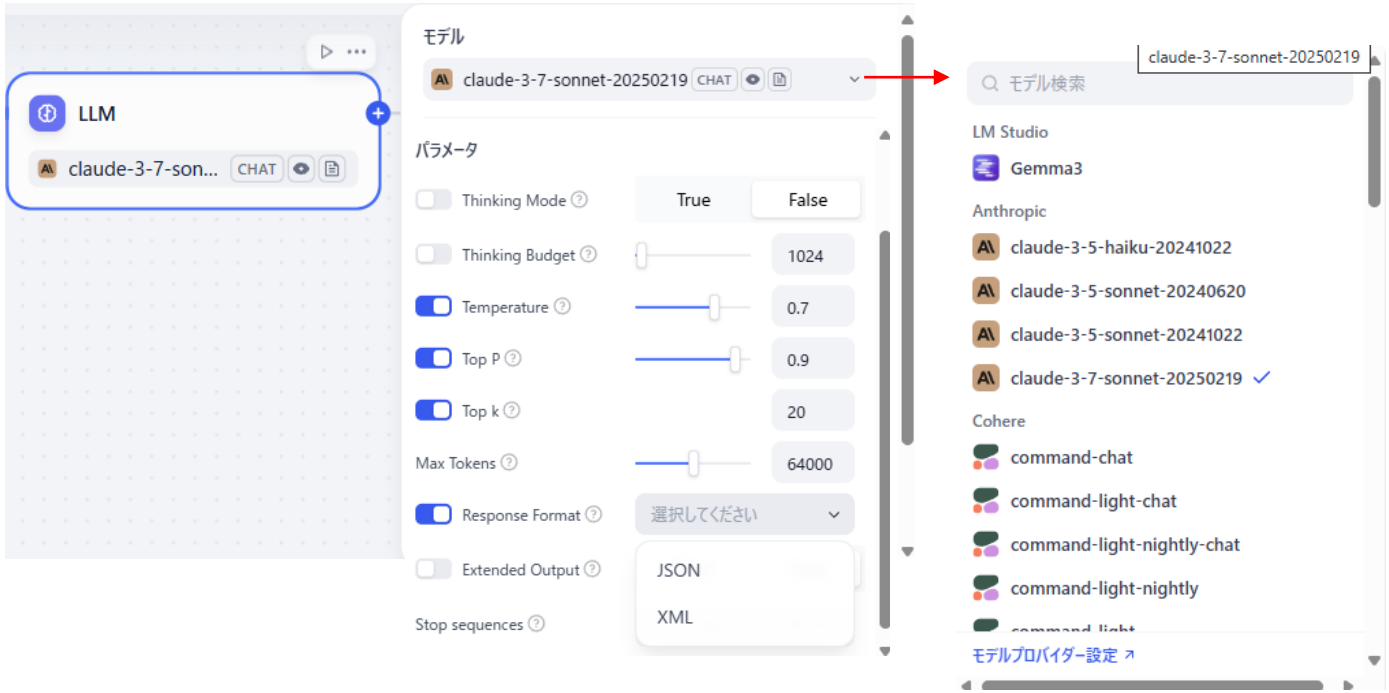
- **AIモデル**：LLMを選択（システム設定> モデルプロバイダでモデルの設定を事前に行う必要あり）
- **コンテキスト**：（オプション）LLMに提供する背景情報
- **SYSTEM**：システムプロンプト
- **＋メッセージ追加**：ユーザープロンプト、アシスタントプロンプトを追加指定

詳細設定

- **メモリ**：ONにすると会話の履歴を保持
- **ビジョン**：マルチモーダル対応（画像処理能力）
- **出力変数**：後続のノードで参照するためのLLMから出力される変数形式の設定（デフォルトでテキストが設定）
- **失敗時再試行**：LLMノードの処理が失敗したとき（ネットワークエラー等）に自動的に再試行する回数を設定
- **例外処理**：エラーが発生したときの代替動作を指定（メッセージ出力、代替分岐）

②「LLM」ノードの設定方法：モデル

LLM - Dify Docs



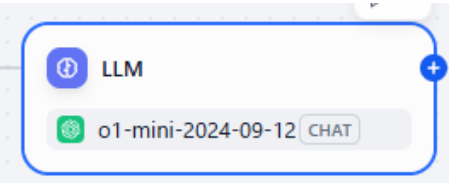
まずはデフォルト設定で進めて、
出力結果に応じて調整しよう

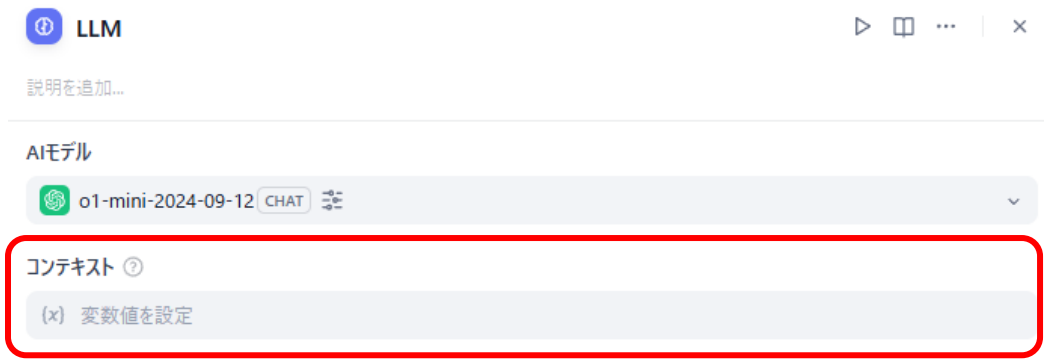
パラメータ設定
（モデルによって設定項目が異なる）

- **Thinking Mode：**推論モード（推論モデルのみ選択可能）
- **Thinking Budget：**内部推論プロセスに割り当てるトークンの上限設定（推論モデルのみ選択可能）
- **Temperature：**次の単語の確率の高さを指定。0に近いほど高確率な単語を、1に近いほど低確率な単語からランダムに出力
- **Top P：**次の単語候補の累積確率がp%を越えるものの中から選択。小さいほど少ない候補の中から確定的に出力、大きいほど多く候補から出力し、多様性・創造性が増加
- **Top K：**確率に関わらず候補上位K個のトークンのみ候補とする。小さいほど確定的、大きいほど多様性増加。
- **Presence Penalty**：モデルが出力候補を選ぶ際に、同じトークン（語彙）ではなく新規語彙を出力するように誘導するためのペナルティ。値を大きくすると出力の多様性が増す。
- **Frequency Penalty**：生成済トークンの出現回数に応じたペナルティ。単語やフレーズの過度な繰り返しを防ぐ。
- **Max Tokens：**出力までのプロセスに使えるトークンの上限を指定。（内部推論プロセスで使うトークンも含まれるため、Max Tokens > Thinking Budgetで設定する必要あり）
- **Response Format：JSON or XML**
- **Extended Output：**「TRUE」にすると、出力トークンの上限を拡張して最大128kトークンまで拡張する
- **Stop Sequences：**モデルに出力させたくない文字列、またはそこで出力を一旦区切りたい場合に、最大4つまで文字列指定し、その出力を含んだタイミングで以降の生成を停止させる
- **JSON Schema**：出力されるJSONのスキーマ（型、プロパティ等）を指定して、関数呼び出しやDB連携の安定性を向上させる

②「LLM」ノードの設定方法：コンテキスト

LLM - Dify Docs

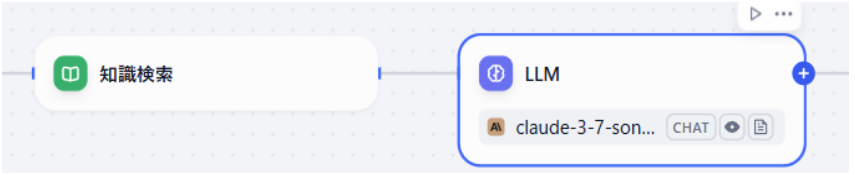




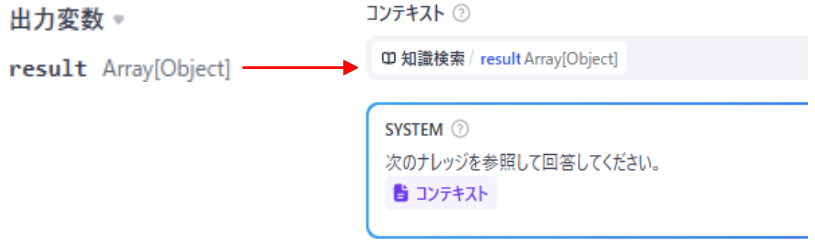
「コンテキスト」とは、
上流ノードで取得したテキスト情報（知識検索結果、ドキュメント抽出結果、会話履歴など）をプロンプトに埋め込むための設定項目です。具体的には、「知識検索」ノードの出力変数(result)や「テキスト抽出」ノードの出力変数(text)、あるいは会話履歴変数を「コンテキスト変数」としてLLMノードにマッピングし、プロンプト内で{{変数名}}の形式で参照します。

（例）RAG：「知識検索」（外部参照）結果を前提知識としてLLMに入力

ノード構成



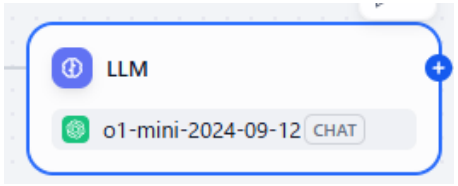
変数設定
コンテキスト



システム
プロンプト

②「LLM」ノードの設定方法：システムプロンプト等

LLM - Dify Docs



プロンプト設定

● **SYSTEM プロンプト**：
•AIモデルの「ペルソナ」や「役割」。「応答スタイル」、「禁止事項」などの基本ルールを定義します。
•ユーザーには表示されない裏側の設定で、モデルの振る舞いや回答の様式など基本的なガイドラインを設定します。
•例：「あなたは親切な日本語アシスタントです。簡潔に応答してください」

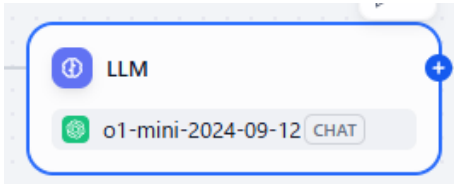
● **USER プロンプト**：
•ユーザー入力をテンプレート化した様式でモデルに渡すための補助文。
•変数や入力フィールドからの情報を含めて指示。
•例：
□ 実際のユーザー入力{{query}}例：「最新の有給休暇取得ポリシー」
□ USERプロンプト例：『{{query}}』について教えてください」
□ モデルに渡されるプロンプト：「最新の有給休暇取得ポリシーについて教えてください」

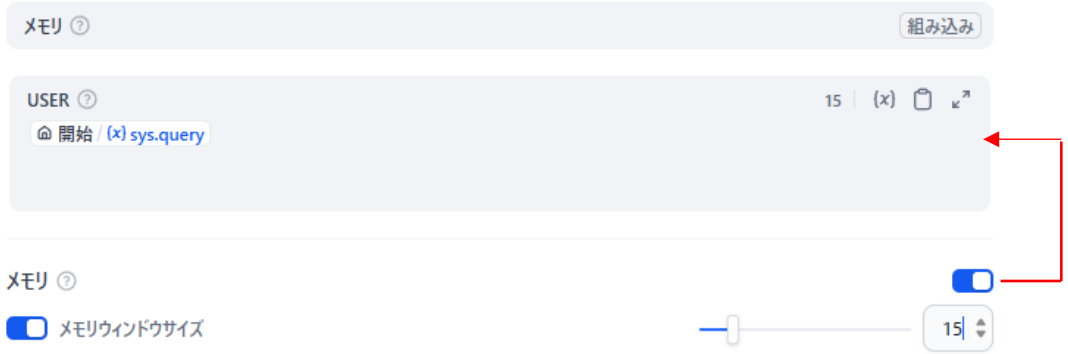
● **ASSISTANT プロンプト**：
•モデルの応答の形式や構造をテンプレート化
•応答の始め方や終わり方、含めるべき情報の構造を指定できます。
•レスポンスの一貫性を保ちつつ、特定のフォーマットに従わせることができます

指定しない場合は、モデルからの回答をそのまま出力

②「LLM」ノードの設定方法：メモリ設定

LLM - Dify Docs





メモリウィンドウサイズ：過去何回分の会話を記憶するか

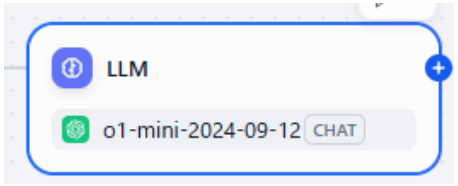
＊多くするとより多くの文脈を理解できるが、トークン消費量が増えて応答が遅くなる

メモリ設定

- メモリ機能をONにすると「USER」設定項目が表示される
- デフォルト設定の「sys.query」は、Difyのシステム変数の一つで、ユーザーが入力した最新のクエリ（質問や指示）の内容を参照するための変数です。この変数には以下のような意味と役割があります：
  - ユーザー入力の取得：ユーザーが送信した最新のメッセージを自動的に取得し、LLMへの入力として使用します。
  - 動的な対応：ユーザーの質問内容に応じて動的にプロンプトを構成できます。
  - メモリとの連携：メモリ機能がONの場合、この変数を使うことで、会話の文脈を維持しながら新しい質問に対応できます。
  - 例えば、USER欄に「{{sys.query}}について詳しく教えてください」というプロンプトがあると、ユーザーが「AIの歴史」と質問した場合、LLMには「AIの歴史について詳しく教えてください」というプロンプトが送られます。

②「LLM」ノードの設定方法：その他

LLM - Dify Docs



失敗時再試行設定

✓ エラー時の最大試行回数と再試行間隔を設定

例外処理設定

✓ エラー発生時のユーザー向けメッセージ設定

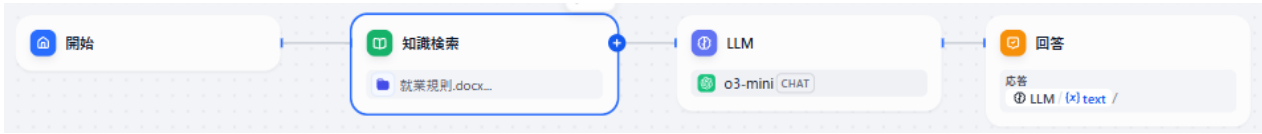
✓ エラー発生時に代替ノードへ処理を移す設定

③「知識検索」ノードの設定方法：

知識検索 - Dify Docs

知識検索：外部データを検索した結果を出力する。

■ 使用例：LLMノードに接続し、ユーザー入力+検索結果を踏まえて回答させる



ユーザーが入力した情報を元に2つのナレッジベース（就業規則.txtと外部Webサイト）を参照して出力する設定例

設定項目

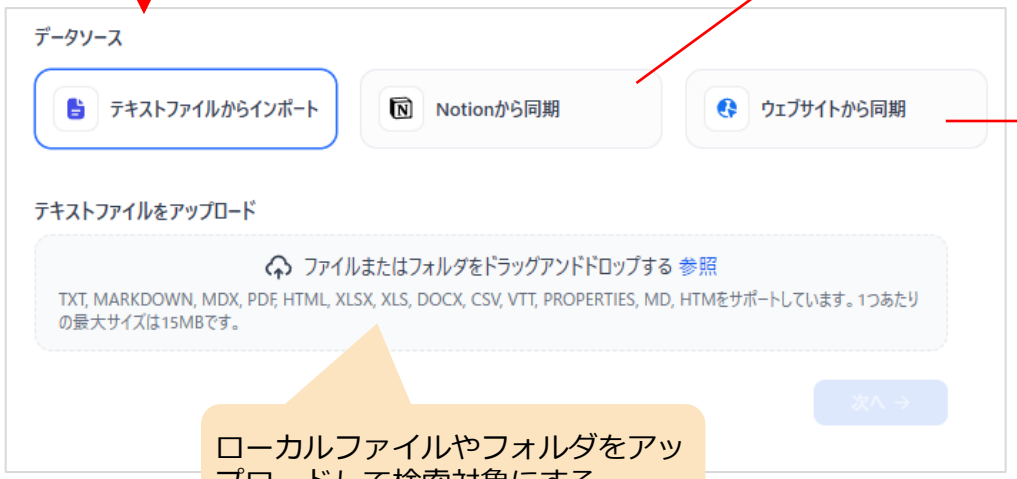
- **検索変数**：参照する元になる内容。ここでは変数「sys.query」（= ユーザーが最初に入力した内容）を元にナレッジベースで検索する。ナレッジベースで参照できる元のクエリは200文字以下。
- **ナレッジベース**：検索先
  - **事前設定必要。**事前にトップメニュー「ナレッジ」から作成して選択できるようにする
- **メタデータフィルタ**：ユーザーに関連性の高いパーソナライズされた情報を提供する、もしくは制御するためのもの
  - 自動生成：ユーザーの履歴に基づいて最適な情報を提示
  - 手動設定：特定条件に基づいて情報へのアクセスを制御する
- **出力変数**：外部検索してマッチした内容の変数。後続するノード（LLMが一般的）に変数として渡してプロンプトとして使用することができます

③「知識検索」ノードの設定方法：

ナレッジベース作成 - Dify Docs

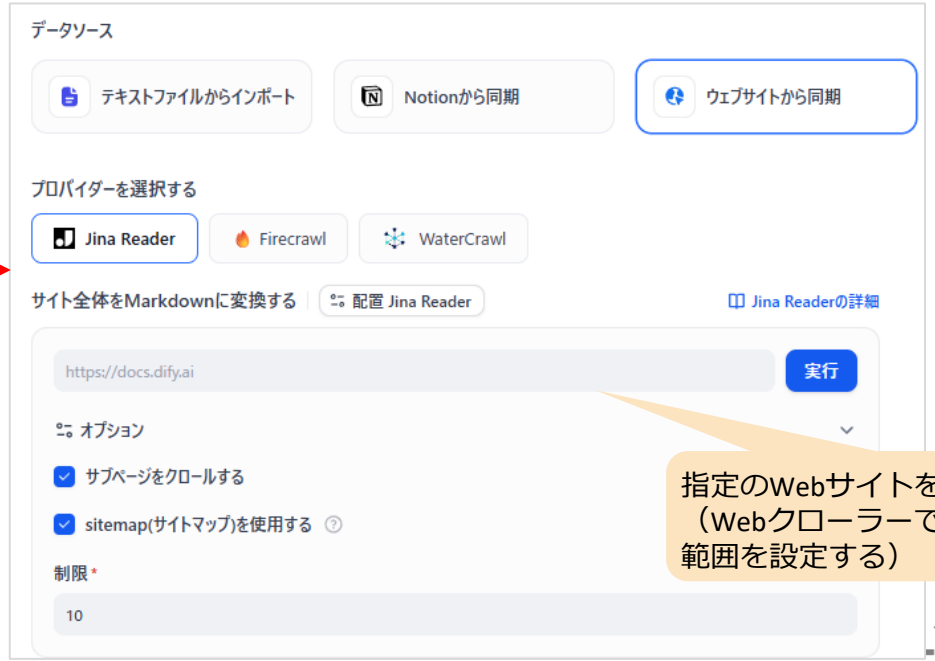
「知識検索」ノードに加えるための事前設定：ナレッジベース作成

トップメニュー「ナレッジ」から「ナレッジベースを作成」



ローカルファイルやフォルダをアップロードして検索対象にする

Notionと接続して検索対象にする



指定のWebサイトを検索対象にする（Webクローラーで事前に取得する範囲を設定する）

③「知識検索」ノードの設定方法：

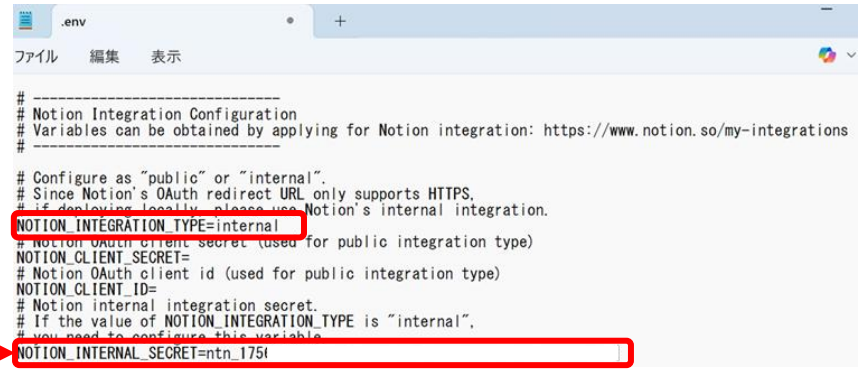
「知識検索」ノードに加えるための事前設定：ナレッジベース作成

Notion

Notionと接続するための設定
（ローカル環境の場合）

Notion上で内部インテグレーションを設定 → **Notion上で内部インテグレーションシークレットを取得** → **Dify/Dockerフォルダ内の.envファイルに設定**

https://www.notion.so/profile/integrations



Dify/Dockerフォルダ内の.envファイルに以下を追加
NOTION_INTEGRATION TYPE = **Internal**
NOTION_INTERNAL_SECRET = **Notionの内部インテグレーションシークレット**

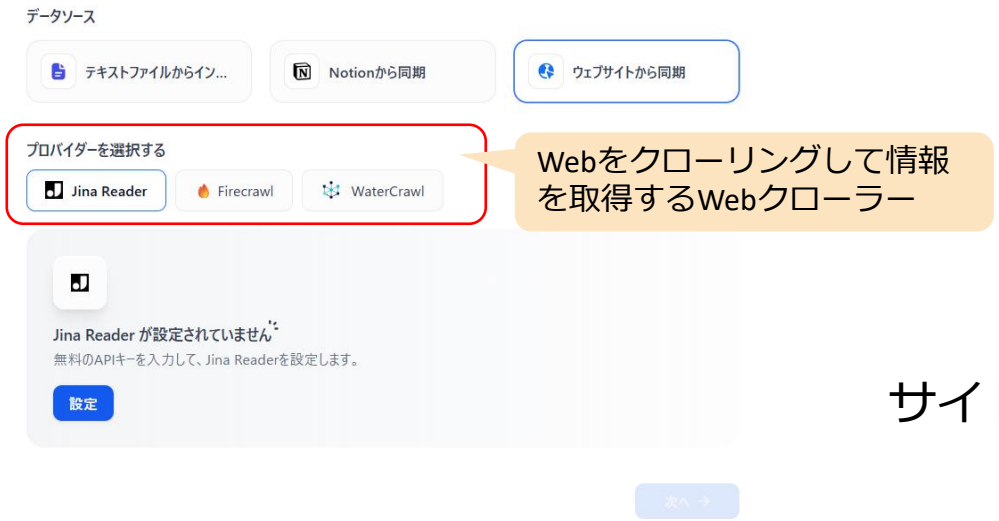
*公開環境の場合には、Notion上で統合の種類をPublicとし、「Client ID」と「Client Secret」を取得して、Difyの.envファイルに設定する
（NOTION_INTEGRATION_TYPE =Public）

③「知識検索」ノードの設定方法：

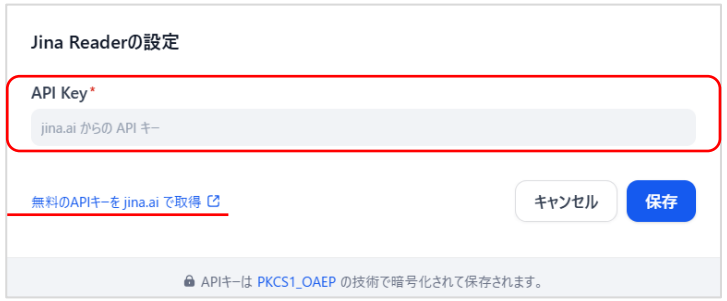
1.2 Webサイトからデータをインポート - Dify Docs

「知識検索」ノードに加えるための事前設定：ナレッジベース作成

ウェブサイトの情報を検索対象とするための設定



サイトからAPIを取得して入力



https://jina.ai/reader/

https://www.firecrawl.dev/app/api-keys

https://app.watercrawl.dev/dashboard/api-keys

③「知識検索」ノードの設定方法：

「知識検索」ノードに加えるための事前設定：ナレッジベース作成

Webクローラーの比較

ツール名	特徴	ユースケース	コスト	拡張性・制御性
Jina Reader	- URL→Markdown変換 - CSSセレクタ／ブラウザエンジン指定 - 画像キャプション生成、Shadow DOM抽出 - ReaderLM-v2による実験的HTML→Markdown/JSON変換	- 単一ページを速やかにMarkdown化 - 特定要素のみ抽出し細かい前処理が必要 - 認証クッキー／プロキシ経由取得	クラウド版：20リクエスト/分/IPまで無料 OSS版：無料（環境構築コストのみ）	高（パラメータ設定多数）
FireCrawl	- サブページ含むサイト全体クロール - クロール深度・ページ上限・除外/含むパス設定 - OSS版で無制限、自前サーバ運用可 - JSブロックやプロキシ対応	- 企業サイトやドキュメントサイト一括取り込み - RAG基盤の初期構築 - 内部ネットワーク含む大量クロール	クラウド版：500ページまで無料 OSS版：無料（環境構築コストのみ）	中（基本設定UIのみ）
WaterCrawl	- JavaScriptレンダリング対応 - PDF化／スクリーンショット生成 - プラグインシステムで独自AI処理パイプライン - 構造化JSON出力、リアルタイムステータス追跡	- SPA/Ajax多用サイトなど動的コンテンツ - フィールド単位での構造化データ抽出 - カスタムプラグイン開発が必要な場合	クラウド版：1000ページ/月 まで無料 OSS版：無料（環境構築コストのみ）	非常に高（プラグイン開発可）

＜参考＞選択する際の主な判断材料

対象サイトの性質
•静的HTML中心：Jina Reader or FireCrawl
•動的・JavaScript多用：WaterCrawl or Jina Reader（Browser Engineオプション）

出力フォーマット
•単純Markdown：Jina Reader or FireCrawl
•構造化JSON／カスタムデータ：WaterCrawl

運用コスト・スケール
•少量頻度：Jina Reader（無料枠活用）
•大量クロール：FireCrawl OSS or WaterCrawl（自社インフラ）

拡張性・制御性
•簡易：FireCrawl
•細かい調整：Jina Reader
•プラグイン開発：WaterCrawl

コストとレート制限
•単発利用：Jina Reader（無料）
•大規模・商用：FireCrawl OSS／WaterCrawl（セルフホスティングプラン）

③「知識検索」ノードの設定方法：

「知識検索」ノードに加えるための事前設定：ナレッジベース作成

ナレッジベースとして追加され使用できる状態になると＜設定＞「データソース」にステータスが反映され、「知識検索」ノードで選択利用可能となります。

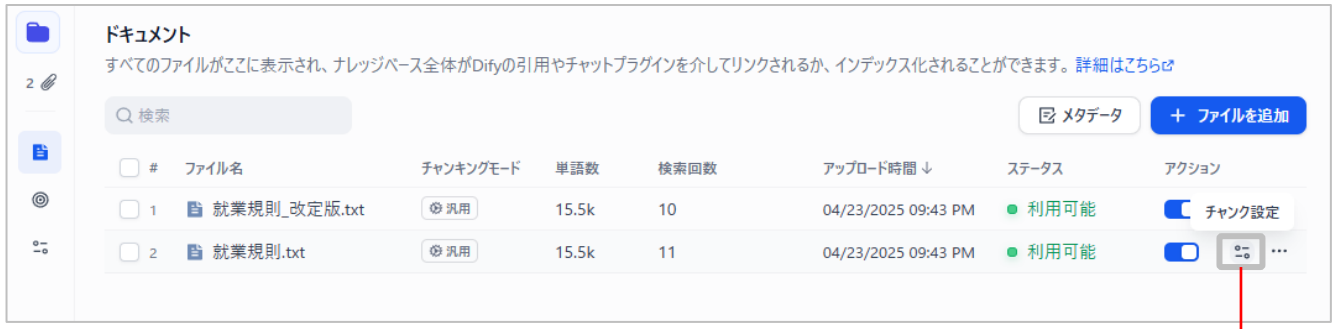
③「知識検索」ノードの設定方法：

ナレッジベースの応用設定：RAG「チャンク設定」

チャンク：意味を持ったテキストの塊。全文検索せずに効率的にマッチする検索結果を見つけるために元のテキストを分割したもの。

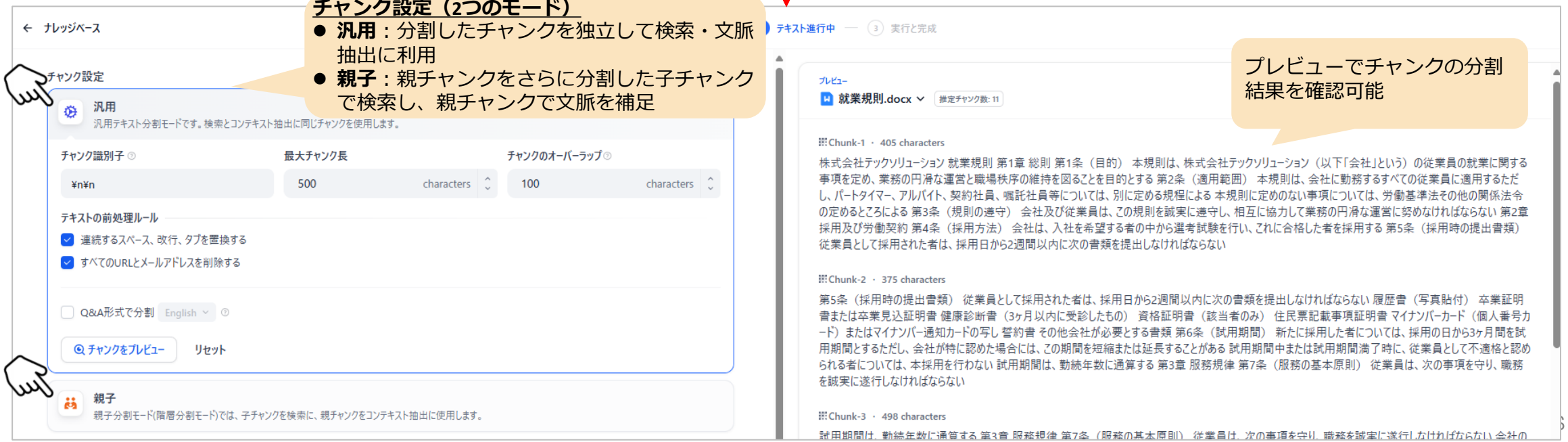
最初に参照ファイルをアップロードしたとき

あとから修正する場合



チャンク設定（2つのモード）

- **汎用**：分割したチャンクを独立して検索・文脈抽出に利用
- **親子**：親チャンクをさらに分割した子チャンクで検索し、親チャンクで文脈を補足



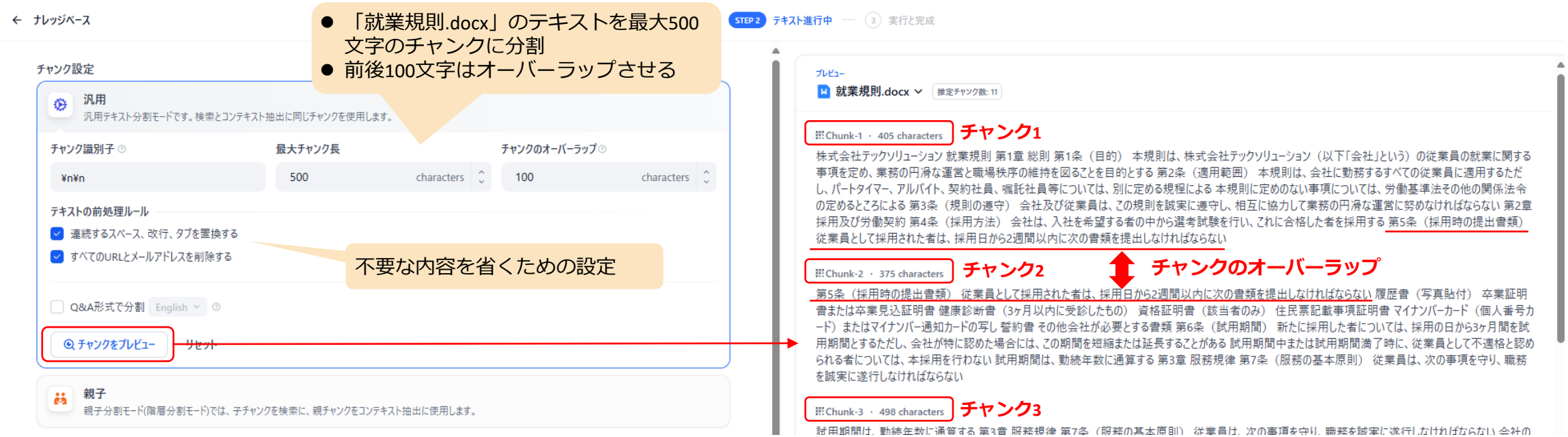
プレビューでチャンクの分割結果を確認可能

③「知識検索」ノードの設定方法：

[2. チャンクモードの指定 - Dify Docs]

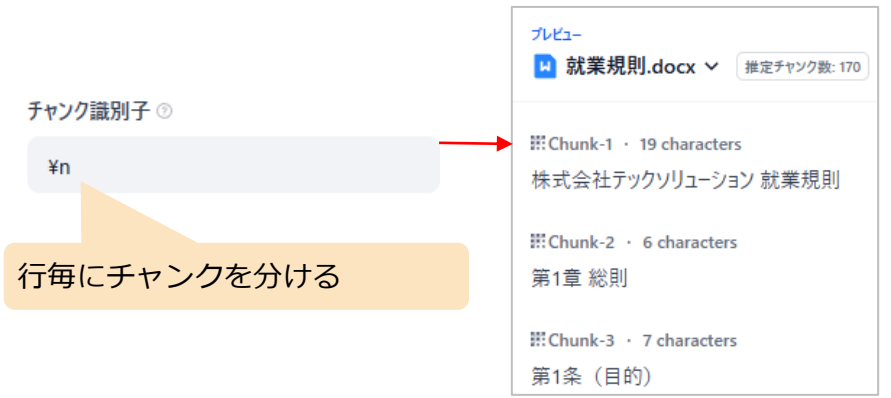
ナレッジベースの応用設定：RAG「チャンク設定」

<汎用モードの場合>



チャンク識別子：
- ¥n¥n（二重改行）：テキスト内の空行（段落区切り）を抽出してチャンクを生成（デフォルト）
- ¥n（改行）：各行をチャンクとして分割
- “文字列”：指定の文字列ごとに分割

最大チャンク長：チャンクの最大文字数（設定できる最大は4000）
チャンクのオーバーラップ：チャンク間で重複して保持するテキストの文字数。テキストの意味のまとまりをチャンク内で保持するために同じ文章をチャンク間で重複して保持させる。

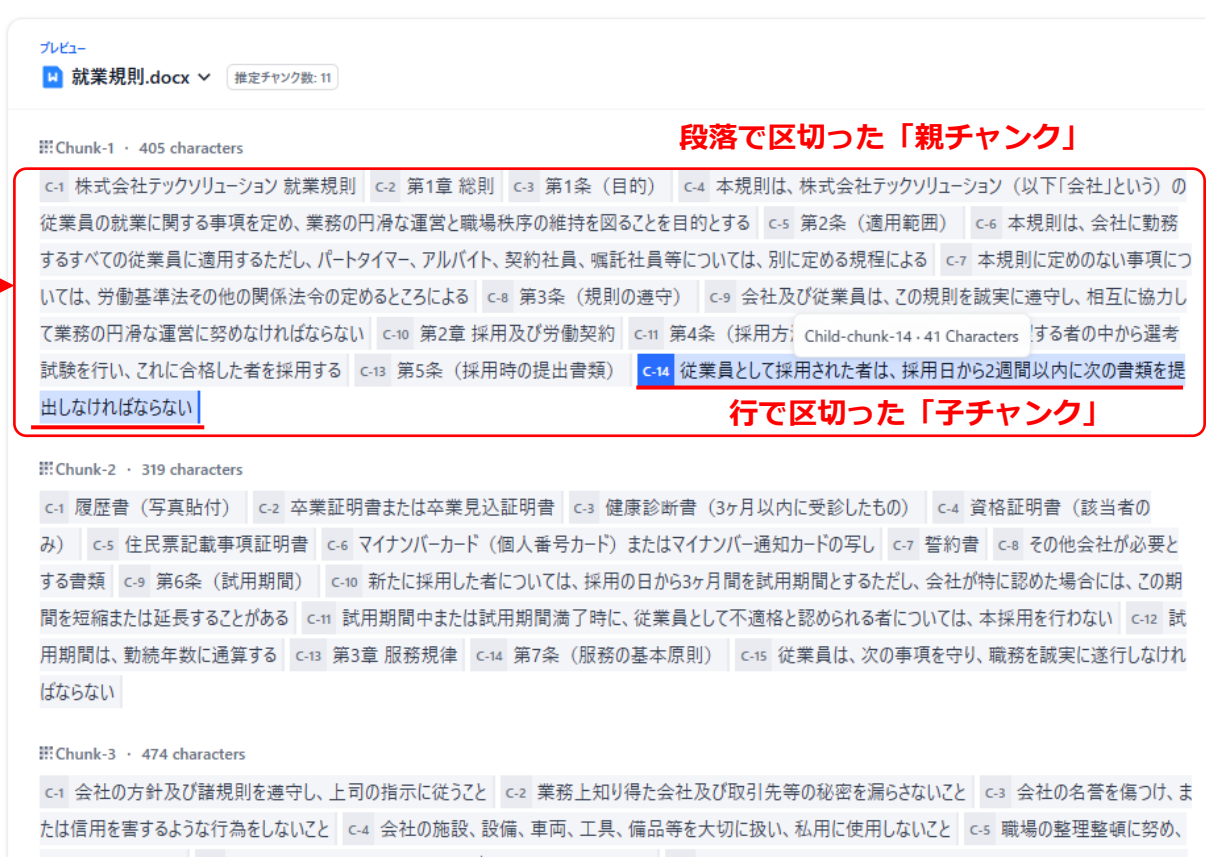


③「知識検索」ノードの設定方法：

[2. チャンクモードの指定 - Dify Docs]

ナレッジベースの応用設定：RAG「チャンク設定」

<親子モードの場合>



親子設定（2つのモード）

- **段落**：識別子（段落等）で親チャンクを区切る設定→親チャンクが過剰にならず処理コストを抑えられる。FAQやマニュアル等段落で論理的に区切られたテキストに最適。
- **全文**：親チャンクをドキュメント全文にする設定→全体を通して関連性を把握したい短文資料に最適。（10,000トークンを越えると末尾が切り捨てられる）

③「知識検索」ノードの設定方法：

[3. インデックス方法と検索設定を指定 - Dify Docs]

ナレッジベースの応用設定：RAG「インデックス方法設定」

インデックス方法設定（2つのモード）

高品質：
- 分割されたテキストチャンクをEmbeddingモデル（例：text-embedding-3-largeなど）で数値ベクトルに変換し、大量のテキスト情報を効率的に圧縮・保存することで、ユーザーの質問とマッチングする精度が向上します。
- 「ベクトル検索」「全文検索」「ハイブリッド検索」の3つのオプションが用意されており、意図やドキュメント特性に応じて最適な手法を選択できます。

経済的：
- 各テキストチャンク内から最大10個のキーワードを抽出し、「逆引きインデックス方式」のみでマッチングを行います。これにより検索精度はやや低下しますが、トークン消費や外部API呼び出しが不要でランニングコストを抑えられます。
- 「転置インデックス」（＝「逆引きインデックス」）でTop-Kのみ設定可能。（Top-Kの値が大きいほど呼び出される候補文の数が多くなります）

③「知識検索」ノードの設定方法：

[3. インデックス方法と検索設定を指定 - Dify Docs]

ナレッジベースの応用設定：RAG「検索設定」

検索設定（3つのモード）

- **ベクトル検索**：ユーザーが入力した質問をベクトル化し、クエリテキストのベクトルを生成し、クエリベクトルとナレッジベース内の対応するテキストベクトル間の距離を比較し、隣接する分割コンテンツを探します。
- **全文検索**：文書内のすべての語彙をインデックス化し、ユーザーが質問を入力した際に、キーワード検索でテキストマッチングしてテキストを抽出します。
- **ハイブリッド検索**：全文検索とベクトル検索、またはRerankモデルを同時に実行し、クエリ結果からユーザーの質問に最もマッチする最良の結果を選択します。

設定項目

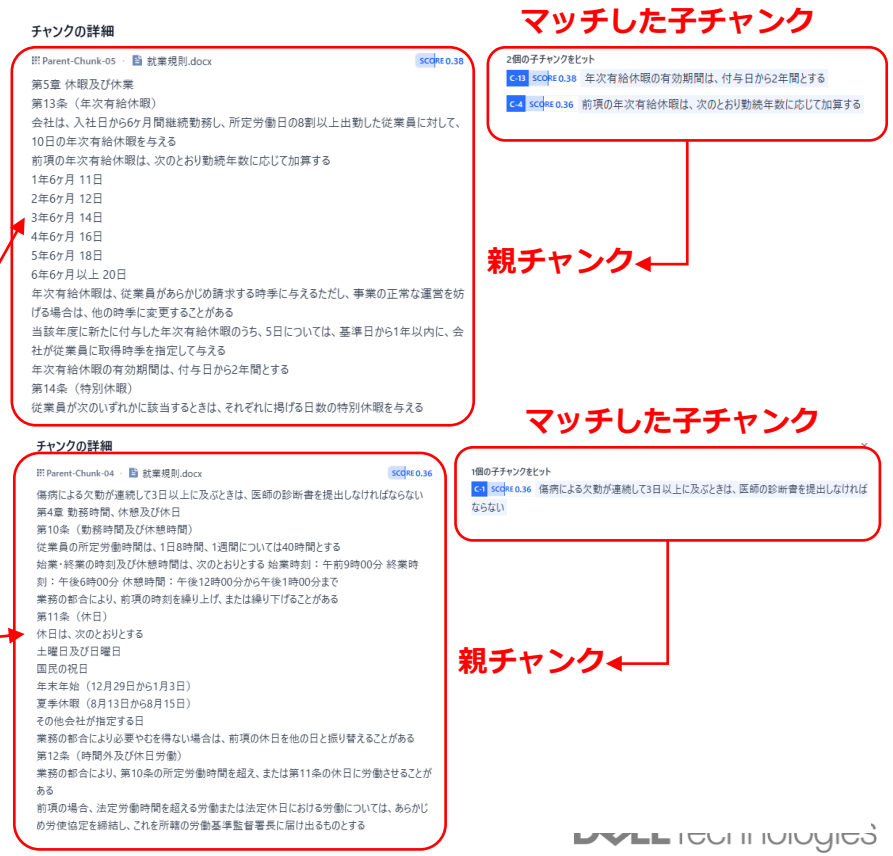
<共通>
- **Rerankモデル**：ベクトル検索で取得した候補チャンクの順位を外部モデルを使用して再評価する（ここではCohereのモデルrerank-v3.5を使用）ことで回答精度を向上させることが可能
- **Top-K**：値が大きいほど呼び出される候補文の数が多くなります。
- **スコア閾値**：抽出するテキストの類似度の閾値。類似度の値が大きいほど候補テキストは少なくなります。

<ハイブリッド検索>
- **ウエイト設定**：セマンティック（意味）検索とキーワード検索のどちらを優先するかの重み付け設定

DELL Technologies

③「知識検索」ノードの設定方法：

ナレッジベース：検索結果のテスト



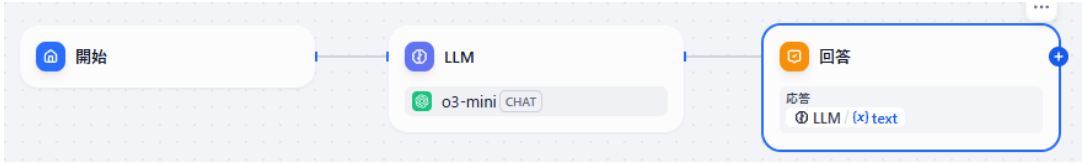
マッチした子チャンクから
親チャンクの文脈を抽出

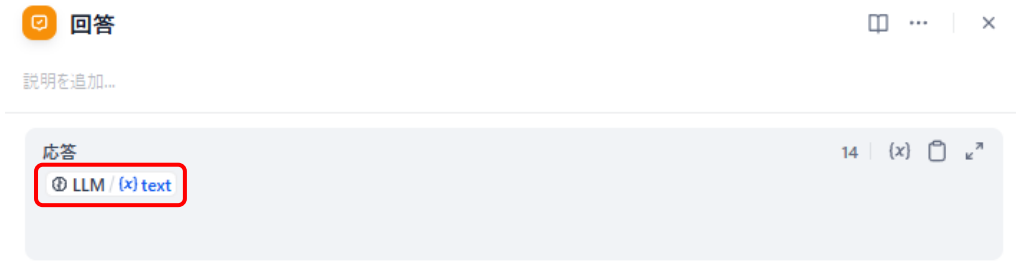
④「回答」ノードの設定方法：

回答 - Dify Docs

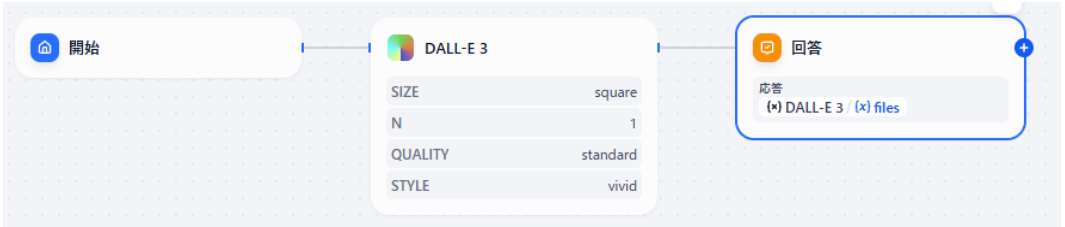
回答：フローの中間や最後にテキスト/画像等の生成結果を出力する。

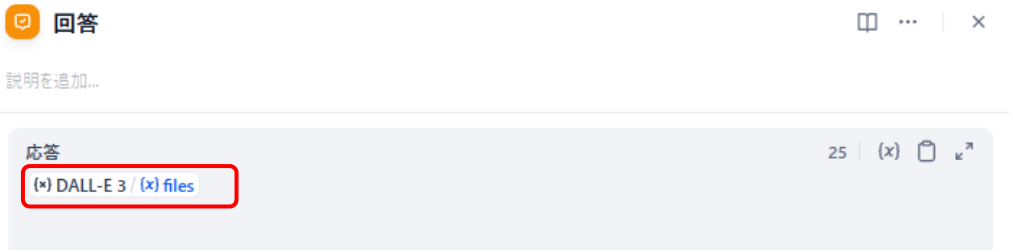
- LLMからの出力はテキストのため、Text変数を指定





- 画像生成用プラグインから画像を出力する場合は、file変数を指定

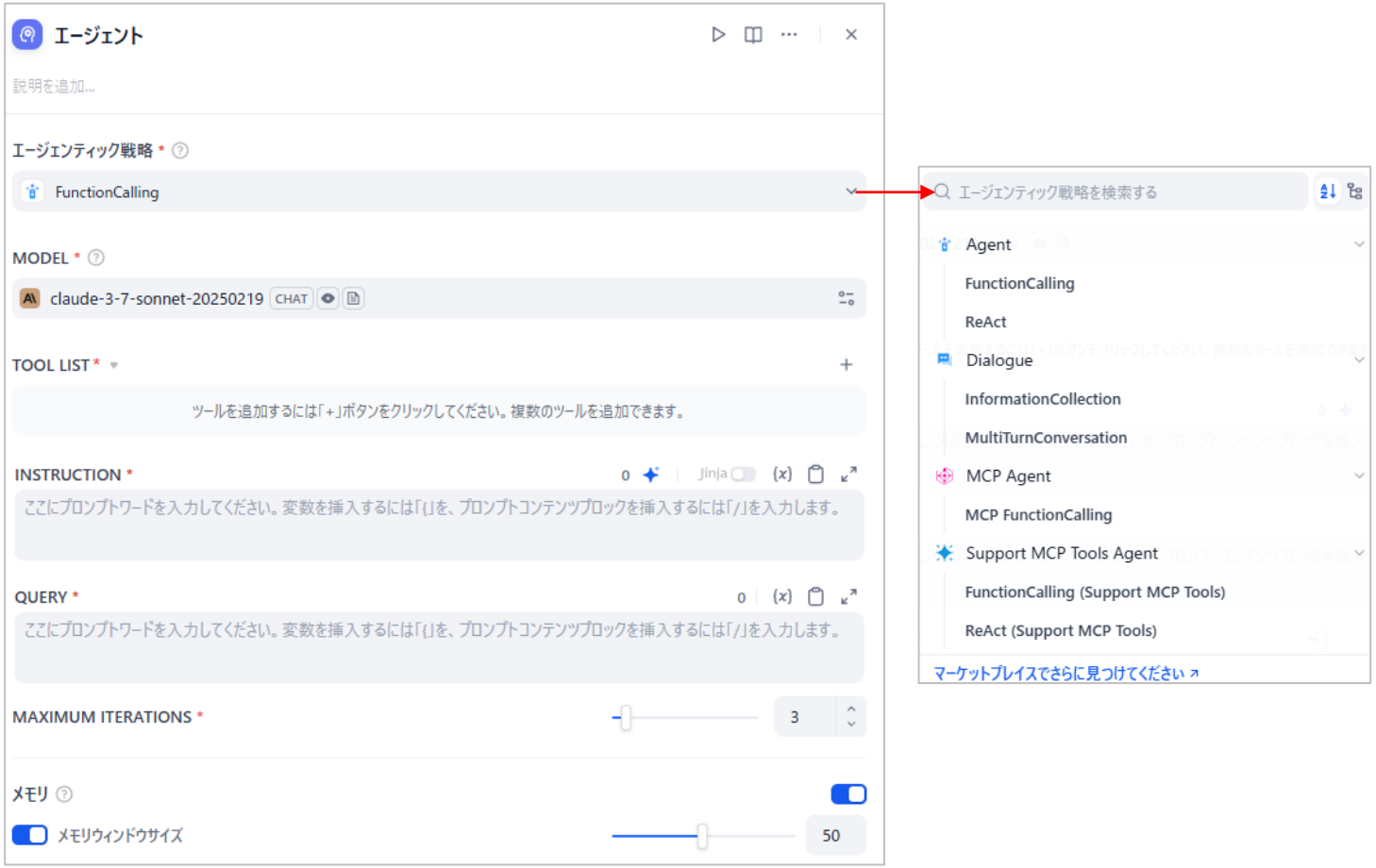




⑤「エージェント」ノードの設定方法：

エージェント - Dify Docs

エージェント：自律的にツールを呼び出す。



設定項目

- **エージェンティック戦略**：ツール呼び出しを実行する方法を設定
  - **事前設定必要。**事前にトップメニュー「プラグイン」からインストールして選択できるようにする

- **MODEL：**エージェントを実行するLLMを選択
- **TOOL LIST**：エージェントから呼び出すツールを設定
- **INSTRUCTION**：タスクの目標とコンテキストを定義。（上流ノードの変数の参照可能）
- **QUERY**：ユーザー入力内容（入力変数）および、エージェントに渡す上で補足しておきたい内容を追加。（なければユーザー入力変数のみでもOK）
- **MAXIMUM ITERATIONS**：最大反復実行回数を設定。
- **メモリウィンドウサイズ**：エージェントが記憶する以前の会話の数を設定。

⑤「エージェント」ノードの設定方法：

プラグインのインストールと利用方法 - Dify Docs

「エージェント」ノードで設定するための事前設定：「エージェント戦略」プラグインのインストール

プラグイン名	Dify Agent Strategies	Agent Strategies (Support MCP Tools)	Dialogue Agent	MCP Agent Strategy
概要	Function Calling や ReAct といった標準的な推論戦略を提供し、LLM が実行時に動的にツールを選択・呼び出せるようにします。	Function Calling・ReAct に加えて MCP プロトコル経由のツール探索・呼び出し（HTTP+SSE/Streamable HTTP）をサポート	構造化された会話を通じて情報を収集できるタスク指向の対話エージェント。動的フィールド検証、マルチフィールド情報抽出、および状態管理をサポート。	Function Calling と互換を保ち、MCP サーバーとのツール呼び出しを統合するプラグイン。
主な機能	- Function Calling：特定ツール（API等）を構造化された関数で呼び出 - ReAct：思考（Reason）と行動（Act）をループしながらツール呼び出し	- MCP対応：HTTP+SSE/Streamable HTTPでMCPサーバー上のツールを発見・呼び出し - ツール接続設定の詳細カスタマイズ	- 入力バリデーション：フォーマット不備時に再入力促進 - 条件分岐：回答に応じて次質問を動的に選択 - 構造化データ出力	- MCPツール呼び出し：SSE/stdio経由でMCPサーバー上のツールをストリーミング操作 - 複数サーバー設定：APIキー・ヘッダー・タイムアウト調整
活用例	- 最新データ検索→グラフ化→レポート自動生成 - 旅行プラン提案（ホテル検索→要約→日程提示）	- リアルタイム株価・天気情報取得 - 社内API連携（CRM/DB照会）	- 申請フォーム自動化（氏名・メールなど順次質問＆検証） - 顧客ヒアリングチャットボット（分岐対話）	- 自動メール配信（CRM→リスト取得→メール生成→配信） - IoTデバイス制御（MCP経由で家電API呼び出し）

⑥「質問分類器」ノードの設定方法：

[質問分類 - Dify Docs]

質問分類器：入力内容を分類して後続ノードに渡し、個別に処理できるようにする。

＊入力の意図をくみ取って分類したい場合に使用

設定項目

- **クラス：入力内容の分類。**
  例）顧客からの問い合わせ窓口：クラス1(製品の使用方法）、クラス2（製品トラブル）、クラス3（その他）
- **高度な設定（オプション）：どのように分類するかの細かい指示**

それぞれの専門分野の知識検索へつなげて専門的な回答をする例

⑥「IF/ELSE」ノードの設定方法：

条件分岐 - Dify Docs

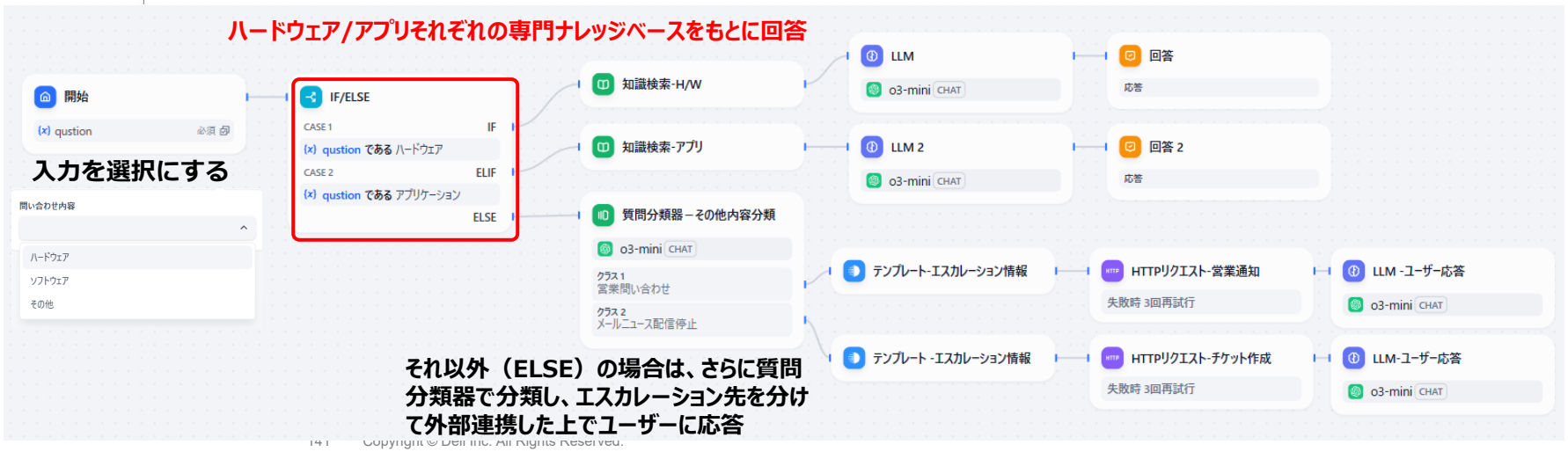
IF/ELSE：条件（IF）に応じて分岐して後続ノードに渡し、個別に処理できるようにする。

＊明確な条件、キーワードで分類したい場合に使用

設定項目

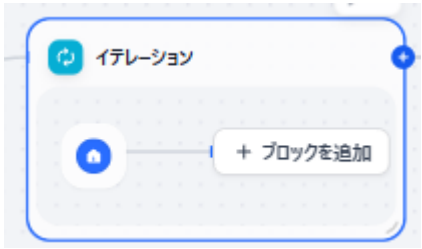
- **IF：条件設定（もしAなら）**
- **ELIF：IFが偽である場合、他の条件設定（もしBなら）**
- **ELSE：条件のすべてが偽である場合（AでもBでもない）**

■ 使用例：問い合わせがIFハードウェア関連なのか、アプリ関連なのか、それ以外なのか分岐



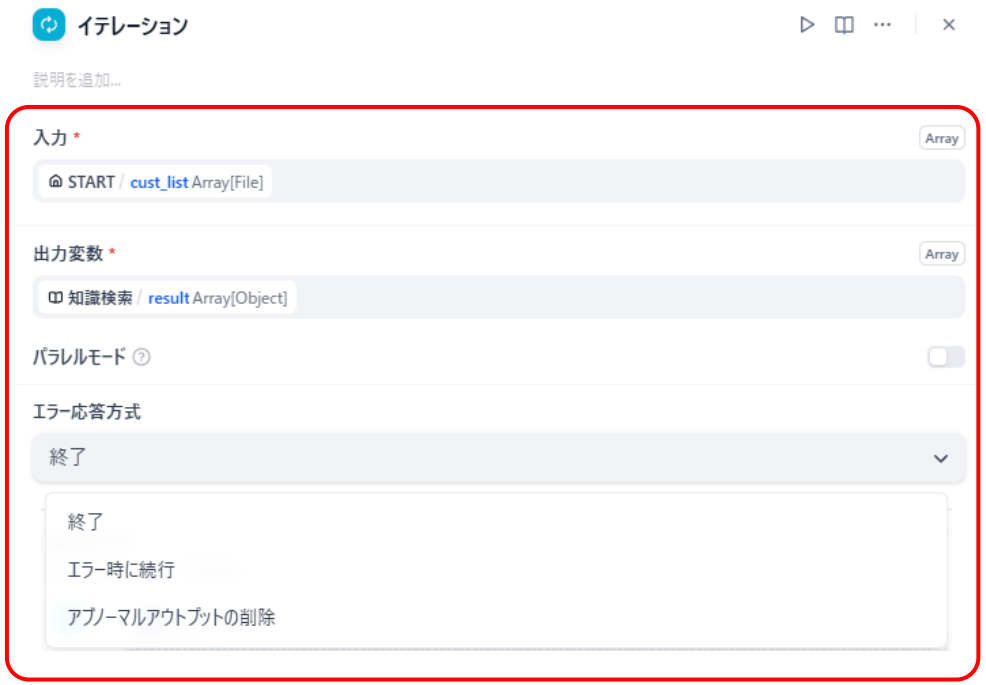
⑦「イテレーション」ノードの設定方法：

反復処理（イテレーション）- Dify Docs



イテレーション：入力リストに対してノード内の処理を順番に繰り返し実行する

■**ユースケース**：イテレーターノードは、記事の段落の一括翻訳、メール処理、または異なるソーシャルメディアプラットフォーム間での日々のマーケティングコンテンツの配信など、繰り返しのステップを必要とするタスクに適しています



設定項目

- **入力**：リスト等の配列型データ（Array）のみ設定可能（直前のノードはArray形式で出力できるものに限られます）
- **出力変数**：リスト等の配列型データ（Array）のみ出力できます。出力が配列型データのため、直接回答として出力することはできません。後続ノードで配列をテキストに変換する必要があります）
- **パラレルモード**：並列処理を実行します。最大10の同時並列処理を実行可能です。
- **エラー応答形式**：処理中にエラーが発生した際の処理方法を設定。
  - **終了**：イテレーションノードを終了し、エラーメッセージを出力。
  - **エラー時に続行**：エラーメッセージを無視して残りの処理を続行。出力には成功した結果と失敗した結果（NULL値）が含まれます。
  - **アブノーマルアウトプットを削除**：エラーメッセージを無視して残りの処理を続行し、出力には成功した結果のみを含みます。

ノード構成のパターン

直前に配置できるノード
- 「コード実行」ノード
- 「パラメータ抽出」ノード
- 「リスト処理」ノード
- 「知識検索」ノード
- 別の「イテレーション」ノード

＊Array（配列）で出入力できるノードのみ前後に配置可能

ノード内に追加できるノード
- 「LLM」ノード
- 「HTTPリクエスト」ノード
- 「コード実行」ノード
- 「テンプレート」ノード
- 「IF/ELSE」ノード
- 「知識検索」ノード

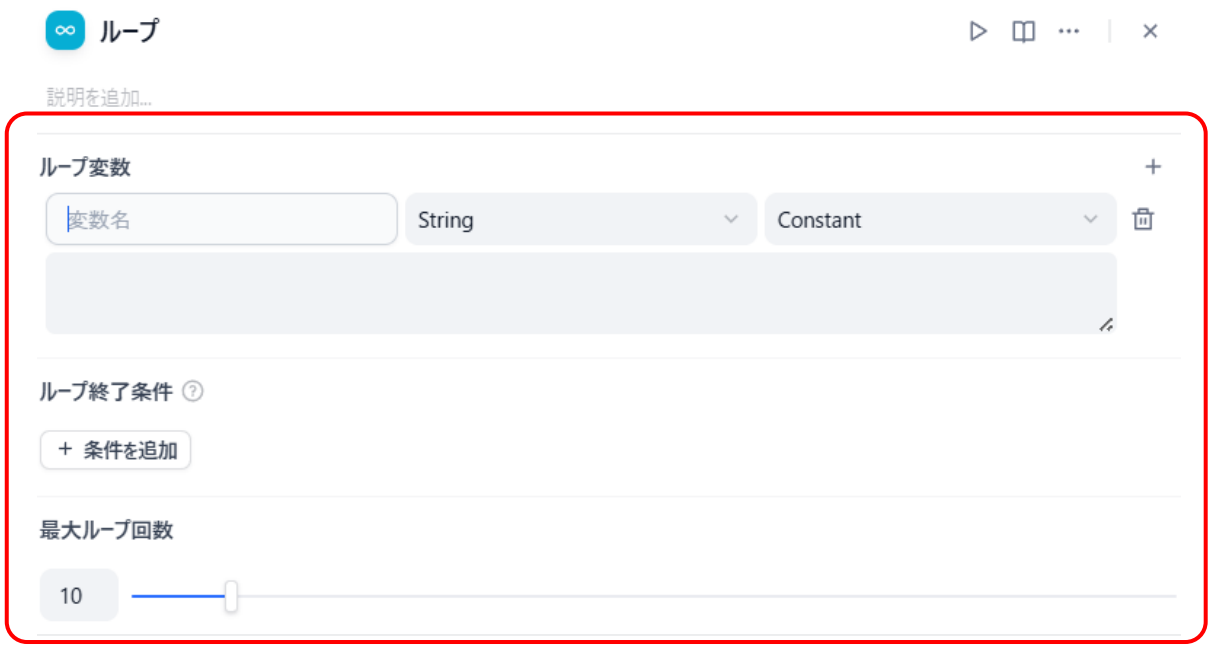
直後に配置できるノード
- 「変数集約器」ノード
- 「テンプレート」ノード
- 「コード実行」ノード
- 「LLM」ノード
- 「IF/ELSE」ノード
- 別の「イテレーション」ノード

⑧「ループ」ノードの設定方法：

繰り返し処理（ループ） - Dify Docs

ループ：結果に基づいてタスクを反復して実行する

＊終了条件を満たすか最大繰り返し回数に達するまで継続



設定項目

- **ループ変数：**ループ内のノードで共有し、複数回のループ処理の間でデータを引き継ぐための変数を設定。ループの継続、終了の条件として使用します。（前回処理結果や累積結果など）
- **ループ終了条件：**ループの終了条件を変数に対する条件で設定します。
  例(1）回答の信頼性評価に基づいた終了：分析用LLMのあとに分析結果の自己評価（0.0-1.0の間で出力）用LLMを配置し、その信頼性評価が>0.8になったときに終了する
  例(2）情報の完全性に基づいた終了：情報収集が十分かLLMにTrue/Falseで回答させて、Trueの値になれば終了する
- **最大ループ回数：**設定した最大ループ回数に達すると終了条件に関わらず強制終了します。（無限ループを防ぎます）

利用例

- 特定の条件が満たされるまでHTTP APIを繰り返し呼び出す
- LLMモデルを使用して、望ましい結果が得られるまで複数回テキスト生成を行う
- データバッチ処理を特定の基準が満たされるまで繰り返す
- 反復的な計算やデータ変換プロセスを実行する

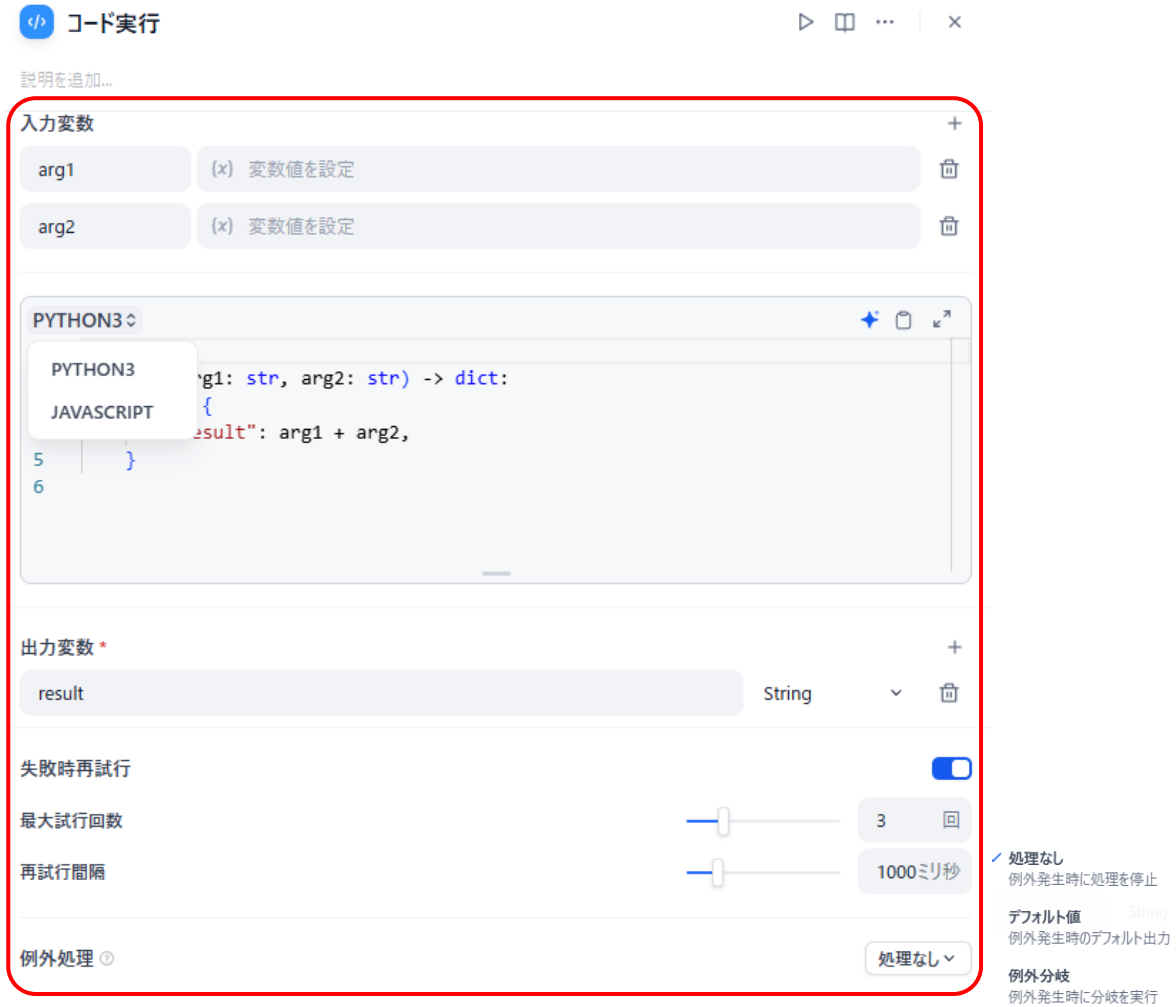
「イテレーション」と「ループ」ノードの違い

項目	「イテレーション」ノード	「ループ」ノード
基本機能	リスト等の配列要素に対して順番に同じ操作を実行し、結果を出力するバッチプロセッサー	終了条件が満たされるか最大回数に達するまで、前の結果に依存する繰り返しタスクを実行
適用シナリオ	バッチ処理、並列データ処理	再帰的操作、最適化問題、条件達成までの反復
入力要件	リストオブジェクト形式の入力値が必要	単一値または変数も可能
反復の特徴	各反復は独立して実行される	各反復は前の反復結果に依存する
終了条件	すべての配列要素が処理されると終了	終了条件の達成、Exit Loopノードの実行、または最大ループ回数に達すると終了
主要構成要素	入力変数、反復ワークフロー、出力変数	ループ終了条件、最大ループ回数、ループ完了ノード

⑨「コード実行」ノードの設定方法：

コード - Dify Docs

コード実行：PythonまたはNode.jsのコードを直接実行してデータ変換や演算処理を行う



設定項目

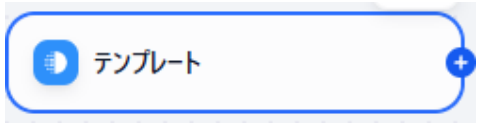
- **入力変数：**上流ノードからのデータを受け取ってコード内で参照させる変数を設定
- **Python/JAVASCRIPT：**実行させるコードを記述
  - Python：科学計算、データ処理、テキスト処理等
  - JAVASCRPT：Web関連の処理やJSON操作等
- **出力変数：**コード実行によって出力された結果を後続ノードに渡すための変数を設定
- **失敗時再試行：コードの実行が失敗したときの自動再試行設定**
  - 最大試行回数：失敗した場合に再試行する最大回数
  - 再試行間隔：各再試行の間隔をミリ秒単位で指定
- **例外処理：**コード実行中にエラーが発生した場合の対応方法
  - 処理なし：処理を停止
  - デフォルト値：指定したデフォルト値を代わりに出力
  - 例外分岐：別の処理フローに分岐し、エラーハンドリングを行う

利用例

- フロー内で非構造化データ処理（JSONの解析、抽出、変換など）を行う
- HTTP応答から特定のデータフィールドを抽出する
- 複雑な数学計算を実行する（配列の分散計算など）
- 複数のデータソースを連結する

⑩「テンプレート」ノードの設定方法：

テンプレート - Dify Docs



テンプレート：前のステップの出力をテキストに変換する

＊Jinja2（Pythonのテンプレート構文）を使って変数を動的にフォーマットして単一のテキストベースの出力に結合します

設定項目

- **入力変数：**上流ノードからのデータを受け取ってテンプレート内で参照させる変数を設定
- **コード：**Jinja2コードを使用してテキストを生成
  ＜設定内容＞
  - テンプレート構文: 静的テキストと動的な変数参照、制御構造（条件分岐、ループなど）の組み合わせ
  - 変数参照: {{ 変数名 }} の形式で変数を参照
  - 制御構造: {% if 条件 %}...{% endif %} などの形式で条件分岐やループを実装
  - フィルター: {{ 変数名|フィルター名 }} の形式で変数を変換（例: 大文字変換、結合など）

利用例

- 複数ソースからのデータを特定の形式に結合
- 条件に基づいて異なる出力を生成
- 繰り返しデータ（リスト、配列）の整形
- テキスト出力の書式設定と構造化
- 後続のノード（LLMノードなど）のための入力テキストの準備

⑪「変数集約器」ノードの設定方法：

[変数集約 - Dify Docs]

変数集約器：複数の出力変数を一つの変数に集約する

設定項目

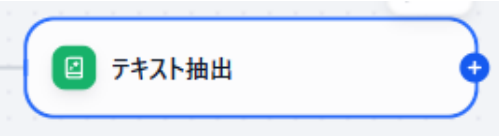
- **Group（1,2…）：**集約する変数のグループを定義。各グループに1つの出力変数（複数の入力変数を含む）を設定できる。
  <設定条件>
  - グループ内の変数（入力、出力）は、同じデータ型である必要があります：
    - Group1：文字列（String）：検索結果スコア
    - Group 2：数字（Number）：信頼性スコア、など

ユースケース例

- **カスタマーサポートの問い合わせ分類と統合された回答：**上流ノードで問い合わせ内容を（以下のような）分野毎に細分化し、それぞれの回答を集約してユーザーに回答する。
  1. 技術ドキュメントを検索して生成するノード
  2. 返品ポリシーを参照して手続き情報を生成するノード
  3. 在庫データベースから情報を取得するノード
- **マルチチャネルマーケティングキャンペーン分析**：複数のマーケティングチャネル（ソーシャルメディア、メール、広告など）のパフォーマンスデータを集約して統合レポートを生成する。
  1. ソーシャルメディア：SNSのAPIからデータ取得・分析
  2. メール：配信システムから開封/クリックデータ取得・分析
  3. 広告：広告プラットフォームからデータ取得・分析
- **顧客プロファイルの統合利用**：複数のシステム（CRM、購買履歴、サポート記録など）のデータを集約してそれらを踏まえた顧客提案内容を生成する
  - 顧客基本データ：企業の基本情報
  - 行動分析：購買パターン・嗜好
  - 対応履歴：サポート状況

⑫「テキスト抽出」ノードの設定方法：

テキスト抽出ツール - Dify Docs



テキスト抽出：様々なファイルの情報をテキストに変換して後続のLLMノードで解釈できるようにする

設定項目

- **入力変数：**ファイルの変数を指定。File（単一ファイル）or File[Array]（複数ファイル）。ファイルを認識・読み取り、情報を抽出し、下流のノードが呼び出せる文字列型の出力変数に変換する。
  <サポートされるファイル形式>
  テキスト抽出ノードは、TXT、Markdown、PDF、HTML、DOCXなどのドキュメント形式の（テキストレイヤーが含まれる）ファイルからのみ情報を抽出できます。画像、音声、動画、その他のファイル形式は処理できません。（画像としてテキストが保存されているPDFは処理できません。）

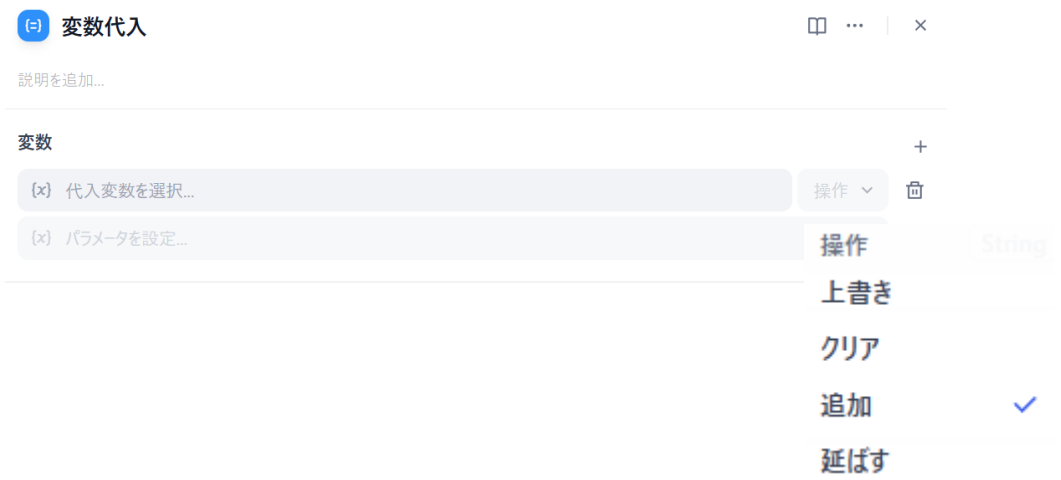
ユースケース例

- **契約書チェックツール：**契約書レビューと重要事項の抽出。開始ノードで契約PDFファイルをアップロードし、テキスト抽出ノードでPDFからテキストを抽出。コード実行ノードで抽出テキストを構造化（セクション分割等）し、LLMノードで、重要事項の特定、リスク分析、要約を生成、終了ノードで分析結果を出力。

⑬「変数代入」ノードの設定方法：

[変数代入 - Dify Docs]

変数代入：書き込み可能な変数に他の変数を代入して後続ノードで活用できるようにする。

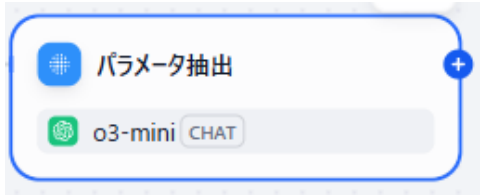


設定項目

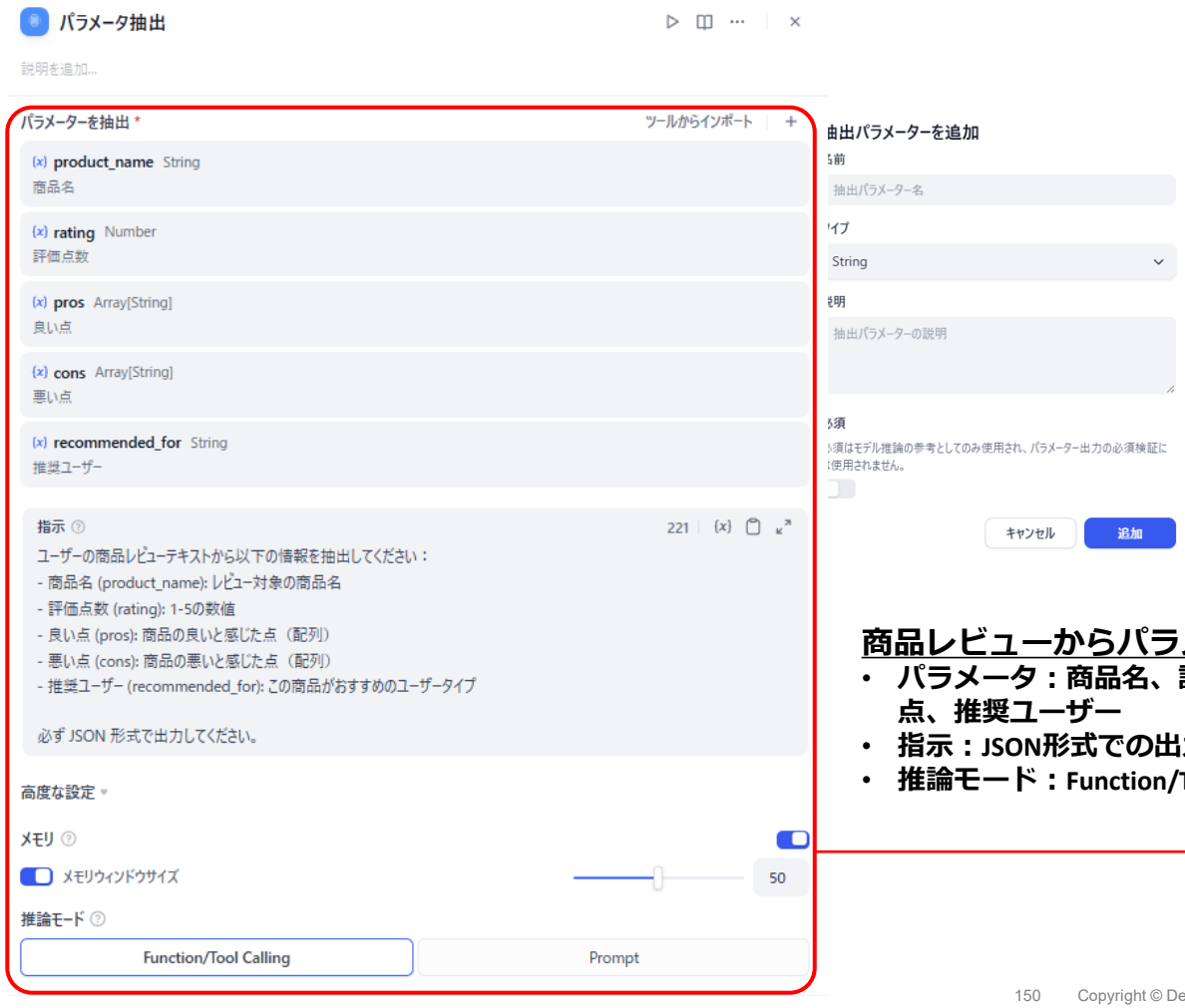
- **変数：**
  - 代入変数を選択：値を代入する変数を設定
  - パラメータを設定：値の取得元を指定し、変数に値を代入する
    - **上書き**：ソース変数でターゲットを直接上書きする。→状態の保持と管理：ユーザープロファイル情報の更新、最新設定値の反映など。
    - **クリア**：選択したターゲット変数の内容をクリアする。→データの加工と変換：プライバシー保護のための情報消去、新しいプロセスの開始前の初期化等。
    - **セット**：ソース変数を必要とせずに手動で値を割り当てる→条件付き処理の実現：ステータスフラグの設定、プロセスの開始条件の設定等。
    - **追加**：ターゲット変数の配列に新しい要素を追加する→パーソナライゼーションの強化：対話履歴の蓄積、顧客購入履歴の追跡など。
    - **延ばす**：ターゲット変数に新しい配列を追加し、複数の要素を一度に追加する→フロー間の情報連携：複数APIからの結果の統合、異なるナレッジベースからの情報の統合など

⑭「パラメータ抽出」ノードの設定方法：

パラメータ抽出 - Dify Docs



パラメータ抽出：自然言語からパラメータを抽出・構造化することで、ツール呼び出しやHTTPリクエストができるようにする



設定項目

- **モデル**：自然言語からパラメータを抽出・構造化するためのモデルを選択
- **入力変数**：パラメータ抽出の対象となる入力データ変数を指定。
- **ビジョン**：画像処理機能を有効化するかどうかの設定。
- **パラメータを抽出**：抽出するパラメータを定義。ツールからインポートまたは手動で追加可能。
- **指示**：LLMがパラメータを抽出するための指示を記述。
- **メモリ**：メモリが有効な場合、質問分類器への各入力には会話のチャット履歴が含まれ、LLMがインタラクティブな対話中にコンテキストを理解し、質問理解を向上させるのに役立ちます。
- **メモリウィンドウサイズ**：参照する会話履歴の数を設定
- **推論モード**：関数/ツール呼び出し機能、またはプロンプトのどちらでパラメータを抽出するか指定。

商品レビューからパラメータ抽出する例

- **パラメータ：商品名、評価点数、良い点、悪い点、推奨ユーザー**
- **指示：JSON形式での出力を指示**
- **推論モード：Function/Tool Calling**

出力例（JSON形式）

```
{
 "product_name": "ノートパソコン",
 "rating": 4,
 "pros": ["処理速度が速い", "画面がきれい"],
 "cons": ["バッテリー持ちが良くない"],
 "recommended_for": "プログラミングをする学生"
}
```

⑮「HTTPリクエスト」ノードの設定方法：

HTTPリクエスト - Dify Docs

HTTPリクエスト：HTTPでサーバーにリクエストを送信し、外部データの取得、ウェブフック、画像生成、ファイルのダウンロードなどを実行する。

設定項目

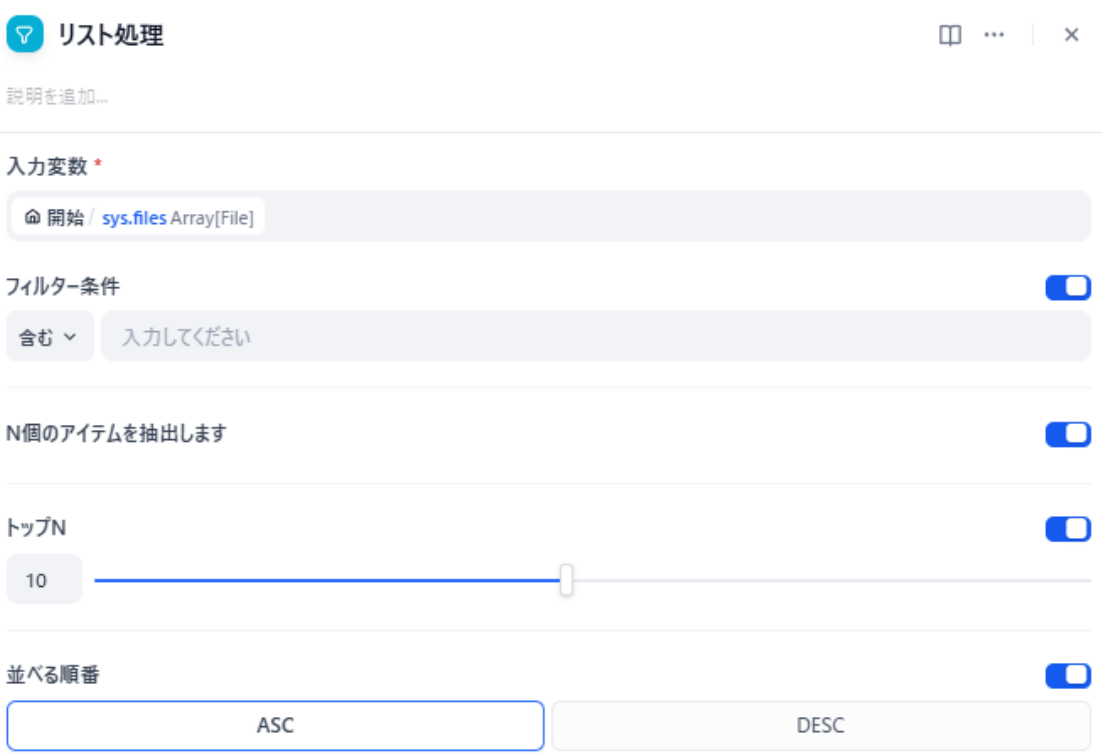
- **API**：HTTPリクエスト先のURLを入力し、GET/POST/PUT/DELETE等のリクエスト種別を選択
- **ヘッダー**：HTTPリクエストのヘッダー情報
  - **キー**：リクエストヘッダーの名前（例: Content-Type, Authorization）
  - **値**：対応するヘッダー値（例: application/json, Bearer token）
- **パラメータ**：クエリパラメータ（URL末尾の?の後に付く値）を設定
  - **キー**：クエリパラメータ名
  - **値**：パラメータ値
- **ボディ**：POSTやPUTリクエストで送信するデータ形式と内容を設定
  - none: データなし
  - form-data: フォームデータ形式
  - x-www-form-urlencoded: URLエンコードされたフォームデータ
  - JSON: JSON形式のデータ
  - raw: 生データ
  - binary: バイナリデータ（ファイル送信など）
- **タイムアウト設定**：リクエストがタイムアウトするまでの時間を設定
  - 接続タイムアウト: サーバーへの接続を待機する最大時間（秒）
  - 読み取りタイムアウト: サーバーからのデータ受信を待機する最大時間（秒）
  - 書き込みタイムアウト: サーバーへのデータ送信を待機する最大時間（秒）
- **失敗時の再試行**：リクエストが失敗した場合の再試行設定
- **例外処理**：リクエスト失敗時の対応方法

「HTTPリクエスト」ノードは、外部APIとの連携、データ取得、Webhookの送信などに活用できます。また、HTTPリクエストの戻り値には、レスポンス本文、ステータスコード、レスポンスヘッダー、ファイルが含まれます。レスポンスにファイルが含まれている場合、このノードは自動的にファイルを保存し、ワークフローの後続ステップで使用できるようにします。

⑯「リスト処理」ノードの設定方法：

[リスト処理 - Dify Docs]

リスト処理：アップロードされたファイルを種別毎に分けて次のノードに渡して個別に処理するために使われる。



設定項目

- **入力変数：**リスト処理する対象の配列変数を指定。リスト処理ノードは、Array[string]、Array[number]、Array[file]の変数のみ受け付けます。
- **フィルター条件：**入力変数の配列からフィルタで指定した条件を満たすすべての配列変数を抽出します。ファイル名やファイルタイプ、ファイルサイズなどに基づいて配列要素をフィルタリングするなど。
  <フィルタ可能な属性>
  - type: ファイルカテゴリ（画像、ドキュメント、音声、動画など）
  - size: ファイルサイズ
  - name: ファイル名
  - url: URLを通じてアップロードされたファイルの完全なURL
  - extension: ファイル拡張子
  - mime_type: MIMEタイプ（例: "text/html"）
  - transfer_method: ファイルアップロード方法（ローカルアップロードまたはURL経由）
- **N個のアイテムを抽出します/トップN：**配列の先頭からN個のアイテムを抽出します。
- **並べる順番：**配列の並び替え→ASC：昇順（アルファベット順：A～Z）、DESC：降順（逆アルファベット順：Z～A）

アップグレード手順

- Difyのバージョンアップ方法

Difyアップグレード手順 – ローカルPC環境

1. カスタム設定(yaml)ファイルのバックアップ
2. 最新コードの取得
3. サービス停止
4. データボリュームのバックアップ
5. サービス再起動
6. 注意点
   - 環境変数の差分確認：新バージョンで.env.exampleに追加項目がある可能性があります。自環境の.envと比較し、必要に応じて変数を追記してください
   - .env初期化（必要な場合）：新規導入や設定ファイルを失った場合は、サンプルをコピーして再作成します。
   - ログ確認：コンテナ起動後にdocker compose logs -f apiなどでマイグレーションやエラー発生をチェックし、問題がないか確認してください。

Difyアップグレード手順 – ローカルPC環境

1. カスタム設定(yaml)ファイルのバックアップ（yamlファイルで設定をカスタムしている場合に実行してください）

<PowerShellまたはコマンドプロンプト>

```
cd dify/docker
$ts = Get-Date -Format "yyyyMMdd_HHmmss“
Copy-Item docker-compose.yaml "docker-compose.yaml.$ts.bak"
```

#タイムスタンプを変数に格納
#バックアップ

<バックアップできているかの確認方法>

```
Test-Path -Path ".¥docker-compose.yaml.$ts.bak"          #バックアップされているか確認
```

➢ “True”と返ってきたらバックアップされています。

Difyアップグレード手順 – ローカルPC環境

2. 最新コードの取得

```
git checkout main
```

次のエラーメッセージが出る場合

```
PS C:\Users\若松信康\dify\docker> git checkout main
fatal: detected dubious ownership in repository at 'C:/Users/若松信康/dify'
'C:/Users/若松信康/dify' is owned by:
        BUILTIN/Administrators (S-1-5-32-544)
but the current user is:
        AzureAD/[illegible] (S-1-12-1-1824770312-1152713296-2884549013-1425004037)
To add an exception for this directory, call:
```

このメッセージは、Git 2.35.2 以降で導入された「リポジトリ所有者がコマンド実行ユーザーと異なる場合、誤操作や脆弱性を防ぐために安全とみなさない」セキュリティ機能によるものです。エラー自体はリポジトリ破損やデータ損失ではなく、所有権の不一致を検出した際の保護措置です。リポジトリが信頼できるものであれば、所有者の変更または「safe.directory」設定に追加することで解消できます。

safe.directory に追加：リポジトリを「このユーザーで安全に扱う」と明示的に登録

```
git config --global --add safe.directory 'C:/Users/若松信康/dify'
```

＊自分の環境のDifyのディレクトリを指定してください。

<確認方法>

```
git config --global --get-all safe.directory
```

正常に追加されていれが該当ディレクトリが表示されます。

```
PS C:\Users\若松信康\dify\docker> git config --global --get-all safe.directory
C:/Users/若松信康/dify
```

```
git pull origin main
```

Difyアップグレード手順 – ローカルPC環境

3. サービス停止

```
docker compose down
```

4. データボリュームのバックアップ

Unixエポック秒を取得する場合は以下を使う。
$timestamp = [int][double]::Parse((Get-Date -UFormat %s))

```
$timestamp = [DateTimeOffset]::Now.ToUnixTimeSeconds()
tar -cvf "volumes-$timestamp.tgz" --exclude="**/.venv" --exclude="**/lib64" volumes
```

タイムスタンプ（日時）を取得

#tarコマンドでバックアップ

<生成ファイルの確認>

```
Get-ChildItem -Filter "volumes-*.tgz"
```

一覧に volumes-1714076403.tgz のようなファイル名が表示されれば成功です

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---          2025/04/26     20:51       75872768 volumes-.tgz
-a---          2025/04/26     20:53       75872768 volumes-1745668382.tgz
```

- タイムスタンプが取れていない（失敗）
- タイムスタンプが取れている（成功）

<ファイルの存在チェック>

```
Test-Path -Path ".¥volumes-1745668382.tgz"
```

```
PS C:\Users\若松信康\dify\docker> Test-Path -Path ".\volumes-1745668382.tgz"
True
```

- True（成功）

Difyアップグレード手順 – ローカルPC環境

5. サービス再起動

```
docker compose pull                                    # イメージを最新化（任意だが推奨）
docker compose up -d
```

6. データベースマイグレーション

```
docker compose exec api flask db upgrade
docker compose exec api flask transform-datasource-credentials    #データソース認証情報の変換（v1.9.0未満からのアップグレード時）
```

注意点

- 環境変数の差分確認：新バージョンで.env.exampleに追加項目がある可能性があります。自環境の.envと比較し、必要に応じて変数を追記してください
- .env初期化（必要な場合）：新規導入や設定ファイルを失った場合は、サンプルをコピーして再作成します。
- ログ確認：コンテナ起動後にdocker compose logs -f apiなどでマイグレーションやエラー発生をチェックし、問題がないか確認してください。

Difyアップグレード手順 – 旧VerのDockerイメージの削除（任意）

バージョンアップしても、古いバージョンのDockerイメージは自動削除されません。

- 削除してもいいケース
  - 旧バージョンのイメージのコンテナすべて停止している
  - ディスク容量を節約したい
  - ロールバック予定がなく、新バージョンを安定運用する場合

- 残したほうがいいケース
  - 旧バージョンで動くテスト環境を並行かどうさせる場合
  - トラブル発生時に旧バージョンでの復旧検証が必要な場合

➢ 古いイメージの確認方法

```
docker images                    # 全イメージ一覧を表示
```

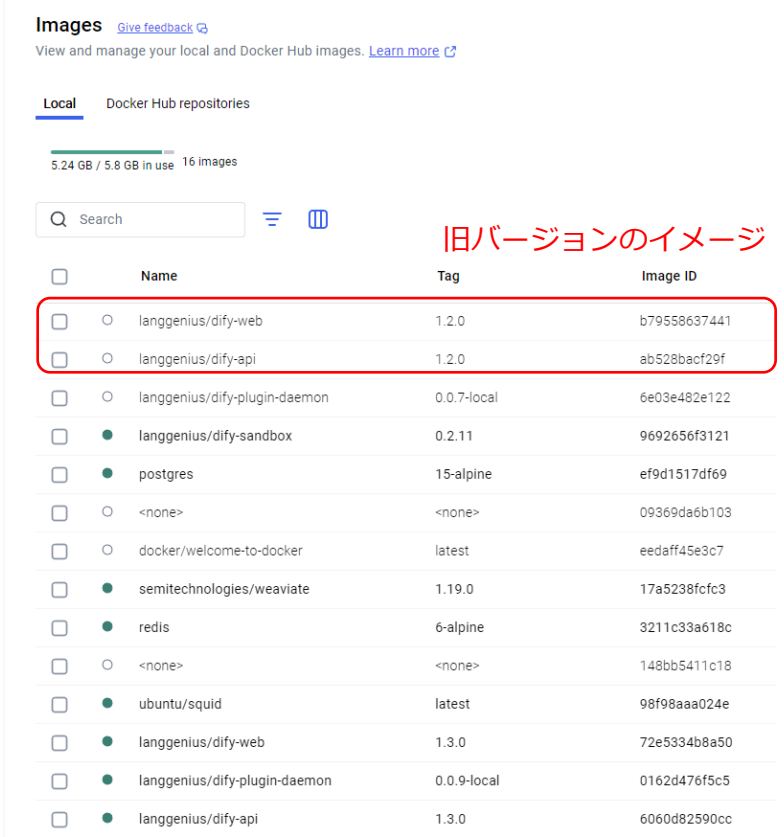
```
PS C:\Users\若松信康\dify\docker> docker images
REPOSITORY                    TAG          IMAGE ID       CREATED         SIZE
redis                         6-alpine     3211c33a618c   2 days ago      45.2MB
langgenius/dify-plugin-daemon 0.0.9-local  0162d476f5c5   2 days ago      1.81GB
langgenius/dify-web           1.3.0        72e5334b8a50   3 days ago      762MB
langgenius/dify-api           1.3.0        6060d82590cc   3 days ago      2.72GB
nginx                         latest       5ed8fcc66f4e   10 days ago     281MB
<none>                        <none>       e97dcded971f   2 weeks ago     333MB
langgenius/dify-web           1.2.0        b79558637441   2 weeks ago     756MB
langgenius/dify-api           1.2.0        ab528bacf29f   2 weeks ago     3.12GB
```

➢ 特定タグを削除する場合

```
docker rmi langgenius/dify:1.2.0
```

➢ 未使用イメージをまとめてクリーンナップする場合

```
docker image prune
```



（参考）Difyの内部構造

- YAML設定ファイル（docker-compose.yml）を読み解く

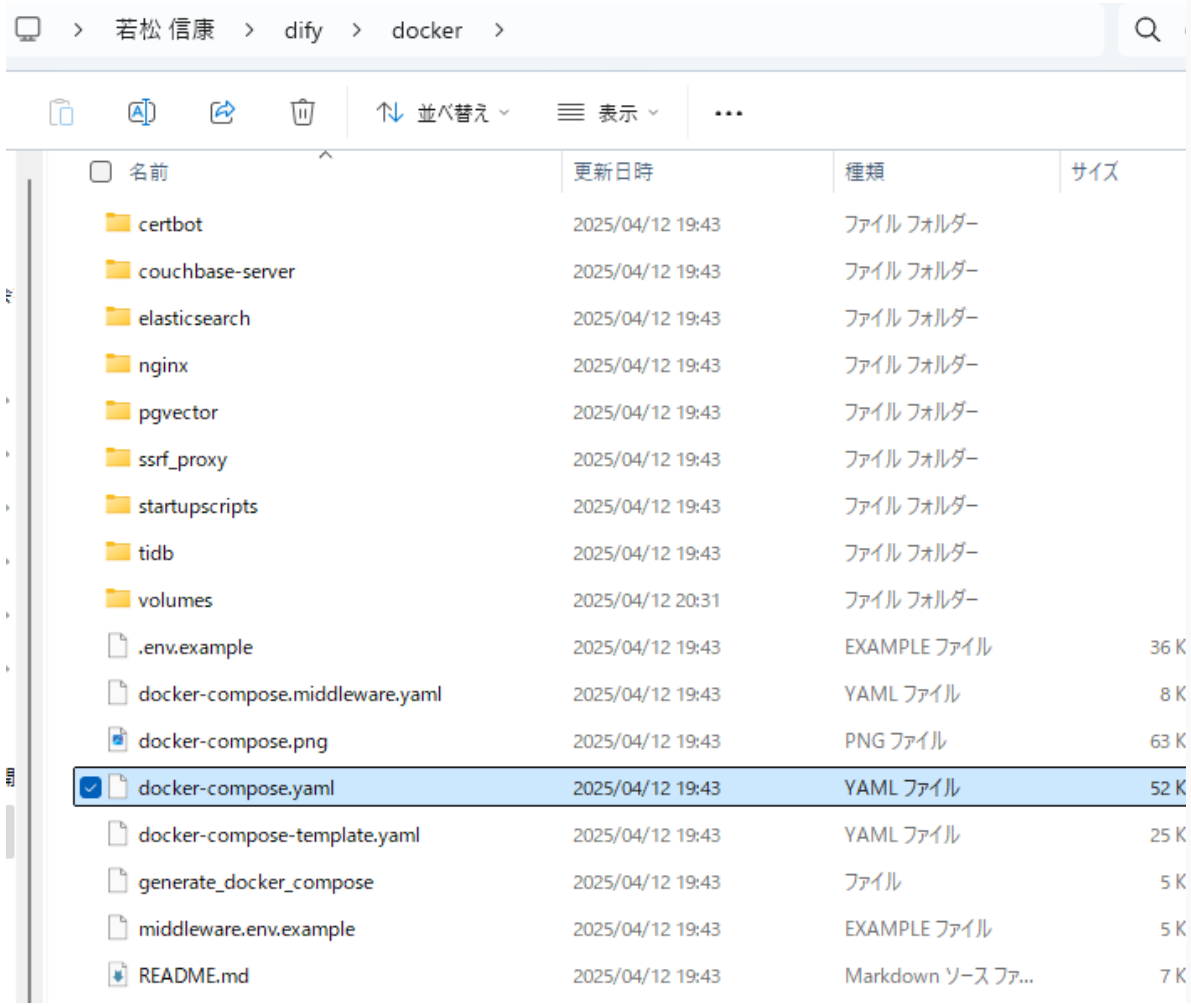
Difyの内部構造を理解するメリット

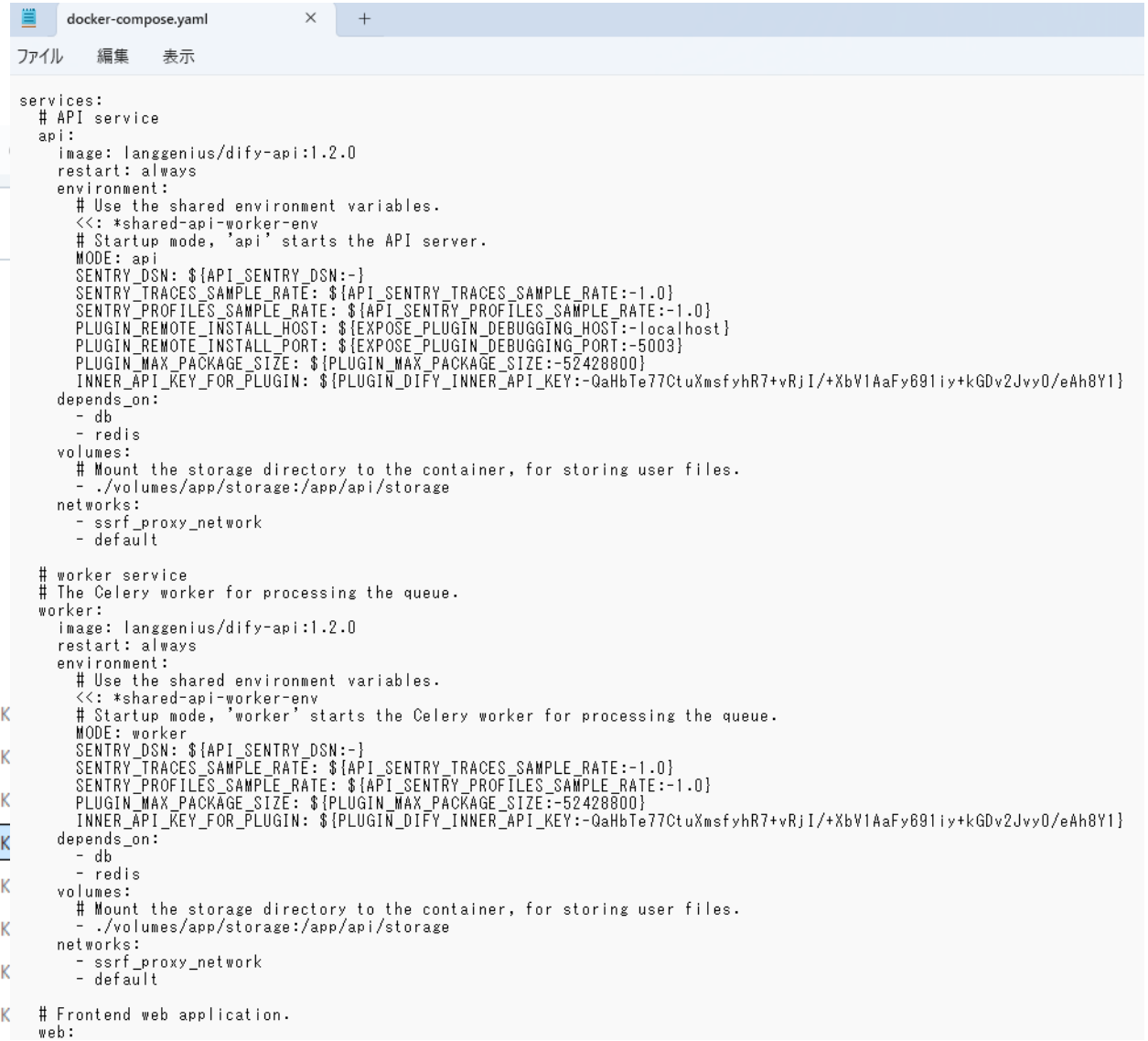
- docker-compose.ymlファイル

1. **環境構築の理解**：Difyがどのようなコンポーネント（データベース、キャッシュ、検索エンジンなど）で構成されているかを把握でき、システム全体のアーキテクチャを理解できます。
2. **カスタマイズの容易さ**：設定を変更することで、メモリ割り当て、ポート設定、ボリューム設定などを自分の環境に合わせて調整できます。
3. **トラブルシューティングの効率化**：問題が発生した際に、どのコンポーネントで問題が起きているかを特定しやすくなります。
4. **スケーリングの計画**：リソース使用量を理解し、必要に応じてスケールアップやスケールアウトの計画を立てやすくなります。
5. **セキュリティ設定の確認**：環境変数や公開ポートなどのセキュリティ関連設定を確認・調整できます。
6. **拡張性の理解**：追加したいサービスやコンポーネントをどのように統合すべきかの判断材料になります。デプロイメントの自動化：CI/CDパイプラインを構築する際の参考になります。

Difyの内部構造

- docker-compose.ymlファイル





Difyの内部構造

- docker-compose.ymlファイル

docker-compose.yaml

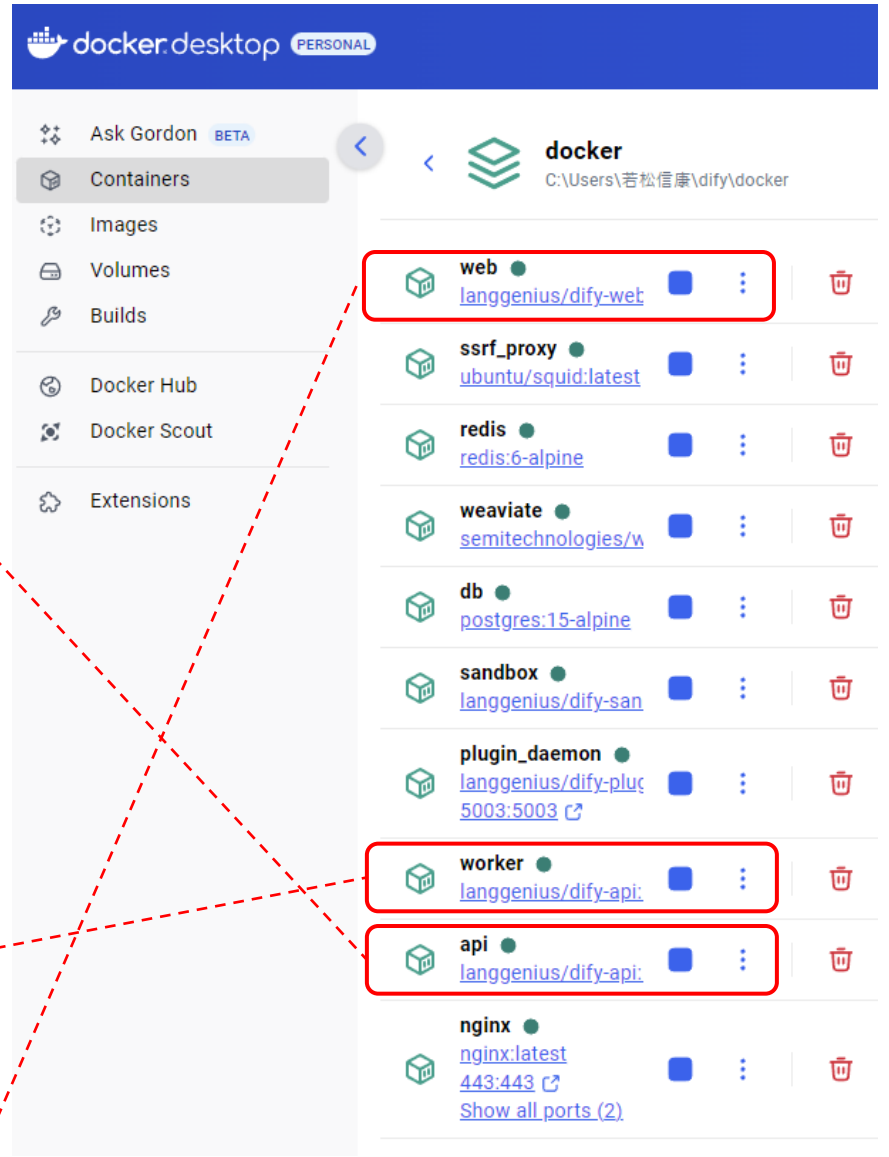
ファイル　編集　表示

```
services:
  # API service
  api:
    image: langgenius/dify-api:1.2.0
    restart: always
    environment:
      # Use the shared environment variables.
      <<: *shared-api-worker-env
      # Startup mode, 'api' starts the API server.
      MODE: api
      SENTRY_DSN: ${API_SENTRY_DSN:-}
      SENTRY_TRACES_SAMPLE_RATE: ${API_SENTRY_TRACES_SAMPLE_RATE:-1.0}
      SENTRY_PROFILES_SAMPLE_RATE: ${API_SENTRY_PROFILES_SAMPLE_RATE:-1.0}
      PLUGIN_REMOTE_INSTALL_HOST: ${EXPOSE_PLUGIN_DEBUGGING_HOST:-localhost}
      PLUGIN_REMOTE_INSTALL_PORT: ${EXPOSE_PLUGIN_DEBUGGING_PORT:-5003}
      PLUGIN_MAX_PACKAGE_SIZE: ${PLUGIN_MAX_PACKAGE_SIZE:-52428800}
      INNER_API_KEY_FOR_PLUGIN: ${PLUGIN_DIFY_INNER_API_KEY:-QaHbTe77CtuXmsfyhR7+vRjI/+XbV1AaFy691iy+kGDv2JvyO/eAh8Y1}
    depends_on:
      - db
      - redis
    volumes:
      # Mount the storage directory to the container, for storing user files.
      - ./volumes/app/storage:/app/api/storage
    networks:
      - ssrf_proxy_network
      - default

  # worker service
  # The Celery worker for processing the queue.
  worker:
    image: langgenius/dify-api:1.2.0
    restart: always
    environment:
      # Use the shared environment variables.
      <<: *shared-api-worker-env
      # Startup mode, 'worker' starts the Celery worker for processing the queue.
      MODE: worker
      SENTRY_DSN: ${API_SENTRY_DSN:-}
      SENTRY_TRACES_SAMPLE_RATE: ${API_SENTRY_TRACES_SAMPLE_RATE:-1.0}
      SENTRY_PROFILES_SAMPLE_RATE: ${API_SENTRY_PROFILES_SAMPLE_RATE:-1.0}
      PLUGIN_MAX_PACKAGE_SIZE: ${PLUGIN_MAX_PACKAGE_SIZE:-52428800}
      INNER_API_KEY_FOR_PLUGIN: ${PLUGIN_DIFY_INNER_API_KEY:-QaHbTe77CtuXmsfyhR7+vRjI/+XbV1AaFy691iy+kGDv2JvyO/eAh8Y1}
    depends_on:
      - db
      - redis
    volumes:
      # Mount the storage directory to the container, for storing user files.
      - ./volumes/app/storage:/app/api/storage
    networks:
      - ssrf_proxy_network
      - default

  # Frontend web application.
  web:
```

APIサービス
コンテナ

Workerサービス
コンテナ

Webサービスコンテナ



Difyの内部構造

- docker-compose.ymlファイル>APIサービスコンテナ

```
services:
  # API service
  api:
    image: langgenius/dify-api:1.2.0
    restart: always
    environment:
      # Use the shared environment variables.
      <<: *shared-api-worker-env
      # Startup mode, 'api' starts the API server.
      MODE: api
      SENTRY_DSN: ${API_SENTRY_DSN:-}
      SENTRY_TRACES_SAMPLE_RATE: ${API_SENTRY_TRACES_SAMPLE_RATE:-1.0}
      SENTRY_PROFILES_SAMPLE_RATE: ${API_SENTRY_PROFILES_SAMPLE_RATE:-1.0}
      PLUGIN_REMOTE_INSTALL_HOST: ${EXPOSE_PLUGIN_DEBUGGING_HOST:-localhost}
      PLUGIN_REMOTE_INSTALL_PORT: ${EXPOSE_PLUGIN_DEBUGGING_PORT:-5003}
      PLUGIN_MAX_PACKAGE_SIZE: ${PLUGIN_MAX_PACKAGE_SIZE:-52428800}
      INNER_API_KEY_FOR_PLUGIN: ${PLUGIN_DIFY_INNER_API_KEY:-QaHbTe77CtuXmsfyhR7+vRjI/
+XbV1AaFy691iy+kGDv2Jvy0/eAh8Y1}
    depends_on:
      - db
      - redis
    volumes:
      # Mount the storage directory to the container, for storing user files.
      - ./volumes/app/storage:/app/api/storage
    networks:
      - ssrf_proxy_network
      - default
```

使用するイメージを指定

問題発生時の動作を指定：常にリスタートを試みる

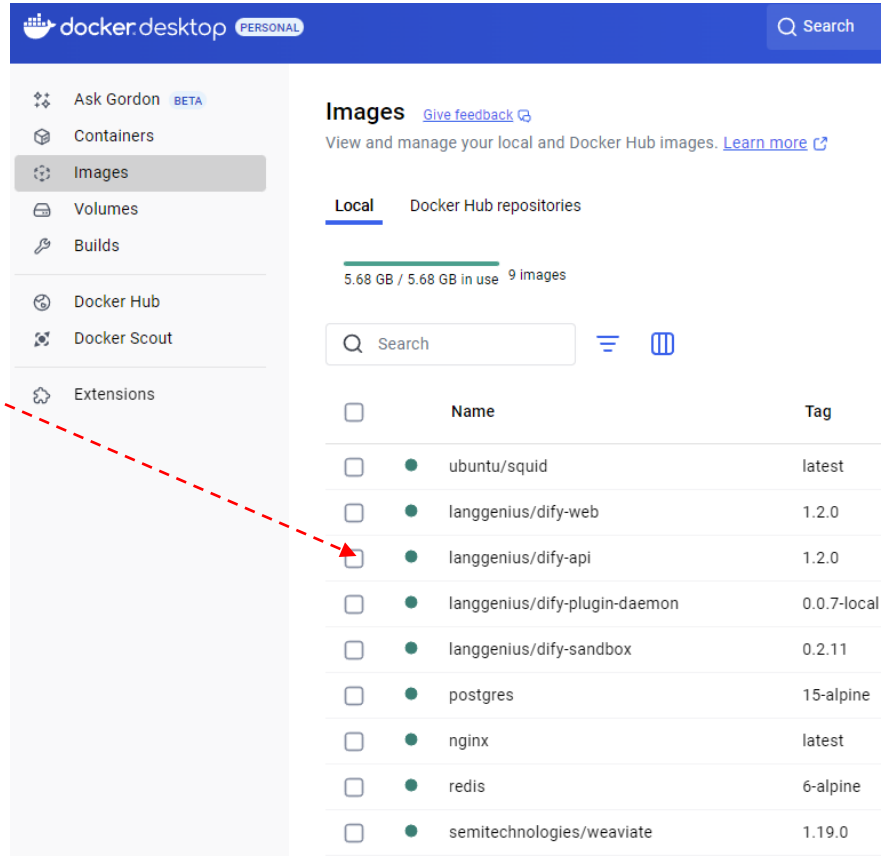
環境設定を指定：共通の設定を挿入

動作モードを指定：APIモード

実行の依存関係の指定：DBとRadisを起動してから起動する

データの保存場所
コンテナ：/app/api/storage
PC：.volumes/app/storage

ネットワーク接続先の指定
・ssrf_proxy_network
・default



ローカル環境考察

1. ローカルでLLMを実行するために必要なスペックの目安
2. ローカルでLLMを実行する場合の考慮点

ローカルでLLMを実行するために必要なスペックの目安

LLMのサイズと実行可能な環境の目安:

- 7B（70億）パラメータモデル: 8GB VRAM以上のGPUか16GB以上のRAM
- 13B（130億）パラメータモデル: 16GB VRAM以上のGPUか32GB以上のRAM
- 70B（700億）パラメータモデル: 量子化しても40GB以上のVRAMか80GB以上のRAMが必要

必要なハードウェア仕様の目安
基本的な計算式
モデル実行に必要なメモリは基本的に次の式で計算できます： VRAM = パラメータ数 × バイト/パラメータ × オーバーヘッド
精度による必要メモリの違い
量子化レベル別の必要メモリ：
•FP16（16ビット）：1Bパラメータあたり2GB →7Bパラメータ：約14GB VRAM
•INT8（8ビット）：1Bパラメータあたり1GB →7Bパラメータ：約7GB VRAM
•INT4（4ビット）：1Bパラメータあたり0.5GB →7Bパラメータ：約3.5GB VRAM

推奨システム構成
GPUインファレンスの場合、RAMの主な役割はモデルの重みをストレージからVRAMにロードすることです。そのため、少なくともVRAMと同じ量のRAMを持ち、できればVRAMの1.5～2倍のRAMを持つことを推奨します
CPU
最新のマルチコアプロセッサ（Intel Core i5/i7/i9、AMD Ryzen 5/7/9など）
RAM
•少なくともVRAMと同量以上（GPUのVRAMが8GBなら最低8GB）
•理想的にはVRAMの1.5～2倍（GPUのVRAMが8GBなら12～16GB）
ストレージ
高速なNVMe SSDが推奨（モデルの読み込み速度に影響）
GPU
•8GBのVRAM: 4ビット量子化した7Bモデルが実行可能
•16GBのVRAM: 8ビット量子化した13Bモデル、または4ビット量子化した24Bモデルが実行可能
•24GBのVRAM: 4ビット量子化されたLlama3-70Bは、2台のA10 24GB GPUで実行可能
•80GBのVRAM: Llama 2 70Bモデルを16ビットモードでサービングするには、2台のA100 80GBで十分
量子化のメリットとトレードオフ
一般的に、8ビット量子化は16ビットを使用する場合と同様のパフォーマンスを達成できます。しかし、4ビット量子化はモデルのパフォーマンスに顕著な影響を与える可能性があります
より詳細な計算や特定のモデルに対する正確なメモリ要件を知りたい場合は、Hugging Faceが提供するようなLLMモデルVRAM計算ツール利用すると便利です。

ローカルPC上でLLMを実行する場合の考慮点：アーキテクチャ

GPU対応の違い

- LLMの実行においては、NVIDIAとAMDのGPUでは選択肢が異なります。NVIDIAが広く推奨される主な理由は、LinuxとWindowsの両方で様々な推論ライブラリにわたるCUDAのサポートが広範囲に及んでいるためです。特にWindows環境では、多くのプロジェクトがNVIDIAカード向けに構築されており、それを前提としています。

X86版Windowsの利点

1. 成熟したエコシステム：より多くのLLMツールとソフトウェアが対応
2. NVIDIA GPUとの互換性：NVIDIAは最近、Windows PCでのLLM推論と開発を加速するための開発者ツールを発表しました。
3. AMD GPUサポートの改善：Ollamaは一部のAMD Radeon 6000および7000シリーズカードのネイティブサポートを開始しました。

推奨アプローチ

現時点では以下のことを考慮すべき：

1. **ハードウェア重視の選択**：
    1. NVIDIA GPUを搭載したx86版Windowsが現状最も互換性が高く、問題が少ない選択肢です
    2. AMDのRyzen AI PCやRadeon 7000シリーズグラフィックスカードでもLLMを実行できるようになってきています
2. **ソフトウェア選択**：
    1. Ollamaなどのクロスプラットフォーム対応ツールを選ぶと将来的な互換性が高まります
    2. WSL（Windows Subsystem for Linux）の活用：一部の開発者は、WSLを使用してUbuntu Linuxをホストし、LLMを微調整するアプローチを取っています
3. **Arm版Windows選択時の注意点**：
    1. ネイティブArmアプリの利用を優先する
    2. エミュレーションによるパフォーマンス低下を考慮する
    3. NPUにオフロードできるライブラリが提供されているか確認する