JAEA-Technology
2025-017
DOI:10.11484/jaea-technology-2025-017

JAEA-Technology

スーパーコンピュータを用いたオンプレミス生成AI基盤の構築と展開

Construction and Deployment of an On-premises Generative AI Infrastructure Using Supercomputers

高久 雄飛　坂爪 駿　木村 英雄

Yuhi TAKAKU, Shun SAKAZUME and Hideo KIMURA

システム計算科学センター

Center for Computational Science & e-Systems

March 2026

Japan Atomic Energy Agency　日本原子力研究開発機構

国立研究開発法人日本原子力研究開発機構　研究開発推進部　科学技術情報課
〒 319-1112　茨城県那珂郡東海村大字村松 4 番地 49
E-mail: ird-support@jaea.go.jp

JAEA-Technology 2025-017

スーパーコンピュータを用いたオンプレミス生成 AI 基盤の構築と展開

日本原子力研究開発機構
システム計算科学センター

高久　雄飛、坂爪　駿、木村　英雄

（2025 年 12 月 1 日受理）

国立研究開発法人日本原子力研究開発機構（原子力機構）では、業務効率化や研究開発におけるアイデア創出といった観点から、生成 AI の活用に対する期待とニーズが高まっていた。しかしながら、ChatGPT をはじめとするクラウド型の外部生成 AI サービスは、入力データを学習に利用するという特性を有するため、セキュリティ上の観点から取り扱うことのできない情報が少なからず存在した。また、利用開始までの申請や手続きが煩雑であり、原子力機構内において生成 AI が十分に普及・活用されているとは言い難い状況であった。

このような背景を踏まえ、我々は原子力機構が保有するスーパーコンピュータ（スパコン）などの既存の計算資源とオープンソースソフトウェアを活用し、導入コストを抑えつつ、安全かつ容易に利用できる生成 AI 基盤を構築し、原子力機構内へ展開した。その結果、日常業務の効率化に一定の効果が見られたほか、生成 AI への関心が原子力機構全体で高まり、生成 AI の活用へ向けた取り組みが拡大するようになった。

原子力科学研究所: 〒319-1195　茨城県那珂郡東海村大字白方 2-4

JAEA-Technology 2025-017

Construction and Deployment of an On-premises Generative AI Infrastructure Using Supercomputers

Yuhi TAKAKU, Shun SAKAZUME and Hideo KIMURA

Center for Computational Science & e-Systems
Japan Atomic Energy Agency
Tokai-mura, Naka-gun, Ibaraki-ken

(Received December 1, 2025)

At the Japan Atomic Energy Agency (JAEA), expectations and demand for generative AI had been increasing, particularly to improve operational efficiency and foster ideas in research and development. However, cloud-based external generative AI services such as ChatGPT typically use input data for learning, which raised security concerns and prevented handling a considerable amount of information. In addition, the required procedures and applications before use were cumbersome, making it hard to say that generative AI was widely adopted or effectively used within JAEA.

To address these issues, we built a generative AI infrastructure using JAEA's existing computing resources, including its supercomputers, and open-source software. This approach kept implementation costs low while ensuring safety and ease of use. After deployment across the organization, we observed notable improvements in daily operational efficiency and a surge in interest in generative AI, leading to expanded initiatives for its utilization.

Keywords: Generative AI, Supercomputers, On-premises

目次

Contents

1. はじめに

近年、生成 AI は急速な発展を遂げており、あらゆる分野において従来の枠組みを根底から変革し得る新たなパラダイムシフトの契機として、世界的に注目を集めている。

国立研究開発法人日本原子力研究開発機構（以下、「原子力機構」という）においても、業務効率化や研究開発におけるアイデア創出といった観点から、生成 AI の活用に対する期待とニーズが高まっていた。しかしながら、原子力機構では、ChatGPT をはじめとするクラウド型の外部生成 AI サービスを利用するまでの申請や手続きが煩雑であり、原子力機構内において生成 AI が十分に普及・活用されているとは言い難い状況であった。また、生成 AI の特性やリスクを十分に理解し、適切かつ効率的に活用する能力（以下、「生成 AI リテラシー」という）を有する人材も多くはなかった。

これらの課題を解決するためには、自由かつ主体的に試行できる生成 AI 環境が必要である。しかし、外部生成 AI サービスは、入力データを学習に利用するという特性を有するため、セキュリティ上の観点から取り扱うことのできない情報が少なからず存在した。そのため、安全かつ柔軟に利用できる生成 AI 環境を原子力機構内に整備することが求められていた。

このような背景を踏まえ、我々は原子力機構が保有するスーパーコンピュータ（以下、「スパコン」という）などの既存の計算資源とオープンソースソフトウェアを活用し、セキュリティ上の懸念を払拭したオンプレミス環境において、安全かつ容易に利用できる生成 AI 基盤を構築し、展開した。

2. 生成 AI の概要

2.1 AI 研究の歴史

今日では、AI（Artificial intelligence、人工知能）という言葉を耳にしない日はないほど、社会全体でかつてない AI ブームが巻き起こっている。

AI について、AI 事業者ガイドライン[1]では次のように示されている。

> AI は Artificial Intelligence（人工知能）を意味し、1956 年にダートマス会議で初めて使用された言葉であるとされている。AI は未だ確立された定義は存在しないが、「人工」・「知能」とあるように、人間の思考プロセスと同じような形で動作するコンピュータープログラム、コンピューター上で知的判断を下せるシステム等を指す。

AI 研究の理論的基盤は、1950 年にイギリスの数学者アラン・チューリングが発表した論文『Computing Machinery and Intelligence』において示され、その後、1956 年にアメリカの計算機科学者ジョン・マッカーシーがダートマス会議において初めて AI という語を用いたとされている。

AI は以来､4 度のブームと停滞を繰り返しながら発展を遂げてきた｡AI の歴史を図 2-1 に示す。

図 2-1　AI の歴史

（総務省「平成 28 年版情報通信白書」[2] を基に作成）

(1) 第 1 次 AI ブーム（1950 年代～1960 年代）

第 1 次 AI ブームは、論理的な「推論」と「探索」により最適な解を導く記号処理型 AI が中心であり、人間の知的思考の模倣が試みられた。しかし、当時のコンピュータの処理能力や記憶容量は現代のコンピュータと比較して極めて低く、AI が処理できるのは単純なパズルや数理的問題にとどまった。複雑な現実問題への応用は困難であり、ブームはやがて沈静化した。

(2) 第 2 次 AI ブーム（1980 年代～1990 年代前半）

第 2 次 AI ブームでは、特定の分野に特化し、専門家と同等の知識を扱うエキスパートシステムが大きな注目を集めた。エキスパートシステムは、専門家の知識をコンピュータに蓄積した「知識ベース」と、それを用いて問題を解決する「推論エンジン」によって構成されていた。しかし、膨大な知識を人手で入力・更新し続ける必要があったため、作業量が次第に限界に達し、システムの維持が困難になった。さらに、当時の計算資源にも限界があったことから、ブームは再び終息していった。

(3) 第 3 次 AI ブーム（2000 年代～）

第 3 次 AI ブームは、2000 年代以降、コンピュータ性能の向上とインターネットの普及によるビッグデータと呼ばれる大量のデータの活用を背景に起こった。大量のデータを基に AI 自身が知識を習得する機械学習が進化し、さらにその一種であるディープラーニング（深層学習）技術の登場により、画像認識や音声認識、機械翻訳などで人間を上回る精度を達成した。この時期の AI は、与えられたデータを基に分類、分析、予測を行う「判断する AI」としての性質が強い。

(4) 第 4 次 AI ブーム（2020 年代～）

第 4 次 AI ブームは、生成 AI の登場によって引き起こされたものであり、2022 年の OpenAI 社による対話型 AI「ChatGPT」の発表を契機として始まったとされている。生成 AI に関する説明は次項で述べる。

2.2　生成 AI と LLM

生成 AI について、令和 6 年版情報通信白書[3)]では次のように説明している。

> 「生成 AI」は、テキスト、画像、音声などを自律的に生成できる AI 技術の総称であり、2022 年の OpenAI による対話型 AI “Chat GPT”の発表を契機に、特に注目された分野である。

従来の AI は、大量のデータからパターンを学習し、未知のデータに対して「どちらに分類されるか」や「次に何が起こるか」を判断・推定する能力を有していたのに対し、生成 AI は、その延長線上にありながらも、自ら新しいコンテンツを生成できる点で大きく異なる。

また、大量のテキストデータから言語構造やパターンを理解する大規模言語モデル（Large Language Model、以下、「LLM」という）や、拡散モデルなどの画像生成モデルなどと組み合わせることで、テキスト・画像・音声・プログラムコードなど、人間の創造活動に類似した成果物を生成できる。

AI、機械学習、ディープラーニング、生成 AI の関係を図 2-2 に示す。

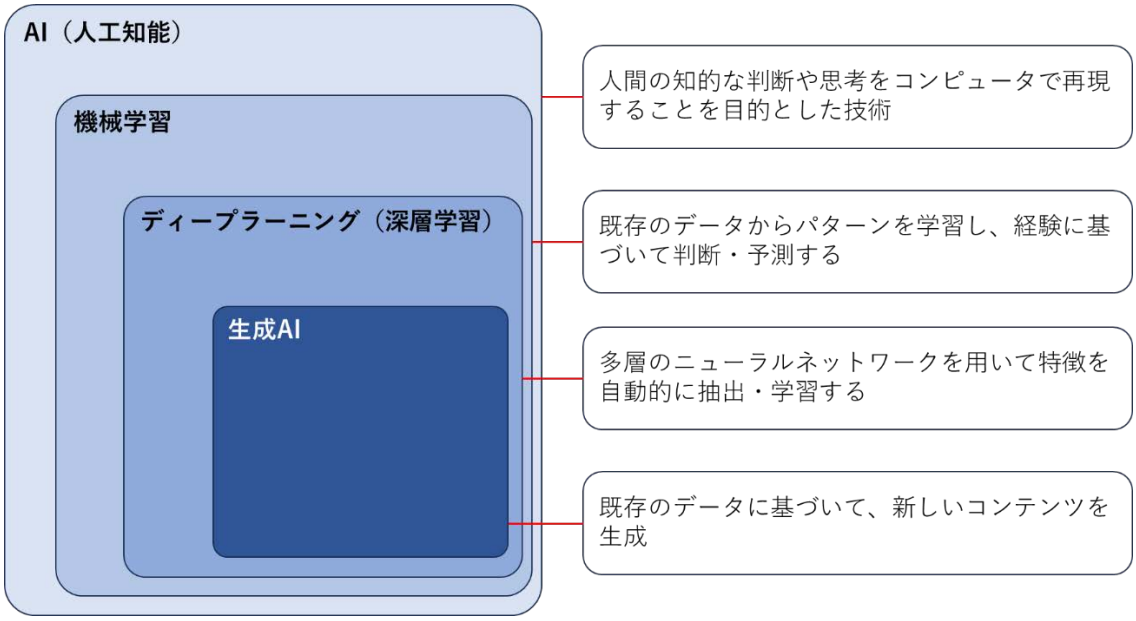

図 2-2　AI、機械学習、ディープラーニング、生成 AI の関係

生成 AI は、適切に活用することで大きな恩恵を享受できる一方、リスクや弱みも内包している。生成 AI の代表的な負の側面として、「ハルシネーション」と呼ばれる現象が挙げられる。ハルシネーションとは、実在しない情報や誤った内容を、あたかも事実であるかのようにもっともらしく回答してしまう現象である。したがって、正確性が求められる場面においては、人間による検証と判断が不可欠となる。

また、生成 AI へ指示を与える際の入力文（以下、「プロンプト」という）に個人情報や秘密情報が含まれている場合、それらの情報がモデルの学習や再利用の過程で外部に漏洩するリスクがある。入力データを学習に利用しない旨を明示している生成 AI サービスも存在するが、利用形態や提供事業者によって取り扱いは異なるため、情報管理上のリスクは依然として残る。このため、生成 AI を利用する際には、取り扱う情報の機密区分に応じた利用ルールの策定が不可欠であり、技術的に入力そのものを防ぐ仕組みの実装は困難な現状である。

生成 AI のうち、最も一般的である ChatGPT をはじめとしたテキスト生成 AI では、人間の言葉である自然言語による処理が可能であり、自然言語処理は LLM によって実現されている。LLM は、膨大なテキストデータを学習し、単語間の統計的関係を把握することで、与えられた文脈から適切な語句や文を予測・生成する言語モデルである。代表的な LLM として、OpenAI 社の GPT、Meta 社の Llama などが挙げられる。

一方で、テキスト生成 AI には LLM が学習していない情報に関する回答は精度が低下するという欠点がある。この欠点を補う技術として、ファインチューニングと RAG（Retrieval-Augmented Generation）がある。

(1) ファインチューニング

ファインチューニングは、あらかじめ学習されたモデルに追加学習を行うことで、特定分野の知識や固有の表現を習得させる技術である。追加学習により回答精度の低下を防ぐには、追加データの量と質が重要となる。ファインチューニングの概要図を図 2-3 に示す。

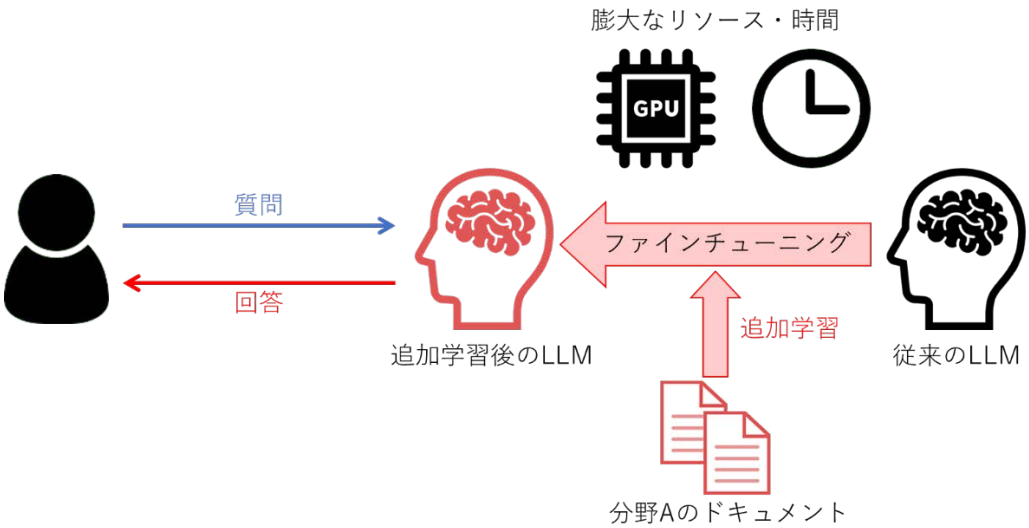

図 2-3　ファインチューニングの仕組み
（出典：IPA「テキスト生成 AI の導入・運用ガイドライン」）[4)]

(2) RAG

RAG は、プロンプトに関連するデータを外部データベース（ベクトル DB）から検索し、その結果をプロンプトに付加して LLM に回答を生成させる技術である。RAG においてもファインチューニングと同様に、用いるデータの量と質が重要となる。RAG の概要図を図 2-4 に示す。

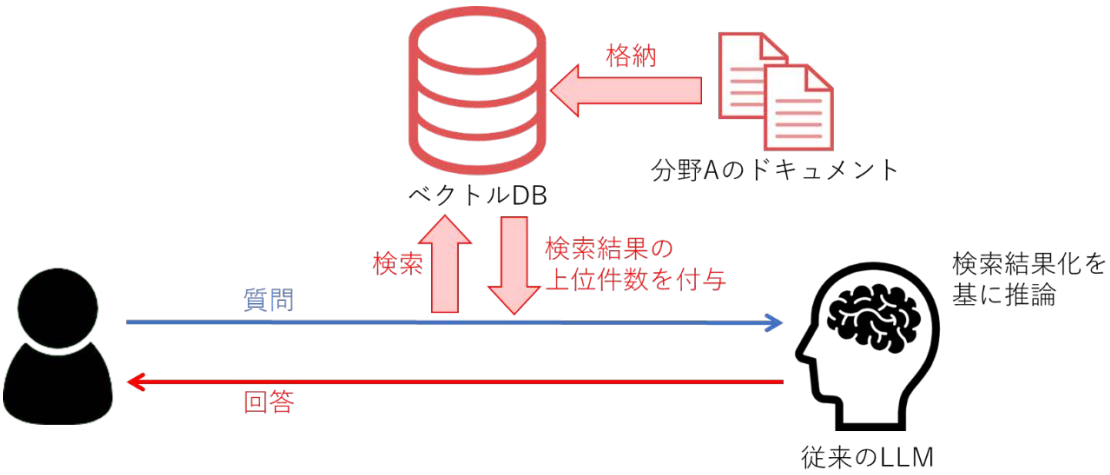

図 2-4　RAG の仕組み

（出典：IPA「テキスト生成 AI の導入・運用ガイドライン」）[4)]

2.3　生成 AI の種類

生成 AI は、生成対象の種類によってテキスト生成型、画像生成型、動画生成型、音声生成型などに分類される。それぞれ異なる学習データと生成手法を用いており、目的に応じて多様な応用が進んでいる。それぞれの特徴と代表的なサービスについて以下に述べ、各生成 AI と LLM の位置付けを図 2-5 に示す。

(1) テキスト生成 AI

テキスト生成 AI は最も一般的な生成 AI であり、LLM を基盤として人間の文章と遜色のない自然な文章を生成することができる。代表的な例として、OpenAI 社の GPT シリーズ、Google 社の Gemini、Microsoft 社の Copilot などが挙げられる。これらは文章生成や要約、翻訳、質問応答など、幅広い用途で活用されている。

(2) 画像生成 AI

画像生成 AI は、拡散モデル（Diffusion Model）などの技術を用いて、テキストから高品質な画像を生成する。代表的な例として、Stability AI 社の Stable Diffusion、OpenAI 社の DALL-E などが挙げられる。近年では、デザインや広告、ゲーム開発などの分野で活用が急速に拡大している。

(3) 動画生成 AI

動画生成 AI は、静止画やテキストから連続した映像を生成する技術である。代表的な例として、OpenAI 社の Sora、Google 社の Veo などが挙げられる。映像制作や広告生成など、エンターテインメント分野を中心に実用化が期待されている。

(4) 音声生成 AI

音声生成 AI は、人間の音声を模倣した自然な発話や、特定の話者の声質を再現する技術である。代表的なサービスとして、Microsoft 社の VALL-E、Meta 社の Voicebox などが挙げられる。音声アシスタントやナレーション、自動応答システムなどへの応用が進んでいる。

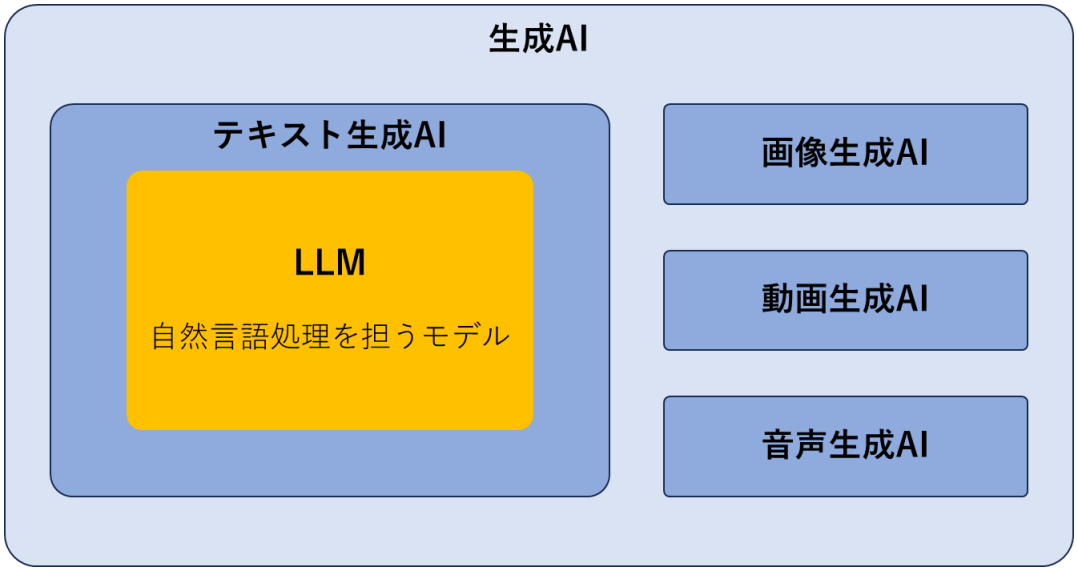

図 2-5　各種生成 AI と LLM の位置付け

（出典：IPA「テキスト生成 AI の導入・運用ガイドライン」）[4)]

2.4　生成 AI の業務への活用

生成 AI は、テキスト、画像、動画、音声など多様なコンテンツを生成できることから、業務効率化から研究開発、クリエイティブ分野まで幅広い分野で応用が進んでいる。生成 AI の代表的な業務への活用について以下に述べる。

(1) 一般業務

生成 AI は、文書作成、要約、翻訳、メール文作成など、日常業務の効率化に広く活用されている。特に、LLM を活用した対話型のテキスト生成 AI は、文脈を踏まえた自然な応答を行うことができ、従来の検索やテンプレート処理では困難だった柔軟な対応を可能にしている。また、社内 FAQ の作成、議事録の要約、問い合わせ対応など、定型業務の自動化にも効果を発揮しており、業務負担軽減に寄与している。

(2) 研究開発支援

研究開発分野においても、生成 AI の導入が進んでいる。論文の要約や翻訳、プログラムコード作成補助、データ解析の支援など、知的作業の補助としても活用されている。GitHub Copilot などのコード補完 AI が開発効率を大きく向上させている。また、研究領域においては、Deep Research などの文献レビューや実験計画立案支援への応用も試みられており、専門知識の習得や試行錯誤の過程を支援することで、研究開発における生産性向上に寄与している。

(3) 創作的業務・コンテンツ制作

画像・動画・音声生成 AI は、デザイン、広告、映像制作などのクリエイティブな分野で広く活用されている。生成 AI を用いることで、短時間で複数の試作案を作成し、企画段階での発想支援や制作コストの削減を実現している。また、自動字幕付与や動画生成、音声合成によるナレーション生成など、表現手段の拡張にも寄与している。このように、生成 AI は人間の創造性を代替するのではなく、創造過程を支援する知的パートナーとしての役割を担い始めている。

2.5 原子力機構における生成 AI の利用状況

生成 AI が世間で大きな注目を集める中、原子力機構内においても業務活用への関心と期待が高まり、生成 AI を業務に取り入れたいという要望が多数寄せられた。こうした状況を受け、原子力機構システム計算科学センターでは、外部の有識者を招いた原子力機構内向けのセミナーを開催するなど、生成 AI の利用促進を目的とした取り組みを実施した。セミナーには各回とも多くの参加者が集い、受講後のアンケートでは「セミナーをきっかけに生成 AI の利用を決めた」といった肯定的な意見が多く見られた。

実際に寄せられた主なニーズとしては、外部提出書類の誤記チェック、論文の翻訳、プログラミング支援、技術データを活用した Q&A ボット、メール文の添削、アイデア創出など、多岐にわたっている。

原子力機構における生成 AI の利用は、ChatGPT 等の生成 AI の業務利用に関する申合せ [5]などに基づき、利用ルールが定められており、利用が認められているクラウド型の外部生成 AI サービスであれば、所定の申請手続きを経て許可を得ることで使用可能となる。ただし、利用に際しては原則として秘密情報を入力しないことが前提である。

一方で、業務上のニーズを満たすためには、秘密情報や内部情報の取り扱いが避けられない場合も少なくない。このため、生成 AI への関心は高いものの、セキュリティ上の懸念から利用を控えるケースや、利用者が不安を感じるケースもあり、結果として生成 AI の利用が広く定着するには至っていない状況であった。

3. オンプレミス生成 AI 基盤の構築

前章で述べたとおり、生成 AI に関する業務上のニーズは高い一方で、秘密情報を扱う際のセキュリティ上の懸念から、活用状況は限定的であった。そこで、これらの課題を解消し、安全に利用できる環境を整備するため、オンプレミス環境に生成 AI 基盤を構築した。本章では、その構築にあたり実施した内容について述べる。

3.1 全体構成

生成 AI 基盤は、原子力機構で稼働しているスパコンの一部を利用して構築した。現在のスパコンは 2020 年 12 月に運用を開始しており、システム全体として総理論演算性能 12.6PFLOPS を有する。また、スパコンは原子力機構のネットワーク内に配置されており、外部との通信は制限されている。スパコンの構成を図 3-1 に、主な性能を表 3-1 に示す。

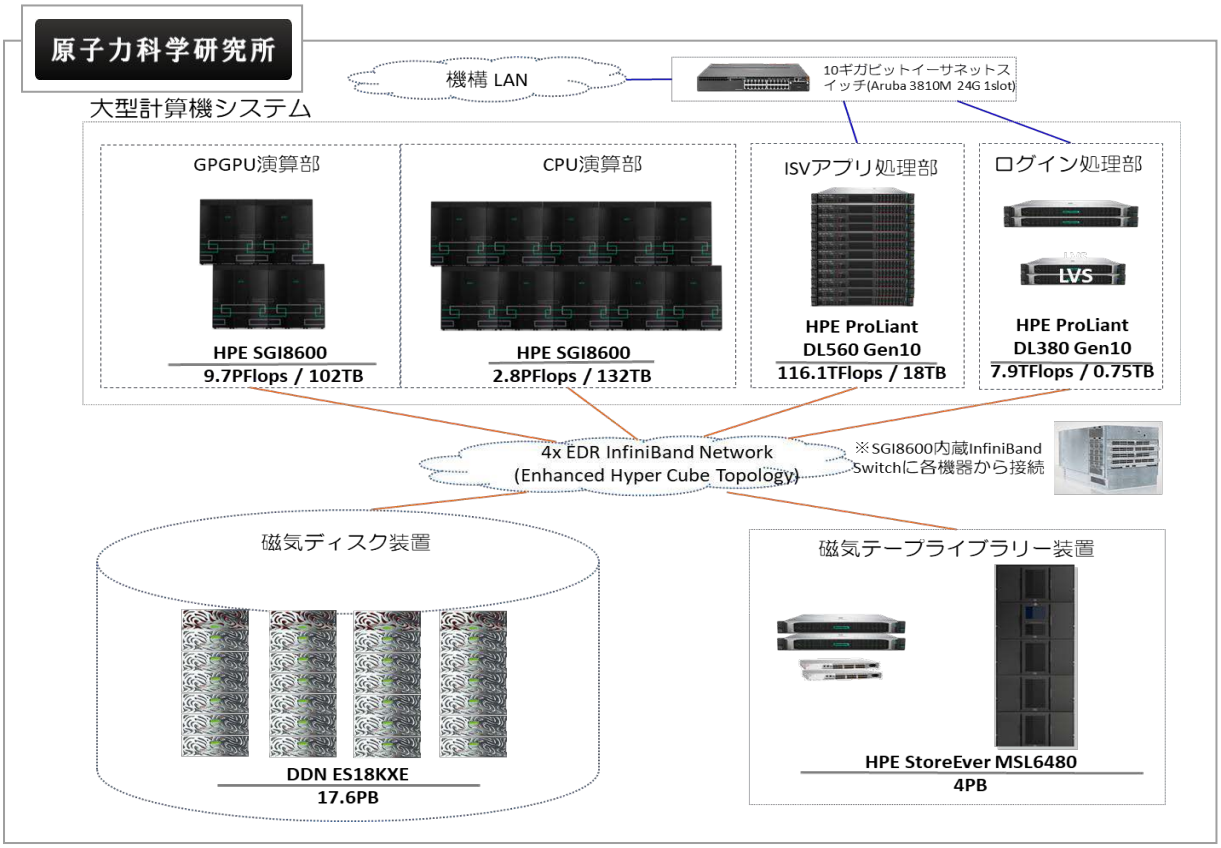

図 3-1　スパコンの構成

（出典：原子力機構「令和 6 年度大型計算機システム利用による研究成果報告集」）[6)]

表 3-1　スパコンの主な性能

（出典：原子力機構「令和 6 年度大型計算機システム利用による研究成果報告集」）[6)]

	GPGPU 演算部 HPE SGI8600		CPU 演算部 HPE SGI8600	ISV アプリ処理部 HPE ProLiant DL540 Gen10	ログイン処理部 HPE ProLiant DL380 Gen10
タイプ	スカラ		スカラ	スカラ	スカラ
総演算性能（TFLOPS）	1.253	8.486	2.801	116.12	7.936
総主記憶容量（TB）	102	-	132.375	18	0.768
コア数／ノード	48		40	112	40
ノード数	272		706	12	2
CPU	Intel Xeon Gold 6248R 24core 3.0GHz ×2CPU	NVIDIA Tesla V100 SXM2 32GB Memory x 4	Intel Xeon Gold 6242R 20core 3.1GHz ×2CPU	Intel Xeon Platinum 8280 28core 2.7GHz ×4CPU	Intel Xeon Gold 6242R 20core 3.1GHz ×2CPU
演算性能／コア（GFLOPS）	96.0	162.5	99.2	86.4	99.2
メモリ／ノード（GB）	384	128	192	1536	384
ノード間通信性能	片方向 50GB/s（全二重）		片方向 50GB/s（全二重）	片方向 25GB/s（全二重）	片方向 25GB/s（全二重）
OS	Red Hat Enterprise Linux 7.7		Red Hat Enterprise Linux 7.8	Red Hat Enterprise Linux 7.8	Red Hat Enterprise Linux 7.8
コンパイラ	Fortran C/C++		Fortran C/C++	Fortran C/C++	Fortran C/C++
バッチシステム	PBS Professional		PBS Professional	PBS Professional	PBS Professional
ファイルシステム	DDN EXAScaler（Lustre）		DDN EXAScaler（Lustre）	DDN EXAScaler（Lustre）	DDN EXAScaler（Lustre）

令和 6 年 3 月末 現在

今回、ISV アプリ処理部内のノード（以下、「ISV ノード」という）に構築した仮想化基盤（以下、「スパコンクラウド」という）上に仮想サーバ（以下、「生成 AI アプリサーバ」という）を立て、GPGPU 演算部内のノード（以下、「GPU ノード」という）を組み合わせることで生成 AI 基盤を構築した。

生成 AI アプリサーバには生成 AI アプリ開発基盤ツールである Dify を導入し、生成 AI の WEB アプリケーション部分を構築した。また、GPU ノードには LLM 実行基盤ツールである Ollama を導入した。GPU ノードは、1 ノードあたり NVIDIA Tesla V100 SXM2 32GB を 4 基搭載しており、本構築ではこれを 2 ノード利用した。

GPU ノードは、生成 AI アプリサーバや原子力機構ネットワーク上の PC などから直接接続することはできず、リバースプロキシ機能を備えた ISV ノードを介してのみ接続可能な閉域構成となっている。この構成により、生成 AI アプリサーバと GPU ノードとの通信はすべてリバースプロキシ機能を備えた ISV ノードを経由して行われ、生成 AI アプリサーバ上の Dify から GPU ノード上の Ollama を呼び出す形で LLM を実行している。

利用者が生成 AI アプリを利用する際には、PC のブラウザを用いて生成 AI アプリサーバ上で稼働する Dify に接続する。入力されたプロンプトは、生成 AI アプリサーバから ISV ノードを経由して GPU ノードへ転送され、GPU ノード上の Ollama が LLM を実行する。結果は生成 AI アプリサーバに返送され、Dify によって整形されたのち、利用者のブラウザ画面に応答として表示される。

生成 AI 基盤の概要図を図 3-2 に示す。

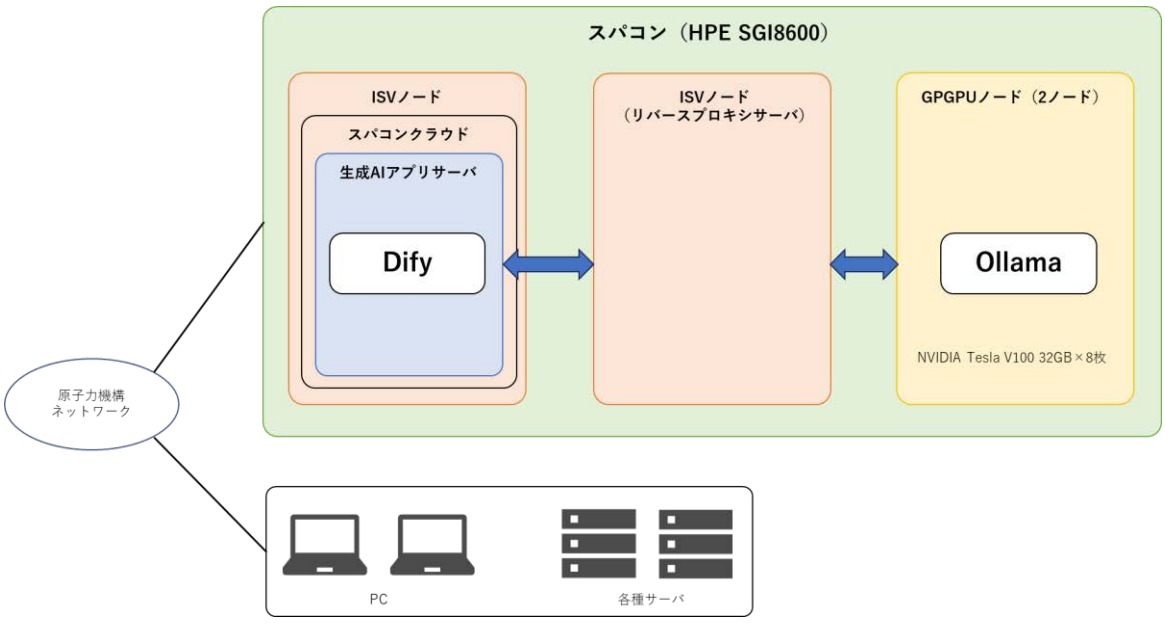

図 3-2　生成 AI 基盤の概要図

3.2　生成 AI アプリ開発基盤ツール（Dify）の構築

生成 AI アプリサーバの主な構成を表 3-2 に示す。

表 3-2　生成 AI アプリサーバの主な構成

項目	内容
OS	Rocky Linux release 8.10 (Green Obsidian)
カーネル	4.18.0-553.el8_10.x86_64
CPU	6 コア
メモリ	32GB
ディスク	75GB（＋NFS 領域 1TB）

今回、上記構成の生成 AI アプリサーバに、生成 AI アプリ開発基盤ツールである Dify を構築した。Dify はプログラミングなどの高度な知識がなくても直感的に生成 AI アプリを作成できる、ローコード型の開発ツールである。Dify にはクラウド版とローカル版が存在し、ローカル版であるコミュニティ版はオープンソースソフトウェアとして公開されている。

Dify の採用にあたっては、LangChain や n8n などの類似ツールとの比較・検討を行った。その結果、Dify は構築および操作が容易であり、短期間で導入・展開できる点で優れている。また、オープンソースソフトウェアであるため、運用中に発生した課題は自ら解決する必要があるが、Dify はユーザコミュニティが活発で情報も豊富であることから、技術的な調査を行いやすい環境が整っている点も利点である。これらの理由から、生成 AI アプリ開発基盤として Dify を採用した。

生成 AI アプリサーバに Dify を構築した際の手順を以下に説明する。

(1) Docker のインストール

Dify の導入に先立ち、コンテナ型仮想化ソフトである Docker を導入する。Dify は WEB アプリ、バックエンド API、データベースなど複数のコンポーネントによって構成されており、これらを Docker 上で実行することで、一元的な管理が可能になる。

①# dnf config-manager --add-repo=https://download.docker.com/linux/centos/docker-ce.repo を実行し、Docker の公式リポジトリを追加する。

②# dnf repolist -v を実行し、Docker の公式リポジトリ（docker-ce-stable）が追加されたことを確認する（図 3-2-1）。

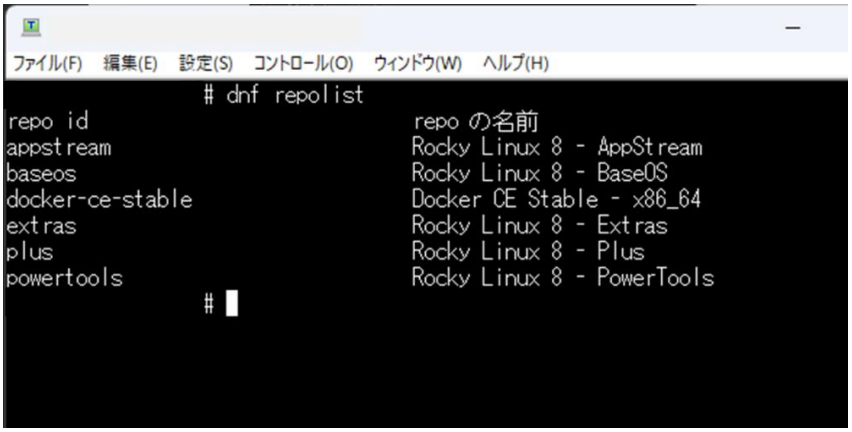

図 3-2-1　Docker の公式リポジトリ（docker-ce-stable）の存在を確認

③# dnf install docker-ce-3:26.1.3-1.el8 を実行してインストールする。

④# docler -v を実行し、Docker のバージョンが表示されることを確認する（図 3-2-2）。

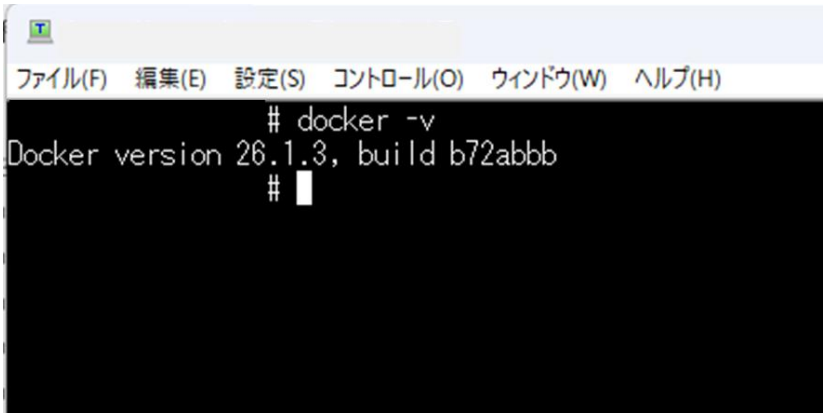

図 3-2-2　Docker のバージョンが表示されることを確認

⑤# systemctl start docker.service を実行し、Docker サービスを起動する。

⑥# systemctl status docker.service を実行し、ステータスが[active (running)]であることを確認する（図 3-2-3）。

⑦# systemctl enable docker.service を実行し、Docker が OS 起動時に自動起動するよう設定する。

⑧# systemctl is-enabled docker.service を実行し、enabled と表示されることを確認する（図 3-2-3）。

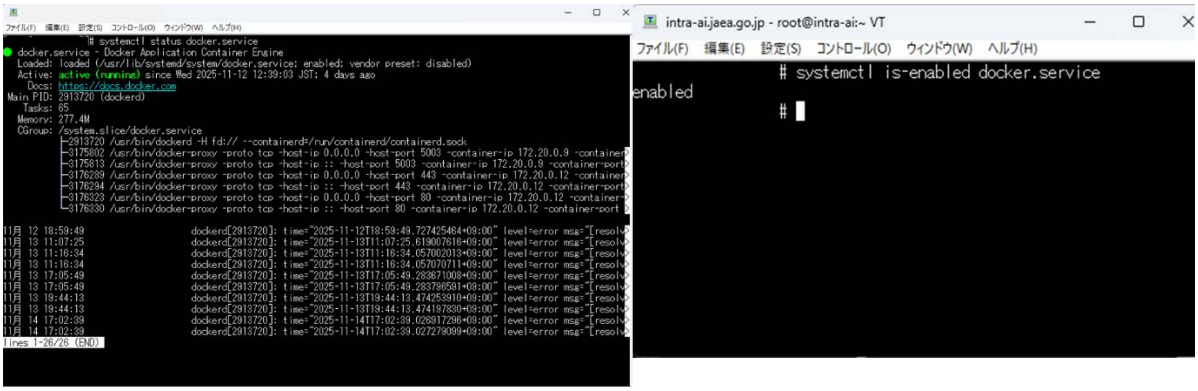

図 3-2-3　Docker のステータス

(2) Dify のインストール

以下の手順で Dify のインストールを実施する。

①# dnf install -y git を実行し、git をインストールする。

②# git clone https://github.com/langgenius/dify.git を実行し、Dify のリポジトリをクローンする。

③# cd /dify/docker を実行し、Dify の Docker 実行用ディレクトリに移動する。

④# cp -p .env.example .env を実行し、環境ファイル（.env）をコピーして作成する。

⑤# systemctl restart docker.service を実行し、Docker サービスを再起動する。

⑥# docker compose up -d を実行し、Dify 関連の Docker コンテナを起動する。

⑦# docker ps を実行し、Dify 関連のコンテナが起動し、STATUS が Up になっていることを確認する（図 3-2-4）。

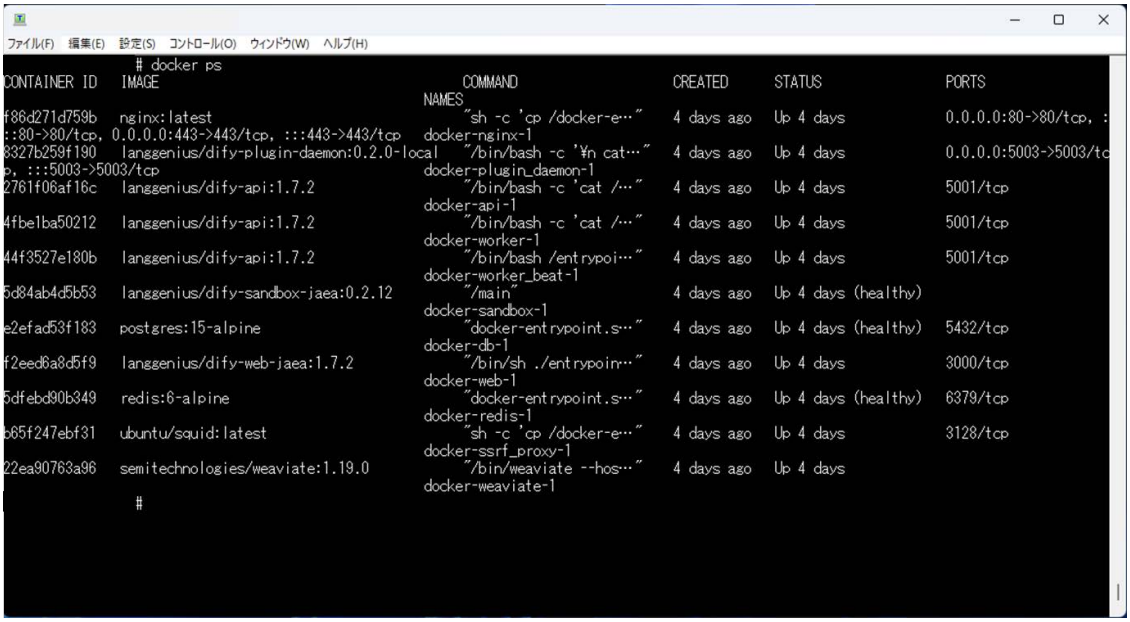

図 3-2-4　Docker のステータス

⑧WEB ブラウザで、“http://生成 AI アプリサーバの IP アドレス”にアクセスし、Dify の初期画面が表示されれば、Dify のインストールは正常に完了している（図 3-2-5）。

図 3-2-5　Dify の初期画面

3.3　LLM 実行基盤ツール（Ollama）の構築

ここでは、GPU ノードに構築した LLM 実行基盤である Ollama の構築手順について説明する。Ollama は、ローカル環境で LLM を実行するためのツールであり、オープンソースソフトウェアとして公開されている。Ollama は LM Studio などの類似ツールと比較してカスタマイズ性に優れており、CUI による操作が可能であることから、運用面での自動化や機能拡張が容易であるという利点を有する。また、ChatGPT と互換性のある API を採用しており、Dify との連携にも適している。これらの理由から、LLM 実行基盤ツールとして Ollama を採用した。Ollama で実行可能な LLM として、Meta 社の Llama シリーズ、Google 社の Gemma シリーズ、OpenAI 社の gpt-oss などがある。

GPU ノードの主な構成は表 3-1 に示すとおりであるが、OS については 2025 年 11 月時点において Red Hat Enterprise Linux 8.8 を使用している。

以下に、GPU ノードに Ollama を構築した際の手順を示す。

なお、外部サイトへのアクセスが必要な箇所については、GPU ノードが外部ネットワークに接続されていないため、ディスク領域を共有する別ノード（以下、「ログインノード」という）上でコマンドを実行している。また、GPU ノードおよびログインノードは管理者権限が付与されていない環境であるため、一般権限でコマンドを実行している。さらに、GPU ノードおよびログインノードは csh/tcsh を標準シェルとして利用する環境である。

(1) Ollama のインストール

以下の手順で Ollama をインストールする。

①$ mkdir -p $HOME/ollama を実行し、Ollama 用のディレクトリを作成する。

②ログインノード上で、$ curl -L https://ollama.com/download/ollama-linux-amd64 -o $HOME/ollama/bin/ollama を実行し、Ollama をダウンロードする。

③$ chmod +x $HOME/ollama/bin/ollama を実行し、実行権限を付与する。

④GPU ノード上で$ vi /$HOME/.cshrc を実行し、環境変数を設定する（図 3-3-1）。

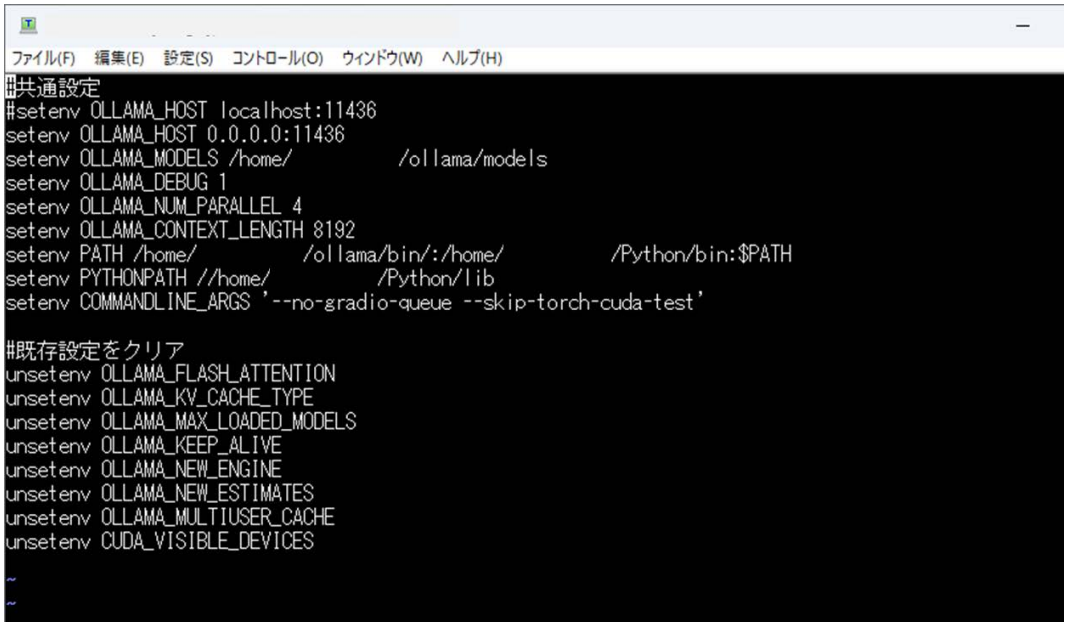

図 3-3-1　GPU ノードの環境変数設定

⑤$ ollama --version を実行し、Ollama のバージョンが表示されることを確認する（図 3-3-2）。

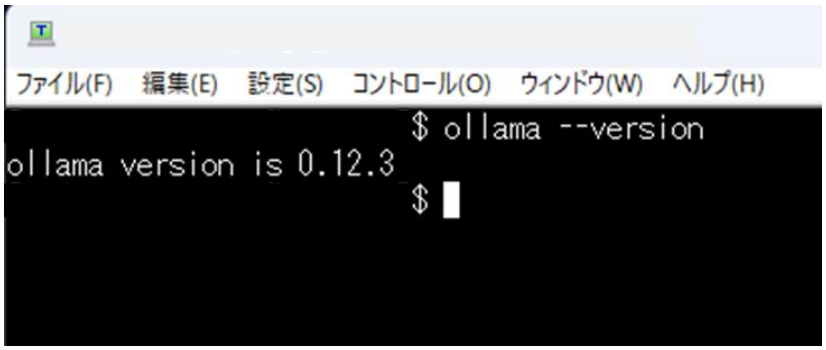

図 3-3-2　Ollama のバージョンが表示されることを確認

(2) Ollama の起動

①Ollama 起動用シェルスクリプトを作成する（図 3-3-3）。

```
ファイル(F)  編集(E)  設定(S)  コントロール(O)  ウィンドウ(W)  ヘルプ(H)
#!/bin/csh

source /home/          cshrc
set datetime = `date "+%Y%m%d%H%M%S"`

#GPU1
setenv OLLAMA_HOST 0.0.0.0:11436
setenv CUDA_VISIBLE_DEVICES 0
setenv OLLAMA_CONTEXT_LENGTH 8192
setenv OLLAMA_NUM_PARALLEL 4
nohup /home/          /ollama/bin/ollama serve >& /home/          /ollama/log/ollama1_${datetime}.log &

#GPU2
setenv OLLAMA_HOST 0.0.0.0:11437
setenv CUDA_VISIBLE_DEVICES 1
setenv OLLAMA_CONTEXT_LENGTH 8192
setenv OLLAMA_NUM_PARALLEL 4
nohup /home/          /ollama/bin/ollama serve >& /home/          /ollama/log/ollama2_${datetime}.log &

#GPU3
setenv OLLAMA_HOST 0.0.0.0:11438
setenv CUDA_VISIBLE_DEVICES 2
setenv OLLAMA_CONTEXT_LENGTH 8192
setenv OLLAMA_NUM_PARALLEL 4
nohup /home/          /ollama/bin/ollama serve >& /home/          /ollama/log/ollama3_${datetime}.log &

#GPU4
setenv OLLAMA_HOST 0.0.0.0:11439
setenv CUDA_VISIBLE_DEVICES 3
#setenv OLLAMA_CONTEXT_LENGTH 32768
setenv OLLAMA_CONTEXT_LENGTH 8192
setenv OLLAMA_NUM_PARALLEL 4
nohup /home/          /ollama/bin/ollama serve >& /home/          /ollama/log/ollama4_${datetime}.log &
~
~
```

図 3-3-3　Ollama の起動用シェルスクリプト

②①で作成した Ollama 起動用シェルスクリプトを実行し、Ollama サービスをデーモンモードで起動する。

③$ curl http://localhost:11436～11439 を実行し、“Ollama is running”と表示されれば、Ollama のインストールと起動は正常に完了している（図 3-3-4）。

```
ファイル(F)  編集(E)  設定(S)  コントロール(O)  ウィンドウ(W)  ヘルプ(H)
                        curl http://localhost:11436
Ollama is running            ~/ollama]$
                        curl http://localhost:11437
Ollama is running            ~/ollama]$
                        curl http://localhost:11438
Ollama is running            ~/ollama]$
                        curl http://localhost:11439
Ollama is running            ~/ollama]$
                ~/ollama]$
```

図 3-3-4　Ollama の起動確認

④Ollama を停止する際は、$ ps -ef | grep -i ollama を実行して Ollama のプロセス ID（PID）を確認し、$ kill [Ollama の PID]で停止する（図 3-3-5）。

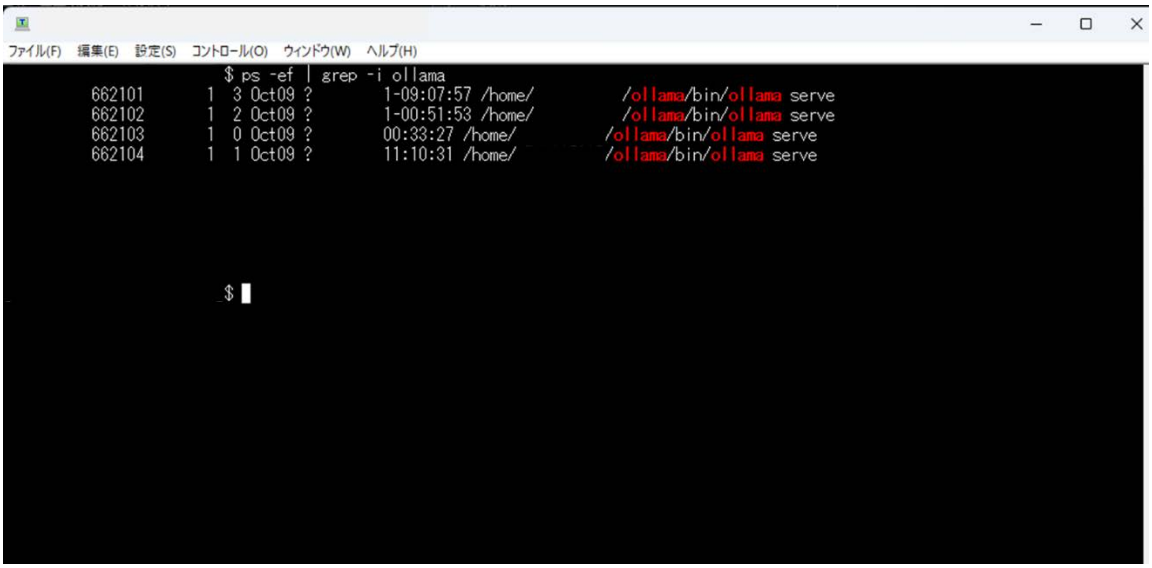

図 3-3-5　Ollama の PID 確認

(3) Ollama への LLM 追加

ここでは、例として Llama3.2:3b を Ollama で実行する手順を説明する。

①ログインノードでの Ollama 起動用シェルスクリプトを作成する（図 3-3-6）。

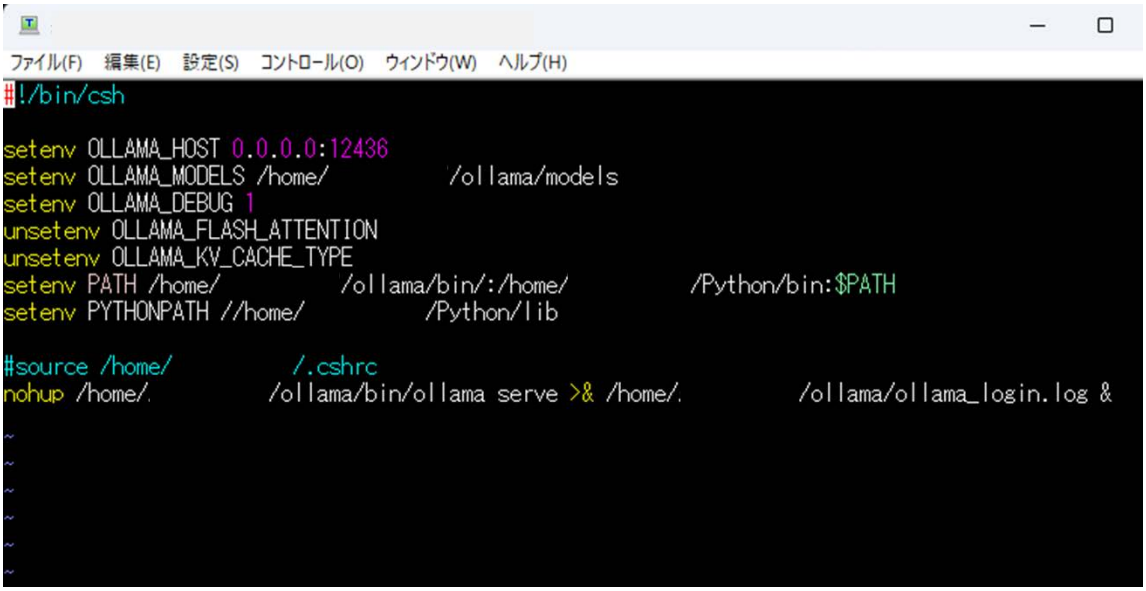

図 3-3-6　ログインノードでの Ollama 起動用シェルスクリプト

②①で作成した Ollama 起動用シェルスクリプトを実行し、Ollama サービスをデーモンモードで起動する。

③$ ollama pull llama3.2:3b を実行し、LLM を追加する。

④$ ollama list を実行し、llama3.2:3b が追加されたことを確認する（図 3-3-7）。

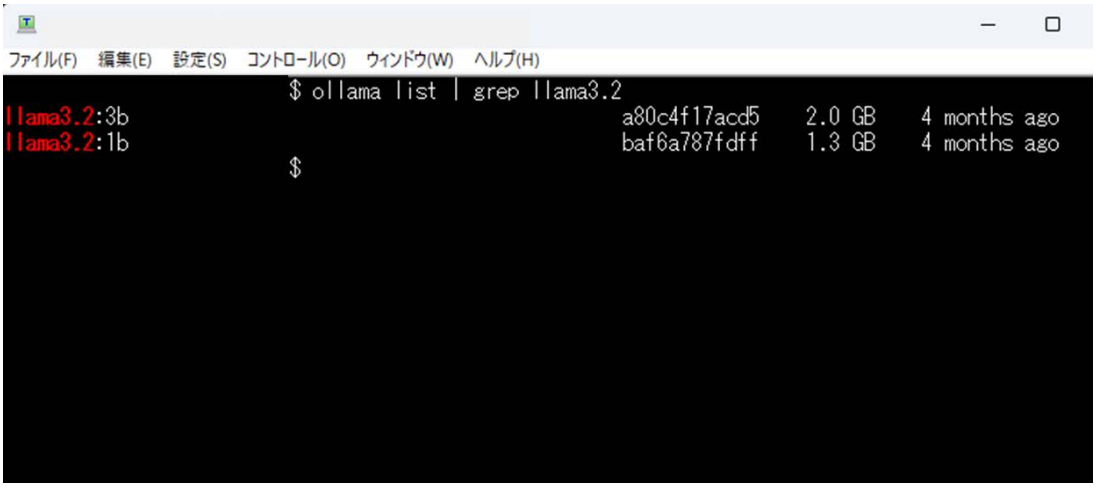

図 3-3-7　llama3.2:3b が追加されたことを確認

⑤GPU ノード上で$ ollama run llama3.2:3b を実行し、プロンプトを入力して出力が返ってくることを確認する（図 3-3-8）。

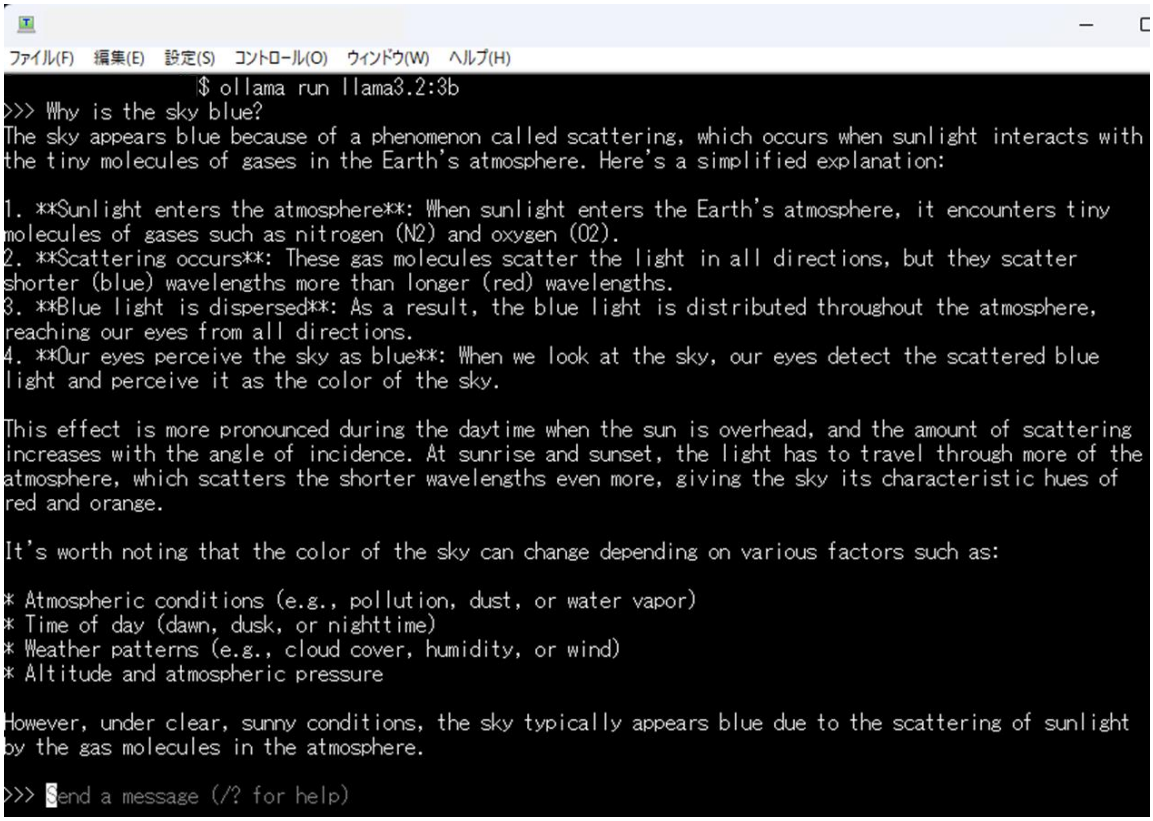

図 3-3-8　llama3.2:3b を実行し出力が返ってくることを確認

3.4　生成 AI アプリの作成

生成 AI アプリサーバに構築した Dify から、GPU ノード上で稼働する Ollama の LLM（llama3.2:3b）を利用する生成 AI アプリの例として、チャットボットの作成手順を以下に示す。

(1) Dify で管理者アカウントを作成しログイン後、スタジオ画面にてアイコンをクリックし、[設定]をクリックする（図 3-4-1）。

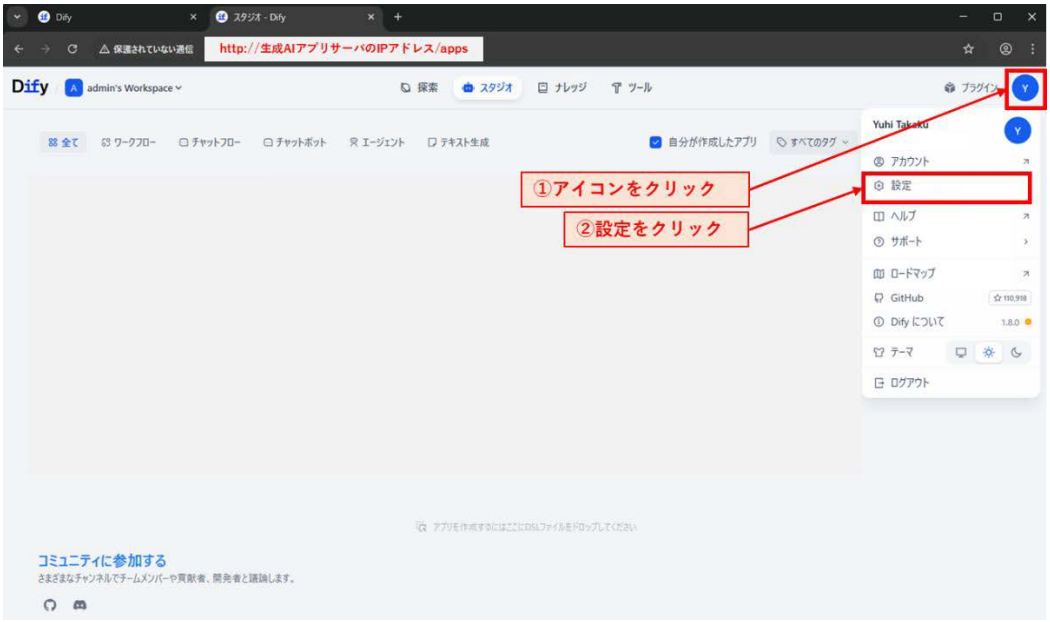

図 3-4-1　Dify の設定画面を開く

(2) [モデルプロバイダー]から[Ollama]をインストールする。（図 3-4-2）。

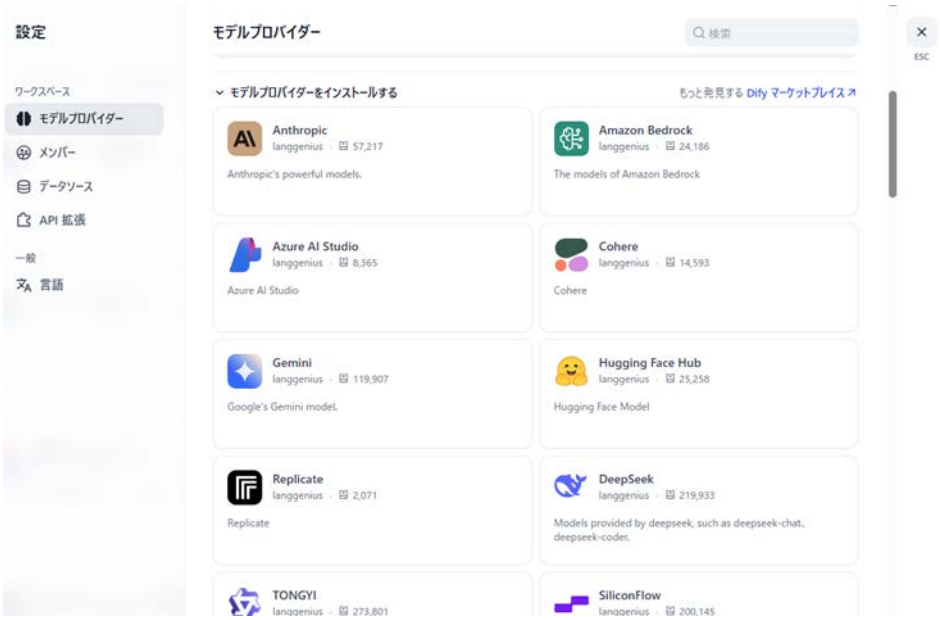

図 3-4-2　Ollama のモデルプロバイダーをインストールする

(3) [モデルの追加]をクリックし、LLM の情報を入力して[追加]をクリックする。（図 3-4-3）。

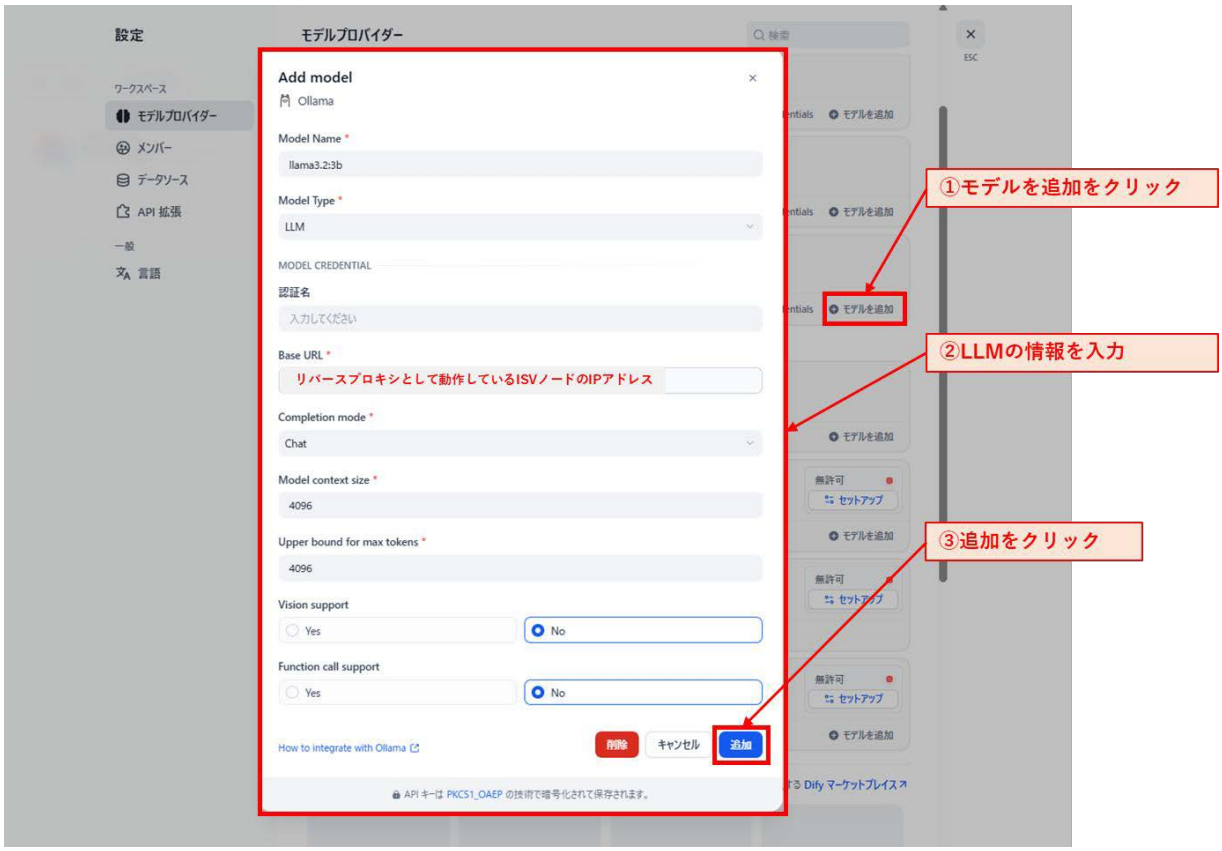

図 3-4-3　LLM の情報を入力する

(4) モデルの追加が完了したら(1)のスタジオ画面に戻り、[最初から作成]をクリックする（図 3-4-4）。

図 3-4-4　最初から作成をクリックする

(5) [初心者向けの基本的なアプリタイプ]をクリックして開き、[チャットボット]をクリックする。
　チャットボットをクリック後、[アプリのアイコンと名前]に作成するチャットボットの名前を入力して[作成する]をクリックする（図 3-4-5）。

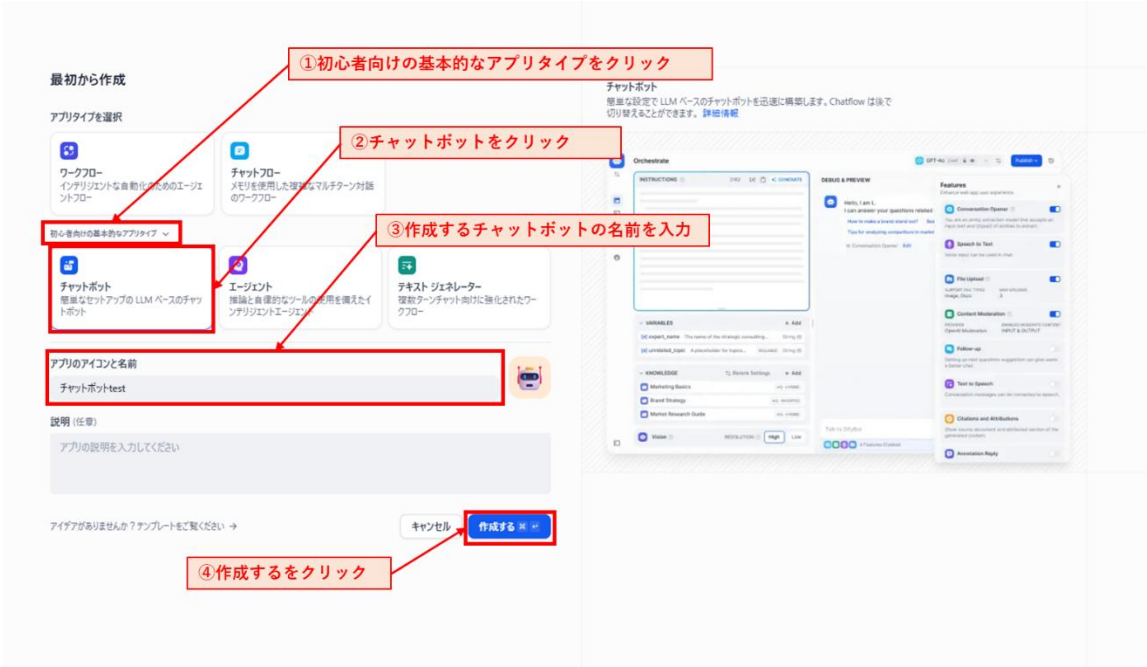

図 3-4-5　作成するをクリックする

(6) オーケストレーションの設定画面に遷移するので、画面右上 Ollama のアイコンをクリックし、[llama3.2:3b]を選択する（図 3-4-6）。

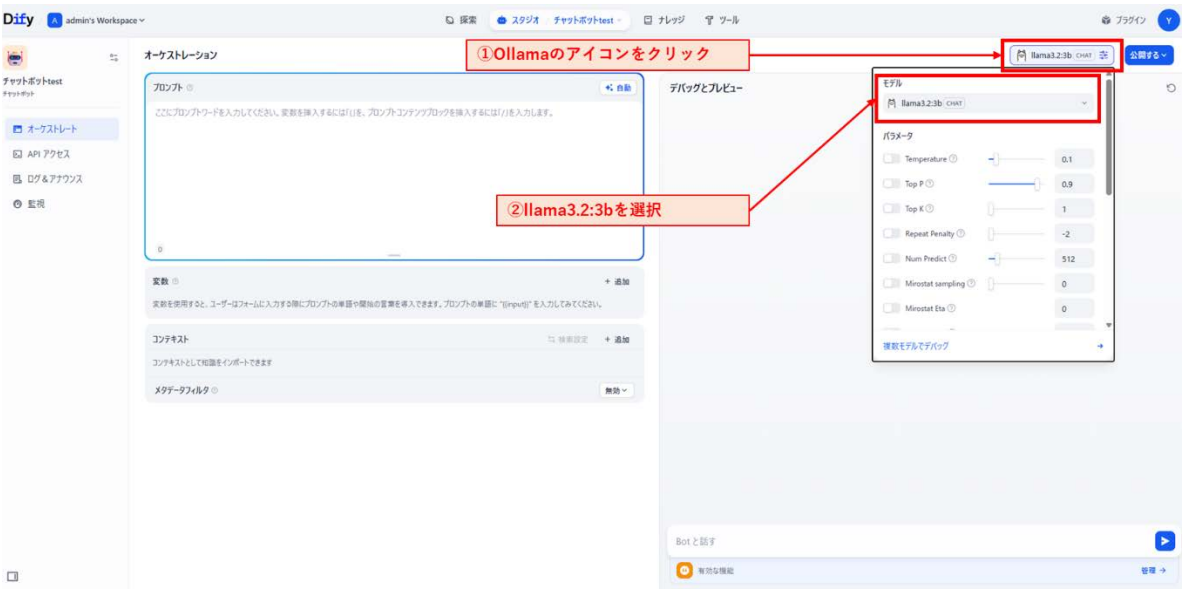

図 3-4-6　llama3.2:3b を選択する

(7) [公開する]、[更新を公開]、[アプリを実行]の順にクリックする（図 3-4-7）。

図 3-4-7　アプリを公開する

(8) 作成したチャットボットのアプリが開くので、プロンプトを入力し、出力が返ってくれば生成 AI アプリの作成は正常に完了している（図 3-4-8）。

図 3-4-8　チャットボットアプリの動作確認

4. オンプレミス生成 AI 基盤の展開

4.1　オンプレミス生成 AI 基盤の展開

前章の手順により構築したオンプレミス生成 AI チャットボットについて、原子力機構システム計算科学センター内で試験運用を実施した後、2025 年 3 月に原子力機構内に展開した。展開直後の利用状況として、1 日あたりの利用者数が 200～300 人、1 か月の累計で 1,000 人以上が利用するなど、一定の利用実績が確認できた。

その後も利用者は増加し、現在では月間 2,000 人以上が利用している。2025 年 3 月から同年 10 月までの月間ユニーク利用者数、月間合計メッセージ数、月間合計トークン数の推移を表 4-1 に整理し、図 4-1 にグラフとして示す。なお、メッセージ数は AI が回答した総数を指し、トークン数はテキストを単語や句読点などの要素に分割した際の、LLM が処理する最小単位の総数を指す。

表 4-1　生成 AI チャットボット利用状況の推移

	2025 年 3 月	2025 年 4 月	2025 年 5 月	2025 年 6 月
月間ユニーク利用者数	1,109	1,595	2,538	2,355
月間合計メッセージ数	14,699	32,360	33,929	42,855
月間合計トークン数	18,858,548	44,498,018	47,999,577	51,407,997
	2025 年 7 月	2025 年 8 月	2025 年 9 月	2025 年 10 月
月間ユニーク利用者数	2,005	1,507	2,030	2,399
月間合計メッセージ数	31,097	21,750	28,792	60,343
月間合計トークン数	31,737,104	23,016,076	32,513,437	111,548,758

運用期間を通じて大きな障害はなく、比較的安定して稼働しているものの、複雑かつ巨大なプロンプトが入力されると GPU ノードの負荷が高まり応答が遅延する場合があるため、プロンプトの文字数に上限を設けるなどの運用上の工夫を行っている。また、利用する LLM については、展開当初は軽量性と回答の安定性の双方を兼ね備えた Google 製の gemma3:12b を採用していた。しかし、利用者の増加に伴い処理速度の低下などが懸念されたため、2025 年 11 月現在では、より軽量な Google 製のモデルである gemma3n:e4b を採用している。

各 LLM を GPU ノード上で実行した際の比較結果を表 4-2 に示す。処理速度は LLM の性能に依存するため、Python プログラムを用いて Ollama を API 経由で実行し、実際の応答時間を計測した。計測に用いたプロンプトとプログラムを表 4-3 および表 4-4 に、出力結果を表 4-5 に示す。

比較には 3 種類のプロンプトを用いた。基本的な応答性能および応答速度を計測する短文応答のプロンプト①、長文入力に対する処理性能および知識性能を計測する長文要約のプロンプト②、思考力や推論の正確性を評価する推論（Chain-of-Thought）形式のプロンプト③の 3 種類である。

表 4-2　計測結果

LLM	gemma3:12b	gemma3n:e4b
プロンプト①の応答速度	4.67 秒	5.03 秒
プロンプト②の応答速度	1.99 秒	1.77 秒
プロンプト③の応答速度	1.87 秒	1.76 秒

表 4-3　計測に用いたプロンプト

プロンプト①
こんにちは。今日の気分を一言で教えてください。
プロンプト②
『走れメロス』の内容を 200 文字以内で要約してください。
プロンプト③
りんご 3 個が 240 円で売られています。 りんご 10 個ではいくらですか？計算過程も示してください。

表 4-4　計測に用いたプログラム

```
import requests
import json
import time
from datetime import datetime

# ====== 設定 ======
url = "http://ISV ノードの IP アドレス/api/generate"
model_name = "計測対象の LLM"
prompt_file = "prompt.txt"
# ==================

# プロンプトファイル読み込み
with open(prompt_file, "r", encoding="utf-8") as f:
    prompt_text = f.read()

# 実行時間計測開始
start_time = time.time()
start_dt = datetime.now()

payload = {
    "model": model_name,
    "prompt": prompt_text,
    "stream": False
}

response = requests.post(url, json=payload)

end_time = time.time()
end_dt = datetime.now()

data = response.json()

# ====== 出力 ======
print("=== 実行結果 ======================================")
print("【モデル】", model_name)
print("【入力文字数】", len(prompt_text))
print("【開始時刻】", start_dt.strftime("%Y-%m-%d %H:%M:%S"))
print("【終了時刻】", end_dt.strftime("%Y-%m-%d %H:%M:%S"))
print("【実行時間】", f"{end_time - start_time:.2f}  秒")
print()
print("=== モデル応答 ====================================")
print(data.get("response", "No response"))
print("====================================================")
```

表 4-5　出力結果

LLM	gemma3:12b	gemma3n:e4b
プロンプト①に対する出力	今日は、穏やかな晴れ空のように、のんびりとした気分です。😊	今日は、**穏やか**です。 😊 何かお手伝いできることはありますか？
プロンプト②に対する出力	ディオニシオスの命令で、メロスは友人であるセリヌンティオを救うため、処刑されるべき怪物と一騎打ちを挑む。メロスは怪物に勝利するものの、処刑は免れず、その姿は伝説となり、人々に勇気と希望を与えた。メロスの献身と友情の精神が、ギリシアの民に深く刻まれた物語。	メロスは、王の命で暗殺を依頼された友軍ゴルガスの脱走を阻止するため、駆け足で急行する。道中、様々な困難や誘惑に遭遇するが、友を救うという強い意志と、人々の助けを借りながら、ついにゴルガスに間に合う。友情と忠誠心の物語。
プロンプト③に対する出力	りんご 3 個が 240 円なので、りんご 1 個あたりの値段は、 240 円 ÷ 3 個 ＝80 円 りんご 1 個 80 円ですから、りんご 10 個の値段は、 80 円 × 10 個 ＝800 円 したがって、りんご 10 個では 800 円です。	りんご 3 個が 240 円で売られているので、1 個あたりの値段は 240 円 ÷ 3 個 ＝80 円/個 りんご 10 個の値段は 80 円/個 × 10 個 ＝800 円 したがって、りんご 10 個では 800 円です。

表 4-2 より、応答速度には大きな差は見られなかった。

表 4-5 より、短文応答のプロンプト①および推論（Chain-of-Thought）形式のプロンプト③においては、両モデルとも適切な応答を示し、大きな差は見られなかった。一方、長文要約のプロンプト②では、両モデルとも内容の改変や情報の欠落など、不正確な出力が見られた。

この結果から、利用者の増加に伴う処理速度の低下を懸念して gemma3n:e4b を採用したことは合理的である。しかし、長文要約など高度な処理に対しては限界が見られた。今後、より複雑なプロンプトに対応するためには、高性能な LLM と、それを動かす計算リソースが不可欠である。

4.2　展開後の機能拡張

2025 年 3 月に生成 AI チャットボットを展開した後も、利用者からの意見を踏まえ、ニーズに応じた機能拡張や最適化を継続的に実施している。

2025 年 8 月、OpenAI 社からローカル版 LLM として gpt-oss シリーズが公開された。なかでも gpt-oss:120b は高い性能を有するが、GPU リソースの要求性能が非常に高く、スパコンの GPU ノード上での稼働ではサービスとしての展開は困難であった。

そこで、原子力機構システム計算科学センターにて保有している GPU サーバ（NVIDIA RTX 6000 Ada 48GB を 4 基搭載）上で稼働する Ollama で gpt-oss:120b を動かし、それを利用した高性能版生成 AI チャットボットを Dify で構築し、原子力機構内に展開した。

GPU サーバ上で表 4-4 のプログラムを実行した際の応答速度と出力結果を表 4-6 に示す。

表 4-6　GPU サーバでの応答速度と出力結果（1/2）

LLM	gpt-oss:120b
プロンプト①の応答速度	58.73 秒
プロンプト②の応答速度	12.92 秒
プロンプト③の応答速度	125.43 秒
プロンプト①に対する出力	ワクワク✨
プロンプト②に対する出力	友情と信義を試すため、王に裏切りの疑いをかけられたメロスは、友人のシラヌスに 3 日間の執行猶予を取り付け、約束の期限までに帰還しようと走り続ける。途中の困難や自らの恐怖に直面しながらも、最後には無事帰り、王はメロスの誠実さに感銘を受けて寛容になる。

表 4-6　GPU サーバでの応答速度と出力結果（2/2）

LLM	gpt-oss:120b
プロンプト③に対する出力	**計算過程** 1. まず、1 個あたりの価格を求めます。 ¥[¥frac{240¥ ¥text{円}}{3¥ ¥text{個}} = 80¥ ¥text{円/個} ¥] 2. 1 個が 80 円なので、10 個の価格は ¥[80¥ ¥text{円/個} ¥times 10¥ ¥text{個} = 800¥ ¥text{円} ¥] **答え** りんご 10 個は **800 円** です。

表 4-6 より、gpt-oss:120b は gemma3:12b および gemma3n:e4b と比較して応答速度は約 7～70 倍程度と劣るものの、両モデルで不正確な情報を含む出力が見られたプロンプト②に対して、より正確な出力を示した。

別途、高性能版生成 AI チャットボット以外にも、画像生成、音声文字起こし、翻訳、イントラ検索システムと連携したチャットボットなど、ニーズに応じた複数の生成 AI アプリを開発し、段階的に展開している。これらのアプリは API 経由でも利用できるようになっており、Dify などの生成 AI アプリ開発基盤ツールやプログラム中から呼び出すことが可能である。

4.3　まとめ

本章では、生成 AI 基盤の原子力機構内への展開、LLM の性能比較および展開後に実施した機能拡張について述べた。基本的なプロンプトに対しては安定した応答が得られており、利用者数も減少することなく推移していることから、業務効率化に一定の効果があると考えられる。

生成 AI 基盤の展開以降、生成 AI を活用した課題解決の個別相談が約 10 部署から寄せられており、API 経由での利用も約 30 箇所に提供している。これらの状況から、生成 AI への関心の高まりが確認された。

また、現在は生成 AI に関する開発者コミュニティも立ち上げており、生成 AI 基盤の構築手順や各開発者による検証結果などの技術的知見を共有しながら、組織横断的な意見交換を行うことで、原子力機構内における生成 AI の活用は継続的に拡大している。

一方で、オンプレミス生成 AI の回答精度は、最新の ChatGPT をはじめとするクラウド型の外部生成 AI サービスと比較すると性能面では劣る。そのため、秘密情報を含まない場合はクラウド

型の外部生成 AI サービスを、含む場合はオンプレミス生成 AI を利用するなど、用途に応じた使い分けが重要である。

4.4 今後の課題

生成 AI の活用が広まる中で、原子力機構内の規程や文書を参照しながら応答する RAG 型の利用ニーズが高まりつつある。Dify にも「ナレッジ」と呼ばれる RAG 機能が備わっているが、現状では探索精度のばらつきや、参照対象となる文書の品質・形式の不統一などにより、十分な精度を実現できていない。今後は、文書や検索クエリをベクトル表現に変換して類似性の高い情報を検索する Embedding モデルや、検索された候補の文書を再評価して適切な順序に並べ替える Rerank モデルなどを検証・活用することで、RAG の精度向上を図る必要がある。そのためには、文書の前処理も含めた改善が重要である。

また、生成 AI 基盤の利用者数は頭打ちになりつつあり、現状の利用者層は生成 AI に関する関心が高い部署に偏っている傾向がある。生成 AI の利用経験が少ない利用者に対しては、生成 AI を実際に活用する機会が十分に提供できていない可能性もあるため、業務における生成 AI の活用を定着させるための導入支援や利用促進の取り組みが求められる。加えて、さらなる利用者の生成 AI リテラシーやプロンプト作成に関するスキルの向上も重要である。

5. おわりに

本報告書では、原子力機構におけるオンプレミス生成 AI 基盤の構築および展開について述べるとともに、LLM の性能評価や機能拡張の取り組みを通じて、生成 AI を業務活用するための条件と課題を明らかにした。原子力機構では、秘密情報を扱う業務が多く存在することから、セキュリティリスクを伴うクラウド型の外部生成 AI サービスに依存せず、安全かつ容易に生成 AI を利用できる環境を提供した点は大きな成果と言える。

本基盤は、新たな設備投資を行うことなく、スパコンをはじめとする既存の計算資源およびオープンソースソフトウェアを活用して構築したものであり、追加コストを要せずに実装できた点も大きな特徴である。さらに、オープンソースソフトウェアを活用することで、試行と改良の過程を通じて、従来の課題を技術的に解決する力を原子力機構内で醸成できた点は、原子力機構における技術的自立性を高めるという観点からも意義深い。

生成 AI はあくまで手段であり、目的そのものではない。本基盤の活用を通じて、業務プロセスの見直しや人材スキルの向上を促し、DX 推進の契機として展開していくことが重要である。すなわち、生成 AI の活用を原子力機構全体の DX 推進へとつなげていく点にこそ、本質的な価値がある。

生成 AI 技術は日進月歩で進化を続けているため、最新技術を積極的に取り入れながら、継続的な機能改善を行うことで、より効果的かつ持続可能な生成 AI 基盤の実現を目指して取り組んでいく所存である。

本報告書の内容が、生成 AI の導入や基盤構築を実施する際の一助となれば幸いである。

謝辞

本報告書を執筆するにあたり、ご指導いただいたシステム計算科学センター業務ＤＸ推進室久野哲也室長に深く感謝の意を表す。

また、オンプレミス生成 AI 基盤の構築と展開にご助力いただいたシステム計算科学センター井戸村泰宏副センター長およびシステム計算科学センターＨＰＣ・ＤＸ基盤技術開発室伊奈拓也技術副主幹に深く感謝の意を表す。

さらに、スパコンの利用に際しご支援いただいたシステム計算科学センター高性能計算技術利用推進室坂本健作室長、増子献児氏および日本ヒューレット・パッカード合同会社の藤敏弘氏に深く感謝の意を表す。

加えて、オンプレミス生成 AI 基盤展開後の機能拡張において、生成 AI による画像生成機能および翻訳機能の構築にご助力いただいたシステム計算科学センターＡＩ・ＤＸ基盤技術開発室奥村雅彦研究主幹および真弓明恵氏に深く感謝の意を表す。

最後に、本報告書の執筆およびオンプレミス生成 AI 基盤の構築と展開に携わっていただいたすべての関係者と利用者に深く感謝の意を表す。

参考文献

1) 総務省・経済産業省, AI 事業者ガイドライン（第 1.1 版）,
https://www.soumu.go.jp/main_content/001002576.pdf
https://www.meti.go.jp/shingikai/mono_info_service/ai_shakai_jisso/pdf/20250328_1.pdf
（参照：2025 年 11 月 7 日）.
2) 総務省, 平成 28 年版情報通信白書,
https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h28/pdf/n4200000.pdf
（参照：2025 年 11 月 7 日）.
3) 総務省, 令和 6 年版情報通信白書,
https://www.soumu.go.jp/johotsusintokei/whitepaper/ja/r06/pdf/n1310000.pdf
（参照：2025 年 11 月 7 日）.
4) 情報処理推進機構, テキスト生成 AI の導入・運用ガイドライン,
https://www.ipa.go.jp/jinzai/ics/core_human_resource/final_project/2024/f55m8k0000003spo-att/f55m8k0000003svn.pdf （参照：2025 年 11 月 12 日）.
5) デジタル庁, ChatGPT 等の生成 AI の業務利用に関する申合せ（第 2.1 版）,
https://www.digital.go.jp/assets/contents/node/basic_page/field_ref_resources/debd5eca-6832-406e-a530-4e98ec032133/c60c5872/20250325_meeting_executive_agreement_07.pdf
（参照：2025 年 11 月 20 日）.

6) 日本原子力研究開発機構, 令和 6 年度大型計算機システム利用による研究成果報告集, JAEA-Review 2025-044, 2026, 140p.
https://doi.org/10.11484/jaea-review-2025-044（参照：2025 年 11 月 14 日）.

This is a blank page.