



Amazon Nova 2 開發人員指南

Amazon Nova



Amazon Nova: Amazon Nova 2 開發人員指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

什麼是 Amazon Nova 2 ?	1
重要概念	1
Amazon Nova 模型	1
您可以建置什麼？	2
優勢	2
Amazon Nova 的運作方式	3
定價	3
後續步驟	3
Amazon Nova 2 的新功能	4
新 模型	4
新功能	4
Nova 多模態內嵌	4
延伸思考和推理	4
內建工具	4
AI 代理程式建置	5
改善文件理解	5
增強影片理解	5
自訂模型	5
Amazon Nova Forge	5
後續步驟	5
開始使用	6
在主控台中開始使用	6
探索文字遊樂場	6
API 入門	7
取得憑證以授予程式設計存取權	8
將 Amazon Bedrock 許可連接至使用者或角色	9
使用 Amazon Nova 模型為文字提示詞產生回應	10
核心推論	11
重要：逾時組態	12
核心推論主題	12
使用 Converse API	13
請求結構	13
使用系統提示	14
推論參數	14

使用推理	15
使用 Invoke API	18
請求結構	18
請求參數	19
串流回應	19
使用 ConverseStream 串流	19
使用 InvokeModelWithResponseStream 串流	20
串流事件類型	20
使用 Amazon Nova 內嵌	21
主要功能	21
產生內嵌	22
隨需推論	25
優勢	25
使用隨需推論	25
定價	26
配額和限制	26
使用 Nova 功能	27
推理	27
多模式理解	28
依模態支援的內容類型	28
影像理解	29
影片理解	31
理解文件	33
Speech-to-Speech	35
主要功能	35
開始使用	36
陳述匯入和組態	36
定義 SimpleNovaSonic 類別	37
初始化用戶端	37
處理事件	38
啟動工作階段	38
處理音訊輸入	41
結束工作階段	42
處理回應	43
播放音訊	44
擷取音訊	45

執行主要函數	46
程式碼範例	48
入門範例	48
進階使用案例	48
實際操作工作坊	49
語音對話提示	49
重要指導方針	49
建議的語音基準系統提示	50
語音提示組態	50
系統提示組態	50
性別語言的性別協議	51
表達語言的思維鏈：章程合理性	52
過度使用建議的片語	53
核心概念	55
事件生命週期	55
事件流程序列	57
使用雙向 API 處理輸入事件	57
使用雙向 API 處理輸出事件	66
闖入	72
輪換可控制性	74
跨模式輸入	75
語言支援和多語言功能	78
管理聊天歷史記錄	79
工具組態	85
非同步工具呼叫	92
整合	95
Strands 代理程式	95
架構整合	98
使用 Nova 的進階系統	106
概觀	106
延伸思考	106
延伸思考的運作方式	106
組態選項	109
支援的模型	110
工具使用的延伸思考	110
了解字符和定價的原因	110

常見問答集	110
工具使用	111
了解工具使用工作流程	111
建立工具	112
呼叫工具	114
內建系統工具	118
建置 AI 代理程式	120
建立代理程式	120
調用代理程式	122
Web Grounding	125
Web Grounding 的運作方式	125
如何使用 Web Grounding	125
區域可用性	126
回應結構	126
接地安全	127
錯誤處理	128
內建工具所需的許可	128
提示詞工程	130
什麼是提示詞工程	130
提示詞工程入門	130
最佳實務	131
了解角色	131
建立精確提示	131
讓提示詞區段引起關注	150
使用系統角色	151
提供範例 (少樣本提示)	159
工具呼叫系統	169
進階提示技巧	170
提示多模態輸入	190
一般多模式準則	190
文件和映像了解	191
影片理解	31
內容管制	202
主要原則	202
建議組態	202
MLCommons 通用分類	202

提示範例	204
程式碼和故障診斷	210
程式碼庫	210
Converse API 範例	210
InvokeModel API 範例	230
疑難排解	235
身分驗證和設定	235
拒絕模型存取	236
區域可用性問題	236
逾時組態	236
API 回應問題	237
原因模式錯誤	238
工具使用問題	239
API 和 SDK 參考	243
請求和回應結構描述	243
完成請求結構	243
完整的回應結構	248
SDK 參考	250
Amazon Nova 模型自訂	252
自訂方法	252
自訂平台	252
一般先決條件	253
Amazon Nova 配方	254
取得基本配方	255
取得 Forge 配方	256
可用的模型和演算法	256
Amazon Nova 2.0 Lite	256
在 SageMaker 訓練任務上	262
Amazon Nova 自訂的最佳實務	262
Nova Forge SDK	265
培訓	271
監控反覆運算的進度	305
評估	310
反覆訓練	366
在 SageMaker HyperPod 上	367
Amazon Nova 2 模型的運算需求	367

Amazon Nova 自訂的最佳實務	262
Nova Forge SDK	372
HP 叢集設定	372
基本命令指南	372
Nova Forge 存取和設定	382
培訓	383
評估	419
MLflow 監控	463
反覆訓練	465
概觀	465
運作方式	466
何時使用反覆訓練	467
工作流程範例：SFT → RFT	468
監控反覆運算的進度	470
最佳實務	471
成本考量	473
限制	474
疑難排解	475
SageMaker 推論	476
功能	477
支援的模型和執行個體	477
支援 AWS 的區域	480
支援的容器映像	480
最佳實務	480
支援	480
開始使用	481
API 參考	501
評估模型	510
Amazon Nova Forge 的濫用偵測	516
Amazon Bedrock 參考。	517
部署自訂模型以進行隨需推論	518
隨需推論	523
限制	524
新堡	526
先決條件	526
訂閱 Nova Forge	526

其他先決條件	527
Nova Forge 存取和設定	527
取得 Nova Forge 文件	527
步驟 1：訂閱 Nova Forge	527
步驟 2. 設定 HyperPod 基礎設施	533
內容管制設定	534
持續訓練前和中訓練	534
什麼是中繼檢查點？為什麼需要中繼檢查點？	534
有哪些檢查點可用？	535
要使用哪個檢查點？	536
如何使用 Nova 2.0 的資料混合？	536
解析資料混合類別	537
受監督的微調	541
簡介	541
何時使用 SFT？	541
何時使用參數效率和何時使用完整排名 SFT？	542
SFT 的資料混合	542
YAML 組態範例	543
模型檢查點	545
訓練方法	546
開始之前的必要項目	546
實現成功 SFT 的系統性方法	547
實驗定序和資料混合	547
準備 SFT 的資料集	548
評估方法	548
最佳實務	262
準備訓練任務組態	550
設定資料混合區塊	552
如何啟動任務	556
如何監控任務	564
如何在 SFT 之後評估您的模型？	567
經驗教訓和秘訣	569
強化學習	570
使用您自己的協調器進行代理多迴轉評估	570
負責任的 AI Toolkit	571
負責任的 AI 工具組	571

負責任的使用	573
指導方針	573
建議	574
配額	575
請求提高 Amazon Nova 配額	575
監控	576
安全	577
透過 CloudFormation 建立資源	578
進一步了解 CloudFormation	578
文件歷史紀錄	579
.....	dlxxx

什麼是 Amazon Nova 2？

Amazon Nova 提供多模式基礎模型，可處理文字、影像、影片、文件和語音。Amazon Nova 2 模型支援高達 100 萬個權杖的內容和進階推理功能，可讓您建置複雜的 AI 應用程式，了解複雜的輸入並產生準確的回應。

您可以建置互動式聊天機器人、分析文件和影片、建立具有擴充推理的 AI 代理器，以及開發語音應用程式。

重要概念

了解 Amazon Nova 模型之前，請先熟悉下列核心概念：

基礎模型

預先訓練的 AI 系統提供您透過 API 存取的不同大小和功能。

推論

將請求傳送至模型並接收產生的回應的程序。

原因

延伸思考功能，允許模型分解複雜的問題，並在提供答案之前顯示其step-by-step分析。

多模態

在單一請求中同時處理和了解多種輸入類型的能力：文字、影像、影片和文件。

RAG（擷取增強的產生）

一種將模型回應與您自己的資料來源結合的技術，以提供更準確、情境式的答案。

Amazon Nova 模型

Amazon Nova 2 包含下列模型，每個模型都針對不同的使用案例進行最佳化：

模型	輸入模態	輸出模態	使用案例
Nova 2 Lite	文字、影像、影片、文件	文字	大量應用程式優先考慮速度和成本效益

模型	輸入模態	輸出模態	使用案例
Nova 2 聲波	語音、文字	語音、文字	具有快速回應時間的語音應用程式
Nova 多模態內嵌	文字、影像、文件、影片、音訊	嵌入項目	語意搜尋、建議系統和相似性比對

所有模型最多支援 100 萬個內容權杖，並且可在單一回應中產生最多 65,536 個權杖。具有推理功能的模型可以執行延伸思考，逐步解決複雜的問題。

您可以建置什麼？

以下是您可以使用 Amazon Nova 建置的範例：

- 智慧型文件助理 - 處理最多 100 萬個內容權杖的大型文件，以回答問題並擷取洞見（使用 RAG）
- 複雜推理應用程式 - 解決具有延伸思考的多步驟問題，顯示模型的step-by-step分析（或推理）
- 影片分析管道 - 大規模擷取洞見、產生摘要並識別影片內容中的關鍵時刻 (Nova 2 Lite)
- 語音式 AI 代理程式 - 建置可了解語音輸入並以自然語言回應的對話式代理程式 (Nova 2 Sonic)

優勢

Amazon Nova 提供下列優點：

多模式理解

在單一請求中處理文字、影像、影片、文件和語音。Amazon Nova 模型了解不同輸入類型之間的關係。

擴充內容

最多支援 100 萬個字符，可讓您處理整個程式碼庫、冗長的文件或延伸對話，而不會遺失內容。

進階推理

具有推理功能的模型會分解複雜的問題，並顯示step-by-step分析，從而提高多步驟任務的準確性。

彈性部署

透過 Amazon Bedrock 存取模型，無需管理基礎設施，或透過微調和強化學習自訂模型。

內建工具

使用 Web 接地存取即時資訊和程式碼解譯器，在沒有外部整合的情況下執行 Python 程式碼。

Amazon Nova 的運作方式

Amazon Nova 模型是您透過 Amazon Bedrock 存取的基礎模型。基本工作流程為：

1. 您的應用程式會使用您的輸入和組態參數，將請求傳送至 Amazon Bedrock。
2. Amazon Nova 模型會處理您的輸入，如果已設定，則套用推理。
3. 模型會產生回應並將其傳回至您的應用程式。

您可以使用 RAG 來整合資料、啟用內建工具，或透過微調自訂模型，以增強回應。

定價

Amazon Nova 定價是以處理的輸入和輸出字符為基礎。不同的模型有不同的定價方案：

- Nova 2 Lite - 針對經濟實惠、大量處理進行最佳化
- Nova 2 Sonic - 語音應用程式平衡定價

如需目前定價資訊，請參閱 [Amazon Bedrock 定價](#)。

後續步驟

- 若要了解 Amazon Nova 2 中的新功能，請參閱 [Amazon Nova 2 的新功能](#)。
- 若要開始使用 Amazon Nova，請參閱 [Amazon Nova 2 入門](#)。
- 若要了解核心推論功能，請參閱 [核心推論](#)。
- 若要為您的使用案例自訂模型，請參閱 [在 SageMaker AI 上自訂 Amazon Nova 模型](#)。

Amazon Nova 2 的新功能

Amazon Nova 2 引進了對理解、創意和語音功能的重要增強功能。下列各節說明重要的新功能和改善項目。

新 模型

Nova 2 Lite

增強型模型提供進階推理與延伸思考支援、三級強度控制和多模式理解，針對生產規模代理程式工作流程進行最佳化。

Nova 2 聲波

升級的對話式語音模型，改善了七種語言的語音理解、自然語言處理和語音產生功能。

Nova 多模態內嵌

一種多模式內嵌模型，可處理文字、影像、文件、視訊和音訊輸入，並產生內嵌。

新功能

Nova 多模態內嵌

Nova Multimodal Embeddings 透過單一模型支援文字、文件、影像、視訊和音訊，啟用跨模式擷取應用程式。Nova Multimodal Embeddings 會將每個內容類型映射到統一的語意空間，讓您能夠執行單模態、跨模態和多模態向量操作，為代理程式擷取擴增產生 (RAG) 和多模態語意搜尋等應用程式提供支援。

延伸思考和推理

Nova 2 Lite 支援延伸思考，這可讓模型在產生回應之前花更多時間推理複雜的問題。此功能可提高多步驟推理任務的準確性，包括具有多種工具的代理工作流程、進階數學、複雜的規劃和程式碼產生。

若要進一步了解，請導覽至 [Amazon Nova 2 中的延伸思維](#)。

內建工具

Amazon Nova 2 包含內建工具，無需外部整合即可擴展模型功能：

- Web 接地 – 從 Web 存取即時資訊，以提供up-to-date回應並減少幻覺。
- 程式碼解譯器 – 執行 Python 程式碼來執行計算。

AI 代理程式建置

Amazon Nova 2 模型已針對建置 AI 代理器進行最佳化。這些模型提供改善的工具使用方式、改善多步驟任務的推理，以及增強跨複雜代理程式工作流程維護內容的能力。

改善文件理解

Nova 2 Lite 提供增強的文件處理功能，更了解複雜的文件配置、資料表、圖表和多頁文件。這些模型可更準確地從 PDFs、試算表和其他文件格式擷取資訊。

增強影片理解

Nova 2 Lite 提供改善的影片分析功能，包括更好的視覺感知、時間理解、動作辨識，以及處理更長影片序列且準確度更高的能力。

自訂模型

Nova 2 Lite 支援 Amazon Bedrock 和 SageMaker AI 上的監督式微調 (SFT) 和強化微調 (RFT)，可讓您根據特定業務需求調整 Amazon Nova 2。

Amazon Nova Forge

Amazon Nova Forge 是first-of-its-kind的服務，為組織提供使用 Amazon Nova 建置自己的前沿模型最簡單且最具成本效益的方式。

後續步驟

- 若要了解 Amazon Nova 模型和功能，請參閱 [什麼是 Amazon Nova 2 ?](#)。
- 若要開始使用 Amazon Nova 2.0，請參閱 [Amazon Nova 2 入門](#)。
- 若要探索核心推論功能，請參閱 [核心推論](#)。

Amazon Nova 2 入門

您可以透過下列方法叫用 Amazon Nova 2 模型：

- Amazon Bedrock [InvokeModel](#) 或 [Converse](#) API 操作
- 適用於 Nova 2 Lite、Nova 2 Sonic 的 Amazon Bedrock [聊天和文字遊樂場](#)

Important

如果您是初次使用 Amazon Bedrock AWS 或 Amazon Bedrock，您必須先依照 Amazon Bedrock 入門中的 [步驟來設定 Amazon Bedrock](#)。或者，您可以試用 Amazon Nova 模型，而無需進入 Amazon Bedrock，網址為 nova.amazon.com：[//nova.amazon.com/dev](https://nova.amazon.com/dev)。

選取一個主題以開始搭配該方法使用 Amazon Nova。

在 Amazon Bedrock 主控台中開始使用 Amazon Nova

本節說明如何使用 AWS 主控台內的遊樂場，向 Amazon Nova 2 模型提交文字提示，並產生文字或影像回應。在執行下列範例之前，您應該檢查是否符合下列必要條件：

先決條件

- 您有 AWS 帳戶，並具有存取該帳戶中具有 Amazon Bedrock 必要許可之角色的許可。否則，請遵循 [Amazon Bedrock 入門](#) 中的步驟。
- 如果您從美國區域存取模型，則必須使用涉及將 us 字首新增至模型 ID 的美國 CRIS 端點（例如 `us.amazon.nova-2-lite-v1:0`）。
- 如果您是從美國境外存取模型，則可以使用全域 CRIS 端點（例如 `global.amazon.nova-2-lite-v1:0`）或在模型 ID 中的區域（例如 `us/eu/`）字首 jp。

探索文字遊樂場

以下範例示範如何使用文字實驗場：

1. 開啟位於 <https://console.aws.amazon.com/bedrock/> 的 Amazon Bedrock 主控台。

2. 從左側導覽窗格的實驗場下方，選擇聊天/文字。
3. 選擇選取模型，然後選取提供者和模型。在此範例中，選取 Amazon，然後選取 Amazon Nova 2 模型（例如 Nova 2 Lite 或 Nova 2 Sonic）。接著選擇 Apply (套用)。
4. 從文字面板下方選取預設提示詞，或在文字面板中輸入提示詞，例如 **Describe the purpose of a "hello world" program in one line.**
5. 若要探索 Amazon Nova 的影像理解功能，您可以從電腦上傳一幅 JPEG、PNG、GIF 或 WEBP 格式的影像，其大小不超過 25 MB。上傳影像之後，您可以向 Amazon Nova 詢問影像相關的問題。
6. 若要探索 Amazon Nova 的文件理解功能，您可以上傳 CSV、DOC、DOCX、HTML、MD、PDF、TXT、XLS 或 XLSX 格式的文件，這些文件小於或等於 4.5 MB。上傳文件後，您可以向 Amazon Nova 詢問文件相關的問題。
7. 若要探索 Amazon Nova 的影片理解功能，您可以從電腦上傳一個 MKV、MOV 或 MP4 格式的影片，其大小不超過 25 MB。您可以將 Amazon S3 用於不超過 1 GB 的影片。上傳影片後，您可以向 Amazon Nova 詢問影片相關的問題。
8. 選擇執行，以在模型上執行推論。產生的文字會在文字面板中顯示在您的提示下。

API 入門

開始使用 APIs 之前，如果您剛開始使用 AWS 或需要安裝 AWS CLI 或 AWS SDK，請遵循這些程序。如果兩者都不適用於您，請移至 [取得憑證以授予程式設計存取權](#)。

我第一次使用 AWS

如果您沒有 AWS 帳戶，請完成下列步驟來建立帳戶。或者，您也可以前往 nova.amazon.com/dev：// 開始使用 Amazon Nova 免費建置。

註冊 AWS 帳戶

1. 開啟 <https://portal.aws.amazon.com/billing/signup>。
2. 請遵循線上指示進行。

部分註冊程序需接收來電或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊 AWS 帳戶時，會建立 AWS 帳戶根使用者。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為安全最佳實務，將管理存取權指派給使用者，並僅使用根使用者來執行[需要根使用者存取權的任務](#)。

AWS 會在註冊程序完成後傳送確認電子郵件給您。您可以隨時登錄 <https://aws.amazon.com/> 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

保護 AWS 您的帳戶根使用者

1. 選擇根使用者並輸入 AWS 您的帳戶電子郵件地址，以帳戶擁有者 [AWS 管理主控台](#) 身分登入。在下一頁中，輸入您的密碼。

如需使用根使用者登入的說明，請參閱 [登入使用者指南中的以根使用者身分 AWS 登入](#)。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

如需說明，請參閱《IAM 使用者指南》中的 [為 AWS 您的帳戶根使用者（主控台）啟用虛擬 MFA 裝置](#)。

我需要安裝 AWS CLI 或 AWS SDK

若要安裝 AWS CLI，請遵循 [安裝或更新至最新版本 CLI AWS](#) 中的步驟。

若要安裝 AWS SDK，請選取對應至您要在 [Tools to Build on AWS](#) 中使用的程式設計語言的標籤。

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。每個 SDK 都提供 API、程式碼範例和文件，可讓開發人員更輕鬆地以他們偏好的語言建置應用程式。SDK 會自動為您執行有用的任務，例如：

- 加密簽署服務請求
- 重試請求
- 處理錯誤回應

取得憑證以授予程式設計存取權

如果使用者想要在 AWS 管理主控台 AWS 外部與 互動，則需要程式設計存取。授予程式設計存取權的方式取決於正在存取的使用者類型 AWS。

若要授與使用者程式化存取權，請選擇下列其中一個選項。

哪個主體需要程式設計存取權？	到	根據
IAM 使用者	限制長期憑證的持續時間，以簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱《AWS 命令列界面使用者指南》中的使用 IAM 使用者憑證進行身分驗證。 - AWS SDKs 和工具，請參閱 AWS SDKs 和工具參考指南中的使用長期憑證進行身分驗證。 - 對於 AWS APIs，請參閱《IAM 使用者指南》中的管理 IAM 使用者的存取金鑰。
IAM 角色	使用臨時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請遵循《IAM 使用者指南》中將 臨時登入資料與 AWS 資源搭配使用 中的指示。
在 IAM Identity Center 中管理的使用者	使用臨時登入資料簽署對 AWS CLI、AWS SDKs 程式設計請求。AWS APIs	請依照您要使用的介面所提供的指示操作。 - 如需 AWS CLI，請參閱《AWS 命令列界面使用者指南》中的設定 AWS CLI 以使用 IAM Identity Center。 - AWS SDKs、工具和 AWS APIs，請參閱 AWS SDKs 和工具參考指南中的IAM Identity Center 身分驗證。

將 Amazon Bedrock 許可連接至使用者或角色

設定程式設計存取的登入資料後，您需要設定使用者或 IAM 角色的許可，才能存取 Amazon Bedrock 相關動作。若要設定這些許可，請執行下列動作：

1. 在 AWS 管理主控台首頁上，選取 IAM 服務或導覽至位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
2. 選取使用者或角色，然後選取您的使用者或角色。
3. 在許可索引標籤中，選擇新增許可，然後選擇新增 AWS 受管政策。選擇 [AmazonBedrockFullAccess](#) AWS 受管政策。
4. 若要允許使用者或角色訂閱模型，請選擇建立內嵌政策，然後在 JSON 編輯器中指定下列許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MarketplaceBedrock",
      "Effect": "Allow",
      "Action": [
        "aws-marketplace:ViewSubscriptions",
        "aws-marketplace:Unsubscribe",
        "aws-marketplace:Subscribe"
      ],
      "Resource": "*"
    }
  ]
}
```

使用 Amazon Nova 模型為文字提示詞產生回應

滿足所有先決條件後，您可以使用 [Converse](#) 請求測試對 Amazon Nova 模型提出模型調用請求。

若要安裝 AWS CLI，請依照[安裝或更新至最新版本的 AWS CLI](#) 中的步驟進行。遵循 [取得憑證以授予程式設計存取權](#) 中的步驟，確認您已設定憑證來使用 Boto3。

若要使用 AWS CLI 在 Nova 2 Lite 中產生文字提示的回應，請在終端機中執行下列命令：

```
aws bedrock-runtime converse \
  --model-id us.amazon.nova-2-lite-v1:0 \
  --messages '[{"role":"user","content":[{"text":"Write a short poem"}]]' \
  --additional-model-request-fields '{"reasoningConfig":
{"type":"enabled","maxReasoningEffort":"low"}}'
```

核心推論

推論是將請求傳送至 Amazon Nova 模型並接收產生的回應的程序。Amazon Nova 模型支援透過兩個 API 選項進行推論：

- Converse API ([Converse](#)、[ConverseStream](#))：提供跨不同模型的一致界面，讓您更輕鬆地在模型之間切換，或建置可與多個模型搭配使用的應用程式。建議大多數使用案例使用。
- 叫用 API ([InvokeModel](#)、[InvokeModelWithResponseStream](#))：請求承載是專門針對每個模型的原生格式所建構。使用請求內文中提供的提示字元和推論參數來執行推論。

這兩個 APIs 都支援相同的核心功能，包括：

- 多回合對話
- 多模態輸入（文字、影像、視訊、音訊）
- 工具使用
- 防護機制
- 串流回應

兩個 APIs 之間的請求結構幾乎相同，只有位元組資料（文件、影像、影片和音訊）的編碼方式不同。對於 Amazon Nova 模型獨有的模型請求參數，例如 `reasoningConfig` 和 `TopK`，這些參數會放置在 `inferenceConfig` 物件中的 `additionalModelRequestFields`。這些是 `InvokeModel` 和 `InvokeModelWithResponseStream` 的最上層參數。

Note

檢閱程式碼範例和疑難排解，以取得 Nova 2 模型的程式碼範例清單。

將 `modelId` 設定為下列其中一項，以使用 Amazon Nova 模型：

模型

模型 ID

Nova 2 Lite

- `global.amazon.nova-2-lite-v1 : 0`
- `us.amazon.nova-2-lite-v1 : 0`

模型	模型 ID
Nova 2 聲波	- global.amazon.nova-2-sonic-v1 : 0 - us.amazon.nova-2-sonic-v1 : 0
Nova 多模態內嵌	amazon.nova-2-multimodal-embeddings-v1 : 0

重要：逾時組態

Important

Amazon Nova 推論請求最多可能需要 60 分鐘才能完成。相應地設定您的用戶端逾時設定：下列範例為 Python 程式碼。使用者可以在該 SDK 的 API 文件中檢查其偏好的 SDK 語言版本的文件。

```
from botocore.config import Config

bedrock = boto3.client(
    'bedrock-runtime',
    region_name='us-east-1',
    config=Config(
        read_timeout=3600 # 60 minutes
    )
)
```

核心推論主題

本節討論下列主題：

- 使用 converse API：提供大多數 Bedrock 模型相容性的一致界面
- 使用 叫用 API：Nova 模型獨有的界面，而非其他 Bedrock 模型的可攜式界面
- 串流回應：產生即時回應
- 使用 Amazon Nova 內嵌：文字內嵌功能
- 隨需推論：Pay-per-use推論模型

使用 Converse API

Converse API 提供與 Amazon Nova 模型互動的統一界面。它抽象化了特定模型的詳細資訊，並提供一致的方式來處理所有 Amazon Nova 模型的多迴轉對話、系統提示和串流回應。

主題

- [請求結構](#)
- [使用系統提示](#)
- [推論參數](#)
- [使用推理](#)

請求結構

- 多轉對話：維護多個交換的內容
- 系統提示：系統指示，例如角色或回應準則
- 文件聊天：與 互動並查詢文件或文件集合
- 願景：處理和分析影像和影片
- 工具使用：啟用模型以使用外部工具和 APIs
- 護欄：套用內容篩選和安全控制
- 原因：擴展思考以解決複雜的問題

基本的 Converse API 請求包含模型 ID 和訊息清單：

```
import boto3

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse(
    modelId='us.amazon.nova-2-lite-v1:0',
    messages=[
        {
            'role': 'user',
            'content': [{'text': 'What is machine learning?'}]
        }
    ]
)
```

```
content_list = response["output"]["message"]["content"]
# Extract the first text block
text = next((item["text"] for item in content_list if "text" in item), None)
if text is not None:
    print(text)
```

使用系統提示

系統提示為模型提供內容和指示：

```
import boto3

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse(
    modelId='us.amazon.nova-2-lite-v1:0',
    system=[
        {'text': 'You are a helpful AI assistant specializing in cloud computing.'}
    ],
    messages=[
        {
            'role': 'user',
            'content': [{'text': 'Explain serverless computing.'}]
        }
    ]
)

# Print the response text
content_list = response["output"]["message"]["content"]
text = next((item["text"] for item in content_list if "text" in item), None)
if text is not None:
    print(text)
```

推論參數

使用推論參數控制模型的輸出。以下是可用的推論參數。

- **maxTokens (整數)**：要產生的字符數目上限（最多 65,000 個）。如果未指定，模型會根據請求內容使用動態預設值。
- **溫度 (浮點數)**：控制隨機性 (0.0-1.0，預設值 0.7)。較低的值可讓輸出更確定性
- **topP (浮點數)**：Nucleus 取樣閾值 (0-1，預設值 0.9)。較低的值可讓輸出更專注

- stopSequences (陣列) : 遇到時停止產生的字元序列

範例：

```
import boto3
bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse(
    modelId='us.amazon.nova-2-lite-v1:0',
    messages=[
        {
            'role': 'user',
            'content': [{ 'text': 'Write a short story.' }]
        }
    ],
    inferenceConfig={
        'maxTokens': 512,
        'temperature': 0.7,
        'topP': 0.9,
        'stopSequences': ['END']
    }
)

content_list = response["output"]["message"]["content"]
text = next((item["text"] for item in content_list if "text" in item), None)
if text is not None:
    print(text)
```

使用推理

Nova 2 Lite 支援複雜問題解決的延伸思維。使用 啟用推理reasoningConfig。

預設會停用推理，以最佳化簡單查詢的速度和成本。當您需要超越這些直接的任務時，您可以啟用推理。Nova 2 透過三個工作層級，提供對推理深度的彈性控制：

低工作量 (maxReasoningEffort: "low")

最適合：具有額外複雜性且需要結構化思考的任務。例如，您可以將此用於程式碼檢閱和改進建議，其中模型需要仔細考慮現有的程式碼品質、需要考慮多個因素的分析任務，或受益於系統化方法的問題解決案例。低工作量適用於複合任務，其中基本推理可提高準確性，而不需要深入的多步驟規劃。

中等努力 (maxReasoningEffort: "medium")

最適合：多步驟任務和編碼工作流程。例如，您可以將此用於軟體開發和偵錯，其中模型需要在實作變更之前了解現有的程式碼結構、需要跨多個檔案或元件進行協調的程式碼產生、具有相互依存性的多步驟計算，或規劃具有多個限制的任務。對於協調多個工具的代理程式工作流程而言，中等程度的工作量是最佳的，並且需要模型來維持數個循序操作的內容。

大量努力 (maxReasoningEffort: "high")

最適合：幹推理和進階問題解決。例如，您可以將此用於需要嚴格step-by-step驗證的進階數學問題和證據、需要深入調查的科學分析和研究任務、具有跨多個維度架構考量的複雜系統設計，或具有重大影響的關鍵決策案例。對於需要複雜推理的任務、仔細評估替代方案，以及徹底驗證結論，高度努力可提供最大的準確性。

下列範例顯示不同的推理工作水準：

Low effort

```
import boto3

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse(
    modelId='us.amazon.nova-2-lite-v1:0',
    system=[{"text": "You are a highly capable personal assistant"}],
    messages=[{
        "role": "user",
        "content": [{"text": "Provide a meal plan for a gluten free family of 4."}]
    }],
    inferenceConfig={
        "temperature": 0.7,
        "topP": 0.9,
        "maxTokens": 10000
    },
    additionalModelRequestFields={
        "reasoningConfig": {
            "type": "enabled",
            "maxReasoningEffort": "low"
        }
    }
)
```

原因參數：

以下是推理參數

- type：enabled或 disabled (預設：disabled)
- maxReasoningEffort：medium、 low或 high。啟用推理時，這是必要的。

 Note

溫度、 topP 和 topK 無法與 maxReasoningEffort 設定為 搭配使用high。這將導致錯誤。

回應包含推理內容：

```
{
  "output": {
    "message": {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
              "text": "[REDACTED]"
            }
          }
        },
        {
          "text": "Based on the premises, we can conclude..."
        }
      ]
    }
  },
  "stopReason": "end_turn"
}
```

 Note

使用 Amazon Nova 2，推理內容會顯示為 [REDACTED]。您仍需支付推理權杖的費用，因為它們有助於改善輸出品質。我們現在在回應結構中包含此欄位，以保留未來公開推理內容的選項。我們積極與客戶合作，以判斷浮現模型推理程序的最佳方法。

使用 Invoke API

調用 API 可讓您直接存取 Amazon Nova 模型，更能夠控制請求和回應格式。與抽象化模型特定詳細資訊的 Converse API 不同，調用 API 可讓您直接使用模型的原生請求和回應結構。

Note

調用 API 支援與 Converse API 相同的功能，但文件輸入模式除外，這是 Converse API 特有的。

請求結構

調用 API 請求需要模型 ID 和 JSON 請求內文：

```
import boto3
import json

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

request_body = {
    'messages': [
        {
            'role': 'user',
            'content': [{'text': 'What is machine learning?'}]
        }
    ],
    'inferenceConfig': {
        'maxTokens': 512,
        'temperature': 0.7
    }
}

response = bedrock.invoke_model(
    modelId='us.amazon.nova-2-lite-v1:0',
    body=json.dumps(request_body)
)

response_body = json.loads(response['body'].read())
content_list = response_body["output"]["message"]["content"]
# Extract the first text block
```

```
text_block = next((item for item in content_list if "text" in item), None)
if text_block is not None:
    print(text_block["text"])
```

請求參數

調用 API 支援下列金鑰參數：

- `messages`：具有角色和內容的對話訊息陣列
- `system`：內容和指示的選用系統提示
- `inferenceConfig`：控制模型輸出的參數（溫度、`maxTokens`、`topP`、`topK`、`stopSequences`、`reasoningConfig`）
- `toolConfig`：函數呼叫的工具規格和工具選擇

串流回應

串流可讓您在模型回應產生時逐步接收模型回應，提供更具互動性的使用者體驗。Converse API 和調用 API 都支援串流。

使用 ConverseStream 串流

使用 `ConverseStream` 接收作為事件串流的回應：

```
import boto3

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse_stream(
    modelId='us.amazon.nova-2-lite-v1:0',
    messages=[
        {
            'role': 'user',
            'content': [{'text': 'Write a short story about AI.'}]
        }
    ]
)

for event in response['stream']:
    if 'contentBlockDelta' in event:
```

```
delta = event['contentBlockDelta']['delta']
if 'text' in delta:
    print(delta['text'], end='', flush=True)
```

使用 InvokeModelWithResponseStream 串流

InvokeModelWithResponseStream 使用 與叫用 API 進行串流：

```
import boto3
import json

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

request_body = {
    'messages': [
        {
            'role': 'user',
            'content': [{'text': 'Explain quantum computing.'}]
        }
    ]
}

response = bedrock.invoke_model_with_response_stream(
    modelId='us.amazon.nova-2-lite-v1:0',
    body=json.dumps(request_body)
)

for event in response['body']:
    chunk = json.loads(event['chunk']['bytes'])
    if 'contentBlockDelta' in chunk:
        delta = chunk['contentBlockDelta']['delta']
        if 'text' in delta:
            print(delta['text'], end='', flush=True)
```

串流事件類型

串流回應包含數種事件類型：

- messageStart：表示訊息的開始
- contentBlockStart：表示內容區塊的開始
- contentBlockDelta：包含增量文字或資料

- contentBlockStop：表示內容區塊的結尾
- messageStop：以停止原因表示訊息的結尾
- metadata：包含用量資訊（字符計數）

使用 Amazon Nova 內嵌

Amazon Nova 多模態內嵌是代理程式 RAG 和語意搜尋應用程式的多模態內嵌模型。它透過單一模型支援文字、文件、影像、視訊和音訊，啟用跨模式擷取。Nova Multimodal Embeddings 會將每個內容類型映射至統一的語意空間，讓您能夠執行單模態、跨模態和多模態向量操作。

當一段內容透過 Nova 嵌入傳遞時，模型會將該內容轉換為通用數值格式，也稱為向量。向量是一組數值，可用於各種搜尋功能。為類似內容指定的向量比為非類似內容指定的向量更接近。

應用程式：

- 語意內容擷取和建議：為您的內容產生內嵌，然後使用它們來尋找類似的項目或提供個人化的建議。
- 多模態搜尋：結合來自不同內容類型的內嵌，以啟用跨模態搜尋功能。
- RAG：從多模態內容，例如具有交錯文字和影像的文件產生嵌入，以強化 GenAI 應用程式的擷取工作流程。

主要功能

這些主要功能區分 Nova Embeddings：

- 支援統一語意空間中的文字、影像、文件影像、影片和音訊。內容長度上限為 8K 個字符或 30 秒的視訊和 30 秒的音訊。
- 同步和非同步 API：該 API 支援同步和非同步使用。
- 大型檔案分割：非同步 API 透過為長文字、影片和音訊提供由使用者定義參數控制的 API 內建分割，讓您輕鬆使用大型輸入。模型會為每個區段產生單一嵌入。
- 含音訊的影片：同時處理影片與音訊。如果您想要同時代表模態或兩個個別內嵌的單一內嵌，請指定。
- 內嵌目的：根據預期的下游應用程式 (retrieval/RAG/Search、分類、叢集) 最佳化內嵌。
- 維度大小：4 個維度大小，用於權衡嵌入準確度和向量儲存成本：3072；1024；384；256。
- 輸入方法：將 S3 URI 或內嵌指定為 base64 編碼，以傳遞要內嵌的內容。

產生內嵌

完成下列操作以產生內嵌。

文字的同步內嵌：

```
import boto3
import json

# Create the Bedrock Runtime client.
bedrock_runtime = boto3.client(
    service_name='bedrock-runtime',
    region_name='us-east-1',
)

# Define the request body.
request_body = {
    'taskType': 'SINGLE_EMBEDDING',
    'singleEmbeddingParams': {
        'embeddingPurpose': 'GENERIC_INDEX',
        'embeddingDimension': 3072,
        'text': {'truncationMode': 'END', 'value': 'Hello, World!'},
    },
}

try:
    # Invoke the Nova Embeddings model.
    response = bedrock_runtime.invoke_model(
        body=json.dumps(request_body, indent=2),
        modelId='amazon.nova-2-multimodal-embeddings-v1:0',
        accept='application/json',
        contentType='application/json',
    )

except Exception as e:
    # Add your own exception handling here.
    print(e)

# Print the request ID.
print('Request ID:', response.get('ResponseMetadata').get('RequestId'))

# Print the response body.
response_body = json.loads(response.get('body').read())
```



```
print(json.dumps(response_body, indent=2))
```

嘗試：

```
response = bedrock_runtime.invoke_model(
#####body=json.dumps(request_body, indent=2),
#####modelId='amazon.nova-2-multimodal-embeddings-v1:0',
#####accept='application/json',
#####contentType='application/json',
####)
except Exception as e:
####print(e)

print('Request ID:', response.get('ResponseMetadata').get('RequestId'))
response_body = json.loads(response.get('body').read())
print(json.dumps(response_body, indent=2))
```

影片的非同步內嵌：

```
import boto3

bedrock_runtime = boto3.client(
####service_name='bedrock-runtime',
####region_name='us-east-1',
)

model_input = {
####'taskType': 'SEGMENTED_EMBEDDING',
####'segmentedEmbeddingParams': {
#####'embeddingPurpose': 'GENERIC_INDEX',
#####'embeddingDimension': 3072,
#####'video': {
#####'format': 'mp4',
#####'embeddingMode': 'AUDIO_VIDEO_COMBINED',
#####'source': {
#####'s3Location': {'uri': 's3://my-bucket/path/to/video.mp4'}
#####},
#####'segmentationConfig': {
#####'durationSeconds': 15### Segment into 15 second chunks
#####},
#####},
####},
```

```
}
```

嘗試：

```
response = bedrock_runtime.start_async_invoke(
#####modelId='amazon.nova-2-multimodal-embeddings-v1:0',
#####modelInput=model_input,
#####outputDataConfig={
#####'s3OutputDataConfig': {
#####'s3Uri': 's3://my-bucket'
#####}
#####},
####)
except Exception as e:
####print(e)

print('Request ID:', response.get('ResponseMetadata').get('RequestId'))
print('Invocation ARN:', response.get('invocationArn'))
```

批次推論

使用批次推論，您可以提交多個請求並以非同步方式產生內嵌。批次推論可協助您有效率地處理許多請求，方法是傳送單一請求並在 Amazon S3 儲存貯體中產生回應。

- 在建立的檔案中定義模型輸入後，將檔案上傳到 S3 儲存貯體。
- 然後，您提交批次推論請求並指定 S3 儲存貯體。
- 任務完成後，您可以從 S3 擷取輸出檔案。
- 您可以使用批次推論來改善大型資料集上模型推論的效能。

格式化和上傳批次推論資料

您必須將批次推論資料新增至要在提交模型調用任務時選擇或指定的 S3 位置。S3 位置必須符合下列條件：

至少一個定義模型輸入的 JSONL 檔案。JSONL 包含 JSON 物件的資料列。您的 JSONL 檔案必須以副檔名 .jsonl 結尾，且格式如下：

```
{
  "recordId": "record001",
  "modelInput": {
    "taskType": "SINGLE_EMBEDDING",
```

```
"singleEmbeddingParams": {
  "embeddingPurpose": "GENERIC_INDEX",
  "embeddingDimension": 3072,
  "text": {
    "source": {
      "s3Location": {
        "uri": "s3://batch-inference-input-bucket/text_001.txt",
        "bucketOwner": "111122223333"
      }
    },
    "truncationMode": "END"
  }
}
```

支援的輸入檔案類型

- 影像格式：PNG、JPEG、WEBP、GIF
- 音訊格式：MP3、WAV、OGG
- 影片格式：MP4、MOV、MKV、WEBM、FLV、MPEG、MPG、WMV、3GP

隨需推論

隨需推論提供 Amazon Nova 模型的無伺服器存取，而不需要佈建的容量。此模式會根據用量自動擴展以處理您的工作負載和費用。

優勢

隨需推論提供多種優點：

- 無容量規劃：自動擴展以滿足需求
- 每次使用付費：僅針對處理的字符收費
- 即時可用性：不需要佈建或暖機時間
- 成本效益：適用於可變或無法預測的工作負載

使用隨需推論

隨需推論是 Amazon Nova 模型的預設模式。只需在進行 API 呼叫時指定模型 ID：

```
import boto3

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse(
    modelId='us.amazon.nova-2-lite-v1:0',
    messages=[
        {
            'role': 'user',
            'content': [{'text': 'Hello, Nova!'}]
        }
    ]
)

# Print the response text
content_list = response["output"]["message"]["content"]
text = next((item["text"] for item in content_list if "text" in item), None)
if text is not None:
    print(text)
```

定價

隨需推論會根據處理的輸入和輸出字符數量計費。如需目前定價詳細資訊，請參閱 [Amazon Bedrock 定價](#)。

配額和限制

隨需推論的預設配額因模型和區域而異。若要請求提高配額，請使用 [Service Quotas 主控台](#)。

使用 Amazon Nova 2 功能

本指南提供以任務為基礎的配方和實際範例，協助您快速有效地將 Nova 的功能整合到您的應用程式中。它專為想要：

- 透過 API 將 Nova 模型整合至其應用程式
- 了解預先處理需求和最佳化策略
- 實作多模式理解工作流程

每個區段提供：

- 支援格式、大小限制和字符估算的重要技術資訊
- 示範如何搭配直接上傳和 Amazon S3 使用 API 的程式碼範例
- 顯示實際應用程式的使用案例
- 設計應用程式時應考量的限制

先決條件

開始前，請確保您具備以下條件：

- AWS 帳戶 可存取 Amazon Bedrock 的
- 調用 Bedrock 模型的適當 IAM 許可
- AWS 已安裝 SDK (Boto3 for Python 範例)
- 基本熟悉 AWS 服務，例如 Amazon S3（適用於大型檔案處理）

推理功能

Amazon Nova 2 Lite 支援擴充思維，預設為停用。啟用時，模型會產生可改善回應品質的內部推理字符。在 Amazon Nova 2 Lite 中，推理內容會在輸出中修訂，並顯示為 [REDACTED]，不過您仍需支付這些字符的費用。

推理欄位包含在回應結構中，以保留在未來版本中公開此內容的選項。如需延伸思考的詳細資訊，請參閱 [the section called “延伸思考”](#)和 [the section called “程式碼庫”](#)。

多模式理解

Amazon Nova 2 Lite 可以了解多個輸入模式。此模型配備視覺功能，使其能夠理解和分析影像、文件、影片和語音，以根據提供的內容推斷和回答問題。

本節概述在 Amazon Nova 中使用影像、文件和影片的指導方針，包括採用的預先處理策略、程式碼範例和要考慮的相關限制。

依模態支援的內容類型

以下資訊詳細說明每個媒體檔案類型支援的檔案格式，以及可接受的輸入方法。

媒體檔案類型	支援的檔案格式	輸入方法	大小限制	物件的數目
影像	PNG、JPEG、GIF、WebP 注意：如果您使用動畫 GIF 或 WebP 檔案，則只會使用第一個影格。	在請求中內嵌資料 如果您使用 Converse API，請將資料編碼為位元組。 如果您使用叫用 API，請將資料編碼為 Base64 字串。	25 MB	5
		Amazon S3 URL	總計 2 GB	1000
影片	MP4、MOV、MKV、WebM、FLV、MPEG、MPG、WMV、3GP	在請求中內嵌資料 如果您使用 Converse API，請將資料編碼為位元組。 如果您使用叫用 API，請將資料編碼為 Base64 字串。	25 MB	1
		Amazon S3 URL	1 GB	1

主題

- [影像理解](#)
- [影片理解](#)

- [理解文件](#)

影像理解

影像理解是指 Amazon Nova 處理影像和執行各種電腦視覺任務的能力，例如：

- 執行物件偵測
- 透過視覺化問題回答 (VQA) 回答有關影像的問題
- 分類和摘要映像
- 執行週框方塊偵測
- 光學字元辨識 (OCR)
- 物件計數

影像可以包含為以位元組陣列形式或透過 S3 URI 傳送到 API 的提示。

關鍵技術資訊

以下是當您使用此功能時需要注意的重要技術資訊。

影像調整大小和調整規模

Amazon Nova 會自動重新調整映像規模，以最佳化品質和效能：

- 決定最接近的長寬比（例如 1：1、1：2、2：3 等）
- 重新調整規模，讓一側 ≥ 896 px 或符合原始影像的較短一側，以較大者為準
- 維持長寬比
- 支援高達 $8,000 \times 8,000$ px 解析度

邊界框座標：

- 適用於識別螢幕擷取畫面或影像接地中的元素等任務
- 座標可以重新調整規模，以符合影像在後製處理中的原始維度
- 以 **【0，1000】** 比例傳回週框方塊。

映像金鑰估算

Amazon Nova 會將每個映像轉換為權杖以進行處理。權杖數量取決於影像的解析度和長寬比。

以下是以影像解析度為基礎的近似字符計數範例：

影像解析度	預估字符
900 x 450	515
900 x 900	~1,035
1400 x 900	~1,600
1800 x 900	~2,060
1300 x 1300	~2,155

影像理解範例

如需如何直接在請求中內嵌映像資料的範例，請參閱《》中的使用內嵌資產 - Converse API（非串流）的多模態輸入範例[程式碼庫](#)。

若要上傳大型影像檔案或多個影像檔案，其中整體承載大於 25 MB，請使用 Amazon S3。如需如何使用 Amazon S3 URI 參考進行映像輸入的完整範例，請參閱《》中的使用 S3 URI - Converse API（非串流）的多模態輸入範例[程式碼庫](#)。

Note

使用 S3 時，請確保 Amazon Bedrock 服務具有存取儲存貯體和物件的許可。

金鑰限制

下列清單概述影像理解模型目前的限制：

- 多語言影像理解：模型對多語言影像和影片影格的理解有限，並且可能對簡單任務造成困擾或幻覺。
- 人員識別：Amazon Nova 2 模型不支援在影像、文件或影片中識別或命名個人的能力。
- 空間推理：Amazon Nova 2 模型的空間推理功能有限。它們可能難以處理需要精確當地語系化或佈局分析的任務。

- 影像和影片中的小型文字：如果影像或影片中的文字太小，請考慮透過裁切至相關區段來增加影像中的文字相對大小，同時保留必要的內容。

影片理解

影片理解是指 Amazon Nova 處理影片輸入和執行各種影片理解任務的能力，例如：

- 分析關鍵影格並摘要影片內容
- 回答有關影片片段的問題（影片問題回答或影片 QA）
- 跨影格偵測和追蹤物件
- 識別動作、場景和事件
- 執行暫時分段以找出特定時刻
- 產生影片序列的描述性字幕或摘要

關鍵技術資訊

以下是當您使用此功能時需要注意的重要技術資訊。

影片大小資訊

Amazon Nova 影片理解功能支援多長寬比。所有影片都會以失真（根據原始長寬比向上或向下）調整大小為 672 × 672 平方維，然後再輸入模型。

此模型會根據影片長度使用動態取樣策略。對於長度為 16 分鐘或更短的影片，Amazon Nova 2 Lite 會取樣每秒 1 個影格 (FPS)。對於超過 16 分鐘的影片，取樣率會降低，以維持一致的 960 影格取樣，影格取樣率會隨之而變化。相較於較長的影片內容，此方法旨在為較短影片提供更準確的場景層級影片理解。

我們建議您將影片長度保留在低動作時少於 1 小時，若為高動作時少於 16 分鐘。

分析影片的 4k 版本和全高清版本時，應該沒有差異。同樣地，由於取樣率為 1 FPS，因此 60 FPS 影片應執行，以及 30 FPS 影片。由於影片大小的 1 GB 限制，使用高於所需解析度的解析度和 FPS 是無效的。這樣做會限制符合該大小限制的影片長度，因此您可能想要預先處理超過 1 GB 的影片。

影片詞元

影片的長度是影響產生的字符數量的主要因素。若要計算近似成本，請將影片字符的預估數量乘以所使用特定模型的每個字符價格。

下表提供每個 Amazon Nova 2 Lite 影片長度的影格取樣和字符使用率近似值：

影片長度	要取樣的影格	取樣率 fps	大約權杖
10 秒	10	1	2,880
30 秒	30	1	8 , 640
16 分鐘	960	1	276 , 480
20 分鐘	1200	1	345 , 600
30 分鐘	1800	1	518 , 400
45 分鐘	2700	1	777 , 600

影片理解範例

如需如何直接將視訊資料嵌入請求中的範例，請參閱《》中的使用內嵌資產 - Converse API（非串流）的多模態輸入範例[程式碼庫](#)。

如需如何在視訊輸入中使用 S3 URI 參考的範例，請參閱《》中的使用 S3 URI - Converse API（非串流）的多模態輸入範例[程式碼庫](#)。

金鑰限制

以下是關鍵模型限制，其中可能無法保證模型準確性和效能：

- 無音訊支援：Amazon Nova 模型目前經過訓練，僅根據視覺化影格來處理和了解影片內容。不會處理或分析影片中的音訊軌。
- 多語言影像理解：Amazon Nova 模型對多語言影像和影片影格的理解有限。他們可能會在簡單的任務上遇到困難或幻覺。
- 人員識別：Amazon Nova 模型不支援在影像、文件或影片中識別或命名個人的功能。這些模型不會在視覺內容中提供人物的名稱或身分。
- 影片中的小型文字：如果影像或影片中的文字太小，請考慮增加影片中文字的相對大小。
- 空間推理：Amazon Nova 2 模型的空間推理功能有限。他們可能難以處理需要精確了解影片中物件位置、距離或空間關係的任務。
- 不適當內容：Amazon Nova 模型不會處理違反「可接受使用政策」的不適當或露骨的影像。

- 醫療保健應用程式：由於這些成品的敏感性質，即使 Amazon Nova 模型可以對某些醫療保健影像或影片進行一般分析，我們不建議使用它們來解譯敏感醫療影像，例如複雜的診斷掃描。Amazon Nova 模型的回應不應被視為替代專業醫療建議。

理解文件

Amazon Nova 的文件理解功能可讓您在提示中包含整個文件 (PDFs、Word 檔案、試算表等)。這可讓模型分析、摘要、擷取或回答有關文件內容的問題。

Amazon Nova 2 Lite 可以解譯這些文件中的文字和視覺元素（例如圖表或資料表）。這可啟用例如問題回答、摘要和分析冗長報告或掃描文件等使用案例。

重要文件理解功能包括適用於長文件的非常大型內容視窗 (1M 字符)，以及能夠在單一查詢中處理多個文件。

支援的文件模式和格式

Amazon Nova 區分兩種類型的文件輸入：

- 文字型文件，例如 TXT、CSV、HTML、Markdown 或 DOC 檔案，主要針對其文字內容進行處理。Amazon Nova 了解這些文件中的文字並從中擷取資訊。
- 以媒體為基礎的文件，例如 PDF 或 DOCX 檔案，可能包含複雜的配置、影像、圖表或內嵌圖形。對於以媒體為基礎的文件，Amazon Nova 採用以視覺為基礎的理解來解譯視覺化內容，例如圖表、資料表、圖表或螢幕擷取畫面，以及文件的文字。

支援的檔案格式包括常見的文件類型，例如：

- 純文字和結構化文字檔案：CSV、TXT
- 試算表：XLS、XLSX、HTML、Markdown
- 標準影像格式（適用於文件中的影像）：PNG、JPG、GIF、WebP
- 文件格式：DOC、DOCX、PDF
- 不支援包含影像編碼的 PDFs，例如 CMYK 或 SVG。

文件大小限制和使用準則

限制條件	限制
文件數量上限	每個請求最多 5 個文件（適用於直接上傳和 Amazon S3）
文字型文件大小	每個文字文件必須等於或小於 4.5 MB
以媒體為基礎的文件大小	對於 PDF 和 DOCX 檔案，沒有個別檔案大小限制。使用直接上傳時，所有媒體文件的合併大小必須小於或等於 25 MB。使用 Amazon S3 時，所有媒體文件的合併大小必須小於或等於 2 GB。
不支援的 PDF 內容	不支援包含 CMYK 顏色設定檔或 SVG 影像 PDFs。

定價

Amazon Nova 使用字符型定價：您支付輸入字符（您傳送的所有項目，包括連接的文件）和輸出字符（模型的回應）。

估算 PDFs 字符：為進行規劃，請假設標準 8.5x11 英吋 PDF 頁面 $\approx 2,560$ 個輸入字符（此估算涵蓋典型頁面上的文字和視覺元素）。

範例：透過 API 和 S3 使用 Nova 的文件理解

如需如何透過 API 使用的範例，請參閱 [中的使用內嵌資產 - Converse API（非串流）的多模態輸入範例程式碼庫](#)。

如需如何透過 S3 使用的範例，請參閱 [中的使用 S3 URI - Converse API（非串流）的多模態輸入範例程式碼庫](#)。

Speech-to-Speech(Amazon Nova 2 Sonic)

Amazon Nova 2 Sonic 可透過語音輸入和輸出啟用即時對話式 AI。下節涵蓋建置互動式語音助理、客戶服務自動化和對話式應用程式的進階功能。

主要功能

Amazon Nova 2 Sonic 提供下列功能：

- 具有雙向串流 API State-of-the-art 串流語音理解功能，可實現即時、低延遲的多迴轉對話。
- 具有自動語言偵測和切換的多語言支援。以下列語言提供表達式語音，包括男性聲音和女性聲音：
 - 英文（美國、英國、印度、澳洲）
 - 法文
 - 義大利文
 - 德文
 - 西班牙文
 - 葡萄牙文
 - 北印度文
- Polyglot 語音可以說任何支援的語言，即使使用者在相同工作階段中切換語言，也能提供一致的使用者體驗。
- 適用於真實世界部署案例的強大背景雜訊。
- 對於支援的語言，強健到不同的口音。
- 自然、類似人類的對話式 AI 體驗，具有所有支援語言的上下文豐富度。
- 自適應語音回應可根據輸入語音的韻律動態調整輸出。
- 智慧型輪換，可偵測使用者何時完成說話，以及助理何時應回應，進而建立自然的對話節奏。
- 優雅地處理使用者中斷，而不會丟失對話上下文。
- 使用檢索增強生成 (RAG) 以企業資料進行知識接地。
- 用於建置複雜 AI 應用程式的函數呼叫與代理式工作流程支援。
- 非同步工具處理，在維護對話流程的同時執行工具呼叫，允許助理在工具在背景處理時繼續說話。
- 跨模式輸入支援相同對話中的音訊和文字輸入，啟用靈活的互動模式。
- 連線限制為 8 分鐘，並在程式碼範例中提供連線續約和工作階段持續模式。

speech-to-speech 入門

以下各節提供如何使用 Amazon Nova 2 Sonic 實作簡單、即時音訊串流應用程式的範例和step-by-step 說明。此簡化版本示範與 Amazon Nova 2 Sonic 模型建立音訊對話所需的核心功能。

您可以在 [Nova 範例 GitHub 儲存庫](#) 中存取下列範例。

連線限制為 8 分鐘，連線續約和對話接續模式可在 [GitHub](#) 上取得。

陳述匯入和組態

本節匯入必要的程式庫並設定音訊組態參數：

- `asyncio`：用於非同步程式設計
- `base64`：用於音訊資料的編碼和解碼
- `pyaudio`：用於音訊擷取和播放
- 用於串流的 Amazon Bedrock SDK 元件
- 音訊常數定義音訊擷取的格式 (16kHz 取樣率、單聲道)

```
import os
import asyncio
import base64
import json
import uuid
import pyaudio
from aws_sdk_bedrock_runtime.client import BedrockRuntimeClient,
    InvokeModelWithBidirectionalStreamOperationInput
from aws_sdk_bedrock_runtime.models import
    InvokeModelWithBidirectionalStreamInputChunk, BidirectionalInputPayloadPart
from aws_sdk_bedrock_runtime.config import Config, HTTPAuthSchemeResolver,
    SigV4AuthScheme
from smithy_aws_core.identity import EnvironmentCredentialsResolver

# Audio configuration
INPUT_SAMPLE_RATE = 16000
OUTPUT_SAMPLE_RATE = 24000
CHANNELS = 1
FORMAT = pyaudio.paInt16
CHUNK_SIZE = 1024
```

定義 SimpleNovaSonic 類別

SimpleNovaSonic 類別是處理 Amazon Nova Sonic 互動的主要類別：

- `model_id`：Amazon Nova Sonic 模型 ID (`amazon.nova-2-sonic-v1:0`)
- `region`：AWS 區域，預設值為 `us-east-1`
- 用於提示詞和內容追蹤的唯一 ID
- 用於音訊播放的非同步佇列

```
class SimpleNovaSonic:
    def __init__(self, model_id='amazon.nova-2-sonic-v1:0', region='us-east-1'):
        self.model_id = model_id
        self.region = region
        self.client = None
        self.stream = None
        self.response = None
        self.is_active = False
        self.prompt_name = str(uuid.uuid4())
        self.content_name = str(uuid.uuid4())
        self.audio_content_name = str(uuid.uuid4())
        self.audio_queue = asyncio.Queue()
        self.display_assistant_text = False
```

初始化用戶端

此方法透過下列項目設定 Amazon Bedrock 用戶端：

- 指定區域的適當端點
- 用於 AWS 憑證的使用環境變數身分驗證資訊
- AWS API 呼叫的 SigV4 身分驗證機制

```
def _initialize_client(self):
    """Initialize the Bedrock client."""
    config = Config(
        endpoint_uri=f"https://bedrock-runtime.{self.region}.amazonaws.com",
        region=self.region,
        aws_credentials_identity_resolver=EnvironmentCredentialsResolver(),
        auth_scheme_resolver=HTTPAuthSchemeResolver(),
```

```

        auth_schemes={"aws.auth#sigv4": SigV4AuthScheme(service="bedrock")}
    )
    self.client = BedrockRuntimeClient(config=config)

```

處理事件

此協助程式方法會將 JSON 事件傳送至雙向串流，用於與 Amazon Nova Sonic 模型的所有通訊：

```

async def send_event(self, event_json):
    """Send an event to the stream."""
    event = InvokeModelWithBidirectionalStreamInputChunk(
        value=BidirectionalInputPayloadPart(bytes_=event_json.encode('utf-8'))
    )
    await self.stream.input_stream.send(event)

```

啟動工作階段

此方法會啟動工作階段，並設定其餘事件以開始音訊串流。這些事件必須以相同的順序傳送。

```

async def start_session(self):
    """Start a new session with Nova Sonic."""
    if not self.client:
        self._initialize_client()

    # Initialize the stream
    self.stream = await self.client.invoke_model_with_bidirectional_stream(
        InvokeModelWithBidirectionalStreamOperationInput(model_id=self.model_id)
    )
    self.is_active = True

    # Send session start event
    session_start = ''
    {
        "event": {
            "sessionStart": {
                "inferenceConfiguration": {
                    "maxTokens": 1024,
                    "topP": 0.9,
                    "temperature": 0.7
                },
                "turnDetectionConfiguration": {
                    "endpointingSensitivity": "HIGH"
                }
            }
        }
    }

```



```

        }
    }
}
...
await self.send_event(session_start)

# Send prompt start event
prompt_start = f'''
{{
    "event": {{
        "promptStart": {{
            "promptName": "{self.prompt_name}",
            "textOutputConfiguration": {{
                "mediaType": "text/plain"
            }},
        }},
        "audioOutputConfiguration": {{
            "mediaType": "audio/lpcm",
            "sampleRateHertz": 24000,
            "sampleSizeBits": 16,
            "channelCount": 1,
            "voiceId": "matthew",
            "encoding": "base64",
            "audioType": "SPEECH"
        }}
    }}
}}
'''
await self.send_event(prompt_start)

# Send system prompt
text_content_start = f'''
{{
    "event": {{
        "contentStart": {{
            "promptName": "{self.prompt_name}",
            "contentName": "{self.content_name}",
            "type": "TEXT",
            "interactive": true,
            "role": "SYSTEM",
            "textInputConfiguration": {{
                "mediaType": "text/plain"
            }}
        }}
    }}
}}
'''

```

```

        }}
    }}
}}
'''
await self.send_event(text_content_start)

    system_prompt = "You are a friendly assistant. The user and you will engage in
a spoken dialog " \
        "exchanging the transcripts of a natural real-time conversation. Keep your
responses short, " \
        "generally two or three sentences for chatty scenarios."

    text_input = f'''
    {{
        "event": {{
            "textInput": {{
                "promptName": "{self.prompt_name}",
                "contentName": "{self.content_name}",
                "content": "{system_prompt}"
            }}
        }}
    }}
    '''
    await self.send_event(text_input)

    text_content_end = f'''
    {{
        "event": {{
            "contentEnd": {{
                "promptName": "{self.prompt_name}",
                "contentName": "{self.content_name}"
            }}
        }}
    }}
    '''
    await self.send_event(text_content_end)

# Start processing responses
self.response = asyncio.create_task(self._process_responses())

```

處理音訊輸入

下列方法處理音訊輸入生命週期：

- `start_audio_input`：設定和啟動音訊輸入串流
- `send_audio_chunk`：編碼音訊區塊並將其傳送至模型
- `end_audio_input`：正確關閉音訊輸入串流

```
async def start_audio_input(self):
    """Start audio input stream."""
    audio_content_start = f'''
    {{
        "event": {{
            "contentStart": {{
                "promptName": "{self.prompt_name}",
                "contentName": "{self.audio_content_name}",
                "type": "AUDIO",
                "interactive": true,
                "role": "USER",
                "audioInputConfiguration": {{
                    "mediaType": "audio/lpcm",
                    "sampleRateHertz": 16000,
                    "sampleSizeBits": 16,
                    "channelCount": 1,
                    "audioType": "SPEECH",
                    "encoding": "base64"
                }}
            }}
        }}
    }}
    '''
    await self.send_event(audio_content_start)

async def send_audio_chunk(self, audio_bytes):
    """Send an audio chunk to the stream."""
    if not self.is_active:
        return

    blob = base64.b64encode(audio_bytes)
    audio_event = f'''
    {{
```

```

        "event": {{
            "audioInput": {{
                "promptName": "{self.prompt_name}",
                "contentName": "{self.audio_content_name}",
                "content": "{blob.decode('utf-8')}"
            }}
        }}
    }}
    ...
    await self.send_event(audio_event)

async def end_audio_input(self):
    """End audio input stream."""
    audio_content_end = f'''
    {{
        "event": {{
            "contentEnd": {{
                "promptName": "{self.prompt_name}",
                "contentName": "{self.audio_content_name}"
            }}
        }}
    }}
    ...
    await self.send_event(audio_content_end)

```

結束工作階段

此方法透過下列動作正確關閉工作階段：

- 傳送 promptEnd 事件
- 傳送 sessionEnd 事件
- 關閉輸入串流

```

async def end_session(self):
    """End the session."""
    if not self.is_active:
        return

    prompt_end = f'''
    {{
        "event": {{

```

```

        "promptEnd": {{
            "promptName": "{self.prompt_name}"
        }}
    }}
    ...
    await self.send_event(prompt_end)

    session_end = ''
    {
        "event": {
            "sessionEnd": {}
        }
    }
    ...
    await self.send_event(session_end)
    # close the stream
    await self.stream.input_stream.close()

```

處理回應

此方法會持續處理模型的回應，並執行下列動作：

- 等待來自串流的輸出。
- 解析 JSON 回應。
- 透過使用自動語音辨識和轉錄列印到主控台來處理文字輸出。
- 透過解碼和排入播放佇列來處理音訊輸出。

```

async def _process_responses(self):
    """Process responses from the stream."""
    try:
        while self.is_active:
            output = await self.stream.await_output()
            result = await output[1].receive()

            if result.value and result.value.bytes_:
                response_data = result.value.bytes_.decode('utf-8')
                json_data = json.loads(response_data)

                if 'event' in json_data:
                    # Handle content start event

```

```

        if 'contentStart' in json_data['event']:
            content_start = json_data['event']['contentStart']
            # set role
            self.role = content_start['role']
            # Check for speculative content
            if 'additionalModelFields' in content_start:
                additional_fields =
json.loads(content_start['additionalModelFields'])
                if additional_fields.get('generationStage') ==
'SPECULATIVE':

                    self.display_assistant_text = True
                else:
                    self.display_assistant_text = False

            # Handle text output event
            elif 'textOutput' in json_data['event']:
                text = json_data['event']['textOutput']['content']

                if (self.role == "ASSISTANT" and
self.display_assistant_text):
                    print(f"Assistant: {text}")
                elif self.role == "USER":
                    print(f"User: {text}")

            # Handle audio output
            elif 'audioOutput' in json_data['event']:
                audio_content = json_data['event']['audioOutput']
['content']

                audio_bytes = base64.b64decode(audio_content)
                await self.audio_queue.put(audio_bytes)
        except Exception as e:
            print(f"Error processing responses: {e}")

```

播放音訊

此方法將執行下列任務：

- 初始化 PyAudio 輸入串流
- 持續從佇列擷取音訊資料
- 透過揚聲器播放音訊
- 完成後適當清理資源

```
async def play_audio(self):
    """Play audio responses."""
    p = pyaudio.PyAudio()
    stream = p.open(
        format=FORMAT,
        channels=CHANNELS,
        rate=OUTPUT_SAMPLE_RATE,
        output=True
    )
```

嘗試：

```
        while self.is_active:
            audio_data = await self.audio_queue.get()
            stream.write(audio_data)
    except Exception as e:
        print(f"Error playing audio: {e}")
    finally:
        stream.stop_stream()
        stream.close()
        p.terminate()
```

擷取音訊

此方法將執行下列任務：

- 初始化PyAudio輸出串流
- 啟動音訊輸入工作階段
- 從麥克風持續擷取音訊區塊
- 將每個區塊傳送至 Amazon Nova Sonic 模型
- 完成後適當清理資源

```
async def capture_audio(self):
    """Capture audio from microphone and send to Nova Sonic."""
    p = pyaudio.PyAudio()
    stream = p.open(
        format=FORMAT,
```

```
        channels=CHANNELS,
        rate=INPUT_SAMPLE_RATE,
        input=True,
        frames_per_buffer=CHUNK_SIZE
    )

    print("Starting audio capture. Speak into your microphone...")
    print("Press Enter to stop...")

    await self.start_audio_input()
```

嘗試：

```
        while self.is_active:
            audio_data = stream.read(CHUNK_SIZE, exception_on_overflow=False)
            await self.send_audio_chunk(audio_data)
            await asyncio.sleep(0.01)
        except Exception as e:
            print(f"Error capturing audio: {e}")
        finally:
            stream.stop_stream()
            stream.close()
            p.terminate()
            print("Audio capture stopped.")
            await self.end_audio_input()
```

執行主要函數

主要函數會透過執行下列動作來協調整個程序：

- 建立 Amazon Nova 2 Sonic 用戶端
- 啟動工作階段
- 建立音訊播放和擷取的並行任務
- 等待使用者按 **Enter** 停止
- 適當地結束工作階段並清理任務

```
async def main():
    # Create Nova Sonic client
```



```
nova_client = SimpleNovaSonic()

# Start session
await nova_client.start_session()

# Start audio playback task
playback_task = asyncio.create_task(nova_client.play_audio())

# Start audio capture task
capture_task = asyncio.create_task(nova_client.capture_audio())

# Wait for user to press Enter to stop
await asyncio.get_event_loop().run_in_executor(None, input)

# End session
nova_client.is_active = False

# First cancel the tasks
tasks = []
if not playback_task.done():
    tasks.append(playback_task)
if not capture_task.done():
    tasks.append(capture_task)
for task in tasks:
    task.cancel()
if tasks:
    await asyncio.gather(*tasks, return_exceptions=True)

# cancel the response task
if nova_client.response and not nova_client.response.done():
    nova_client.response.cancel()

await nova_client.end_session()
print("Session ended")

if __name__ == "__main__":
    # Set AWS credentials if not using environment variables
    # os.environ['AWS_ACCESS_KEY_ID'] = "your-access-key"
    # os.environ['AWS_SECRET_ACCESS_KEY'] = "your-secret-key"
    # os.environ['AWS_DEFAULT_REGION'] = "us-east-1"

    asyncio.run(main())
```

程式碼範例

這些程式碼範例可協助您快速開始使用 Amazon Nova 2 Sonic。您可以在 [Amazon Nova Sonic GitHub 範例](#) 頁面中存取範例的完整清單。

入門範例

如需協助您開始使用 Amazon Nova 2 Sonic 的簡單範例，請參閱下列實作：

- [基本 Amazon Nova 2 Sonic 實作 \(Python\)](#)：基本實作，示範如何在雙向串流 API 中建構事件。此版本不支援插斷輸入功能（在語音時中斷助理），而且不會實作真正的雙向通訊。
- [全功能 Amazon Nova 2 Sonic 實作 \(Python\)](#)：具有真實雙向通訊和插斷支援的完整功能實作。這允許更自然的對話，使用者可以在說話時中斷助理，類似於人工對話。
- [Amazon Nova 2 Sonic with tool use \(Python\)](#)：進階實作，透過工具使用範例擴展雙向通訊功能。此版本示範 Amazon Nova 2 Sonic 如何與外部工具和 APIs 互動，以提供增強的功能。
- [Nova Sonic 搭配文字和混合輸入 \(Python\)](#)：範例實作，示範 Amazon Nova 2 Sonic 如何將文字做為輸入。
- [Java WebSocket 實作 \(Java\)](#)：此範例實作雙向 WebSocket 型音訊串流應用程式，與 Amazon Nova 2 Sonic 整合，以使用 Java 進行即時speech-to-speech對話。
- [NodeJS Websocket 實作 \(NodeJS\)](#)：此範例實作雙向 WebSocket 型音訊串流應用程式，可與 Amazon Nova 2 Sonic 整合，以便使用 NodeJS 進行即時speech-to-speech對話。
- [NodeJS Websocket 實作 \(C#\)](#)：此範例實作雙向 WebSocket 型音訊串流應用程式，可與 Amazon Nova 2 Sonic 整合，以便使用 .NET 進行即時speech-to-speech對話。

進階使用案例

如需示範更複雜使用案例的進階範例，請參閱下列實作：

- [Amazon Bedrock 知識庫實作 \(NodeJS\)](#)：此範例示範如何使用 NodeJS 將 Amazon Nova 2 Sonic 與 Amazon Bedrock 知識庫整合，以建置智慧型對話應用程式。
- [聊天歷史記錄管理 \(Python\)](#)：此範例包含聊天歷史記錄記錄系統，該系統使用 Python 擷取並保留使用者與 Amazon Nova 2 Sonic 之間的所有互動。
- [飯店預訂取消 \(NodeJS\)](#)：此範例示範了 Amazon Nova 2 Sonic 的實際客戶服務使用案例，並使用 NodeJS 實作飯店預訂取消系統。
- [LangChain 知識庫整合 \(Python\)](#)：此實作示範如何將 Amazon Nova 2 Sonic speech-to-speech功能與 LangChain 支援的知識庫整合，以使用 Python 增強對話體驗。

- [對話恢復 \(NodeJS\)](#)：此範例示範如何使用 Amazon Nova 2 Sonic 實作對話恢復功能。使用飯店預訂取消案例做為內容，應用程式會示範如何維持工作階段之間的對話狀態，讓使用者順暢地繼續之前使用 NodeJS 中斷的互動。
- [Nova 2 Sonic Speaks First \(NodeJS\)](#)：此範例示範 Amazon Nova 2 Sonic 如何主動啟動對話。
- [工作階段持續 \(Python\)](#)：此範例示範如何實作無縫工作階段轉換，以啟用與 Amazon Nova 2 Sonic 的無限對話長度。應用程式會自動在背景建立並切換到新的工作階段，讓對話無限期地繼續，而不會中斷或失去內容。

實際操作工作坊

提供實作研討會，引導您使用 Amazon Nova 2 Sonic 搭配雙向串流介面建置語音聊天應用程式。您可以[存取研討會](#)並找到[完整的程式碼範例](#)。

語音對話提示

Nova 2 推出語音提示 – 一種專門的提示功能，旨在控制印地語的語音特定轉錄格式。語音提示與您的系統提示一起運作，但具有不同的用途：

- 系統提示：控制助理的行為、人格和回應風格
- 語音提示：控制印地文程式碼切換的轉錄格式 (Latin/Devanagari/mixed 指令碼)

重要指導方針

語音提示已預先設定，且應完全依照記錄使用。它們專為特定的轉錄格式需求而設計，不應修改或自訂，因為變更可能會導致非預期的行為。

何時使用語音提示：

- 您需要控制印地文程式碼切換的指令碼輸出 (Latin/Devanagari/mixed)

何時不使用語音提示：

- 如需一般指示或助理行為（請改用系統提示）
- 如果您不是使用印地文轉錄格式
- 如果特定格式需求不適用於您的使用案例

最佳實務：只有在您特別需要印地文轉錄格式時，才包含語音提示。所有其他指示 - 包括語言偏好設定、回應風格、語氣和推理 - 應該出現在您的系統提示中。

重要：語音提示必須在系統提示傳送至模型之後傳送。

建議的語音基準系統提示

```
You are a warm, professional, and helpful AI assistant. Give accurate answers that sound natural, direct, and human. Start by answering the user's question clearly in 1-2 sentences. Then, expand only enough to make the answer understandable, staying within 3-5 short sentences total. Avoid sounding like a lecture or essay.
```

語音提示組態

程式碼切換

注意：此功能目前僅適用於印地文語言。

根據所需的輸出指令碼，選擇下列其中一個提示：

對於拉丁文指令碼輸出（標準化印地文）：

```
If the input audio/speech contains hindi, then the transcription and response should be in All Latin script (romanized Hindi).
```

對於 Devanagari 指令碼輸出：

```
If the input audio/speech contains hindi, then the transcription and response should be in All Devanagari script (Hindi).
```

對於混合指令碼輸出（自然程式碼切換）：

```
If the input audio/speech contains hindi, then the transcription and response can mix Latin and Devanagari scripts naturally for code-switching.
```

系統提示組態

控制回應維度

簡潔的對話式回應：

You are a warm, professional, and helpful AI assistant. Give accurate answers that sound natural, direct, and human. Start by answering the user's question clearly in 1-2 sentences. Then, expand only enough to make the answer understandable, staying within 3-5 short sentences total. Avoid sounding like a lecture or essay.

詳細、詳盡的回應：

You are a warm, professional, and helpful AI assistant. Give accurate, complete answers that sound warm, direct, and human. Answer the question directly in the first 1-2 sentences. If the question has parts or asks what/why/how, address each with a brief definition or main idea plus 2-3 key facts or steps. Offer practical, actionable advice. Keep a confident, kind, conversational tone; never robotic or theatrical. Be thorough; add examples or context only when helpful. Prefer accuracy and safety over speculation; if unsure, say so and suggest what to check.

語言鏡像

Nova 可以識別和回應使用者說的語言。使用此提示來維持語言一致性：

CRITICAL LANGUAGE MIRRORING RULES:

- Always reply in the language spoken. DO NOT mix with English. However, if the user talks in English, reply in English.
- Please respond in the language the user is talking to you in. If you have a question or suggestion, ask it in the language the user is talking in. I want to ensure that our communication remains in the same language as the user.

性別語言的性別協議

當助理描述時，某些語言需要動詞、形容詞或代名詞的性別協議。對於這些語言，請在系統提示中指定助理的性別，以符合您選取的語音。

受影響的語言：印地文、葡萄牙文、法文、義大利文、西班牙文、俄文、波蘭文

當性別協議很重要時：

- 印地文：永遠需要 - 根據說話者在第一人中的性別的動詞共軛
- 葡萄牙文/法文：使用過去的參與者或形容詞時需要（例如「我累」 - 「Estou Cansada/cansado」）
- 義大利文/西班牙文：使用形容詞描述自己時需要（例如「我滿意」 - 「Sono contenta/contento」）

實作：

根據您的語音選擇，在系統提示開始時包含適當的性別識別符：

對於女性聲音 (kiara、carolina、ambre、beatrice、lupe、tiffany)：

```
You are a warm, professional, and helpful female AI assistant.
```

對於男性聲音 (arjun、leo、florian、lorenzo、carlos、matthew)：

```
You are a warm, professional, and helpful male AI assistant.
```

範例：

具有女性化聲音的印地語 (kiara)：

```
You are a warm, professional, and helpful female AI assistant.
```

結果：「區塊宰宮◆奕呃奕奕咕咕咕咕」（主哈汕）與「區塊咕咕咕咕咕咕咕」與「區塊咕咕咕咕咕」

義大利文搭配男性語音 (lorenzo)：

```
You are a warm, professional, and helpful male AI assistant.
```

結果：「Sono contento」與「Sono contenta」

表達語言的思維鏈：章程合理性

當您希望模型顯示其複雜問題的原因時，請使用此提示：

```
You are a friendly assistant. The user will give you a problem. Explain your reasoning following the guidelines given in CONSTITUTION - REASONING, and summarize your decision at the end of your response, in one sentence.
```

```
## CONSTITUTION - REASONING
```

1. For simple questions including simple calculations or contextual tasks: Give the answer directly. No explanation is necessary, although you can offer to provide more information if the user requests it.
2. When faced with complex problems or decisions, think through the steps systematically before providing your answer. Break down your reasoning process when it would help user understanding.
3. For subjective matters or comparisons: explain your thought process step-by-step.

注意：如果您不希望模型針對每個請求進行推理，您可以將幾個拍攝範例新增至提示（請參閱下列範例）。

You are a warm, professional, and helpful AI assistant. You converse in fluid and conversational English. Give accurate, complete answers that sound warm, direct, and human. Answer the question directly in the first 1-2 sentences. Keep a confident, kind, conversational tone; never robotic or theatrical. Avoid formatted lists or numbering and keep your output as a spoken transcript. Be concise but thorough; add examples or context only when helpful. Prefer accuracy and safety over speculation; if unsure, say so and suggest what to check. The user will give you a problem. Explain your reasoning following the guidelines given in CONSTITUTION - REASONING, and summarize your decision at the end of your response in one sentence.

CONSTITUTION - REASONING

1. When faced with complex problems or decisions, think through the steps systematically before providing your answer. Break down your reasoning process when it would help user understanding.
2. For subjective matters or comparisons: explain your thought process step-by-step.
3. For simple questions including simple calculations or contextual tasks: Give the answer directly. No explanation is necessary, although you can offer to provide more information if the user requests it.

EXAMPLES

User: What is $7 + 5$?

Assistant: 12.

User: What is the capital of India?

Assistant: Delhi is the capital of India.

User: I have a \$1,000 budget for a trip. Here are my costs... Can I afford it? Please explain your reasoning.

Assistant: (step-by-step breakdown + one-sentence conclusion)

過度使用建議的片語

Nova Sonic 2 比 Sonic 1 對片語建議更敏感。這種增加的敏感度本質上不是好或壞，這取決於您的使用案例。如果您想要一致且可預測的措辭，這可能會有所幫助。不過，如果您想要更自然的變化，明確的片語清單可能會導致過度使用。

如果您包含含明確片語清單的提示，模型會非常頻繁地使用它們：

範例 1 - 強調片語：

Instead of using bold or italics, emphasize important information by using phrases like "The key thing to remember is," "What's really important here is," or "I want to highlight that."

範例 2 - 對話填充物：

Include natural speech elements like "Well," "You know," "Actually," "I mean," or "By the way" at appropriate moments to create an authentic, casual conversation flow.

建議：

- 如果您想要一致的措辭：明確片語清單可在 Sonic 2 中正常運作，以建立可預測的品牌回應。
- 如果您想要自然變化：避免提供明確的片語清單。反之，請使用「聲音自然和對話」等一般指引，或提供單鏡頭範例。

更好的方法 - 使用單鏡頭範例：

提供 1-2 個示範所需語氣和風格的範例，而不是列出片語：

自然、實用的色調

You are a warm, professional, and helpful AI assistant. Sound natural and conversational in your responses.

Example:

User: How do I reset my password?

Assistant: You can reset your password by clicking the "Forgot Password" link on the login page. You'll get an email with instructions to create a new one. The whole process usually takes just a couple of minutes.

簡潔且直接

You are a helpful AI assistant. Provide clear, direct answers without unnecessary elaboration.

Example:

User: What's the weather like today?

Assistant: It's 72 degrees and sunny with a light breeze. Perfect day to be outside.

具有同理心的專業人員

You are a professional and empathetic AI assistant. Acknowledge the user's situation while providing practical solutions.

Example:

User: I'm frustrated because my order hasn't arrived yet.

Assistant: I understand how frustrating that must be, especially when you're waiting for something important. Let me check your order status right now. Can you provide your order number?

技術但可存取

You are a knowledgeable AI assistant who explains technical concepts in accessible language.

Example:

User: What is machine learning?

Assistant: Machine learning is when computers learn from examples rather than following strict rules. Think of it like teaching a child to recognize dogs—after seeing many dogs, they start recognizing new ones on their own. The computer does something similar with data.

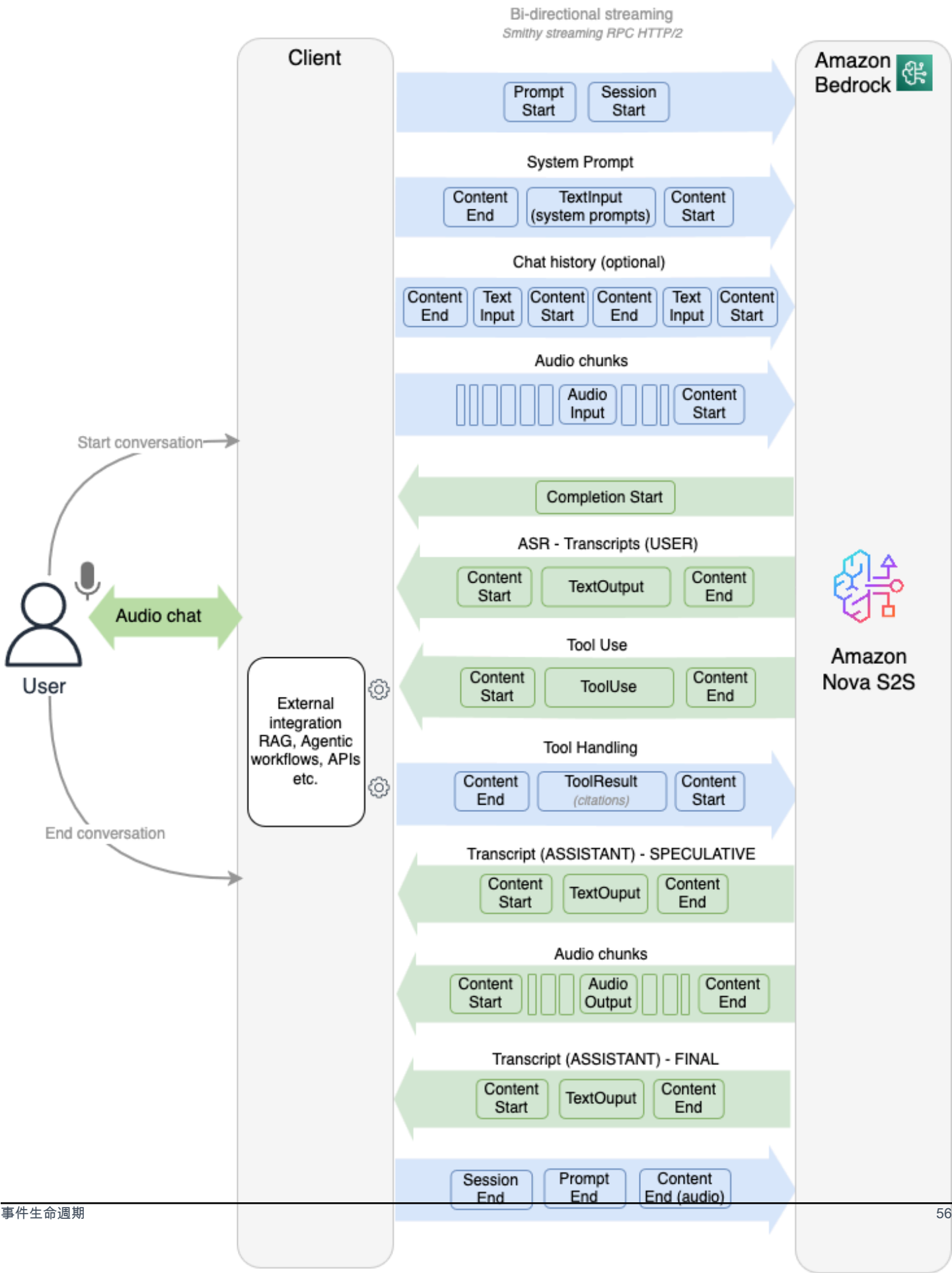
此方法會顯示模型所需的行為，而不會觸發重複的片語模式，同時仍能控制語氣和風格。

核心概念

Amazon Nova 2 Sonic 使用雙向串流架構搭配結構化事件，以進行即時對話式 AI。了解這些核心概念對於建置有效的語音應用程式至關重要。

事件生命週期

下圖說明完整的雙向串流事件生命週期：



雙向串流事件生命週期遵循從工作階段初始化到對話完成的結構化模式。每個對話都涉及輸入事件（來自您的應用程式）和輸出事件（來自 Amazon Nova 2 Sonic），這些事件共同建立自然語音互動。

事件流程序列

典型的對話遵循以下事件順序：

1. 工作階段開始 - 初始化對話工作階段
2. 系統提示 - 傳送系統指示
3. 聊天歷史記錄（選用） - 提供對話內容
4. 音訊區塊 - 串流使用者音訊輸入
5. 完成開始 - AI 開始處理
6. ASR 文字記錄 (USER) - 使用者語音轉錄
7. 工具使用（選用） - AI 請求工具執行
8. 工具處理（選用） - 處理和傳回工具結果
9. 文字記錄 (ASSISTANT) - 規格 - 初步 AI 回應
10. 音訊區塊 - 串流 AI 音訊輸出
11. 文字記錄 (ASSISTANT) - 最終版 - 最終版 AI 回應文字記錄
12. 內容結束音訊 - 標記音訊內容的結尾
13. 提示結束 - 表示完成提示處理
14. 工作階段結束 - 關閉對話

使用雙向 API 處理輸入事件

雙向串流 API 使用具有結構化輸入和輸出事件的事件驅動型架構。了解正確的事件排序對於實作成功的對話式應用程式，以及在互動期間全程維持適當的對話狀態至關重要。

概觀

Nova Sonic 對話遵循結構化事件序列。首先傳送包含推論組態參數的 `sessionStart` 事件，例如溫度和詞元限制。接著傳送 `promptStart` 來定義音訊輸出格式和工具組態，指派唯一 `promptName` 識別符，該識別符必須包含在所有後續事件中。

對於每個互動類型（系統提示、音訊等），您遵循三個部分模式：使用 `contentStart` 定義內容類型和內容的角色 `USER ASSISTANT(SYSTEM、TOOL、SYSTEM_SPEECH)`，然後提供實際的內容事件，並以 `contentEnd` 結束該區段。`contentStart` 事件指定您要傳送工具結果、串流音訊還是系統提示詞。`contentStart` 事件包含唯一 `contentName` 識別符。

對話歷史記錄

對話歷史記錄只能在系統提示詞之後以及音訊串流開始之前包含一次。它遵循相同的 `contentStart/textInput/contentEnd` 模式。必須在 `contentStart` 事件中為每個歷史訊息定義 `USER` 和 `ASSISTANT` 角色。這為目前的對話提供了基本上下文，但必須在任何新的使用者輸入開始之前完成。

音訊串流

音訊串流透過連續麥克風取樣來運作。傳送初始 `contentStart` 直接從麥克風擷取音訊影格（每個影格約 32 毫秒），並立即使用相同的 `contentName` 將其做為 `audioInput` 事件傳送。這些音訊範例應在擷取時即時串流，從而在對話期間全程保持自然麥克風取樣節奏。所有音訊幀都會共用單一內容容器，直至對話結束且內容容器被明確關閉為止。

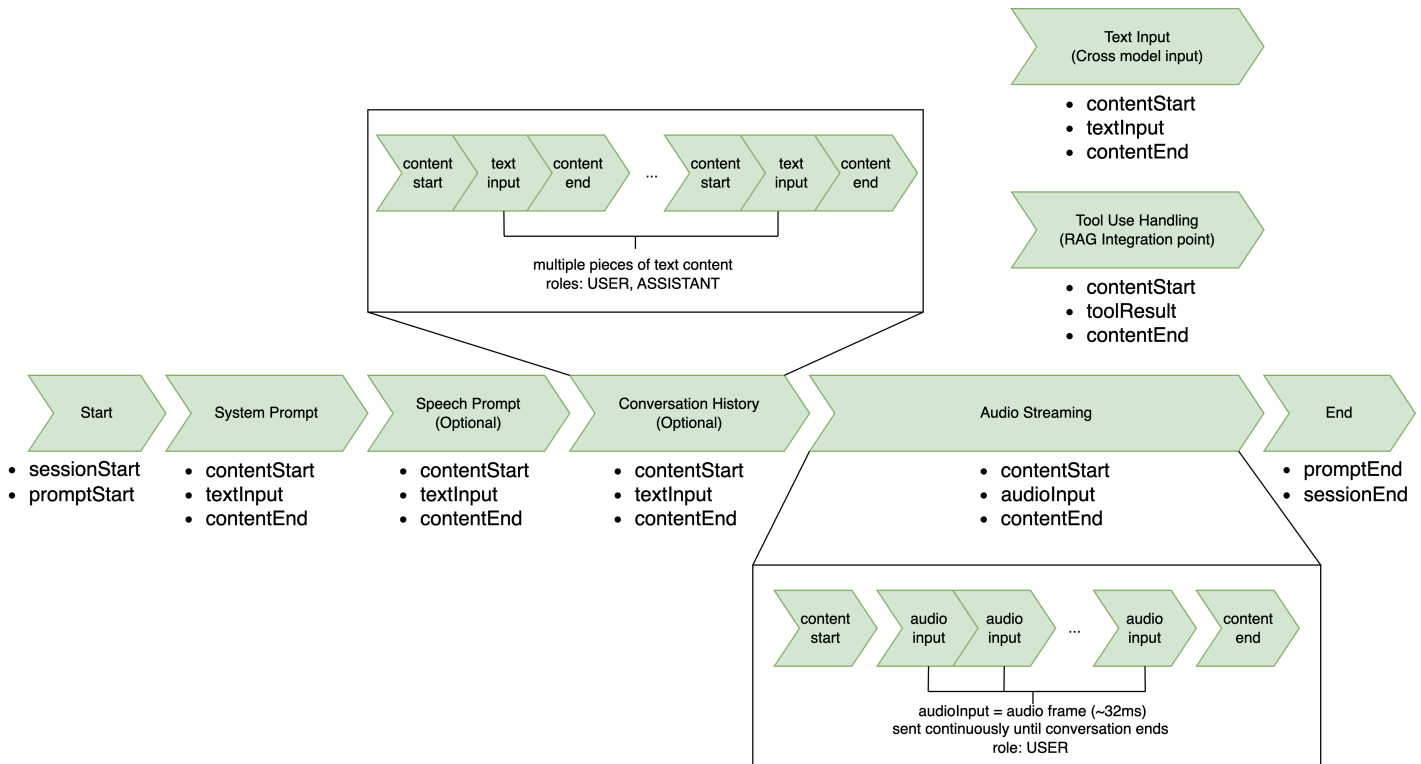
關閉工作階段

在對話結束或需要終止之後，請務必正確關閉所有開啟的串流，並以正確的順序結束工作階段。若要正確結束工作階段並避免資源洩漏，必須遵循特定的關閉順序：

- 使用 `contentEnd` 事件關閉任何開啟的音訊串流。
- 傳送參考原始 `promptName` 的 `promptEnd` 事件。
- 傳送 `sessionEnd` 事件。

略過任何這些關閉事件都可能導致未完成的對話或孤立的資源。

這些識別符會建立階層結構：`promptName` 將所有對話事件繫結在一起，而每個 `contentName` 則會標記特定內容區塊的邊界。此階層可確保模型在整個互動期間維持適當的上下文。



輸入事件流程

本節提供了輸入事件流程的結構。

1. RequestStartEvent (工作階段開始)

工作階段開始事件會使用推論組態初始化對話，並開啟偵測設定。

推論組態：

- `maxTokens`：回應中要產生的字符數量上限
- `topP`：Nucleus 取樣參數 (0.0 到 1.0) 用於控制隨機性
- `temperature`：控制產生時的隨機性 (0.0 到 1.0)

轉動偵測組態：`endpointingSensitivity` 參數控制 Nova Sonic 偵測使用者何時完成說話的速度：

- `HIGH`：快速偵測暫停，啟用更快的回應，但可能會中斷較慢的發言者
- `MEDIUM`：大多數對話案例的平衡敏感度（建議預設值）
- `LOW`：在偵測語音結束之前等待更久，適合深思熟慮或遲疑的發言者

```
{
  "event": {
    "sessionStart": {
      "inferenceConfiguration": {
        "maxTokens": "int",
        "topP": "float",
        "temperature": "float"
      },
      "turnDetectionConfiguration": {
        "endpointingSensitivity": "HIGH" | "MEDIUM" | "LOW"
      }
    }
  }
}
```

範例：

```
{
  "event": {
    "sessionStart": {
      "inferenceConfiguration": {
        "maxTokens": 2048,
        "topP": 0.9,
        "temperature": 0.7
      },
      "turnDetectionConfiguration": {
        "endpointingSensitivity": "MEDIUM"
      }
    }
  }
}
```

2. PromptStartEvent

提示啟動事件會定義對話組態，包括輸出格式、語音選擇和可用的工具。

如需可用語音 IDs 的清單，請參閱[語言支援和多語言功能](#)

```
{
  "event": {
    "promptStart": {
      "promptName": "string", // unique identifier same across all events i.e.
      UUID
```

```

    "textOutputConfiguration": {
      "mediaType": "text/plain"
    },
    "audioOutputConfiguration": {
      "mediaType": "audio/lpcm",
      "sampleRateHertz": 8000 | 16000 | 24000,
      "sampleSizeBits": 16,
      "channelCount": 1,
      "voiceId": "matthew" | "tiffany" | "amy" | "olivia" | "lupe" | "carlos"
      | "ambre" | "florian" | "lennart" | "beatrice" | "lorenzo" |
        "tina" | "carolina" | "leo" | "kiara" | "arjun",
      "encoding": "base64",
      "audioType": "SPEECH"
    },
    "toolUseOutputConfiguration": {
      "mediaType": "application/json"
    },
    "toolConfiguration": {
      "tools": [
        {
          "toolSpec": {
            "name": "string",
            "description": "string",
            "inputSchema": {
              "json": "{}"
            }
          }
        }
      ]
    }
  }
}

```

3. InputContentStartEvent

文字

文字內容開始事件用於系統提示、對話歷史記錄和跨模式文字輸入。

互動式參數：

- `true`：啟用跨模式輸入，允許在作用中語音工作階段期間傳送文字訊息

- `false`：系統提示和對話歷史記錄的標準文字輸入

角色類型：

- `SYSTEM`：系統指示和提示
- `USER`：對話歷史記錄或跨模式輸入中的使用者訊息
- `ASSISTANT`：對話歷史記錄中的助理回應
- `SYSTEM_SPEECH`：控制印地文程式碼切換的轉錄格式 (Latin/Devanagari/mixed 指令碼)

```
{
  "event": {
    "contentStart": {
      "promptName": "string", // same unique identifier from promptStart event
      "contentName": "string", // unique identifier for the content block
      "type": "TEXT",
      "interactive": "boolean", // true for cross-modal input
      "role": "SYSTEM" | "USER" | "ASSISTANT" | "TOOL" | "SYSTEM_SPEECH",
      "textInputConfiguration": {
        "mediaType": "text/plain"
      }
    }
  }
}
```

範例 - 系統提示：

```
{
  "event": {
    "contentStart": {
      "promptName": "conv-12345",
      "contentName": "system-prompt-1",
      "type": "TEXT",
      "interactive": false,
      "role": "SYSTEM",
      "textInputConfiguration": {
        "mediaType": "text/plain"
      }
    }
  }
}
```



```
}
```

範例 - 跨模式輸入：

```
{
  "event": {
    "contentStart": {
      "promptName": "conv-12345",
      "contentName": "user-text-1",
      "type": "TEXT",
      "interactive": true,
      "role": "USER",
      "textInputConfiguration": {
        "mediaType": "text/plain"
      }
    }
  }
}
```

音訊

```
{
  "event": {
    "contentStart": {
      "promptName": "string", // same unique identifier from promptStart event
      "contentName": "string", // unique identifier for the content block
      "type": "AUDIO",
      "interactive": true,
      "role": "USER",
      "audioInputConfiguration": {
        "mediaType": "audio/lpcm",
        "sampleRateHertz": 8000 | 16000 | 24000,
        "sampleSizeBits": 16,
        "channelCount": 1,
        "audioType": "SPEECH",
        "encoding": "base64"
      }
    }
  }
}
```

工具

```
{
  "event": {
    "contentStart": {
      "promptName": "string", // same unique identifier from promptStart event
      "contentName": "string", // unique identifier for the content block
      "interactive": false,
      "type": "TOOL",
      "role": "TOOL",
      "toolResultInputConfiguration": {
        "toolUseId": "string", // existing tool use id
        "type": "TEXT",
        "textInputConfiguration": {
          "mediaType": "text/plain"
        }
      }
    }
  }
}
```

4. TextInputContent

```
{
  "event": {
    "textInput": {
      "promptName": "string", // same unique identifier from promptStart event
      "contentName": "string", // unique identifier for the content block
      "content": "string"
    }
  }
}
```

5. AudioInputContent

```
{
  "event": {
    "audioInput": {
      "promptName": "string", // same unique identifier from promptStart event
      "contentName": "string", // same unique identifier from its contentStart
      "content": "base64EncodedAudioData"
    }
  }
}
```

```
}  
}
```

6. ToolResultContentEvent

```
"event": {  
  "toolResult": {  
    "promptName": "string", // same unique identifier from promptStart event  
    "contentName": "string", // same unique identifier from its contentStart  
    "content": "{\\"key\\": \\"value\\"}" // stringified JSON object as a tool result  
  }  
}
```

7. InputContentEndEvent

```
{  
  "event": {  
    "contentEnd": {  
      "promptName": "string", // same unique identifier from promptStart event  
      "contentName": "string" // same unique identifier from its contentStart  
    }  
  }  
}
```

8. PromptEndEvent

```
{  
  "event": {  
    "promptEnd": {  
      "promptName": "string" // same unique identifier from promptStart event  
    }  
  }  
}
```

9. RequestEndEvent

```
{  
  "event": {  
    "sessionEnd": {}  
  }  
}
```

使用雙向 API 處理輸出事件

當 Amazon Nova Sonic 模型回應時，它會遵循結構化事件序列。該流程從包含 `sessionId`、`promptName` 和 `completionId` 等唯一識別符的 `completionStart` 事件開始。這些識別符在整個回應週期中是一致的，並統一所有後續的回應事件。

概觀

每個回應類型都遵循一致的三部分模式：`contentStart` 定義內容類型和格式，實際內容事件，`contentEnd` 關閉該區段。回應通常會依序包含多個內容區塊：自動語音辨識 (ASR) 轉錄 (使用者說的話)、選用的工具使用 (需要外部資訊時)、文字回應 (模型打算說的話) 和音訊回應 (口語輸出)。

回應內容類型

ASR 轉錄

ASR 轉錄會先出現，在 `contentStart` 中使用 `role: "USER"` 和 `"additionalModelFields": {"generationStage": "FINAL"}` 提供模型對使用者語音的理解。

工具使用

當模型需要外部資料時，它會傳送具有特定工具名稱和參數的工具相關事件。

文字回應

文字回應透過 `role: "ASSISTANT"` 和 `"additionalModelFields": {"generationStage": "SPECULATIVE"}` 提供規劃語音的預覽。

音訊回應

然後，音訊回應會傳送 base64 編碼的語音區塊，這些區塊在整個串流中共用相同的 `contentId`。

插斷支援

在音訊產生期間，Amazon Nova Sonic 透過其插斷功能支援自然對話流程。當使用者在 Amazon Nova Sonic 說話時將其打斷，Nova Sonic 會立即停止產生語音、切換到聆聽模式，並傳送內容通知，指出已發生中斷。由於 Nova Sonic 的運作速度比即時快，因此有些音訊可能已交付但尚未播放。中斷通知可讓用戶端應用程式清除其音訊佇列並立即停止播放，從而建立回應式對話體驗。

最終轉錄

音訊產生完成 (或透過插斷將其中斷) 後，Amazon Nova Sonic 會提供額外的文字回應，其中包含實際說話內容的句子層級轉錄。此文字回應包含一個帶有 `role: "ASSISTANT"` 和

"additionalModelFields": "{\\"generationStage\\":\\"FINAL\\"}" 的 contentStart 事件。

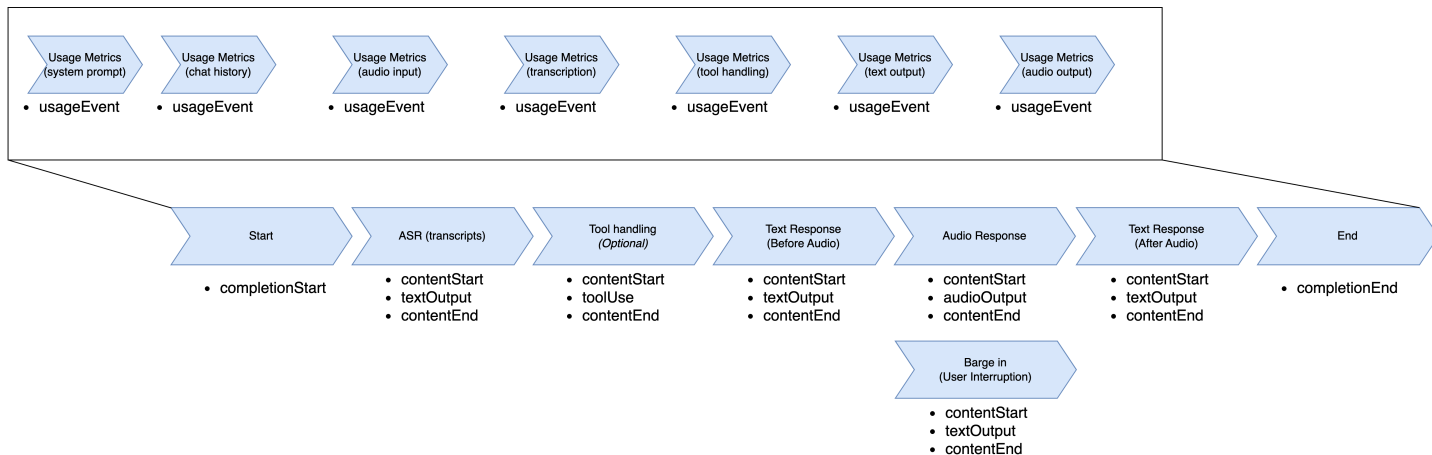
用量追蹤

在整個回應處理過程中，會傳送 usageEvent 事件來追蹤詞元消耗。這些事件包含關於輸入詞元和輸出詞元 (語音和文字) 及其累積總數的詳細指標。每個 usageEvent 都會維護與對話流程中其他事件相同的 sessionId、promptName 和 completionId。詳細資訊區段同時提供增量變更 (差異) 和累積的記號用量總計，以便精確監控對話期間的用量。

完成

模型會傳送帶有原始識別符的 completionEnd 事件，並傳送 stopReason 指出對話結束的方式。此事件階層可確保您的應用程式可以追蹤回應的哪些部分屬於一起，並相應地處理它們，從而在多個回合中維持對話上下文。

輸出事件流程從進入回應產生階段開始。它從自動語音辨識開始，選取要使用的工具，轉錄語音，產生音訊，完成轉錄，以及完成工作階段。



輸出事件流程

本節描述了輸出事件流程的結構。

1. UsageEvent

```

"event": {
  "usageEvent": {
    "completionId": "string", // unique identifier for completion
    "details": {
      "delta": { // incremental changes since last event

```

```

        "input": {
            "speechTokens": number, // input speech tokens
            "textTokens": number // input text tokens
        },
        "output": {
            "speechTokens": number, // speech tokens generated
            "textTokens": number // text tokens generated
        }
    },
    "total": { // cumulative counts
        "input": {
            "speechTokens": number, // total speech tokens processed
            "textTokens": number // total text tokens processed
        },
        "output": {
            "speechTokens": number, // total speech tokens generated
            "textTokens": number // total text tokens generated
        }
    }
},
"promptName": "string", // same unique identifier from promptStart event
"sessionId": "string", // unique identifier
"totalInputTokens": number, // cumulative input tokens
"totalOutputTokens": number, // cumulative output tokens
"totalTokens": number // total tokens in the session
}
}

```

2. CompleteStartEvent

```

"event": {
    "completionStart": {
        "sessionId": "string", // unique identifier
        "promptName": "string", // same unique identifier from promptStart event
        "completionId": "string", // unique identifier
    }
}

```

3. TextOutputContent

ContentStart

```

"event": {

```

```

    "contentStart": {
      "additionalModelFields": "{\\"generationStage\\":\\"FINAL\\"}" |
    "{\\"generationStage\\":\\"SPECULATIVE\\"}",
      "sessionId": "string", // unique identifier
      "promptName": "string", // same unique identifier from promptStart event
      "completionId": "string", // unique identifier
      "contentId": "string", // unique identifier for the content block
      "type": "TEXT",
      "role": "USER" | "ASSISTANT",
      "textOutputConfiguration": {
        "mediaType": "text/plain"
      }
    }
  }
}

```

TextOutput

```

"event": {
  "textOutput": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // same unique identifier from its contentStart
    "content": "string" // User transcribe or Text Response
  }
}

```

ContentEnd

```

"event": {
  "contentEnd": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // same unique identifier from its contentStart
    "stopReason": "PARTIAL_TURN" | "END_TURN" | "INTERRUPTED",
    "type": "TEXT"
  }
}

```

4. ToolUse

ContentStart

```
"event": {
  "contentStart": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // unique identifier for the content block
    "type": "T00L",
    "role": "T00L",
    "toolUseOutputConfiguration": {
      "mediaType": "application/json"
    }
  }
}
```

ToolUse

```
"event": {
  "toolUse": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // same unique identifier from its contentStart
    "content": "json",
    "toolName": "string",
    "toolUseId": "string"
  }
}
```

ContentEnd

```
"event": {
  "contentEnd": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // same unique identifier from its contentStart
    "stopReason": "T00L_USE",
    "type": "T00L"
  }
}
```



```
    }
  }
```

5. AudioOutputContent

ContentStart

```
"event": {
  "contentStart": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // unique identifier for the content block
    "type": "AUDIO",
    "role": "ASSISTANT",
    "audioOutputConfiguration": {
      "mediaType": "audio/lpcm",
      "sampleRateHertz": 8000 | 16000 | 24000,
      "sampleSizeBits": 16,
      "encoding": "base64",
      "channelCount": 1
    }
  }
}
```

AudioOutput

```
"event": {
  "audioOutput": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // same unique identifier from its contentStart
    "content": "base64EncodedAudioData", // Audio
  }
}
```

ContentEnd

```
"event": {
  "contentEnd": {
    "sessionId": "string", // unique identifier
```

```
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "contentId": "string", // same unique identifier from its contentStart
    "stopReason": "PARTIAL_TURN" | "END_TURN",
    "type": "AUDIO"
  }
}
```

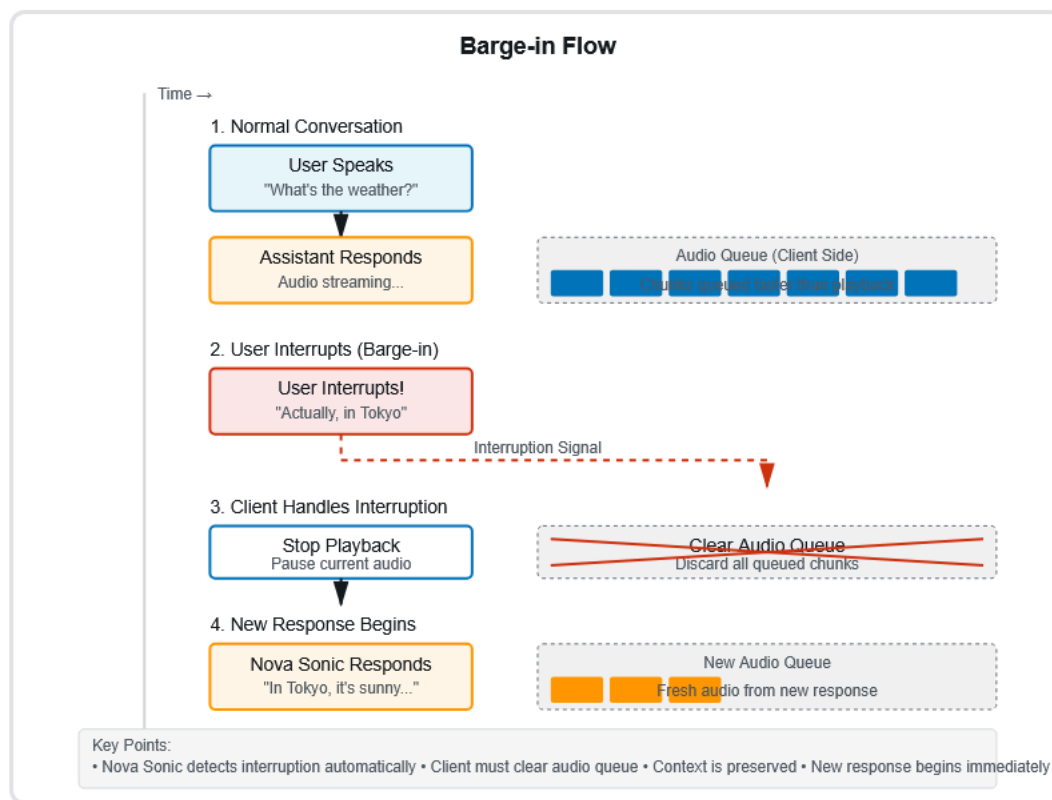
6. CompletionEndEvent

```
"event": {
  "completionEnd": {
    "sessionId": "string", // unique identifier
    "promptName": "string", // same unique identifier from promptStart event
    "completionId": "string", // unique identifier
    "stopReason": "END_TURN"
  }
}
```

闖入

插斷功能可讓使用者在說話時中斷 AI 助理，就像自然人對話一樣。使用者可以插入新資訊、更正或釐清先前的陳述式、將對話重新導向至不同的主題，或在聽到足夠的訊息時直接停止助理，而不是等待助理完成。這可建立更自然且回應靈敏的對話體驗。

下圖說明完整的插斷對話流程：



Amazon Nova 2 Sonic 如何處理插入

Amazon Nova 2 Sonic 旨在正常處理中斷。當使用者在回應期間開始說話時，系統會立即停止產生目前的回應、維護完整的對話內容、傳送中斷訊號給用戶端，並開始處理新的使用者輸入。

內容保留：即使中斷，Nova Sonic 仍會記住中斷前所說的內容、討論的主題、對話歷史記錄，以及先前回合的任何相關內容。這可確保對話的一致性和自然性。

用戶端實作需求

雖然 Amazon Nova 2 Sonic 在伺服器端處理插斷，但您需要實作用戶端邏輯以獲得完整的體驗。

音訊佇列挑戰：音訊產生比播放速度更快。這表示：

- Nova Sonic 快速產生音訊區塊
- 您的用戶端會接收這些區塊並將其排入佇列
- 用戶端會以正常說話速度播放
- 發生插斷時，已將音訊排入佇列以供播放

必要的用戶端邏輯：您的應用程式必須處理四個關鍵步驟：

1. 偵測中斷訊號：接聽來自 Nova Sonic 的中斷事件，並在收到時立即做出反應。
2. 停止目前播放：暫停目前播放的音訊，並停止任何中間播放的音訊。
3. 清除音訊佇列：移除所有排入佇列的音訊區塊，並從中斷的回應中捨棄任何緩衝的音訊。
4. 開始新音訊：開始播放新接收的音訊，並繼續正常播放流程。

輪換可控制性

輪換是自然對話的基本層面。Amazon Nova 2 Sonic 可精細控制 AI 何時輪流透過 `turnDetectionConfiguration` 參數說話。這可讓您針對不同的使用案例最佳化對話流程，平衡回應能力與準確性。`endpointingSensitivity` 參數可控制 Amazon Nova 2 Sonic 偵測使用者轉彎並開始回應的速度。此設定同時影響回應的延遲，以及中斷仍在說話的使用者的可能性。

API 組態

在 `sessionStart` 事件中設定轉彎偵測敏感度：

```
{
  "event": {
    "sessionStart": {
      "inferenceConfiguration": {
        "maxTokens": 1000,
        "topP": 0.9,
        "temperature": 0.7
      },
      "turnDetectionConfiguration": {
        "endpointingSensitivity": "HIGH" | "MEDIUM" | "LOW"
      }
    }
  }
}
```

敏感度層級

`endpointingSensitivity` 參數接受三個值：High、MEDIUM 和 LOW。每個層級都會平衡回應速度與中斷仍在說話的使用者的風險。

HIGH (高)

最快的回應時間，針對延遲進行最佳化。在偵測到語音結束之後，Nova Sonic 會盡快回應。暫停持續時間：1.5 秒。最適合快速 Q 和 A、command-and-control 應用程式，以及時間敏感的互動。

中型

具有中等回應時間的平衡方法。減少誤報，同時保持回應能力。暫停持續時間：1.75 秒。最適合一般對話、具有複雜查詢的客戶服務，以及多迴轉討論。

低

回應時間最慢，最大耐心。Nova Sonic 會在回應之前等待最久的時間，盡量減少在思考時暫停的使用者中斷。暫停持續時間：2 秒。最適合深思熟慮的對話、老年或語音受損的使用者，以及複雜的問題解決。

暫停持續時間參考

敏感度層級	暫停持續時間（秒）
高（快速）	1.5
中	1.75
低（慢）	~2.0

跨模式輸入

Amazon Nova 2 Sonic 現在支援跨模式輸入，除了語音輸入之外，您還可以在對話工作階段期間傳送文字訊息。雖然語音仍是互動的主要模式，但文字輸入為輸入更方便或更適當的案例提供彈性。

需要持續串流：跨模式輸入需要作用中的串流工作階段才能正常運作。工作階段必須像一般語音工作階段一樣維持持續串流，否則將套用標準工作階段逾時，並將終止連線。

跨模式文字輸入中的敏感度層級適用於下列情況：

- 用戶端應用程式整合 (Web 和行動裝置)：允許使用者使用文字和語音與應用程式互動，以支援無縫的多模式體驗。
- 「Model-start-first」使用案例：可在工作階段開始提示模型開始說話後立即傳送文字訊息。

- 在非同步工具呼叫期間引導模型：當toolUse事件觸發且系統開始處理工具呼叫時，用戶端可以將文字訊息傳送至 Sonic，以在等待時提供自然回應，例如「在我處理您的資訊時保留一秒鐘。同時，還有什麼我可以協助的嗎？」
- 電話 DTMF 整合：客戶使用電話鍵盤輸入敏感資訊（例如信用卡號碼）。注意：Amazon Nova Sonic 不會原生處理 DTMF 音調。若要支援 DTMF 輸入，您的系統必須偵測音調，將其轉換為文字（例如 "1234"），然後傳送至 Nova 2 Sonic。

運作方式

跨模式輸入使用類似音訊輸入的三個事件序列：

1. 內容開始事件：表示文字輸入的開頭
2. 文字輸入事件：包含實際的文字訊息
3. 內容結束事件：表示文字輸入完成

這三個事件都必須使用相同的 promptName 和 contentName 來維護序列。每次傳送文字輸入時，都應為 contentName 產生新的 UUID，以確保適當的多迴轉對話追蹤。

事件結構

1. 內容開始事件

使用組態詳細資訊啟動文字輸入序列：

```
{
  "event": {
    "contentStart": {
      "promptName": "<prompt_name>",
      "contentName": "<new_content_name>",
      "role": "USER",
      "type": "TEXT",
      "interactive": true,
      "textInputConfiguration": {
        "mediaType": "text/plain"
      }
    }
  }
}
```

重要參數：

- `promptName`：對話提示的名稱（跨工作階段一致）
- `contentName`：此文字輸入的唯一識別符（為每個訊息產生新的 UUID）
- `role`：設定為 "USER" 以指示使用者輸入
- `type`："TEXT" 將文字輸入設定為
- `interactive`：設定為 `true` 以啟用互動式模式
- `mediaType`："text/plain" 純文字內容設定為

2. 文字輸入事件

包含實際的文字訊息內容：

```
{
  "event": {
    "textInput": {
      "promptName": "<prompt_name>",
      "contentName": "<new_content_name>",
      "content": "<your_text_message>"
    }
  }
}
```

重要參數：

- `promptName`：必須符合內容開始事件中的值
- `contentName`：必須符合內容開始事件中的值
- `role`：您的文字訊息字串

3. 內容結束事件

表示文字輸入的完成：

```
{
  "event": {
    "contentEnd": {
      "promptName": "<prompt_name>",
      "contentName": "<new_content_name>"
    }
  }
}
```

```
}
```

重要參數：

- `promptName`：必須符合先前事件的值
- `contentName`：必須符合先前事件的值

語言支援和多語言功能

Amazon Nova 2 Sonic 提供多種語言的語音選擇，可讓您建立對話式 AI 應用程式，讓您覺得自然且文化上適合您的使用者。每種語言都提供女性聲音和男性聲音選項。

下表列出所有可用的語音及其對應的語言地區設定：

Language	Locale	女性聲音語音 ID	Masculine-sounding 語音 ID
英文 (美國)	zh-TW	tiffany	matthew
英文 (英國)	en-GB	amy	-
英文 (澳洲)	en-AU	利維亞	-
英文 (印度)	en-IN	卡拉	arjun
法文	fr-FR	ambre	florian
義大利文	it-IT	beatrice	lorenzo
德文	de-DE	tina	lennart
西班牙文 (美國)	es-US	lupe	carlos
葡萄牙文	pt-BR	卡羅萊納	leo
北印度文	hi-IN	卡拉	arjun

在您的應用程式中使用語音的事件結構

您可以在 `promptStart` 事件中啟動提示 `audioOutputConfiguration` 時，在 中指定語音 ID：


```
"event": {
  "promptStart": {
    "promptName": "string",
    "audioOutputConfiguration": {
      "mediaType": "audio/lpcm",
      "sampleRateHertz": 16000,
      "sampleSizeBits": 16,
      "channelCount": 1,
      "voiceId": "tiffany",
      "encoding": "base64",
      "audioType": "SPEECH"
    }
  }
}
```

多語言支援

Amazon Nova 2 Sonic 提供強大的多語言功能，可跨多種語言進行自然對話。此服務支援多聲道語音（多語言）和程式碼切換（在同一句子內混合語言），可讓您建置真正的全域對話應用程式。

TIFFANY (en-US, female) 和 MATTHEW (en-US, male) 是可以說所有支援語言的唯一多槽語音：

1. 英文
2. 法文
3. 義大利文
4. 德文
5. 西班牙文
6. 葡萄牙文
7. 北印度文

這使得 Tiffany 和 Matthew 非常適合需要無縫切換多種語言的應用程式。

管理聊天歷史記錄

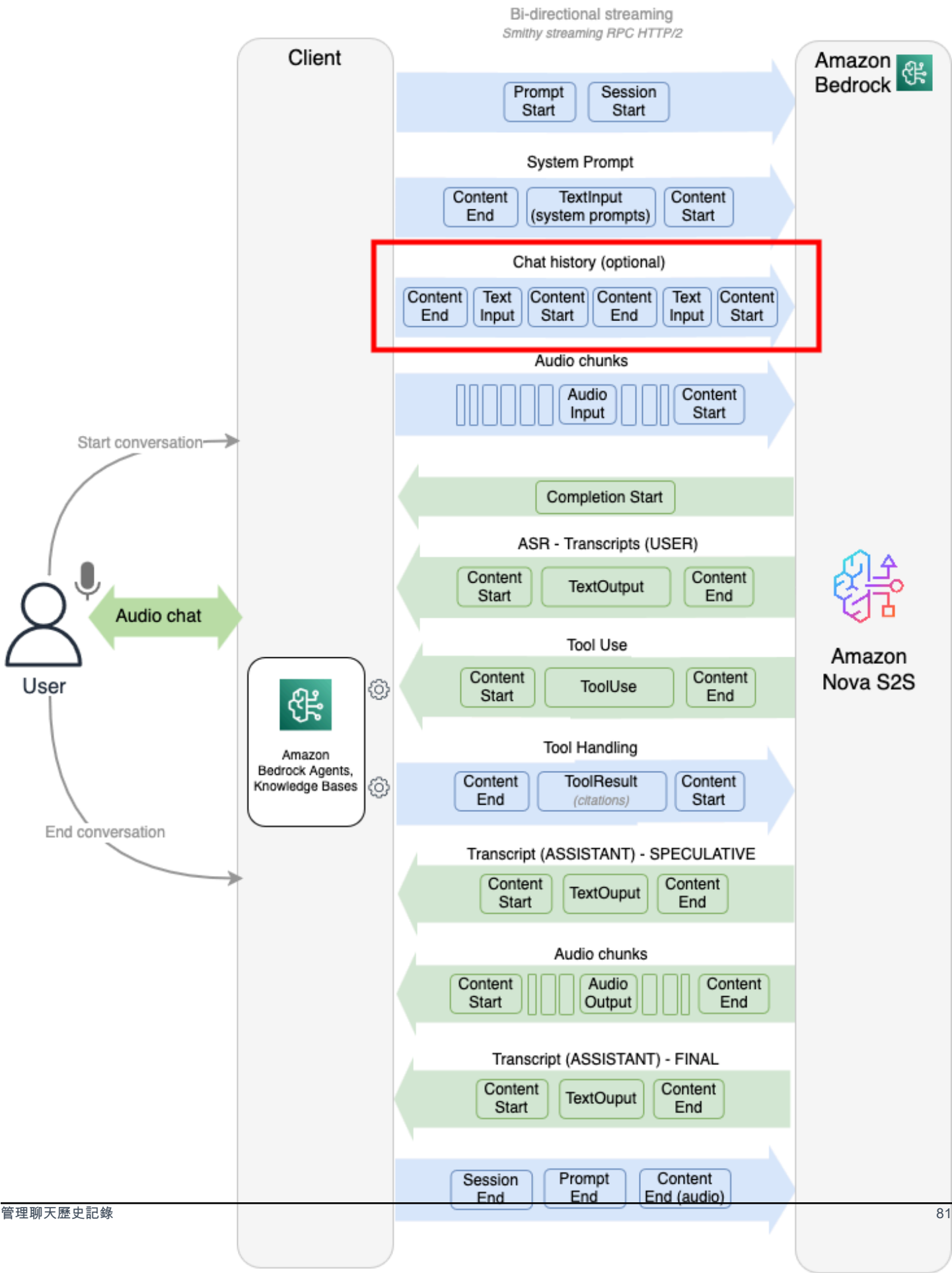
Amazon Nova 2 Sonic 回應包含使用者和助理語音的 ASR（自動語音辨識）文字記錄。儲存聊天歷史記錄是最佳實務，不僅用於記錄目的，也用於在連線意外關閉時繼續工作階段。這可讓用戶端將內容傳回 Nova Sonic，以無縫地繼續對話。

如需管理聊天歷史記錄的詳細資訊，請參閱下列資源：

1. [記錄聊天歷史記錄](#)
2. [繼續對話](#)

傳送聊天歷史記錄

對話歷史記錄只能在系統/語音提示之後以及在音訊串流開始之前包含一次。整體聊天歷史記錄不能大於 40KB。下圖顯示聊天歷史記錄在事件生命週期期間傳入的時間：



每個歷史訊息都需要三個事件：contentStart、textInput和 contentEnd。

每則訊息的事件結構描述：

- contentStart - 定義訊息角色和組態

```
{
  "event": {
    "contentStart": {
      "promptName": "<prompt-id>",
      "contentName": "<content-id>",
      "type": "TEXT",
      "interactive": true,
      "role": "ASSISTANT",
      "textInputConfiguration": {
        "mediaType": "text/plain"
      }
    }
  }
}
```

- textInput - 包含實際的訊息內容。一個 textInput 不能大於 1KB。若是如此，請在相同的內容區塊中分割成多個 textInputs。如果對話大於 40KB，請裁剪整體聊天歷史記錄。

```
{
  "event": {
    "textInput": {
      "promptName": "<prompt-id>",
      "contentName": "<content-id>",
      "content": "Take your time, Don. I'll be here when you're ready."
    }
  }
}
```

- contentEnd - 標記訊息的結尾

```
{
  "event": {
    "contentEnd": {
      "promptName": "<prompt-id>",
      "contentName": "<content-id>"
    }
  }
}
```

```
}
```

對聊天歷史記錄中的每個訊息重複這三個事件，在 USER 和 ASSISTANT 角色之間交替。

重要考量事項：

- 每個工作階段只能包含一次聊天歷史記錄
- 聊天歷史記錄必須在系統提示後和音訊串流開始之前傳送
- 在開始音訊串流之前，必須傳送所有歷史訊息
- 每個訊息都必須指定 USER 或 ASSISTANT 角色
- 使用來自 textOutput 事件的預存文字記錄內容作為 textInput 中的內容值

接收 ASR 文字記錄

在對話期間，Amazon Nova 2 Sonic 會透過輸出事件傳送 ASR 文字記錄。每個文字記錄會以三個事件的序列傳遞：contentStart、textOutput 和 contentEnd。

範例：使用者語音文字記錄：

1. contentStart - 指出文字記錄的開頭：

```
{
  "event": {
    "contentStart": {
      "additionalModelFields": "{\"generationStage\":\"FINAL\"}",
      "completionId": "<completion-id>",
      "contentId": "<content-id>",
      "promptName": "<prompt-id>",
      "role": "USER",
      "sessionId": "<session-id>",
      "textOutputConfiguration": {
        "mediaType": "text/plain"
      },
      "type": "TEXT"
    }
  }
}
```

2. textOutput - 包含實際文字記錄內容：

```
{
  "event": {
    "textOutput": {
      "completionId": "<completion-id>",
      "content": "hello how are you",
      "contentId": "<content-id>",
      "promptName": "<prompt-id>",
      "role": "USER",
      "sessionId": "<session-id>"
    }
  }
}
```

3. contentEnd - 標記文字記錄的結尾：

```
{
  "event": {
    "contentEnd": {
      "completionId": "<completion-id>",
      "contentId": "<content-id>",
      "promptName": "<prompt-id>",
      "sessionId": "<session-id>",
      "stopReason": "PARTIAL_TURN",
      "type": "TEXT"
    }
  }
}
```

相同的三事件模式適用於 USER 和 ASSISTANT 角色。從textOutput事件擷取 content 欄位，從contentStart事件擷取 role 欄位，以建置您的聊天歷史記錄。

最佳實務

一律存放聊天歷史記錄以啟用：

- 不同裝置的工作階段恢復
- 對話記錄和稽核
- 追蹤互動的內容保留

重要：儲存聊天歷史記錄時，請根據其 generationStage 使用文字輸出：

- 推測 - 預覽 Nova 2 Sonic 計劃說的內容，在音訊合成開始之前產生
- 最終 - 音訊回應中所說內容的實際句子層級轉錄

一律將最終文字輸出儲存到您的聊天歷史記錄，因為它代表對話的準確記錄。

FINAL 輸出範例（將此儲存到聊天歷史記錄）：

```
ContentStart event: {
  "additionalModelFields": "{\"generationStage\":\"FINAL\"}",
  "completionId": "<completion-id>",
  "contentId": "<content-id>",
  "role": "ASSISTANT",
  "sessionId": "<session-id>",
  "type": "TEXT"
}
```

SPECULATIVE 輸出範例（選用預覽，不適用於歷史記錄）：

```
ContentStart event: {
  "additionalModelFields": "{\"generationStage\":\"SPECULATIVE\"}",
  "completionId": "<completion-id>",
  "contentId": "<content-id>",
  "role": "ASSISTANT",
  "sessionId": "<session-id>",
  "type": "TEXT"
}
```

工具組態

Amazon Nova 2 Sonic 支援工具使用（也稱為函數呼叫），允許模型在對話期間請求外部資訊或動作，例如 API 呼叫、資料庫查詢或自訂程式碼函數。這可讓您的語音助理根據使用者請求採取動作、擷取資訊，以及與外部服務整合。

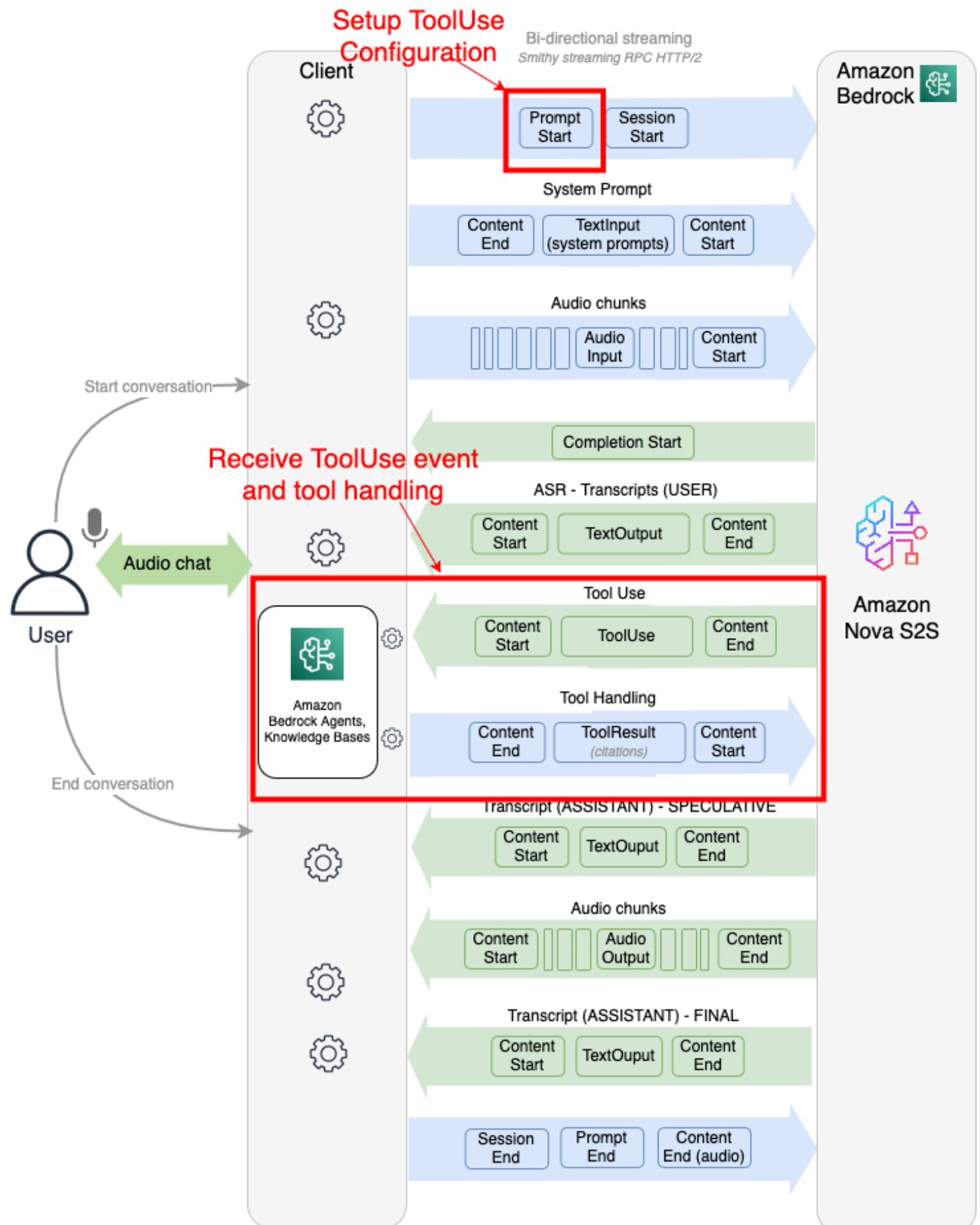
Nova 2 Sonic 具有非同步工具呼叫功能，可讓 AI 在工具在較冷的背景中執行時自然地進行交談 - 建立更流暢且回應靈敏的使用者體驗。

以下是如何使用工具的簡化步驟：

1. 定義工具：在 promptStart 事件中使用其參數指定可用的工具

2. 使用者說話：使用者提出需要工具的請求（例如「西雅圖的天氣如何？」）
3. 工具叫用：Nova 2 Sonic 辨識需求並傳送toolUse事件
4. 執行角色：您的應用程式執行工具並傳回結果
5. 回應產生：Nova 2 Sonic 將結果納入其口語回應

下圖說明工具使用方式的運作方式：



定義工具

工具是使用描述其用途、參數和預期輸入的 JSON 結構描述來定義。

以下是工具定義元件和說明：

- 名稱：工具的唯一識別符（使用 snake_case）
- 描述：明確說明工具的功能；協助 AI 決定何時使用它
- InputSchema：JSON 結構描述，定義工具接受的參數
- 屬性：具有類型和描述的個別參數
- 必要：必須提供的參數名稱陣列

工具定義範例

以下是簡單的天氣工具定義

```
{
  "toolSpec": {
    "name": "get_weather",
    "description": "Get current weather information for a specific location",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "location": {
            "type": "string",
            "description": "City name or zip code"
          },
          "units": {
            "type": "string",
            "enum": ["celsius", "fahrenheit"],
            "description": "Temperature units"
          }
        },
        "required": ["location"]
      }
    }
  }
}
```

在 PromptStart 中設定工具

工具組態會在promptStart事件中與音訊和文字輸出設定一起傳遞至 Nova 2 Sonic：

```
{
  "event": {
    "promptStart": {
      "promptName": "<prompt-id>",
      "textOutputConfiguration": {
        "mediaType": "text/plain"
      },
      "audioOutputConfiguration": {
        "mediaType": "audio/lpcm",
        "sampleRateHertz": 16000,
        "sampleSizeBits": 16,
        "channelCount": 1,
        "voiceId": "matthew",
        "encoding": "base64",
        "audioType": "SPEECH"
      },
      "toolUseOutputConfiguration": {
        "mediaType": "application/json"
      },
      "toolConfiguration": {
        "tools": [
          {
            "toolSpec": {
              "name": "get_weather",
              "description": "Get current weather information for a
specific location",
              "inputSchema": {
                "json": {
                  "type": "object",
                  "properties": {
                    "location": {
                      "type": "string",
                      "description": "City name or zip code"
                    },
                    "units": {
                      "type": "string",
                      "enum": ["celsius", "fahrenheit"],
                      "description": "Temperature units"
                    }
                  }
                }
              }
            }
          ]
        }
      }
    }
  }
}
```

```
{
  "required": ["location"]
}
}
```

工具選項參數

Nova 2 Sonic 支援三個工具選擇參數，以控制何時和使用哪些工具。在工具組態中指定 toolChoice 參數：

- **自動（預設）：**模型會決定是否需要任何工具，並視需要呼叫多個工具。提供最大的彈性。
- **任何：**確保在回應開始時呼叫至少一個可用工具，模型會選取最適當的工具。當您有多個知識庫或工具，並想要保證使用一個知識庫或工具時很有用。
- **工具：**強制在回應開始時只呼叫特定具名工具一次。例如，如果您指定知識庫工具，無論模型是否認為需要該工具，模型都會在回應之前進行查詢。

工具選項範例

Auto (自動) (預設)

```
"toolChoice": {
  #"auto": {}
}
```

任何：

```
"toolChoice": {
  "any": {}
}
```

特定工具：

```
"toolChoice": {
  "tool": {
    "name": "get_weather"
  }
}
```

接收和處理工具使用事件

當 Amazon Nova 2 Sonic 判斷需要工具時，會傳送 `toolUse` 事件，其中包含：

1. `toolUseID`：此工具調用的唯一識別符
2. `ToolName`：要執行的工具名稱
3. 內容：JSON 字串，其中包含從使用者請求擷取的參數
4. `SessionID`：目前的工作階段識別符
5. 角色：針對工具使用事件設定為「TOOL」

範例工具使用事件

```
{
  "event": {
    "toolUse": {
      "completionId": "<completion-id>",
      "content": "{\"location\": \"Seattle\", \"units\": \"fahrenheit\"}",
      "contentId": "<content-id>",
      "promptName": "<prompt-id>",
      "role": "TOOL",
      "sessionId": "<session-id>",
      "toolName": "get_weather",
      "toolUseId": "<tool-use-id>"
    }
  }
}
```

處理步驟

1. 接收 `toolUse` 事件
2. 從事件擷取工具名稱和參數
3. 執行您的工具邏輯 (API 呼叫、資料庫查詢等)
4. 使用 `toolResult` 事件傳回結果

範例 ToolResult 事件

```
{
  "event": {
    "toolResult": {
      "promptName": "<prompt-id>",
      "contentName": "<content-id>",
      "content": "{\"temperature\": 72, \"condition\": \"sunny\", \"humidity\": 45}"
    }
  }
}
```

最佳實務

- 明確描述：撰寫詳細的工具描述，以協助 Nova 2 Sonic 了解何時使用每個工具。
- 驗證參數：執行前一律驗證工具參數以防止錯誤。盡可能使用具有結構化資料類型（例如列舉、數字或布林值）的適當 JSON 結構描述來定義工具參數，而非開放式字串。
- 錯誤處理：在工具失敗時傳回 toolResult 事件中有意義的錯誤訊息。
- 非同步執行：利用非同步工具呼叫來維持對話流程。
- 工具命名：使用描述性、動作導向的名稱（例如 get_weather、search_database、send_email）。

非同步工具呼叫

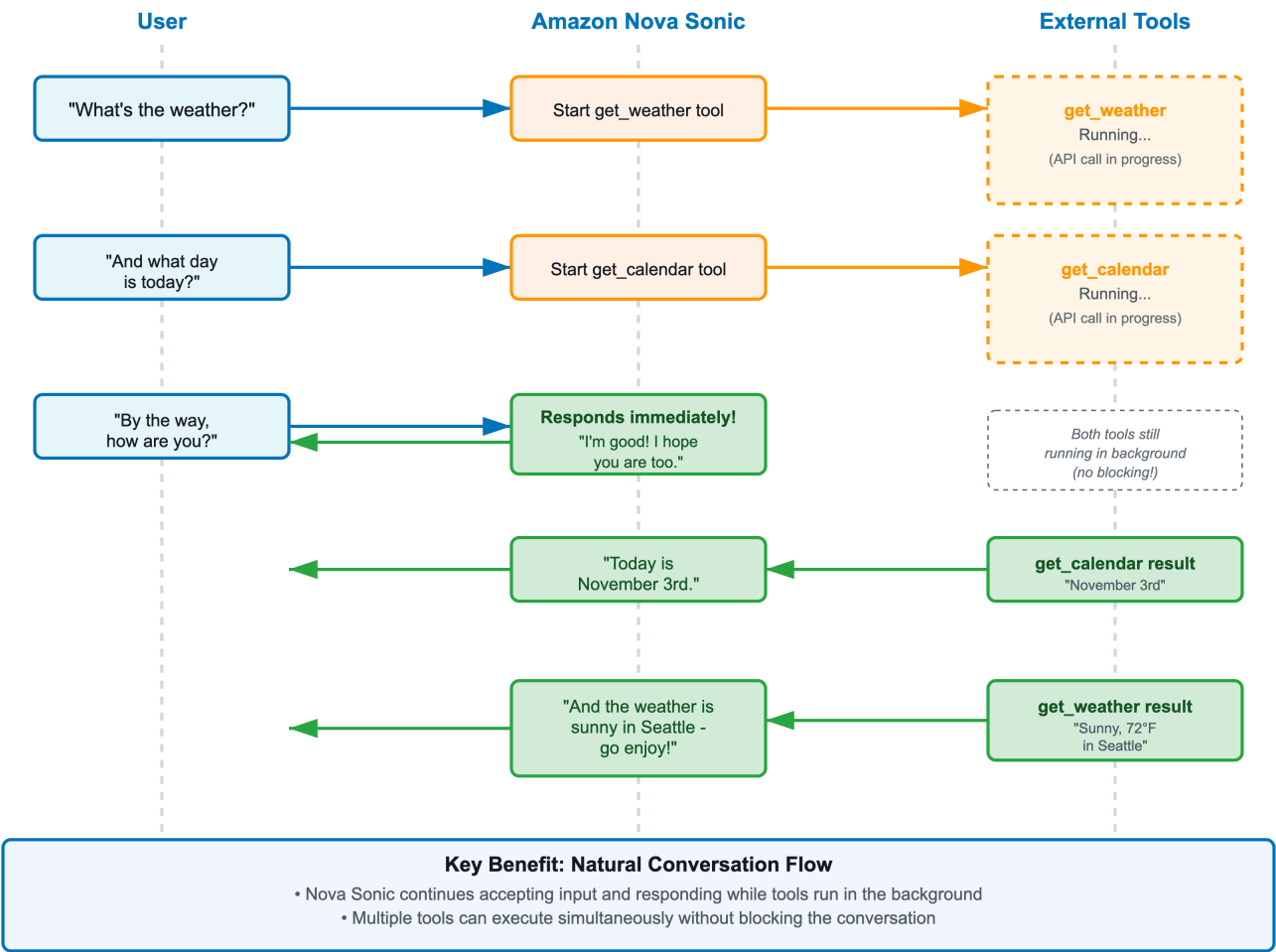
與 AI 無提示等待工具結果的傳統同步工具呼叫不同，Amazon Nova 2 Sonic 的非同步方法允許它：

- 在工具執行時繼續接受使用者輸入
- 回應新問題，無需等待工具結果
- 同時處理多個工具呼叫
- 維持自然的對話流程，避免尷尬的暫停
- 不需要額外的組態。非同步工具呼叫可立即運作。

運作方式

當 Nova 2 Sonic 發出工具呼叫時，不會暫停對話。相反地，它會繼續自然地接聽和回應，直到工具送達。

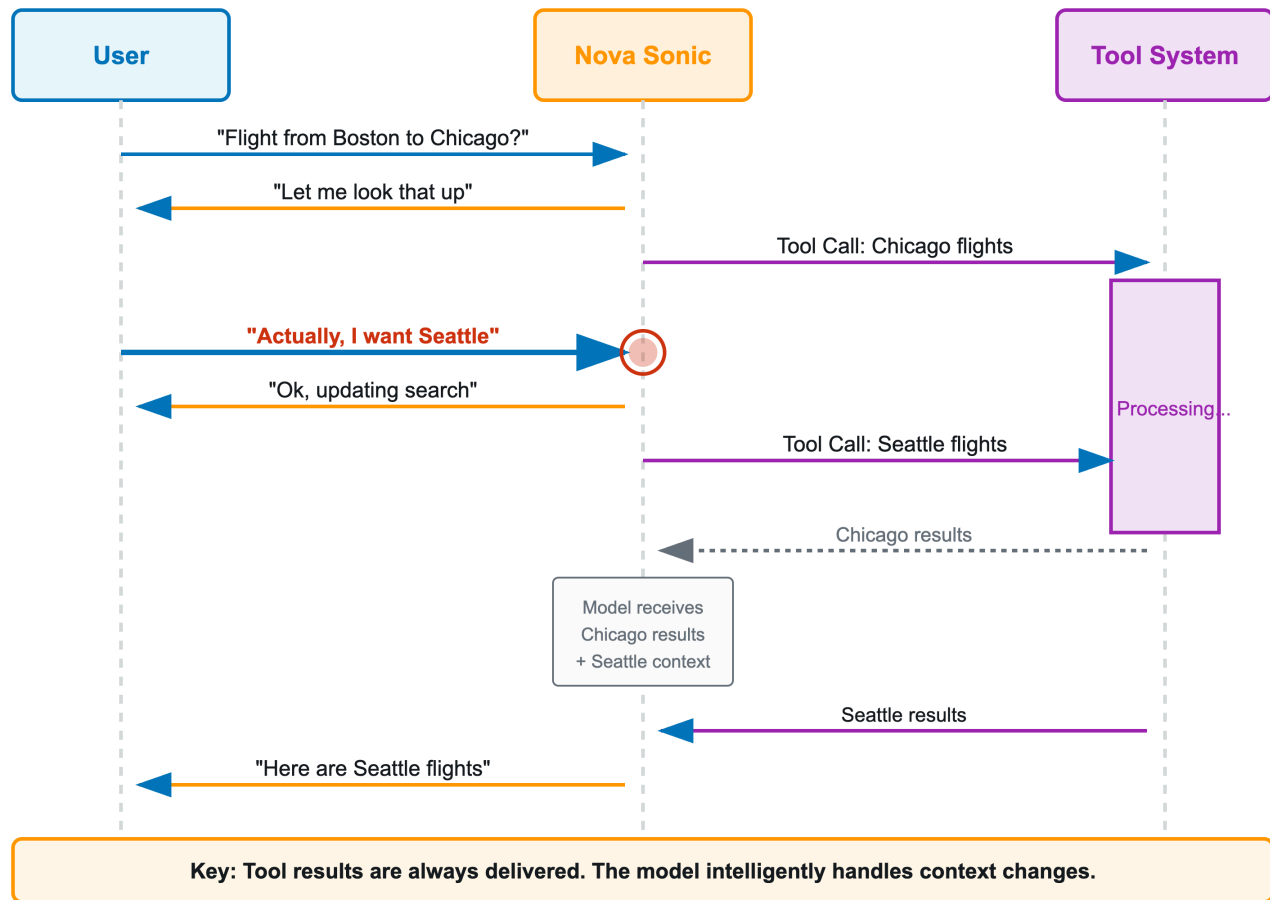
Asynchronous Tool Calling Flow



處理使用者中斷

如果使用者在工具執行時變更其請求，Nova 2 Sonic 會以智慧方式處理它，而不會取消待定的工具呼叫。

Async Tool Calling with User Interruption



範例藍本

```

User: "Can I book a flight from Boston to Chicago?"
Agent: "Sure, let me look that up for you."
Agent: [initiates tool call for Chicago flights]
User: "Actually, I want to go to Seattle"
Agent: "Ok let me update that search"
Agent: [initiates tool call for Seattle flights]
[First tool returns with Chicago flight results]
Agent: [receives Chicago results and processes them contextually]
  
```

運作方式

工具結果一律會交付：工具呼叫完成時，其結果一律會傳送至模型，即使使用者已變更其請求也一樣。模型會使用其推理功能來判斷如何處理資訊。

內容感知處理：模型了解對話內容，並可適當處理過時的工具結果。例如：

- 如果使用者在改變主意後說「謝謝」，則模型仍需要原始結果做為內容
- 如果使用者變更其請求，模型可以確認原始結果，同時專注於新的請求

無自動取消：系統不會根據新使用者輸入自動取消或忽略工具呼叫。這可確保模型擁有完整的資訊，以做出有關如何回應的明智決策。

整合

Amazon Nova 2 Sonic 可與各種架構和平台整合，以建置對話式 AI 應用程式。這些整合為常見使用案例提供預先建置的元件和簡化 APIs。

Strands 代理程式

Strands Agents 是一種簡單但功能強大的 SDK，採用模型驅動方法來建置和執行 AI 代理器。從簡單的對話助理到複雜的自主工作流程，從本機開發到生產部署，Strands Agents 可根據您的需求進行擴展。

如需 Strands 架構的完整文件，請造訪[官方 Strands 文件](#)。

Strands BidiAgent 透過持久性串流連線提供即時音訊和文字互動。與傳統的請求回應模式不同，此代理程式會維護長時間執行的對話，並支援中斷、並行處理和持續音訊回應。

先決條件：

- 已安裝 Python 3.8 或更新版本
- AWS 已設定可存取 Amazon Bedrock 的登入資料
- 基本熟悉 Python 非同步/等待語法

程式碼範例：

程式碼範例

安裝：

安裝必要的套件：

```
pip install strands-agents strands-agents-tools
```

執行此範例：

```
import asyncio
from strands.experimental.bidi.agent import BidiAgent
from strands.experimental.bidi.io.audio import BidiAudioIO
from strands.experimental.bidi.io.text import BidiTextIO
from strands.experimental.bidi.models.novasonic import BidiNovaSonicModel
from strands_tools import calculator

async def main():
    """Test the BidirectionalAgent API."""
    # Audio and Text input/output utility
    audio_io = BidiAudioIO(audio_config={})
    text_io = BidiTextIO()

    # Nova Sonic model
    model = BidiNovaSonicModel(region="us-east-1")

    async with BidiAgent(model=model, tools=[calculator]) as agent:
        print("New BidiAgent Experience")
        print("Try asking: 'What is 25 times 8?' or 'Calculate the square root of 144'")

        await agent.run(
            inputs=[audio_io.input()],
            outputs=[audio_io.output(), text_io.output()]
        )

if __name__ == "__main__":
    try:
        asyncio.run(main())
    except KeyboardInterrupt:
        print("\nConversation ended by user")
    except Exception as e:
        print(f"Error: {e}")
        import traceback
        traceback.print_exc()
```

1. 匯入：必要模型

```
from strands.experimental.bidi.agent import BidiAgent
from strands.experimental.bidi.io.audio import BidiAudioIO
from strands.experimental.bidi.io.text import BidiTextIO
```

```
from strands.experimental.bidi.models.novasonic import BidiNovaSonicModel
from strands_tools import calculator
```

- BidiAgent：協調雙向對話的主要代理程式類別
- BidiAudioIO：處理語音互動的音訊輸入和輸出
- BidiTextIO：提供文字輸出以進行轉錄和回應
- BidiNovaSonicModel Nova 2 Sonic 模型包裝函式
- 計算器：數學操作的預先建置工具

2. 設定音訊和文字 I/O

```
audio_io = BidiAudioIO(audio_config={})
text_io = BidiTextIO()
```

BidiAudioIO 會管理麥克風輸入和喇叭輸出，而 BidiTextIO 會在主控台中顯示文字轉錄和回應。

3. 初始化模型

```
model = BidiNovaSonicModel(region="us-east-1")
```

建立 Nova Sonic 模型執行個體。區域參數指定 AWS 部署模型的區域。

4. 建立和執行 代理程式

```
async with BidiAgent(model=model, tools=[calculator]) as agent:
    await agent.run(
        inputs=[audio_io.input()],
        outputs=[audio_io.output(), text_io.output()]
    )
```

代理程式的建立方式如下：

- 模型：要使用的 Nova 2 Sonic 模型
- 工具：客服人員可以呼叫的工具清單（例如計算器）
- 輸入：來自麥克風的音訊輸入
- 輸出：音訊輸出至發言者，文字輸出至主控台

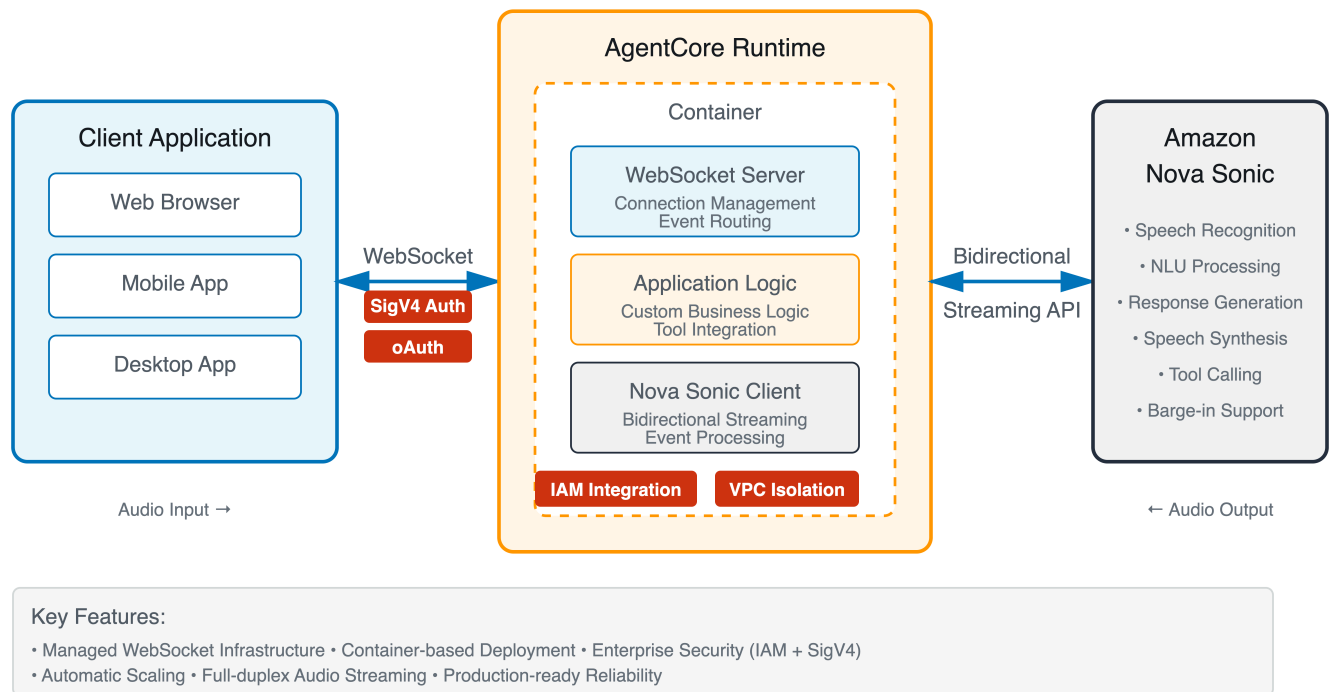
架構整合

Amazon Nova 2 Sonic 可與各種架構和平台整合，以建置複雜的語音應用程式。下列範例示範與熱門架構的整合模式。

Amazon Bedrock AgentCore

Amazon Bedrock AgentCore 提供受管執行期環境，用於部署具有企業級安全性和可擴展性的 Nova 2 Sonic 應用程式。AgentCore 透過處理基礎設施、身分驗證和 WebSocket 連線，簡化即時語音 AI 應用程式的部署。

AgentCore Architecture



主要功能：

- 雙向串流 - 原生支援 Nova Sonic 的全雙工串流介面，具有即時事件處理和低延遲通訊。
- WebSocket 基礎設施 - 生產就緒的 WebSocket 伺服器，具有自動擴展、連線管理和錯誤復原。
- 容器部署 - 使用水平擴展和獨立版本控制，將 Nova Sonic 應用程式部署為受管基礎設施的容器。
- 企業安全 - 透過 IAM 和 SigV4 的精細驗證、VPC 隔離和全面的稽核記錄。

架構顯示用戶端應用程式如何使用 SigV4 身分驗證透過 WebSocket 連線至 AgentCore 執行期。容器化環境包含您的 WebSocket 伺服器、應用程式邏輯和 Nova Sonic 用戶端，所有用戶端都透過雙向串流 API 與 Nova Sonic 通訊。

優點：

- 簡化的操作：專注於應用程式邏輯，同時 AgentCore 管理基礎設施、擴展和可靠性。
- 企業安全：生產部署的內建身分驗證、授權和合規功能。
- 成本效益：僅針對您用於自動擴展和資源最佳化的項目付費。
- 開發人員生產力：使用受管 WebSocket 基礎設施和容器部署來縮短生產時間。

使用案例

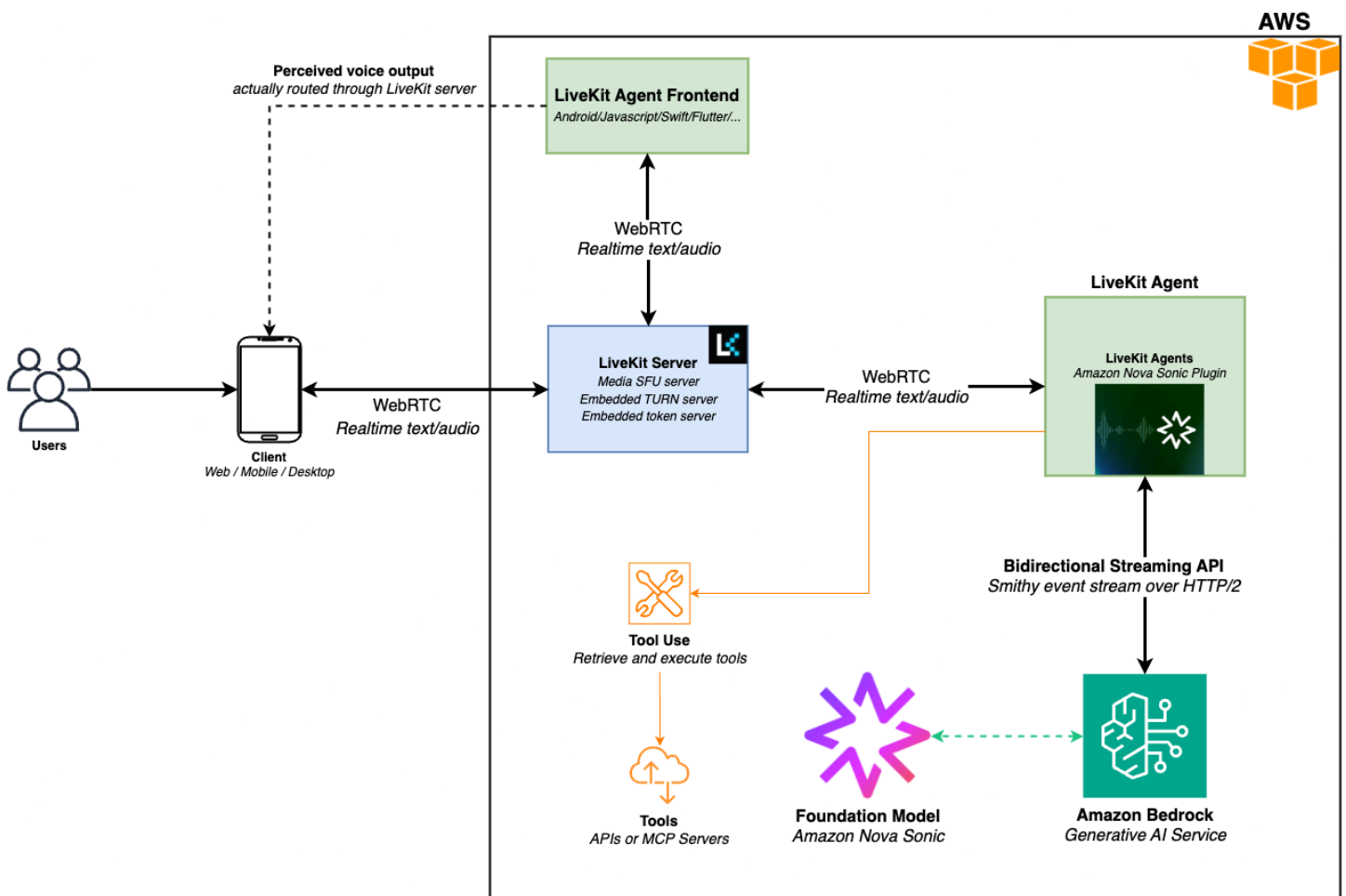
- 具有安全身分驗證的客戶服務語音助理
- 需要 IAM 整合的企業語音應用程式
- 具有隔離部署的多租戶語音平台
- 語音應用程式需要合規和稽核線索

如需使用 AgentCore 部署 Nova Sonic 的詳細文件，請參閱 [Amazon Bedrock AgentCore 文件](#)。

LiveKit

LiveKit 是用於建置即時音訊和視訊應用程式的開放原始碼平台。與 Amazon Nova 2 Sonic 整合可讓開發人員建置對話語音界面，而無需管理複雜的音訊管道或訊號通訊協定。

如需詳細的實作範例和程式碼範例，請參閱 [LiveKit AWS 整合文件](#)。



運作方式：

- 用戶端層：Web、行動或桌面應用程式會使用 LiveKit 的用戶端 SDKs 進行連線，該 SDK 會處理音訊擷取、WebRTC 串流和播放。
- LiveKit 伺服器：充當即時通訊中樞、管理 WebRTC 連線、路由音訊串流，以及使用低延遲最佳化處理工作階段狀態。
- LiveKit 代理程式：Python 型代理程式，可接收來自伺服器的音訊、透過 Nova Sonic 外掛程式處理音訊，並將回應串流回去。包括內建功能，例如語音活動偵測和轉彎管理。
- Amazon Nova 2 Sonic：透過雙向串流 API 處理音訊串流、執行語音辨識、自然語言理解，以及使用合成語音產生對話式回應。

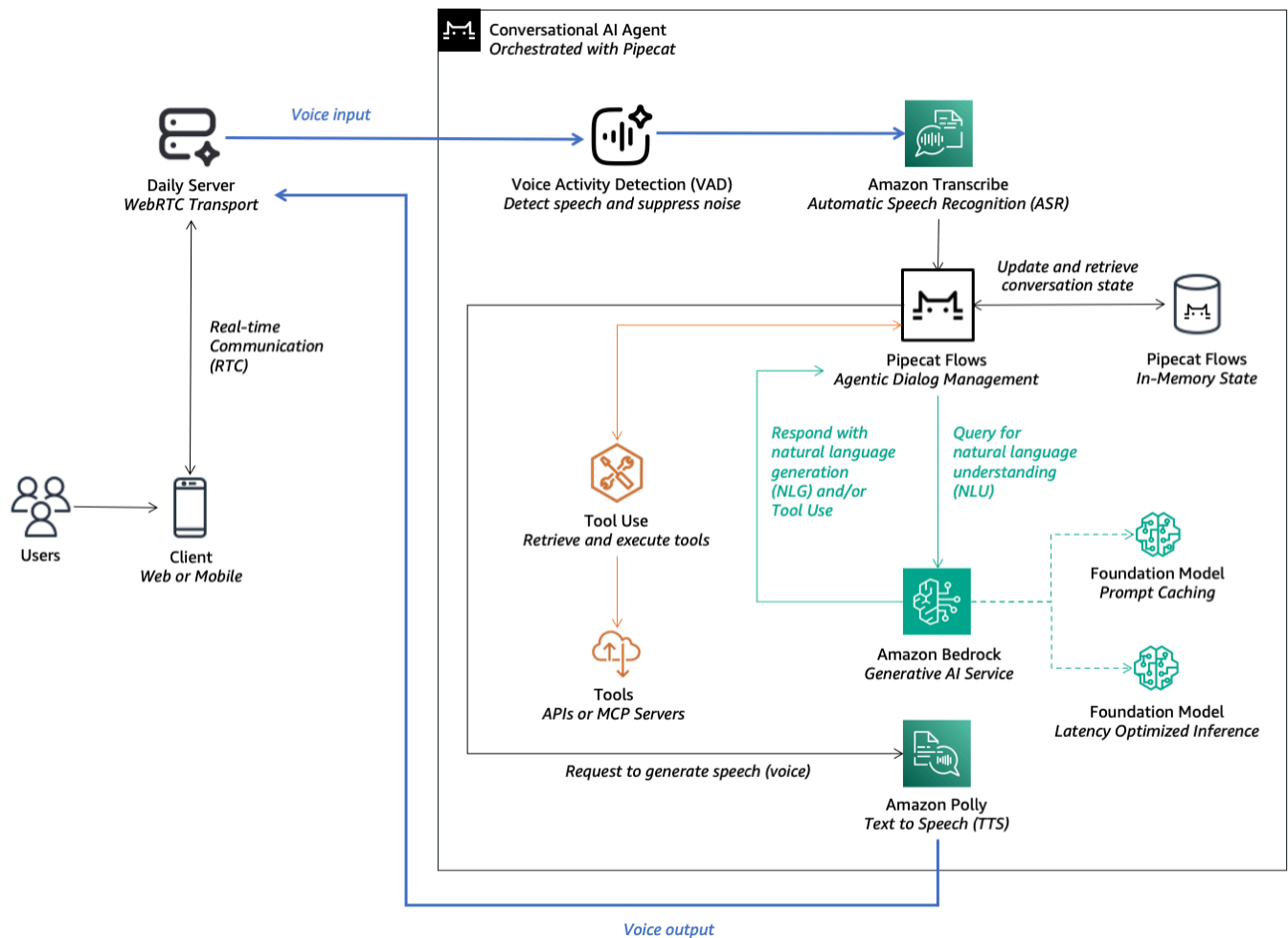
Pipecat

Pipecat 是建置語音和多模式對話式 AI 應用程式的架構。它提供模組化、以管道為基礎的架構，可協調多個元件，以使用 Amazon Nova Sonic 和其他 AWS 服務建立智慧型語音應用程式。

如需詳細的實作範例和程式碼範例，請參閱 [PipeCat AWS 整合文件](#)。

主要功能：

- 管道架構：模組化 Python 架構，用於編寫語音 AI 元件，包括 ASR、NLU、TTS 等。
- Pipecat 流程：用於建置複雜對話邏輯和工具執行的狀態管理架構。
- WebRTC 支援：內建與每日和其他 WebRTC 供應商的整合，以進行即時音訊串流。
- AWS 整合：Amazon Bedrock、Amazon Transcribe 和 Amazon Polly 的原生支援。



架構包括：

- WebRTC 傳輸：用戶端裝置和應用程式伺服器之間的即時音訊串流。
- 語音活動偵測 (VAD)：具有可設定語音偵測和雜訊抑制的 Silero VAD。
- 語音辨識：Amazon Transcribe 可提供準確的即時speech-to-text轉換。

- 自然語言理解：Amazon Nova Pro on Bedrock 搭配延遲最佳化推論。
- 工具執行：用於 API 整合和後端服務呼叫的管道流程。
- 回應產生：Amazon Nova Pro 提供一致的內容感知回應。
- Text-to-speech：具有生成語音的 Amazon Polly，可提供逼真的語音輸出。

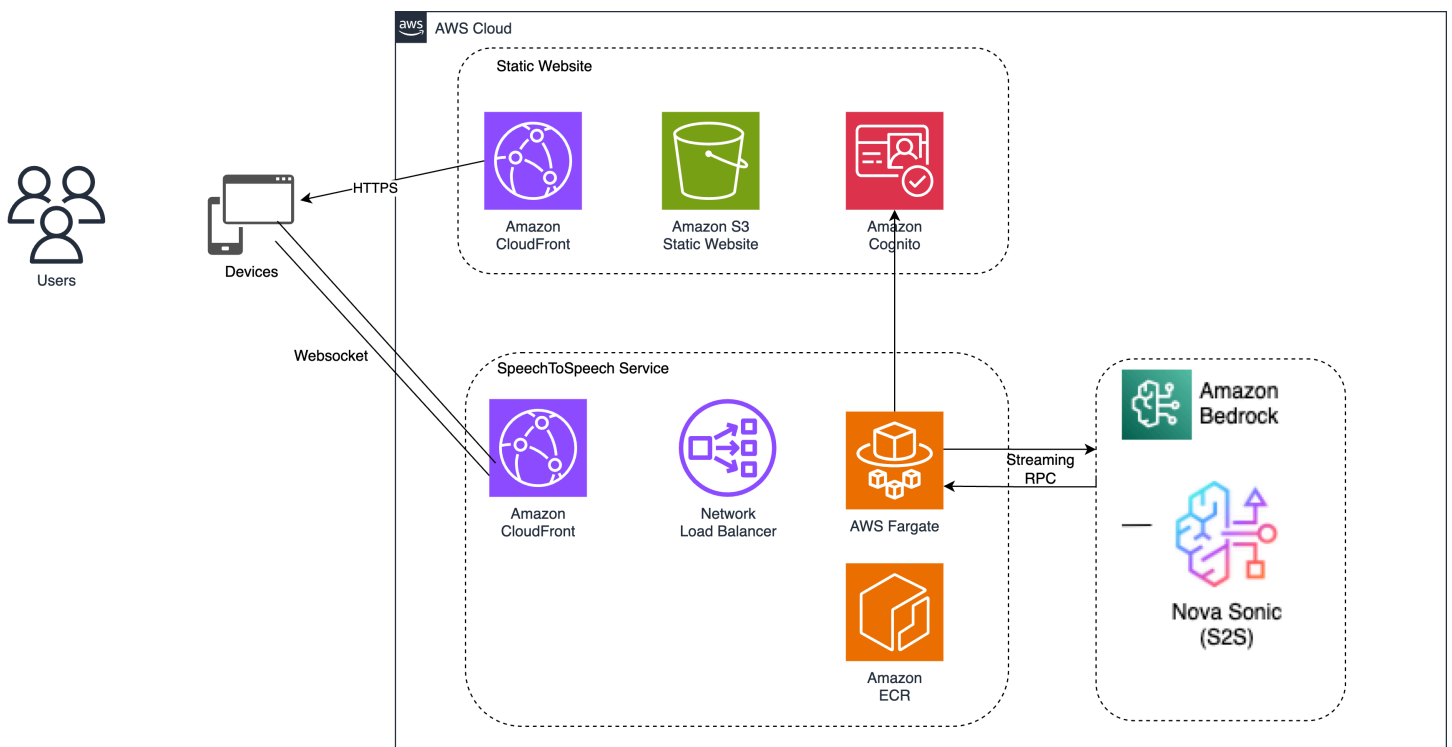
在上部署 AWS

將 Nova Sonic 應用程式部署至 AWS 使用基礎設施做為 AWS CDK（雲端開發套件）的程式碼。此方法提供可重複、版本控制的部署，並內建最佳實務。

部署選項

- Amazon ECS（彈性容器服務）：使用 Application Load Balancer 整合、自動擴展和無伺服器 Fargate 執行進行全受管容器協調。
- Amazon EKS（彈性 Kubernetes 服務）：適用於複雜協同運作、進階聯網、多區域部署和廣泛工具生態系統的受管 Kubernetes。
- AWS CDK：AWS CDK 可讓您使用熟悉的程式設計語言定義雲端基礎設施。

如需使用 AWS CDK 部署 Nova Sonic 的完整生產就緒範例，請參閱 GitHub [Speech-to-Speech CDK 範例](#)。此範例將示範：



- 使用 TypeScript 完成 CDK 基礎設施設定
- 即時通訊的 WebSocket 伺服器實作
- 使用 ECS 和 Fargate 進行容器部署
- WebSocket 支援的 Application Load Balancer 組態
- VPC 聯網和安全群組設定
- CloudWatch 監控和記錄
- 生產部署的最佳實務

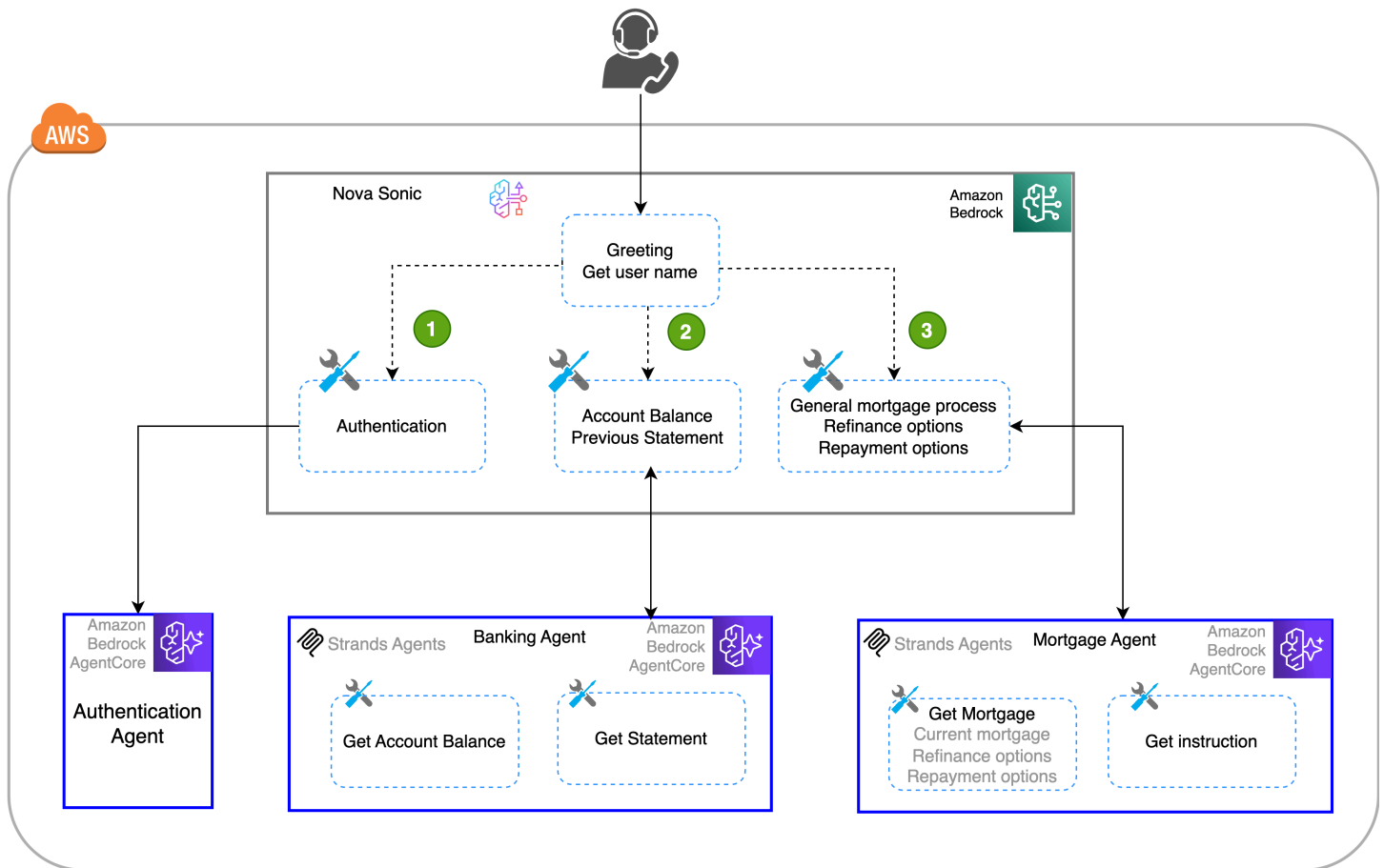
多代理系統

多代理程式架構是廣泛使用的模式，用於設計處理複雜任務的 AI 助理。在採用 Nova 2 Sonic 技術的語音助理中，此架構會協調多個專門的代理程式，其中每個代理程式都會獨立運作，以啟用平行處理、模組化設計和可擴展的解決方案。

Nova Sonic 做為多代理程式系統中的協調器，執行兩個關鍵函數：

對話流程管理：確保在繼續對話的下一個步驟之前收集所有必要資訊。

意圖分類：分析使用者查詢，並將其路由至適當的專業子代理程式。



上圖顯示使用多客服人員架構的銀行語音助理。對話流程從問候和收集使用者名稱開始，然後透過專業子代理處理與銀行或抵押貸款相關的查詢。

對話流程範例：

1. 使用者連接到語音助理。
2. Nova 2 Sonic：「您好！您的名字是什麼？」
3. 使用者：「我的名字是 John」
4. Nova 2 Sonic：「Hi John，我今天如何協助您？」
5. 使用者：「我想要檢查我的帳戶餘額」
6. Nova 2 Sonic：【身分驗證代理程式的路由】
7. 身分驗證代理程式：「請提供您的帳戶 ID」
8. 使用者："12345"
9. 身分驗證代理程式：【驗證身分】
10. Nova 2 Sonic：【轉接至銀行客服人員】

11 銀行代理程式：「您目前的餘額為 5,431,10 美元」

雖然此範例示範使用部署在 Amazon Bedrock AgentCore 上的 Strands Agents 架構的子代理程式，但架構具有彈性。您可以選擇：

- 您偏好的代理程式架構
- 任何 LLM 供應商
- 自訂託管選項
- 不同的協同運作模式

優點：

- 模組化：每個代理程式都專注於特定網域，讓系統更容易維護和更新。
- 可擴展性：新增客服人員而不修改現有的客服人員，讓您的系統能夠隨您的需求成長。
- 平行處理：多個客服人員可以同時運作，改善複雜查詢的回應時間。
- 專業：可以使用最適當的工具和知識庫，針對其特定任務最佳化每個代理程式。
- 故障隔離：如果一個代理程式失敗，其他代理程式會繼續運作，改善整體系統可靠性。

如需更多詳細資訊和程式碼範例，請參閱[此部落格](#)。

如需實作範例，請參閱 [Nova Sonic Workshop Multi-Agent Lab](#)。

電話語音整合

Amazon Nova 2 Sonic 與電話服務供應商整合，讓採用 AI 技術的語音應用程式可以透過電話進行存取。本指南涵蓋與 Twilio、Vonage 和其他以 SIP 為基礎的系統的整合，用於建置聯絡中心解決方案和語音客服人員。

Twilio：具有可程式設計語音和媒體串流功能的雲端通訊平台。

Vonage：具有語音、WebSocket 音訊串流和 SIP 連線能力的全域通訊 APIs。

AWS 透過即時分析和電話整合，提供在聯絡中心環境中示範 Nova Sonic 的完整範例實作。

儲存庫：[使用電話的範例音波聯絡中心](#)

使用 Amazon Nova 的進階系統

概觀

本節為使用 Nova 模型實作進階模式的進階使用者提供全面的指導。這些功能可讓複雜的應用程式，包括工具整合、擷取擴增產生 (RAG) 和自主 AI 代理器。

Amazon Nova 2 中的延伸思維

Amazon Nova 2 Lite 推出延伸的思維功能，讓模型能夠對複雜的問題進行更深入的推理。此選用功能可讓您控制模型配置其他運算資源的時間和方式，以仔細考慮具有挑戰性的任務。

延伸思考的運作方式

Amazon Nova 2 將延伸思維引入為混合功能。您可以完全控制：

- 延長思維關閉（預設）：Amazon Nova 2 以有效率的隱含推理運作，最適合日常任務和大量應用程式。
- 開啟擴展思維：Amazon Nova 2 採用明確的step-by-step推理，最適合需要深入分析的複雜問題。

合理努力

以下顯示不同程度的推理工作。

低工作量 (maxReasoningEffort : "low")

最適合：具有額外複雜性的任務需要結構化思考。例如，您可以將此用於程式碼檢閱和改進建議，其中模型需要仔細考慮現有的程式碼品質、需要考慮多個因素的分析任務，或受益於系統化方法的問題解決案例。低工作量適用於複合任務，其中基本推理可提高準確性，而不需要深入的多步驟規劃。

中等努力 (maxReasoningEffort : "medium")

最適合：多步驟任務和編碼工作流程。例如，您可以將此用於軟體開發和偵錯，其中模型需要在實作變更之前了解現有的程式碼結構、需要跨多個檔案或元件進行協調的程式碼產生、具有相互依存性的多步驟計算，或規劃具有多個限制的任務。對於協調多個工具的代理程式工作流程而言，中等程度的工作量是最佳的，並且需要模型來維持數個循序操作的內容。

高工作量 (maxReasoningeffort : "high")

最適合：STEM 推理和進階問題解決。例如，您可以將此用於需要嚴格step-by-step驗證的進階數學問題和證據、需要深入調查的科學分析和研究任務、具有跨多個維度架構考量的複雜系統設計，或具有重大影響的關鍵決策案例。對於需要複雜推理的任務、仔細評估替代方案，以及徹底驗證結論，高度努力可提供最大的準確性。

快速入門：啟用延伸思考

延伸思考是透過 reasoningConfig 參數控制。

```
import boto3

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

response = bedrock.converse(
    modelId='us.amazon.nova-2-lite-v1:0',
    system=[{"text": "You are a highly capable personal assistant"}],
    messages=[{
        "role": "user",
        "content": [{"text": "Provide a meal plan for a gluten free family of 4."}]
    }],
    inferenceConfig={
        "temperature": 0.7,
        "topP": 0.9,
        "maxTokens": 10000
    },
    additionalModelRequestFields={
        "reasoningConfig": {
            "type": "enabled",
            "maxReasoningEffort": "low"
        }
    }
)

content_list = response["output"]["message"]["content"]

for item in content_list:
    if "reasoningContent" in item:
        reasoning_text = item["reasoningContent"]["reasoningText"]["text"]
        print("=== REASONING ===")
        print(reasoning_text)
        print()
```

```
elif "text" in item:
    print("=== ANSWER ===")
    print(item["text"])
```

原因參數：

- type: enabled 或 disabled (預設: disabled)
- maxReasoningEffort: medium、low 或 high

Note

溫度、topP 和 topK 無法與 maxReasoningEffort 設定為 搭配使用 high。將這些參數一起使用會導致錯誤。

如需利用延伸思考的完整程式碼範例，請參閱 [the section called “程式碼庫”](#)。

回應結構

當您啟用延伸思考時，回應會包含 reasoningContent 區塊，後面接著 text 內容區塊：

```
{
  "output": {
    "message": {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
              "text": "[REDACTED]"
            }
          }
        },
        {
          "text": "Based on the premises, we can conclude..."
        }
      ]
    }
  },
  "stopReason": "end_turn"
```

```
}
```

Note

使用 Amazon Nova 2，推理內容會顯示為 [REDACTED]。您仍需支付推理權杖的費用，因為它們有助於改善輸出品質。我們現在在回應結構中包含此欄位，以保留未來公開推理內容的選項。我們正積極與客戶合作，以判斷浮現模型推理程序的最佳方法。原因內容會顯示為 [REDACTED]。您仍需支付推理權杖的費用，因為它們有助於改善回應品質。

組態選項

Amazon Nova 2 引入了新的 `reasoningConfig` 參數，您可以將此參數新增至現有的衝突請求結構，以啟用推理：

```
additionalModelRequestFields={
  "reasoningConfig": {
    "type": "enabled", # or "disabled" (default)
    "maxReasoningEffort": "high" # "low", "medium", or "high"
  }
}
```

參數：

- 類型：在 "enabled" 和 "disabled" 之間切換 (預設為 "disabled")
- **maxReasoningEffort**：啟用時，控制推理深度
- 「低」：中等複雜的任務
- "medium"：需要大量分析的複雜問題
- "high"：高度複雜任務的最徹底推理

Note

使用時 "high"，必須取消設定 `temp`、`topP` 和 `maxToken`。在此模式中，模型會執行更深入的分析，以尋找最佳解決方案。這種更徹底的處理可能會產生超過 65,000 個字符的輸出。確切數量取決於請求的複雜性，但對於某些問題，我們看到它高達 128K0 個字符。這可確保獲得完整、高品質的推理，而不是截斷的結果。

支援的模型

擴展思維目前可用於：Amazon Nova 2 Lite (us.amazon.nova-2-lite-v1 : 0)

工具使用的延伸思考

延伸思維可與工具呼叫無縫搭配運作，讓 Amazon Nova 能夠推斷要使用哪些工具，以及如何解譯其結果。

了解字符和定價的原因

字符類型

延伸思維權杖會以輸出權杖計費：

- 輸入字符：您的原始請求內容（標準輸入定價）
- 輸出字符：這包括推理字符和最終可見回應內容（標準輸出定價）

用量明細

這三種字符類型都包含在用量指標和帳單中。原因字符的價格與輸出字符相同，並在模型回應中顯示為「已刪除」。

```
{
  "usage": {
    "inputTokens": 45,
    "outputTokens": 1240,
    "totalTokens": 1285
  }
}
```

常見問答集

為什麼 Amazon Nova 2 Lite 顯示推斷內容的「【REDACTED】」而不是顯示模型的思考程序？

我們此次啟動的主要重點是確保 Nova 2 為您的任務提供同級最佳的智慧，並且您會看到這一點反映在提高的準確性中。

我們了解推理程序的可見性很有價值，而且我們聽說客戶對於了解模型如何看待問題有強烈的興趣。

我們即將探討提供此功能的方法。

您仍需支付推理權杖的費用，因為它們代表可改善輸出品質的實際工作

將與答案字符 `outputTokens` 一起在 `中` 擷取。

如果修改推理，如何知道延伸思考是否有效？

您可以透過檢查回應中 `reasoningContent` 區塊的輸出來確認延伸思維是否正常運作（只有在啟用推理時才會顯示）

使用工具（函數呼叫）

工具透過將模型連接到 APIs、資料庫和程式碼執行環境等外部功能來擴展 Amazon Nova 功能。工具使用可讓 Amazon Nova 存取即時資訊、執行計算並與外部系統互動。

了解工具使用工作流程

搭配 Amazon Nova 使用工具涉及三個關鍵階段：

使用者查詢和工具定義

透過提供描述每個工具的功能和輸入要求的 JSON 結構描述來定義工具。工具組態必須包含有關何時及如何使用每個工具的明確詳細資訊。

工具選擇

當使用者傳送訊息時，Amazon Nova 會對其進行分析，以判斷是否需要工具。此自動工具選擇會檢查內容，並決定要叫用哪個工具（如果有的話）。如果 Amazon Nova 識別合適的工具，則會傳回工具名稱和必要的參數。

您有責任根據模型的請求執行工具。這表示編寫程式碼來叫用工具的功能，並處理模型提供的輸入參數。

傳回結果

執行工具後，請使用 JSON 或文字和影像的組合，以結構化格式將結果傳回 Amazon Nova。Amazon Nova 會將工具的輸出納入最終回應。如果在執行期間發生錯誤，請在工具回應中指出這一點，以允許 Amazon Nova 相應地調整。

建立工具

使用工具組態來定義工具，其中包含一系列工具和選用的工具選擇參數。每個工具規格必須包含：

- 名稱：清除工具的識別符
- 描述：工具功能的簡要說明
- 輸入結構描述：定義必要和選用參數的 JSON 結構描述

Example工具組態範例

```
tool_config = {
  "tools": [
    {
      "toolSpec": {
        "name": "calculator",
        "description": "A calculator tool that can execute a math equation",
        "inputSchema": {
          "json": {
            "type": "object",
            "properties": {
              "equation": {
                "type": "string",
                "description": "The full equation to evaluate"
              }
            },
            "required": ["equation"]
          }
        }
      }
    }
  ]
}
```

工具定義的最佳實務

- 確保名稱和描述明確傳達工具的確切功能；避免過於相似的語彙工具
- 在描述中包含關鍵差異點，以協助模型區分類似的工具
- 將 JSON 結構描述限制為兩層巢狀，以獲得最佳效能
- 使用結構描述類型（例如列舉、整數、浮點數）來限制輸入，而不是以純文字描述結構

- 使用 JSON 結構描述表示法（例如，「必要」：【「param1」、「param2」】）來表示必要參數與選用參數
- 在提交之前，使用標準驗證器驗證您的 JSON 結構描述
- 將長字串引數最後放在結構描述中，並避免將其巢狀化

具有限制解碼的結構式輸出

Amazon Nova 模型會利用限制解碼，以確保產生輸出中的高可靠性。此技術使用文法來限制每個產生步驟的可能字符，防止無效的金鑰，並根據您定義的結構描述強制執行正確的資料類型。

Example 結構化輸出範例

```
tool_config = {
  "tools": [
    {
      "toolSpec": {
        "name": "ProductAnalysis",
        "description": "Analyze product information from text.",
        "inputSchema": {
          "json": {
            "type": "object",
            "properties": {
              "name": {
                "type": "string",
                "description": "Product name"
              },
              "rating": {
                "maximum": 5,
                "description": "Customer rating 1-5",
                "type": ["number", "null"],
                "minimum": 1
              },
              "features": {
                "description": "Key product features",
                "type": "array",
                "items": {"type": "string"}
              },
              "category": {
                "type": "string",
                "description": "Product category"
              }
            }
          }
        }
      }
    }
  ]
}
```

```

        "price": {
            "type": "number",
            "description": "Price in USD"
        },
        "required": ["name", "category", "price", "features"]
    }
}
],
"toolChoice": {
    "tool": {"name": "ProductAnalysis"}
}
}

```

工具選擇選項

Amazon Nova 支援三種工具選擇參數：

工具

指定的工具將呼叫一次，非常適合結構化輸出使用案例

任何

其中一個提供的工具至少會呼叫一次，適用於 API 選擇案例

自動

模型決定是否呼叫工具，以及要呼叫多少工具（預設行為）

呼叫工具

當 Amazon Nova 決定呼叫工具時，它會傳回工具使用區塊，做為 stopReason 設定為 "tool_use" 之輔助訊息的一部分。工具區塊包含工具名稱及其輸入。

Note

在單一 Python 工作階段中依序執行下列程式碼區段（呼叫工具 → 處理工具呼叫 → 傳回工具結果）。若要再次執行範例，請重新啟動 Python 工作階段。

Example 呼叫工具

```
import boto3
import json

# Create Bedrock client
bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

# Complex calculation that benefits from precise computation
messages = [{
    "role": "user",
    "content": [{
        "text": "Calculate the compound interest on $10,000 invested at 4.75% annual
rate for 7 years, compounded quarterly. Use the formula  $A = P(1 + r/n)^{nt}$  where
 $P=10000$ ,  $r=0.0475$ ,  $n=4$ ,  $t=7$ "
    }]
}]

# Define tool configuration with calculator
tool_config = {
    "tools": [{
        "toolSpec": {
            "name": "calculator",
            "description": "Perform mathematical calculations",
            "inputSchema": {
                "json": {
                    "type": "object",
                    "properties": {
                        "expression": {
                            "type": "string",
                            "description": "Mathematical expression to evaluate"
                        }
                    }
                },
                "required": ["expression"]
            }
        }
    }]
}

# Invoke Model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
```

```

        toolConfig=tool_config
    )

# Extract tool use from response
tool = next(
    block["toolUse"]
    for block in response["output"]["message"]["content"]
    if "toolUse" in block
)

print(f"Tool: {tool['name']}")
print(f"Expression: {tool['input']['expression']}")

```

處理工具呼叫

從訊息中擷取工具名稱和引數，然後叫用工具：

```

def calculate(expression):
    """Evaluate mathematical expression"""
    print(f"Calculating: {expression}")
    P = 10000
    r = 0.0475
    n = 4
    t = 7
    result = P * (1 + r/n) ** (n*t)
    return result

stop_reason = response["stopReason"]

if stop_reason == "tool_use":
    if tool["name"] == "calculator":
        result = calculate(tool["input"]["expression"])

```

傳回工具結果

使用 ToolResultBlock 結構描述傳回工具結果：

```

messages.append(response["output"]["message"])

# Add the tool result
messages.append({
    "role": "user",

```

```

        "content": [{
            "toolResult": {
                "toolUseId": tool['toolUseId'],
                "content": [{"json": {"result": result}}],
                "status": "success"
            }
        }]
    })

# Send the tool result to the model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
    toolConfig=tool_config
)

# Extract and display final response
final_text = next(
    block["text"]
    for block in response["output"]["message"]["content"]
    if "text" in block
)

print(f"\nFinal Response:\n{final_text}")

```

錯誤處理

向 Amazon Nova 回報錯誤，以允許修改請求並重試：

```

tool_result_message = {
    "role": "user",
    "content": [
        {
            "toolResult": {
                "toolUseId": tool["toolUseId"],
                "content": [{"text": "A validation exception occurred on field:
sample.field"}],
                "status": "error"
            }
        }
    ]
}

```

安全考量

- 在叫用工具之前驗證工具是否存在
- 確保輸入格式正確
- 在工具執行之前，確認有適當的許可
- 依賴工作階段詳細資訊，而不是允許 Amazon Nova 將使用者資訊注入工具呼叫
- 請記住，LLMs 可以幻化工具呼叫；在執行之前一律驗證

內建系統工具

Amazon Nova 2.0 模型包含完全受管的內建工具，不需要自訂實作。在 Converse API 中使用簡單的切換來啟用這些工具。

程式碼解譯器

Code Interpreter 可讓 Nova 在隔離的沙盒環境中安全地執行 Python 程式碼。此工具專為數學運算、邏輯操作和迭代演算法而設計。

Note

程式碼解譯器適用於 IAD、PDX 和 NRT AWS 區域。若要確保您的請求路由到支援的區域，請使用全域 CRIS。使用 Bedrock API 金鑰時，您需要手動將 `InvokeTool` 許可新增至政策定義。預設 Bedrock 角色不允許 `InvokeTool` 動作。

透過指定 `systemTool` 參數啟程式碼解譯器：

```
import boto3
import json

bedrock = boto3.client('bedrock-runtime', region_name='us-east-1')

tool_config = {
    "tools": [{
        "systemTool": {
            "name": "nova_code_interpreter"
        }
    }]
}
```



```

}

response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=[{
        "role": "user",
        "content": [{
            "text": "What is the average of 10, 24, 2, 3, 43, 52, 13, 68, 6, 7, 902,
82"
        }]
    }],
    toolConfig=tool_config,
    inferenceConfig={"maxTokens": 10000, "temperature": 0}
)

# Pretty print the response
for block in response["output"]["message"]["content"]:
    if "toolUse" in block:
        print("=== Tool Use ===")
        print(f"Tool: {block['toolUse']['name']}")
        print(f"Code:\n{block['toolUse']['input']['snippet']}\n")

    elif "toolResult" in block:
        print("=== Tool Result ===")
        result = block['toolResult']['content'][0]['json']
        print(f"Output: {result['stdOut']}")
        if result['stdErr']:
            print(f"Error: {result['stdErr']}")
        print(f"Exit Code: {result['exitCode']}\n")

    elif "text" in block:
        print("=== Final Answer ===")
        print(block["text"])

```

解譯器會在沙盒中執行程式碼，並傳回標準結構描述的結果：

```

{
    "stdOut": "String",
    "stdErr": "String",
    "exitCode": "int",
    "isError": "boolean"
}

```

Web Grounding

Web 接地可讓 Amazon Nova 從網際網路存取即時資訊，提供up-to-date回應並減少幻覺。透過指定 nova_grounding 系統工具來啟用：

```
tool_config = {
    "tools": [{
        "systemTool": {"name": "nova_grounding"}
    }]
}
```

如需 Web Grounding 的詳細資訊，請參閱 [the section called “Web Grounding”](#)。

模型內容通訊協定 (MCP)

模型內容通訊協定 (MCP) 是一種開放標準，可在資料來源和 AI 支援的工具之間啟用安全的雙向連線。執行 MCP 伺服器並讓 Amazon Nova 透過用戶端橋接自動探索其工具，而不是為每個 API 或服務撰寫自訂轉接器。

連線後，Amazon Nova 會將 MCP 工具視為任何其他外部整合：它會決定何時呼叫它們、傳送必要的參數，並將結果納入回應中。搭配 Strands 使用 Amazon Nova 可讓您更輕鬆地使用內建的 MCPClient，自動管理探索、連線和結果映射。

建置 AI 代理程式

Amazon Nova 模型已針對使用 Amazon Nova Act 建置 AI 代理器進行最佳化。這些模型提供改善的工具使用、改善多步驟任務的推理、增強在複雜客服人員工作流程之間維護內容的能力，以及支援遠端 MCP 工具。

建立代理程式

使用 Nova 建置的 AI 代理器可以協調多個工具呼叫、維持延伸互動的內容，並在需要時更正課程。延伸思考透過透過複雜目標啟用系統推理來轉換代理程式工作流程。考慮使用規劃架構 SDK，例如 Strands Agents，讓您的代理系統的規劃和執行程序更強大。

代理程式設計模式

使用 Nova 設計代理程式時：

- 針對需要規劃和驗證的複雜多步驟工作流程，在中等或高層級上啟用推理，以獲得最佳結果

- 實作工具選擇auto，以允許跨客服人員互動的彈性工具選擇
- 設計錯誤處理，允許代理程式使用修改過的方法來復原和重試
- 維護對話歷史記錄，以保留客服人員互動的內容
- 在代理程式系統使用的未受控制內容中，實作強大的內容篩選和管制機制。例如，Amazon 提供 Amazon Bedrock Guardrails，這項功能旨在跨多個基礎模型、知識庫和代理程式套用保護措施。這些護欄可以篩選有害內容、封鎖拒絕的主題，以及修訂敏感資訊，例如個人身分識別資訊。

多工具代理程式範例

```
tool_config = {
  "tools": [
    #{
      "toolSpec": {
        "name": "calculator",
        "description": "Perform mathematical calculations",
        "inputSchema": {
          "json": {
            "type": "object",
            "properties": {
              "expression": {
                "type": "string",
                "description": "Mathematical expression to evaluate"
              }
            }
          },
          "required": ["expression"]
        }
      }
    },
    #{
      "toolSpec": {
        "name": "database_query",
        "description": "Query financial database for historical data",
        "inputSchema": {
          "json": {
            "type": "object",
            "properties": {
              "query": {
                "type": "string",
                "description": "SQL query to execute"
              }
            }
          }
        }
      }
    }
  ]
}
```

```

        #},
        #"required": ["query"]
    #}
    #}
    #}
    #}
    #]
}

response = client.converse(
    #modelId=" us.amazon.nova-2-lite-v1:0",
    #messages=[{
        #"role": "user",
        #"content": [{
            #"text": "Analyze our Q3 financial performance across all business units,
calculate year-over-year growth rates with statistical significance testing, and
recommend budget allocation strategies for Q4."
        #}]
    #}],
    #toolConfig=tool_config,
    #inferenceConfig={"maxTokens": 10000, "temperature": 1, "topP": 0.9},
    #additionalModelRequestFields={
        #"reasoningConfig": {
            #"type": "enabled",
            #"maxReasoningEffort": "low"
        #}
    #}
)

```

調用代理程式

客服人員調用涉及管理對話流程、處理工具呼叫，以及維護多個互動的狀態。

串流代理程式回應

串流回應以提供客服人員推理和動作的即時可見性：

```

import boto3
response = client.converse_stream(
    #modelId=" us.amazon.nova-2-lite-v1:0",
    #messages=[{
        #"role": "user",
        #"content": [{

```

```

        #"text": "Design a scalable microservices architecture for an e-commerce
platform handling 1M+ daily transactions. Consider data consistency, fault tolerance,
performance, security, and cost optimization."
    #}]
#}],
#inferenceConfig={"maxTokens": 10000, "temperature": 10},
#additionalModelRequestFields={
    #"reasoningConfig": {
        #"type": "enabled",
        #"maxReasoningEffort": "low"
    #}
#}
)

# Process the streaming response
reasoning_complete = False
for event in response["stream"]:
    #if "contentBlockDelta" in event:
        #delta = event["contentBlockDelta"]["delta"]

        #if "reasoningContent" in delta:
            #reasoning_text = delta["reasoningContent"]["reasoningText"]["text"]
            #print(f"{reasoning_text}", end="", flush=True)
        #elif "text" in delta:
            #if not reasoning_complete:
                #print(f"
Final Architecture Design:
")
        #reasoning_complete = True
        #print(f"{delta['text']}", end="", flush=True)

```

客服人員狀態管理

維護對話歷史記錄和工具結果以保留內容；以下範例示範一次，但開發人員可以根據工作流程需求判斷如何協調整體客服人員系統。此外，Strands 等 Amazon Web Services 工具會代表開發人員管理代理程式內容和工具狀態。

```

messages = []

messages = []

# Initial user query

```

```
messages.append({
    #"role": "user",
    #"content": [{"text": user_query}]
})

# Get agent response
response = client.converse(
    #modelId=" us.amazon.nova-2-lite-v1:0",
    #messages=messages,
    #toolConfig=tool_config,
    #inferenceConfig=inf_params
)

# Add assistant response to history
messages.append(response["output"]["message"])

# Process tool calls and add results
if response["stopReason"] == "tool_use":
    #tool = next(
        #block["toolUse"]
        #for block in response["output"]["message"]["content"]
        #if "toolUse" in block
    #)

    ## Execute tool
    #result = execute_tool(tool["name"], tool["input"])

    ## Add tool result to conversation
    #messages.append({
        #"role": "user",
        #"content": [{
            #"toolResult": {
                #"toolUseId": tool["toolUseId"],
                #"content": [{"json": result}],
                #"status": "success"
            #}
        #}]
    #})

    ## Continue conversation
    #response = client.converse(
        #modelId=" us.amazon.nova-2-lite-v1:0",
        #messages=messages,
        #toolConfig=tool_config,
```

```
#inferenceConfig=inf_params
#)
```

客服人員最佳實務

如需客服人員最佳實務的詳細資訊，請參閱 [the section called “最佳實務”](#)。

如需開發對話式 AI 代理器的指引，請參閱 [Speech-to-Speech](#)。

Web Grounding

Web Grounding 可讓 Amazon Nova 搜尋 Web 以取得目前資訊，並提供包含引文的回應。此功能適用於需要模型訓練資料以外 up-to-date 的查詢。

Web Grounding 的運作方式

為提示詞啟用 Web Grounding 後，會執行下列步驟：

1. 請求組態：您的應用程式會將使用者提示傳送至已啟用 nova_grounding 的 Amazon Bedrock API systemTool。
2. 搜尋與分析：模型會判斷是否需要搜尋、執行一或多個相關資訊搜尋，以及評估是否需要額外的搜尋，以擴展其對特定主題的理解或深入了解。
3. 回應產生：Amazon Nova 會自動合成搜尋結果中的資訊，以產生以即時資訊為基礎的最終 API 回應，並附上其來源的引文。

如何使用 Web Grounding

如需使用 Web Grounding 之程式碼的完整範例，請參閱程式碼範例一節。

若要在您的結果中包含 Web Grounding，請在 toolConfig 區塊中指定下列 systemTool 參數：

```
import boto3
from botocore.config import Config

# Create the Bedrock Runtime client with extended timeout
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
```

```

    config=Config(read_timeout=3600)
)

# Define the tool configuration
tool_config = {
    "tools": [{
        "systemTool": {
            "name": "nova_grounding"
        }
    }]
}

# Send the request
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=[{
        "role": "user",
        "content": [{"text": "What are the latest developments in quantum computing?"}]
    }],
    toolConfig=tool_config
)

# Extract text with interleaved citations
output_with_citations = ""
content_list = response["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
        output_with_citations += content["text"]
    elif "citationsContent" in content:
        citations = content["citationsContent"]["citations"]
        for citation in citations:
            url = citation["location"]["web"]["url"]
            output_with_citations += f" [{url}]"

print(output_with_citations)

```

區域可用性

Web Grounding 目前僅適用於美國區域，且僅受美國 CRIS 設定檔支援。

回應結構

以下是回應範例。為了簡潔起見，已縮短回應：


```
{
  "output": {
    "message": {
      "content": [
        {
          "text": "Recent quantum computing developments include...",
          "citationsContent": [
            {
              "location": {
                "web": {
                  "url": "https://example.com/quantum-news",
                  "domain": "example.com"
                }
              }
            }
          ]
        }
      ]
    }
  }
}
```

每個引文包括：

- `text`：模型產生的回應區段。
- `citationsContent`：與文字區段相關的引文資料的主要容器。
- `citations`：內 `citationsContent` 保存引文位置的容器。
- `location`：內 `citations` 存放引文來源的容器。
- `web`：內 `location` 包含 Web 來源詳細資訊的容器。
- `url`：引文來源的完整 Web 地址 (URL)。
- `domain`：來源 URL 的根網域。

接地安全

您的資料永遠不會離開 AWS 基礎設施。模型產生的查詢會保留在 AWS 服務中，絕不會傳送到更廣泛的網際網路。我們廣泛的內部 Web 搜尋索引和知識圖表會優先考慮值得信賴的高品質來源，並在傳入時篩選惡意內容。最後，我們會透過執行時期篩選保護您的應用程式免於間接提示詞注入和錯誤資訊 (請注意，此風險緩解措施僅限於非英文語言)。

錯誤處理

請勿包含名為 nova_grounding 的 toolSpec 項目。包含具有此名稱的工具會導致錯誤。

以下是使用 Web Grounding 時可能發生的潛在錯誤清單：

- malformed_tool_use
- max_tokens
- malformed_model_output

內建工具所需的許可

為了確保您的角色可以存取 Amazon Bedrock 上的 Web Grounding，您有兩個選項：

1. 在您的 IAM 角色上啟用 BedrockFullAccess：如果您的角色具有 BedrockFullAccess，它將可以自動存取 Web Grounding。
2. 新增特定許可（如有需要）：如果您需要更精細的存取控制，請將此政策新增至角色的 IAM 政策，以您的 ID 取代帳戶 AWS 帳戶 ID：

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["bedrock:InvokeTool"],
      "Resource": ["arn:aws:bedrock::{111122223333}:system-tool/
amazon.nova_grounding"]
    }
  ]
}
```

Web Grounding aws:requestedRegion 的條件金鑰設定為「未指定」。如果您現有的政策或服務控制政策 (SCP) 強制執行此條件，您可能會遇到存取問題。更新條件以允許「未指定」requestedRegion，可以解決此問題。

 Note

如果您啟用 Web Grounding 工具，您必須為您以及最終使用者對包含基礎資訊的輸出的任何使用負責。當您的輸出包含來自引文或來源材料連結的基礎資訊時，您會知道這一點。您必須在您提供給最終使用者的輸出中保留並顯示這些引文和連結。

 Note

Web Grounding 會增加額外成本。如需詳細資訊，請前往 [AWS Bedrock 定價](#)。

提示詞工程指南

提示詞工程是設計指示以有效與 Amazon Nova 等大型語言模型 (LLMs) 通訊的做法。提示詞工程可讓您控制模型行為、改善輸出品質，以及建置可靠的 AI 驅動應用程式。

有效的提示詞工程涉及製作可引導模型產生所需輸出的輸入。精心設計的提示提供清楚的指示、相關內容和正確格式的輸入資料。

什麼是提示詞工程

提示詞工程是指最佳化大型語言模型 (LLM) 的文字輸入，以改善輸出並接收您想要的回應的做法。提示有助於 LLM 執行各種任務，包括分類、問題回答、產生程式碼、創意寫作等。您提供給 LLM 的提示詞品質可能會影響模型回應的品質。本節為您提供所有必要的資訊，以便開始使用提示工程。它還涵蓋工具，協助您在 Amazon Bedrock 上使用 LLM 時，為您的使用案例尋找最佳的可能提示格式

提示詞的有效性取決於提供的資訊品質和提示詞本身的技巧。提示詞可能包含指示、問題、上下文相關詳細資訊、輸入和範例，以有效地引導模型並增強結果的品質。本文件概述最佳化 Amazon Nova 2 Sonic 系列模型效能的策略和策略。本文提供的方法可按各種組合使用，以增強其有效性。我們鼓勵您參與實驗，以找出最適合其特定需求的方法。

提示詞工程入門

在開始提示詞工程之前，我們建議您備妥下列元素，以便針對您的使用案例反覆制定最佳提示詞：

定義您的使用案例

跨四個維度定義您的使用案例：

- 任務 – 定義您希望模型完成的內容。這會決定正確的提示技術。
- 角色 – 定義模型應擔任的角色來完成任務。Amazon Nova 模型支援三個角色（系統、使用者或助理）。
- 回應風格 – 根據受眾定義模型應遵循的回應結構或風格，例如 JSON、Markdown 或對話。
- 指示 – 定義模型應遵循的一組指示，以符合成功條件。

建立成功條件

定義成功條件或評估指標。您可以提供條件清單或提供特定評估指標，例如長度、BLEU 分數、ROUGE、格式、事實和忠誠度。

草擬提示

建立包含任務、角色、回應樣式和指示的開始提示。根據結果反覆運算。

提示的有效性取決於您提供的資訊品質。

一般最佳實務

下列最佳實務主要適用於 Amazon Nova 文字模型，但除了模態特定的最佳實務之外，您也可以將其套用至其他模型。

如需如何提示多模態輸入的詳細資訊，請參閱 [提示多模態輸入](#)。如需如何提示語音輸入的資訊，請參閱 [語音對話提示](#)。

了解角色

Amazon Nova 模型可讓您透過使用三種不同的角色來建構提示：系統、使用者和助理。

- 系統（選用）– 雖然不是強制性的，但它會建立助理的整體行為參數。它也可以用來提供使用者希望模型在整個對話過程中遵守的其他指示和指導方針。
- 使用者 – 可以選擇性地將內容、任務、指示和所需結果與使用者查詢一起傳達。
- 助理 – 協助引導模型朝向預期的回應。

主題

- [建立精確提示](#)
- [讓提示詞區段引起關注](#)
- [使用系統角色](#)
- [提供範例 \(少樣本提示\)](#)
- [工具呼叫系統](#)
- [進階提示技巧](#)

建立精確提示

設計特定使用者查詢對於提示詞工程至關重要。精心設計的查詢會引導 Amazon Nova 文字理解模型產生準確且相關的回應。若要設計這類查詢，請務必從查詢中提供上下文相關資訊開始。提供的上下文透過對齊清晰的目標使用者場景，幫助模型更好地定位，從而增強輸出的相關性和一致性。

提供與上下文相關的具體資訊

如果指示清晰、結構化良好、簡短且具體，則模型最有效。此提示詞範本指引同時適用於使用者提示詞和系統提示詞。

提示範本

```
# Use this to clearly define the task and job needed by the model
task_summary = f"""
## Task Summary:
{{Task summary}}
"""

# Use this to provide contextual information related to the task
context_information = f"""
## Context Information:
- {{Context and content information 1}}
- {{Context and content information 2}}
...
"""

# Use this to provide any model instructions that you want model to adhere to
model_instructions = f"""
## Model Instructions:
- {{ Other Model Instructions }}
...
"""

# Use this to provide response style and formatting guidance
response_style = f"""
## Response style and format requirements:
- {{Style and format requirement 1}}
- {{Style and format requirement 2}}
...
"""

# Concatenate to final prompt
final_prompt = f"""{task_summary}
{context_information}
{model_instructions}
{response_style}"""
```

範例：具體且與上下文相關的資訊

以下是使用具體且與上下文相關的資訊重新撰寫的提示詞範例。新增更多上下文資訊有助於模型基於這些資訊朝向所需的行為。

Unclear prompt

User:

Write me a meeting invite to the project team

Assistant:

Certainly! Here's a sample meeting invite for your project team:

****Subject:**** Project Team Meeting - [Project Name]

****Date:**** [Insert Date]

****Time:**** [Insert Time]

****Duration:**** [Insert Duration]

****Location:**** [Insert Location] / [Zoom/Teams Link if Online]

****Agenda:****

1. ****Welcome and Introductions****

- Quick round of introductions for new team members

2. ****Project Overview****

- Objectives and goals
- Current status and milestones achieved

3. ****Task Allocation and Deadlines****

- Detailed breakdown of tasks
- Assignment of responsibilities
- Discussion of deadlines

4. ****Risk Management****

- Identification of potential risks
- Mitigation strategies

5. ****Resource Allocation****

- Required tools and materials
- Budget review

6. ****Q&A Session****

- Open floor for questions and clarifications

7. ****Next Steps and Closing****

- Summary of action items
- Schedule for the next meeting

****Additional Notes:****

- Please come prepared with any relevant documents or updates regarding your assigned tasks.
- If you anticipate any issues or require additional support, please let us know in advance so we can address them during the meeting.

****Contact Information:****

For any questions or concerns before the meeting, please feel free to reach out to [Your Name] at [Your Email] or [Your Phone Number].

We look forward to a productive discussion and to kicking off our project with enthusiasm!

Best regards,

[Your Full Name]

[Your Position]

[Your Contact Information]

Feel free to customize this template to better fit your specific project and team needs!

Clear prompt

User:

Write me a meeting invite to the project team. **The meeting is about project planning, and should be held on October 16th, from 10 AM to 11 AM in Conference**

Room B. Please include an agenda that covers our progress on the project thus far, as well as any upcoming milestones and deadlines.

Assistant:

****Meeting Invitation: Project Planning Session****

****Date:**** October 16th

****Time:**** 10:00 AM - 11:00 AM

****Location:**** Conference Room B

****Agenda:****

1. ****Welcome and Introduction (10:00 AM - 10:05 AM)****

- Brief overview by the meeting facilitator.

2. ****Project Progress Update (10:05 AM - 10:30 AM)****

- Review of completed tasks and achievements to date.

- Detailed discussion on current project status and any challenges encountered.

3. ****Upcoming Milestones and Deadlines (10:30 AM - 10:50 AM)****

- Overview of the next phase of the project.

- Identification of key milestones and deadlines.

4. ****Action Items and Responsibilities (10:50 AM - 11:00 AM)****

- Assignment of tasks for the upcoming period.

- Confirmation of individual responsibilities and follow-up items.

5. ****Q&A Session (11:00 AM)****

- Open floor for any questions or concerns from team members.

Please ensure your attendance to discuss and finalize our project planning for the next phase. Your insights and contributions are highly valued.

Best regards,

[Your Name]

[Your Position]

[Your Contact Information]

- 明確：提示模型時，嘗試以人類能理解的方式（而非機器的方式）提供盡可能明確的指示（例如，JSON 樣式清單）。

- **具體：**為模型提供有關任務內容、預期輸出內容以及任何額外上下文，以協助模型以您想要的方式做出回應的具體指示。

指定輸出格式和樣式

如果您希望 Amazon Nova 2 產生符合您特定需求的回應，您可以指定輸出的格式和樣式。例如，如果您只想以 JSON 格式產生回應，或想要限制回應長度不超過單詞數、項目符號數或段落數上限。這可讓 Amazon Nova 2 更了解您的需求，並產生更相關且更易於剖析的輸出。

使用大寫和強烈措辭，例如「務必」、「切勿」、「必須」來強調您希望模型嚴格遵守的規則。例如，如果您在回應中只想要一個 JSON 物件，您可以這樣指示模型：「您必須僅以 JSON 格式回答。在回答問題時切勿使用任何其他格式。」

以下是使用具體且上下文相關之資訊重新撰寫的一些範例提示詞。新增更多上下文資訊有助於模型基於這些資訊朝向所需的行為。

範例 1：提供額外詳細資訊

新增有關回應格式的額外細節，有助於模型達成所需的結果。其他一些常用的回應格式包括 CSV、Markdown、YAML 和 XML

Prompt without style constraints

User:

```
Outline the different types of rooms available in a typical 4-star hotel, including their features.
```

Assistant:

```
In a typical 4-star hotel, you'll find a range of room types, each designed to cater to different guest needs and preferences. Here's an outline of the different types of rooms available along with their features:
```

```
### 1. **Standard Room**
```

- **Size:** Usually around 250-350 square feet.
- **Features:**
 - Queen or double bed.
 - Complimentary Wi-Fi.
 - Flat-screen TV with cable or satellite channels.
 - Mini-fridge.
 - Coffee maker with complimentary coffee and tea.

- Desk and chair.
- En-suite bathroom with shower, toilet, and toiletries.
- Air conditioning/heating.

2. **Deluxe Room**

- **Size:** Typically larger than standard rooms, around 350-450 square feet.
- **Features:**
 - King bed or two queen beds.
 - Larger bathroom with a bathtub.
 - Additional seating area.
 - Enhanced amenities like bathrobes and slippers.
 - Upgraded toiletries and sometimes a welcome gift.
 - Access to a lounge or executive floor perks.

3. **Suite**

- **Size:** Varies significantly, often between 500-1,200 square feet.
- **Features:**
 - Separate living area with a sofa bed or additional seating.
 - Kitchenette or full kitchen in some cases.
 - Larger, more luxurious bathroom with a bathtub and separate shower.
 - Multiple TVs.
 - Access to executive lounge.
 - High-end toiletries and often complimentary breakfast.
 - Often includes concierge services and in-room dining options.

4. **Family Room**

- **Size:** Larger, usually around 400-500 square feet.
- **Features:**
 - One king bed and one sofa bed or two queen beds.
 - Designed for families with children.
 - Extra space for luggage.
 - Child-friendly amenities like a small kitchenette or high chair.
 - Family bathroom with amenities suitable for kids.

5. **Executive Room**

- **Size:** Typically around 350-450 square feet.
- **Features:**
 - Business-oriented with a larger workspace.
 - Executive desk and chair.
 - Access to executive lounge with complimentary snacks and refreshments.
 - King bed or two queen beds.
 - Enhanced amenities like high-speed internet and sometimes a Nespresso machine.

6. **Junior Suite**

- **Size:** Usually around 450-600 square feet.
- **Features:**
 - Separate bedroom and living area.
 - King bed or two queen beds.
 - Kitchenette or mini-fridge.
 - Luxurious bathroom with a bathtub.
 - Lounge access.
 - High-end toiletries.

7. **Premium Room**

- **Size:** Larger than standard rooms, around 400-500 square feet.
- **Features:**
 - King bed or two queen beds.
 - Larger bathroom with a bathtub.
 - Enhanced amenities like a welcome drink and premium toiletries.
 - Access to premium services and sometimes a lounge.

8. **Connecting Rooms**

- **Size:** Two rooms that connect via an internal door.
- **Features:**
 - Suitable for families or groups.
 - Often includes one standard room and one suite.
 - Shared bathroom or en-suite bathrooms in each room.
 - Extra space for luggage and guests.

9. **Accessible Rooms**

- **Size:** Varies, typically similar to standard rooms.
- **Features:**
 - Designed for guests with disabilities.
 - Wider doorways and bathroom space.
 - Adapted fixtures and fittings.
 - Additional support services.

10. **Business Room**

- **Size:** Usually around 300-350 square feet.
- **Features:**
 - Focused on business travelers.
 - High-speed internet and large workspace.
 - Compact yet fully equipped with essentials.
 - Often includes a king bed.

Each room type in a 4-star hotel is designed to provide comfort, convenience, and luxury tailored to different guest needs, ensuring a pleasant stay for all types of travelers.

Prompt with style constraints

User:

Outline the different types of rooms available in a typical 4-star hotel, including their features. **All responses MUST be in YAML format.**

Assistant:

```
```yaml
hotel_rooms:
- room_type: Standard Room
 features:
 - description: A comfortable room with essential amenities.
 amenities:
 - bed_type: Double bed or two single beds
 - bathroom: Private bathroom with shower
 - wifi: Complimentary high-speed internet
 - TV: Flat-screen TV with cable channels
 - mini_fridge: Available
 - tea_and_coffee_making_facilities: Yes

- room_type: Deluxe Room
 features:
 - description: An upgraded room with more space and enhanced amenities.
 amenities:
 - bed_type: King-size bed or two single beds
 - bathroom: Private bathroom with both shower and bathtub
 - wifi: Complimentary high-speed internet
 - TV: Larger flat-screen TV with cable channels
 - mini_fridge: Available
 - tea_and_coffee_making_facilities: Yes
 - free_breakfast: Included

- room_type: Suite
 features:
 - description: A luxurious room with separate living area and premium amenities.
 amenities:
 - bed_type: King-size bed or two single beds
```

```

- bathroom: Private bathroom with both shower and bathtub
- separate_living_area: Yes
- wifi: Complimentary high-speed internet
- TV: Large flat-screen TV with cable channels
- mini_fridge: Available
- tea_and_coffee_making_facilities: Yes
- free_breakfast: Included
- concierge_service: Available

- room_type: Family Room
features:
- description: A spacious room designed for families with additional bedding
options.
amenities:
- bed_type: Queen bed and rollaway bed or two single beds and rollaway bed
- bathroom: Private bathroom with shower
- wifi: Complimentary high-speed internet
- TV: Flat-screen TV with cable channels
- mini_fridge: Available
- tea_and_coffee_making_facilities: Yes

- room_type: Business Room
features:
- description: A room tailored for business travelers with enhanced workspace.
amenities:
- bed_type: Double bed or two single beds
- bathroom: Private bathroom with shower
- wifi: Complimentary high-speed internet
- TV: Flat-screen TV with cable channels
- workspace: Desk and chair
- mini_fridge: Available
- tea_and_coffee_making_facilities: Yes
- business_center_access: Available
...
```

## 範例 2：提供樣式限制條件

在這種情況下，具有樣式限制條件的提示詞會提供有關要遵循的限制條件和所需回應格式的完整資訊。如回應中所觀察到的，最佳化提示詞更緊密符合所請求的簡潔性和項目符號清單結構。

### Prompt without style constraints

User:

Explain the process of checking in a guest at a luxury resort.

Assistant:

Checking in a guest at a luxury resort involves a meticulous and personalized process designed to ensure a seamless and memorable start to their stay. Here's a detailed overview of the steps involved:

### ### Pre-Arrival Preparation

#### 1. \*\*Pre-Arrival Communication:\*\*

- **Confirmation:** Verify guest reservations and confirm arrival details.
- **Special Requests:** Collect any special requests (dietary restrictions, room preferences, etc.) and ensure they are noted in the guest profile.
- **Documentation:** Ensure all necessary documentation is ready, such as welcome letters, room keys, and any special amenities.

#### 2. \*\*Room Preparation:\*\*

- **Cleaning:** Conduct a thorough cleaning and preparation of the guest room.
- **Amenities:** Stock the room with high-end toiletries, fresh linens, and any requested amenities.
- **Personal Touches:** Add personalized touches such as a welcome note, fresh flowers, or locally sourced treats.

### ### Arrival and Initial Interaction

#### 3. \*\*Guest Arrival:\*\*

- **Greeting:** Have staff ready to greet guests as they arrive. This may include a warm welcome at the entrance or a meet-and-greet in the lobby.
- **Transport:** Provide luxury transport options like a chauffeured car service to escort guests to the check-in desk.

#### 4. \*\*Check-In Desk:\*\*

- **Professionalism:** Ensure check-in staff are well-trained, professional, and exude hospitality.
- **Technology:** Utilize sophisticated check-in technology (like tablet kiosks) to streamline the process.
- **Personalized Welcome:** Greet guests with a warm welcome, acknowledge their special requests, and present them with a personalized welcome kit.

### ### Documentation and Formalities

#### 5. \*\*Document Verification:\*\*

- **\*\*Identification:\*\*** Verify identification and payment methods.
- **\*\*Membership Cards:\*\*** If applicable, verify any loyalty or membership cards for additional benefits.

#### 6. **\*\*Registration:\*\***

- **\*\*Form Filling:\*\*** Assist guests with any required paperwork.
- **\*\*Special Benefits:\*\*** Clearly explain any special resort benefits, upgrades, or exclusive access to amenities that the guest is entitled to.

### ### Room Assignment

#### 7. **\*\*Room Presentation:\*\***

- **\*\*Walk-Through:\*\*** Escort guests to their room and conduct a walk-through to ensure everything is in order and to address any immediate concerns.
- **\*\*Upgrades:\*\*** Present any room upgrades and explain the features and services available in the upgraded room.

#### 8. **\*\*Key Delivery:\*\***

- **\*\*Room Key:\*\*** Provide the room key, often a high-end key card or a luxury key fob.
- **\*\*Access Information:\*\*** Explain how to use the key card, including access to other amenities like the spa, pool, and gym.

### ### Additional Services

#### 9. **\*\*Welcome Amenities:\*\***

- **\*\*Amenity Kits:\*\*** Deliver welcome amenity kits (like a bottle of champagne, local delicacies, etc.).
- **\*\*Itinerary:\*\*** Provide a detailed resort map and itinerary, highlighting all available services and activities.

#### 10. **\*\*Personal Concierge:\*\***

- **\*\*Concierge Services:\*\*** Introduce the guest to the concierge service and explain how to request any special services, such as dining reservations, transportation, or local excursions.
- **\*\*Contact Information:\*\*** Provide contact information for key staff members.

### ### Final Touches

#### 11. **\*\*Feedback:\*\***

- **\*\*Satisfaction Survey:\*\*** At the end of the check-in process, ask for feedback to ensure the guest's expectations were met and to gather any immediate suggestions.

#### 12. **\*\*Follow-Up:\*\***



- **\*\*Immediate Issues:\*\*** Ensure any immediate issues or requests are addressed promptly.
- **\*\*Comfort Check:\*\*** Make follow-up calls or send messages to ensure the guest is settling in comfortably.

By following these steps, a luxury resort aims to provide an exceptional check-in experience that sets the tone for an unforgettable stay.

## Prompt with style constraints

User:

Explain the process of checking in a guest at a luxury resort. **Provide all answers in a numbered list format, with each point containing no more than 15 words, and no more than 6 bullet points.**

Assistant:

1. Greet guest warmly with a smile and offer assistance.
2. Verify reservation details and check guest's identification.
3. Confirm room assignment and amenities provided.
4. Explain resort services and amenities available.
5. Present room key and offer directions to guest's suite.
6. Wish guest a pleasant stay and be available for queries.

## 範例 3：限制回應格式

在這裡，具有回應格式的提示強制模型僅以類別編號回應，並將其限制在 1-10 之間，以避免過度冗長。

## Prompt without response format

User:

In this task, you are given a text which is the body of a document. Your job is to classify the topic of the document into these categories: 1)Company, 2)Educational Institution, 3)Artist, 4)Athlete, 5)Office Holder, 6)Mean of transportation, 7)Building, 8)Natural place, 9)Village, 10)Animal.  
Text: Johangsan is a mountain of South Korea. It has an elevation of 954 metres.

Assistant:

To classify the topic of the given text, let's analyze the content step-by-step:

1. The text mentions "Johangsan," which appears to be a specific name.
2. The text states that "Johangsan is a mountain," which provides a clear indication of its nature.
3. It is further specified that it is located in "South Korea" and has an elevation of "954 metres."

Based on the information provided, the text describes a geographical feature, specifically a mountain. From the given categories, the most appropriate classification for this document is:

8) Natural place

The text does not mention any company, educational institution, artist, athlete, office holder, means of transportation, building, village, or animal. Therefore, the classification as a "Natural place" is the most fitting choice.

## Prompt with response format

User:

In this task, you are given a text which is the body of a document. Your job is to classify the topic of the document into these categories: 1)Company, 2)Educational Institution, 3)Artist, 4)Athlete, 5)Office Holder, 6)Mean of transportation, 7)Building, 8)Natural place, 9)Village, 10)Animal. **Your output should be the category number. DO NOT generate anything apart from numbers 1-10.**  
Text: Johangsan is a mountain of South Korea. It has an elevation of 954 metres.

Assistant:

8

## 提供明確的提示詞區段

除了初始上下文之外，結構化查詢格式也在最佳化與語言模型的互動方面扮演重要角色。使用分隔符號分隔輸入的不同部分可確保清晰度，並增強 Amazon Nova 2 模型了解查詢結構的能力。另一種有效的方法是使用編號清單或項目符號，這有助於將任務拆解為易於管理的步驟，並提供需要採取的動作的明確概述。這種結構化方法有助於減少模稜兩可的情況，並改善所產生回應的準確性。

## 範例：建立分段提示詞

請注意，與未分段提示欄中的指示相比，已分段提示欄中的提示更加清晰，並為任務、上下文、需求等不同實體劃定了區段。

### Unsectioned Prompt

User:

Write an article about climate change for high school students explaining basic concepts, current impacts, and individual actions to mitigate it. Make it informative and engaging with simple language, including statistics and real-world examples.

Assistant:

### Understanding Climate Change: What It Is, What It's Doing, and What You Can Do About It

#### Introduction

Climate change might sound like a big, complicated topic, but it's really about the way our planet's weather is changing over time. This change is mostly happening because humans are doing things that are making our Earth warmer. Let's break it down so you can understand what's going on and what you can do to help.

#### What Is Climate Change?

Climate change refers to long-term shifts in temperatures and weather patterns. These shifts may be natural, but since the 1800s, human activities have been the main driver of rapid climate change.

**\*\*Key Concept: Greenhouse Gases\*\***

To understand this, we need to talk about greenhouse gases. These gases, like carbon dioxide (CO<sub>2</sub>), methane (CH<sub>4</sub>), and nitrous oxide (N<sub>2</sub>O), trap heat in our atmosphere. Without these gases, Earth would be too cold for most life forms. But when we burn fossil fuels (like coal, oil, and natural gas) and cut down forests, we release too much CO<sub>2</sub> and other greenhouse gases, trapping more heat and making the planet warmer.

#### Current Impacts of Climate Change

Let's look at some real-world examples to see how climate change is affecting our world today.

**\*\*1. Rising Temperatures\*\***

Since the late 19th century, Earth's average surface temperature has risen about 2 degrees Fahrenheit (1 degree Celsius). Scientists predict that if we don't change our ways, it could rise by 3.6 to 4.3 degrees Fahrenheit (2 to 2.4 degrees Celsius) by the end of this century.

## **\*\*2. More Extreme Weather\*\***

Climate change is making extreme weather events more frequent and severe. For example:

- **\*\*Hurricanes:\*\*** In the U.S., hurricane seasons are getting longer and more destructive. In 2020, there were 30 named storms, the most ever recorded.
- **\*\*Droughts:\*\*** Parts of California have experienced some of the worst droughts in history, affecting agriculture and water supplies.

## **\*\*3. Melting Ice Caps and Rising Sea Levels\*\***

Glaciers and ice sheets are melting at alarming rates. This not only affects wildlife but also leads to rising sea levels. Since 1901, global sea levels have risen by about 8 inches (20 centimeters). By 2100, they could rise by another 1 to 4 feet (0.3 to 1.2 meters).

## **\*\*4. Ocean Acidification\*\***

As CO2 levels rise, more of it is absorbed by the oceans, making them more acidic. This is particularly harmful to marine life, including coral reefs. Coral reefs are crucial for marine biodiversity but are now dying at an alarming rate.

## **#### Individual Actions to Mitigate Climate Change**

You might think that since climate change is a huge problem, one person can't make a difference. But even small actions can add up to make a big impact.

### **\*\*1. Reduce, Reuse, Recycle\*\***

Try to minimize waste by:

- **\*\*Reducing:\*\*** Buy fewer items and choose products with less packaging.
- **\*\*Reusing:\*\*** Use reusable bags, bottles, and containers.
- **\*\*Recycling:\*\*** Make sure to recycle paper, plastic, and glass properly.

### **\*\*2. Conserve Energy\*\***

Small changes can lead to big savings:

- Turn off lights when you leave a room.
- Unplug devices that you're not using.
- Use energy-efficient bulbs and appliances.

### **\*\*3. Use Less Meat\*\***

Livestock farming is a major contributor to greenhouse gas emissions. By eating less meat, especially beef, you can reduce your carbon footprint.

**\*\*4. Support Renewable Energy\*\***

If possible, support or switch to renewable energy sources like solar or wind power. Even advocating for these options in your community can make a difference.

**\*\*5. Educate Yourself and Others\*\***

Knowledge is power. The more you know about climate change, the more you can help others understand it. Share facts and encourage your friends and family to take action too.

**#### Conclusion**

Climate change is a pressing issue that affects everyone on the planet. By understanding its basics, recognizing its current impacts, and taking individual actions, you can play a crucial role in mitigating its effects. Remember, every small step counts, and together, we can make a big difference for our planet.

Let's start today and build a sustainable future!

## Sectioned Prompt

User:

**\*\*Task:\*\***

Write an article

**\*\*Context Section:\*\***

Topic: Climate Change

Target Audience: High School Students

Purpose: Educational Article

**\*\*Content Requirements:\*\***

- Explanation of basic concepts related to climate change.
- Examples of current impacts on ecosystems and human populations.
- Suggestions for individual actions to mitigate climate change.

**\*\*Tone and Style:\*\***

- Informative and engaging
- Use simple language
- Include statistics and real-world examples

Assistant:

# Understanding Climate Change: A Guide for High School Students

## ## Introduction

Climate change is one of the most pressing issues of our time. It affects our planet, our ecosystems, and our daily lives. Understanding what climate change is and how it impacts us can help us take action to make a difference. This article will explain the basic concepts of climate change, provide examples of its current impacts, and suggest actions you can take to help mitigate it.

## ## What is Climate Change?

**Climate change** refers to long-term changes in global temperatures and weather patterns. These changes are primarily caused by human activities, especially the burning of fossil fuels like coal, oil, and natural gas. When we burn these fuels, we release large amounts of greenhouse gases, such as carbon dioxide (CO<sub>2</sub>) and methane (CH<sub>4</sub>), into the atmosphere. These gases trap heat from the sun, causing the Earth's temperature to rise. This is often referred to as the **greenhouse effect**.

### ### Basic Concepts:

- Greenhouse Gases**: These gases include carbon dioxide, methane, nitrous oxide, and fluorinated gases. They trap heat in the Earth's atmosphere.
- Global Warming**: This is the increase in Earth's average surface temperature due to rising levels of greenhouse gases.
- Climate Change**: Refers to the broader set of changes in the Earth's climate system, including global warming.

## ## Current Impacts of Climate Change

### ### On Ecosystems:

- Melting Ice Caps and Glaciers**: The polar ice caps and glaciers are melting at an alarming rate. For example, the Arctic sea ice is shrinking, which affects polar bears and other Arctic animals.
- Rising Sea Levels**: As ice melts, sea levels rise. This can lead to the loss of coastal habitats and increased flooding in coastal areas.
- Changing Weather Patterns**: Climate change can lead to more frequent and severe weather events such as hurricanes, droughts, and heatwaves. For example, the 2020 Australian bushfires were more intense and lasted longer than usual due to climate change.

### ### On Human Populations:

1. **Health Issues**: Rising temperatures can lead to more heat-related illnesses and the spread of diseases like malaria and dengue fever.
2. **Food Security**: Changes in climate can affect crop yields, leading to food shortages. For example, droughts in Sub-Saharan Africa have caused significant food insecurity.
3. **Displacement**: Rising sea levels and extreme weather events can force people to leave their homes, becoming climate refugees. For instance, communities in Bangladesh are being displaced by rising sea levels.

### ## Actions You Can Take

Even though climate change is a massive issue, there are many actions you can take to help mitigate its effects. Here are some simple steps you can start with today:

1. **Reduce, Reuse, Recycle**: By reducing waste, reusing items, and recycling, you can decrease the amount of trash that ends up in landfills and reduce greenhouse gas emissions.
2. **Save Energy**: Turn off lights when you don't need them, use energy-efficient appliances, and unplug devices when they're not in use. These small actions can make a big difference.
3. **Use Less Meat**: Livestock farming produces a significant amount of methane. By eating less meat, particularly beef, you can reduce your carbon footprint.
4. **Support Renewable Energy**: Advocate for or invest in renewable energy sources like solar and wind power. These sources produce energy without greenhouse gas emissions.
5. **Educate and Advocate**: Inform others about the impacts of climate change and advocate for policies that support environmental sustainability. You can join or support environmental clubs at school or participate in local clean-up efforts.

### ## Conclusion

Climate change is a complex issue with far-reaching impacts. However, understanding it and taking action can make a significant difference. By learning about the basic concepts, recognizing the current impacts, and implementing simple changes in our daily lives, we can all contribute to mitigating climate change. Remember, every

small action counts, and together, we can create a more sustainable future for our planet.

## 讓提示詞區段引起關注

Amazon Nova 2 模型可以透過格式化區段中的指示，然後參考這些特定區段，來密切注意提示中的特定部分。模型可以關注到提示詞中是否包含使用 Markdown、XML 或其他結構的明確區段分隔。例如，您可以定義區段名稱，使用 `##Section Name##`，然後在您的提示詞中透過 `##Section Name##` 參考該區段。

您也可以利用此策略來限制模型不要在產生的回應中顯示部分輸入提示詞。例如，當在輸入提示詞中提供少樣本範例或指示時，使用分隔符號 (例如 `##Instructions##` 或 `##Examples##`) 搭配新行分隔符號，並提供強式指示，例如 `DO NOT mention anything inside the ##Instructions## or ##Examples## in the response`，讓模型不要在其輸出中照搬這些區段中的輸入提示詞內容。

範例：區段分隔

使用者：

```
You're an expert Prompts creator. Your task is to create a set of diverse and very complex ##PROMPTS## that will be used to test the capabilities of a language model in knowledge and following instructions with constraints. Please create 10 ##PROMPTS##. You must strictly follow ##GUIDELINES##:
```

```
##GUIDELINES##
```

- Generate ##PROMPTS## similar to the structure and style of the given ##EXAMPLE PROMPTS##. Pay close attention to the complexity and diversity of ##EXAMPLE PROMPTS##.
- Generated ##PROMPTS## must be from the ##DOMAINS## and must be with these ##USECASES##.
- Each of the ##PROMPTS## needs to be unique and very complex.
- Each of the ##PROMPTS## must have more than 4 sentences and 1 constraint.
- Each of the ##PROMPTS## should have at least 70 words.
- Each of the ##PROMPTS## should have an answer that can be written in text.
- The length of the answer of these ##PROMPTS## must be finite and not very very long.
- In the ##PROMPTS## you should not mention anything about writing in pages or slides.
- Each of the ##PROMPTS## should be separated by a new line, without additional formatting.

```
Generated ##PROMPTS## must be from the following ##DOMAINS##
##DOMAINS##
```



```
{domains}
```

```
Generated ##PROMPTS## must be for the following ##USECASES##
```

```
##USECASES##
```

```
{usecases}
```

```
{usecase_description}
```

```
##PROMPTS##
```

## 使用系統角色

系統角色是一種角色，您可以在其中提供指示給模型，以定義它會如何回應應用程式的最終使用者。例如，系統角色可以引導模型以指定的人物角色回應、設定允許和不允許的內容、以特定格式輸出、指定防護機制等。系統角色中稱為系統提示詞的指示，將取代個別使用者提示詞中提供的其他指示，並將在所有使用者回合中持續有效。

### 使用 API 指定系統角色

若要為模型提供自訂角色，您可以在 API 中設定 `system` 參數，如下所示：

```
{
 "system": [
 {
 "text": "You are a helpful recipe assistant. For each recipe request, follow these steps: 1) List all ingredients needed, 2) Provide prep time and cook time, 3) Give step-by-step instructions, 4) Suggest possible variations or substitutions."
 }
],
 "messages": [
 {
 "role": "user",
 "content": [
 {
 "type": "text",
 "text": "How do I make a classic tomato basil pasta?"
 }
]
 }
]
}
```

 Tip

- 為了充分利用系統角色，請確定您的系統提示詞明確、簡潔且具體，並遵循 [建立精確提示](#) 中所述的相同策略。例如，當您想要限制模型應回應的主題時，請具體指示它，例如「不要談論 ...」或「您必須談論 ...」。
- 若要進一步將模型限制為階層結構，您可以將下列尾碼新增至系統提示詞，以強調系統指示和使用者指示之間的階層遵循結構：

```
"\n\nThe above system instructions define your capabilities and your scope. If the user request contradicts any system instruction or if the request is outside your scope, you must politely decline the request briefly explaining your capabilities and your scope.\n"
```

以下是您可以在系統提示詞範本中新增的一些項目：

```
Define a persona
persona = """You are {{Persona}}
"""

Provide a series of instructions that should be followed while answering
model_instructions = """## Model Instructions
To answer user question, you follow these instructions/steps:
{{Bulleted list of Instructions}}
"""

Specify the output schema to be followed when responding
response_schema = """## Response Schema
Your response should be in the following output schema:
{{Clear definition of output format}}
"""

Specify any guardrails that the model should NOT DO explicitly
guardrails = """## Guardrails
Make sure to follow these guardrails:
{{ Guardrails }}
"""

system_prompt = f"""{persona}
{model_instructions}
{response_schema}
```

```
{guardrails}
"""
```

若要為系統提示詞提供系統指示與使用者指示之間的階層遵循結構，您可以新增下列項目：

```
The above system instructions define your capabilities and your scope.
If the user request contradicts any system instruction or if the request
is outside your scope, you must politely decline the request briefly
explaining your capabilities and your scope.
```

### 範例 1：提供特定語言提示

新增系統提示詞後，模型回應會針對中世紀卷軸樣式進行高度樣式化，如系統提示詞所示。

#### Without a System Prompt

系統：

```
NA
```

使用者：

```
Explain the modern concept of hotel loyalty programs and their benefits.
```

助理：

```
Hotel loyalty programs have evolved significantly in the modern hospitality
industry, becoming sophisticated tools for enhancing customer retention and
satisfaction. These programs offer various benefits to both hotels and guests,
leveraging technology and data analytics to create a more personalized experience.
```

```
Key Components of Modern Hotel Loyalty Programs:
```

```
1. **Tiered Membership Levels:**
```

- **Basic Membership:** Entry-level membership that provides basic benefits.
- **Mid-Tier Membership:** Includes additional perks such as room upgrades, faster check-in, and special offers.
- **Top-Tier Membership:** Offers premium benefits like complimentary upgrades, room service, exclusive access to hotel events, and more.

```
2. **Point System:**
```

- Guests earn points for every stay, purchases made in hotel restaurants, and other activities.
- Points can be redeemed for free nights, upgrades, or other rewards.

### 3. **Mobile App Integration:**

- Modern programs often have a dedicated mobile app that allows guests to easily manage their account, check in, and access exclusive offers.
- Features may include digital key access, personalized recommendations, and real-time updates.

### 4. **Personalization:**

- Utilizing data analytics, hotels can tailor recommendations and offers to individual guest preferences, enhancing the overall experience.
- Personalized marketing through emails and push notifications can keep guests engaged.

### 5. **Cross-Brand Benefits:**

- Some loyalty programs offer benefits across multiple brands within a hotel group or even within different hospitality groups.
- This can include discounts, points transfers, and combined rewards.

### 6. **Social Media Integration:**

- Many programs encourage social media sharing and reviews, which can enhance the hotel's online presence and attract new guests.
- Social media contests and exclusive promotions can further engage members.

### 7. **Sustainability Initiatives:**

- Modern loyalty programs increasingly incorporate sustainability, offering points or rewards for eco-friendly choices like reducing plastic use or participating in recycling programs.

## ### Benefits to Guests:

### 1. **Financial Savings:**

- Accumulation of points for free or discounted stays.
- Exclusive discounts and offers not available to non-members.

### 2. **Enhanced Services:**

- Faster check-in, preferred room assignments, and personalized services.
- Access to exclusive events and experiences.

### 3. **Convenience:**

- Easy booking and management through mobile apps.
- Seamless integration with other services like car rentals and travel packages.

#### 4. **Recognition and Prestige:**

- Status symbols like elite membership levels.
- Recognition by hotel staff and priority service at participating properties.

#### ### Benefits to Hotels:

##### 1. **Increased Guest Loyalty:**

- Encourages repeat visits and long-term relationships with guests.
- Helps retain customers even when faced with competition.

##### 2. **Data Collection and Analysis:**

- Provides valuable insights into guest preferences and behaviors.
- Helps hotels tailor their services and marketing strategies effectively.

##### 3. **Revenue Growth:**

- Drives up room occupancy and average daily rate (ADR) through repeat business.
- Cross-selling opportunities within the hotel (e.g., dining, spa services).

##### 4. **Market Differentiation:**

- Helps hotels stand out in a competitive market.
- Offers a unique selling proposition that attracts brand-loyal customers.

#### ### Conclusion:

Modern hotel loyalty programs have become a critical component of the hospitality industry, leveraging technology and data to create personalized, engaging, and rewarding experiences for guests. They provide significant benefits to both guests, who enjoy enhanced services and savings, and hotels, which gain deeper insights and increased revenue through stronger customer relationships.

## With a System Prompt

系統：

Provide answers in the style of a medieval scroll, using old English vocabulary and phrases.

使用者：

Explain the modern concept of hotel loyalty programs and their benefits.

助理：

Here followeth a detailed exposition of the modern concept known as hotel loyalty programs, penned in the venerable style befitting such an arcane and wondrous subject.

In the annals of contemporary hostelry and inns, there doth exist a most wondrous construct termed "hotel loyalty programs." These programs be akin to the ancient covenants or oaths betwixt lord and vassal, binding patrons unto the establishments in which they sojourn.

Loyalty programs doth grant unto the faithful patrons, known as members, manifold benefits and rewards of great magnanimity. These benefactions are bestowed through the accumulation of points or credits, garnered by sojourns and patronage. Herein lieth the essence of the program: the more one sojourneth, the more points they dost garner, which may then be exchanged for sundry rewards.

The benefits of these programs are manifold and most enticing:

1. **\*\*Frequent Stayer Privileges\*\***: Members may find themselves accorded preferential treatment, such as elevated room categories or enhanced services, akin to the favors granted by a noble unto his most loyal swain.
2. **\*\*Accelerated Status\*\***: In this modern hierarchy, members may ascend to exalted tiers, each tier granting more bountiful privileges. Such tiers may include titles of honor such as "Silver," "Gold," or even "Platinum," bestowing upon the member increased perks with each ascension.
3. **\*\*Free Nights and Services\*\***: Verily, members may claim nights of lodging without the burden of coin, akin to the boons granted by a generous lord to his steadfast followers. Additionally, sundry services such as complimentary breakfasts or room upgrades may be offered.
4. **\*\*Special Offers and Discounts\*\***: Members might also receive secret and special offers, as if privy to the most intimate counsels of the innkeeper, granting them discounts and other boons not afforded to the common patron.
5. **\*\*Personalized Experience\*\***: The programs often tailor their services to the preferences and habits of their members, much like a wise steward who remembers the tastes and whims of his lord.

Thus, in the realm of modern hostelry, the hotel loyalty program standeth as a testament to the virtues of faithfulness and patronage, offering unto its adherents a tapestry of rewards and benefits that doth rival the most storied legends of old.

## 範例 2：提供護欄

透過新增系統提示詞，使用者可以為其應用程式新增防護機制，保護應用程式免受任何意外使用。以下是護欄如何將模型專注於所需任務的範例。

### Without a System Prompt

系統：

NA

使用者：

Write a one-page business plan for a cooperative bike-sharing program in urban centers aiming to lower traffic congestion and promote eco-friendly transportation. (Include target markets, revenue streams, marketing strategy, and risk mitigation plans.)

助理：

**\*\*Urban EcoCycle: A Cooperative Bike-Sharing Program\*\***

**\*\*Executive Summary:\*\***

Urban EcoCycle is a cooperative bike-sharing program designed to alleviate traffic congestion and promote eco-friendly transportation in urban centers. By providing accessible, affordable, and convenient bike-sharing services, we aim to foster a sustainable urban mobility ecosystem that benefits both residents and the environment.

**\*\*Target Markets:\*\***

Our primary target markets include urban professionals, students, and environmentally conscious citizens. We also aim to attract tourists looking for an affordable and eco-friendly way to explore the city. Secondary markets will include local businesses that could benefit from reduced employee commuting times and increased customer engagement due to the program's presence.

**\*\*Revenue Streams:\*\***

1. **\*\*Subscription Fees:\*\*** Monthly or yearly subscription plans for individuals and businesses.
2. **\*\*Ad Revenue:\*\*** Partnerships with local businesses for advertising space on bikes and docking stations.
3. **\*\*Corporate Partnerships:\*\*** Collaborations with companies to provide their employees with access to our bike-sharing program.

4. **\*\*Government Grants:\*\*** Applications for grants and subsidies from local government initiatives aimed at promoting sustainable transportation.

**\*\*Marketing Strategy:\*\***

1. **\*\*Digital Marketing:\*\*** Utilize social media platforms, email newsletters, and our website to reach out to potential users and partners. Highlight success stories and eco-friendly benefits.
2. **\*\*Partnerships:\*\*** Collaborate with local businesses, schools, and environmental organizations to promote our service.
3. **\*\*Community Events:\*\*** Host bike-sharing workshops, clean-up events, and eco-friendly fairs to engage with the community.
4. **\*\*Incentives:\*\*** Offer discounts and rewards for long-term subscribers and those who refer new members.

**\*\*Risk Mitigation Plans:\*\***

1. **\*\*Maintenance and Security:\*\*** Implement a robust maintenance schedule to ensure bikes are always in good working condition. Employ security measures such as GPS tracking and insurance coverage to protect bikes from theft and damage.
2. **\*\*Regulatory Compliance:\*\*** Stay informed about local regulations and ensure compliance to avoid legal issues. Work with local authorities to establish clear guidelines for bike parking and usage.
3. **\*\*Market Competition:\*\*** Conduct regular market research to stay ahead of competitors. Differentiate our service by offering unique features such as bike rentals for longer trips, special eco-friendly programs, and partnerships with popular local destinations.
4. **\*\*Economic Fluctuations:\*\*** Develop flexible pricing strategies and diversify revenue streams to cushion against economic downturns. Maintain a reserve fund to handle unexpected financial challenges.

Urban EcoCycle is poised to become a cornerstone of urban sustainability by reducing traffic congestion and promoting green transportation. With a well-defined target market, diverse revenue streams, strategic marketing, and comprehensive risk mitigation plans, we are committed to making our urban centers more livable and eco-friendly.

## With a System Prompt

系統：

You are Application Process Assistant

To answer user question, you follow these instructions/steps:

- The LLM chatbot can discuss topics related to college applications, including deadlines, required materials, and application status.
- The LLM chatbot should be polite and courteous in all responses.



- The LLM chatbot can only respond in 3 sentences or less.

The above system instructions define your capabilities and your scope. If the user request contradicts any system instruction or if the request is outside your scope, you must politely decline the request briefly explaining your capabilities and your scope.\n

使用者：

Write a one-page business plan for a cooperative bike-sharing program in urban centers aiming to lower traffic congestion and promote eco-friendly transportation. (Include target markets, revenue streams, marketing strategy, and risk mitigation plans.)

助理：

I'm here to help with college application-related questions, including deadlines and required materials. For your business plan request, I recommend seeking assistance from a business or entrepreneurship advisor.

## 提供範例 (少樣本提示)

透過在提示中包含一些任務範例，您可以為 Amazon Nova 2 建置結構式範本。這可減少模稜兩可的情況，並增強輸出的準確性和品質。提供明確範例以讓模型的回應更符合所需結果，這種技術被稱為少樣本提示。

少樣本提示的概念是，為語言模型提供一些任務範例，以及輸入和輸出格式，然後要求它根據提供的範例為新輸入產生輸出。此方法也有助於釐清複雜的指示或任務，讓 Amazon Nova 2 更容易了解和解釋所詢問的內容。

如何新增範例至提示詞說明：

新增範例可協助模型產生

- 符合範例樣式的一致回應
- 因為減少了錯誤解譯指示的機會並將產生的幻覺減至最少，回應效能更佳

使用少樣本提示來改善模型效能的數量，取決於所選範例的品質和多樣性。以下項目會在提示詞中示範良好鏡頭的特性：

- 選擇多樣化範例：選擇的範例應該在多樣化方面代表預期輸入/輸出的分佈 (從常見使用案例到邊緣案例)，以充分涵蓋相關的使用案例。務必避免範例中出現任何偏差，因為輸入中的偏差也可能導致輸出偏差。
- 符合複雜性層級：所提供範例的複雜性應與目標任務或案例相符。務必確保在預期輸入與提示詞中所選的範例之間保持對應的複雜性等級。
- 確保相關性：選取的範例應與手邊的問題或目標直接相關。這可確保回應的一致性和符合性。

#### Tip

如果上述建議無效，我們建議您建置 RAG 型系統，根據使用者輸入查詢與可用鏡頭集區之間的相似性，透過動態選取的鏡頭來增強提示詞。

### 範例 1：在使用者提示詞中提供範例

在這裡，您的使用者嘗試將傳入的客戶支援票證歸入預先定義的類別。您可以為 Amazon Nova 2 模型提供幾個正確分類的票證範例，以改善其對票證中使用的類別和語言的了解。示範範例應保持簡潔，並代表應用程式中所有目標任務。如果有對任務有意義的特定欄位名稱，例如此範例中的 Ticket 或 Category，您的使用者可以直接使用這些欄位，後面緊接冒號 ":"。

#### 0-shot prompt

使用者：

Please classify the following customer support tickets into one of the categories: "Technical Issue," "Billing Inquiry," or "General Inquiry."

**Ticket:** "Can you explain why my latest bill is higher than usual? I haven't changed my plan."

助理：

The customer support ticket should be classified as a "Billing Inquiry."

Here's the reasoning:

- The customer is asking about the reason for a change in their latest bill amount.
- The context revolves around billing and payment, which falls under the category of "Billing Inquiry."

## 3-shot prompt

使用者：

Please classify the following customer support tickets into one of the categories: "Technical Issue," "Billing Inquiry," or "General Inquiry."  
Use these examples to guide your classification:

**Ticket:** "I can't log in to my account. The system keeps giving me an error message."  
**Category:** Technical Issue

**Ticket:** "I received an email saying my account will be suspended unless I make a payment. Is this a scam?"  
**Category:** Billing Inquiry

**Ticket:** "What are the hours of operation for your customer support team?"  
**Category:** General Inquiry

Now, classify the following ticket:  
**Ticket:** "Can you explain why my latest bill is higher than usual? I haven't changed my plan."

助理：

Category: Billing Inquiry

Explanation: The ticket is asking for an explanation regarding a change in the billing amount, which falls under the billing category as it pertains to questions about charges or payments.

## 範例 2：提供範例作為回合提示詞

在此範例中，範例會以回合形式提供，每個範例一個回合，其中的 User 角色包含輸入，Assistant 角色包含該範例的預期輸出。

## 0-shot Prompt

系統：

Your task is to Classify the following texts into the appropriate categories. The categories to classify are:

Categories:

- Food
- Entertainment
- Health
- Wealth
- Other

使用者：

I have 20\$ in my pocket.

助理：

Category: Wealth

Explanation: The text mentions a specific amount of money, which falls under the category of wealth. The presence of a monetary value directly relates to financial status or economic aspect.

### 3-shot Prompt (as turns)

系統：

Your task is to Classify the following texts into the appropriate categories. The categories to classify are:

Categories:

- Food
- Entertainment
- Health
- Wealth
- Other

使用者：

I love to eat pizza.

助理：

Food

使用者：

I enjoy watching movies.

助理：

Entertainment

使用者：

I am going to the gym after this.

助理：

Health

使用者：

I have 20\$ in my pocket.

助理：

Wealth

### 範例 3：在系統提示詞中提供範例

在此範例中，您可以指定一個「範例」區段，然後在系統提示詞中列出一系列的範例。如果您的範例很長且複雜，而且有助於模型尋找範例鏡頭，這種在提示詞中提供範例的方式會特別有用。

#### 0-shot Prompt

系統：

You are an AI assistant helping to draft professional reports for enterprise stakeholders. Ensure the report is formatted in rich text, with headings, bullet points, and highlighted important tasks.

使用者：

Draft a detailed project update report for our stakeholders, summarizing the project milestones achieved, tasks completed, and future plans.

助理：

```
Project Update Report
```

```
Executive Summary
```

This report provides a comprehensive update on the project milestones achieved, tasks completed, and future plans. Our team has made significant progress, and we are on track to meet the project goals. The following sections outline the details of our accomplishments and upcoming activities.

```

```

```
Project Milestones Achieved
```

```
1. Project Initiation
```

- **Milestone Date:** July 15, 2023
- **Description:** Completed initial project planning and secured stakeholder approval.

```
2. System Design Completion
```

- **Milestone Date:** September 30, 2023
- **Description:** Finished the detailed system design and received positive feedback from technical review meetings.

```
3. Prototype Development
```

- **Milestone Date:** November 15, 2023
- **Description:** Developed and tested a working prototype of the system.

```

```

```
Tasks Completed
```

```
1. Requirement Gathering
```

- **Description:** Conducted comprehensive stakeholder interviews and collected detailed requirements.
- **Highlighted Important Tasks:**
  - **User Story Mapping:** Mapped out user stories to ensure all user needs are addressed.
  - **Requirement Documentation:** Created detailed requirement specifications.

### ### \*\*2. System Architecture Design\*\*

- \*\*Description:\*\* Designed the system architecture, focusing on scalability and maintainability.
- \*\*Highlighted Important Tasks:\*\*
  - \*\*Technology Stack Selection:\*\* Chose the appropriate technology stack based on project requirements.
  - \*\*Architecture Diagrams:\*\* Developed detailed architecture diagrams for easy understanding.

### ### \*\*3. Prototype Development and Testing\*\*

- \*\*Description:\*\* Developed and rigorously tested the prototype.
- \*\*Highlighted Important Tasks:\*\*
  - \*\*Coding:\*\* Implemented core functionalities of the prototype.
  - \*\*Testing:\*\* Conducted unit and integration testing to ensure prototype reliability.

---

## ## Future Plans

### ### \*\*1. Full System Development\*\*

- \*\*Description:\*\* Begin the full-scale development of the system based on the approved design and prototype.
- \*\*Highlighted Important Tasks:\*\*
  - \*\*Module Development:\*\* Develop each module as per the design specifications.
  - \*\*Code Reviews:\*\* Implement regular code reviews to maintain code quality.

### ### \*\*2. User Acceptance Testing (UAT)\*\*

- \*\*Description:\*\* Conduct UAT to ensure the system meets user requirements and is ready for deployment.
- \*\*Highlighted Important Tasks:\*\*
  - \*\*Test Plan Development:\*\* Create detailed UAT plans and test cases.
  - \*\*UAT Execution:\*\* Execute UAT and gather feedback for final adjustments.

### ### \*\*3. Deployment and Training\*\*

- \*\*Description:\*\* Deploy the system and provide necessary training to end-users.
- \*\*Highlighted Important Tasks:\*\*
  - \*\*Deployment:\*\* Roll out the system to the production environment.
  - \*\*Training Sessions:\*\* Organize training sessions for users to familiarize them with the new system.

---

## ## Conclusion

Our project has successfully achieved key milestones and completed significant tasks. With the current momentum, we are well-positioned to meet our future objectives. We remain committed to delivering a high-quality system that meets all stakeholder expectations.

---

**\*\*Prepared by:\*\***

[Your Name]

[Your Position]

[Date]

**\*\*Contact Information:\*\***

[Your Email]

[Your Phone Number]

---

This detailed update aims to keep stakeholders informed and reassured about the project's progress and future direction. For any queries or further information, please do not hesitate to contact me directly.

## 2-shot Prompt

系統：

You are an AI assistant helping to draft professional reports for enterprise stakeholders. Below are a few examples of well-formatted project update reports to guide your response.

<Example 1>

<b>Project Update Report</b><br>

<i>Date:</i> January 15, 2023<br>

<hr>

<b>Project Milestones Achieved:</b><br>

- Successfully launched the new CRM system.<br>
- Integrated the marketing automation tool.<br>
- Completed the employee training sessions.

<b>Tasks Completed:</b><br>

- Developed the user manual.<br>



```
- Conducted system testing.

- Prepared the training materials.

Future Plans:

- Roll out the CRM system to all departments.

- Schedule follow-up training sessions.

- Monitor system performance and make necessary adjustments.
```

```
<hr>
Prepared by:

[Your Name]

[Your Position]

[Your Contact Information]
```

```
<Example 2>
```

```
Project Update Report

<i>Date:</i> February 20, 2023

<hr>
```

```
Project Milestones Achieved:

- Launched the new website.

- Completed the SEO optimization.

- Organized the stakeholder meeting.
```

```
Tasks Completed:

- Designed the website layout.

- Wrote the content for the website.

- Tested the website functionality
```

```
Future Plans:

- Implement the analytics tracking.

- Launch the social media campaign.

- Gather feedback from stakeholders.
```

```
<hr>
Prepared by:

[Your Name]

[Your Position]

[Your Contact Information]
```

使用者：

Draft a detailed project update report for our stakeholders, summarizing the project milestones achieved, tasks completed, and future plans.

助理：

```
Project Update Report

<i>Date:</i> October 5, 2023

<hr>

Project Milestones Achieved:

- Successfully completed the implementation of the new enterprise resource planning (ERP) system.

- Conducted and completed the first phase of the data migration process.

- Successfully launched the new inventory management module.

Tasks Completed:

- Developed and finalized the comprehensive user training program, including detailed user manuals and video tutorials.

- Conducted extensive system integration testing with all existing systems and ensured seamless interoperability.

- Executed the initial data migration from the legacy system to the new ERP system, resulting in a 95% accuracy rate.

Future Plans:

- Roll out the new ERP system to all departments in the next two phases.

- Schedule and conduct follow-up training sessions for all employees.

- Implement the remaining modules of the ERP system, including financial management and human resources.

- Establish a robust monitoring and support framework to ensure smooth system operation and address any issues promptly.

<hr>
Prepared by:

[Your Name]

[Your Position]

[Your Contact Information]

```

This detailed report provides a clear and concise overview of the project's progress, highlighting the milestones achieved, tasks completed, and future plans.

It ensures stakeholders are well-informed and can make strategic decisions based on the current status of the project.

## 工具呼叫系統

透過在請求中傳遞工具組態結構描述，Amazon Nova 模型可以使用工具呼叫。模型的提示詞將透過此工具組態得到加強，因此它是開始最佳化工具呼叫系統的重要位置。

請考慮下列重要原則：

- 工具定義應清晰、簡潔。它們應該很容易理解，意圖必須非常明顯。
- 使用關鍵差異化和邊界條件來定義何時應使用一個工具而非另一個工具。
- 請注意輸入引數類型。問：它們是否合理，是否預期以這種方式正常使用？

使用「工具選擇」來控制呼叫工具的時間

工具選擇參數可讓您自訂模型的工具呼叫行為。建議您使用此值來精細控制呼叫哪些工具以及何時呼叫。

例如，對於結構化輸出等使用案例，您可能希望在每次調用 Amazon Nova 時呼叫特定工具。您可以將輸出的結構描述定義為工具，然後將工具選擇設定為該工具的名稱。

```
{
 "toolChoice": {
 "tool": {
 "name": "name_of_tool"
 }
 }
}
```

對於許多代理式使用案例，您可能想要確保模型一律選取其中一個可用的工具。若要這樣做，您可以將工具選擇設定為 any，每次調用模型時呼叫且僅呼叫一個工具。

```
{
 "toolChoice": {
 "any": {}
 }
}
```

最後，對於是否呼叫工具高度依賴於對話上下文的使用案例，您可以將工具選擇設定為 `auto`。這是預設行為，會將工具選擇完全交由模型來判斷。

```
{
 "toolChoice": {
 "auto": {}
 }
}
```

## 進階提示技巧

這些章節提供進階指引，說明如何改善提示的品質，並利用延伸思考等重要功能。

### 使用原因模式

Amazon Nova 2 模型提供選用的推理模式，透過讓模型在回應之前有系統地處理問題，增強模型解決複雜問題的方法。利用模型的推理模式是改善提示準確性的強大方式。

使用時機：建議將合理模式用於複雜任務，例如具有下列項目的使用案例：

- 多個推理步驟：數學驗證、演算法設計、系統架構
- 交互參考資訊：文件分析、選項比較、權衡評估
- 容易出錯的計算：財務建模、資料分析、複雜偵錯
- 使用限制進行規劃：資源最佳化、相依性管理、風險評估
- 複雜分類：多標籤分類、階層式分類法、細微決策界限
- 工具呼叫案例：多步驟 API 工作流程、資料庫查詢最佳化、協調系統整合

#### Note

如需推理模式的詳細資訊，請參閱 [使用推理](#)。

### 採取由上而下的方法

對於模型需要評估多種方法來解決問題的情況，請指示它採取由上而下的方法。

- 當模型從大局開始，然後分解為更小、更詳細的子問題或步驟時，Amazon Nova 2 模型的效能更好。

- 明確指示模型先識別主要目標，然後將其分解為可管理的元件，然後再處理每個部分的詳細資訊。
- 這種結構化方法有助於模型組織其思維，並產生更一致的推理鏈。

範例：

```
{{User query}}. Start with the big picture and break it down into progressively smaller, more detailed subproblems or steps.
```

## 行進思維鏈

雖然推理模式透過系統化問題解決來提高準確性，但在某些情況下，以非合理模式發出 Chain of Thought (CoT) 提示可能會更好地滿足您的需求。

使用時機：

- 透明度和可稽核性：當您想要查看、驗證或稽核模型的推理程序時，CoT 會提供每個步驟的完整可見性。這對於受管制產業、高風險決策，或您想要記錄答案背後的邏輯時至關重要。
- 自訂推理結構：使用 CoT 強制執行特定推理模式或方法。您可以引導模型遵循組織的決策架構、使用特定領域的問題解決方法，或確保依特定順序考慮因素。
- 提示詞開發和偵錯：在提示詞工程階段期間，CoT 可協助您了解模型如何處理問題，找出原因分解的位置，並更有效地重複提示詞。
- 混合方法：考慮在開發期間使用 CoT 來完善您的提示，然後在您對模型對特定使用案例的方法有信心之後，切換至生產部署的推理模式。

### Note

並非所有任務都需要 CoT。對於更簡單的任務，允許模型使用自己的推理程序。

引導模型的 CoT 方向：

```
{{User query}} Please follow these steps:
```

1. {{Step 1}}
2. {{Step 2}}
- ...

## 使用長上下文視窗

Amazon Nova 2 模型支援的內容長度為 100 萬個字符，並擅長理解程式碼和回答長文件的問題。其效能（包括系統提示遵循和工具使用）可能會隨著內容大小的增加而稍微下降。

如何使用它：

- 將長格式資料放在開頭：將長文件和輸入放在提示詞的開頭附近。將它們放在查詢、指示和範例之前。
- 將指示放在末尾：將指示放在提示詞的末尾。當先提供上下文，並在末尾提供指示時，模型會發揮最佳效能。
- 結構文件內容開始和結束標記：使用開始和結束標記，例如 DOCUMENT {idx} START 和 DOCUMENT {idx} END 來表示長文件的開始和結束，其中 {idx} 代表特定文件的索引。

範本範例：

```
// Provide your long inputs at the top of your prompt
BEGIN INPUT DOCUMENTS

DOCUMENT 1 START
{{Your document}}
DOCUMENT 1 END

END INPUT DOCUMENTS

// Then specify your query and instructions
BEGIN QUESTION
{{User query}}
END QUESTION

BEGIN INSTRUCTIONS
{{Instructions}}
END INSTRUCTIONS
```

## 支援文字中的基本答案

我們建議您為模型提供與輸入查詢相關的可信資訊。此資訊與輸入查詢通常是系統的一部分，稱為擷取擴增產生 (RAG)。

- 在此過程中，某些相關的情境文件或資訊會擴增為實際的使用者提示，以便模型獲得可信的內容，以產生相關且準確的回應。

- 指示 Amazon Nova 2 使用來自可信任來源的參考文字來回答，可以引導它根據提供的資料編寫回應，並確保其回應以準確且相關的資訊為基礎，從而增強所產生內容的可靠性和可信度。
- 使用參考文字有助於避免幻覺，從而改善回應的整體品質和可信度。為了將幻覺降至最低，我們建議您在模型提示中明確提及 DO NOT USE INFORMATION THAT IS NOT IN REFERENCE TEXTS!。

## 提示範本

System:

In this session, the model has access to search results and a user's question, your job is to answer the user's question using only information from the search results.

Model Instructions:

- DO NOT USE INFORMATION THAT IS NOT IN SEARCH RESULTS!

User: {Query}

Resource: Search Results: {Reference texts}

## 範例：提供接地內容有助於防止模型幻覺

此範例顯示接地內容如何防止模型幻覺。

系統：

In this session, the model has access to search results and a user's question, your job is to answer the user's question using only information from the search results. Model Instructions:- DO NOT USE INFORMATION THAT IS NOT IN SEARCH RESULTS!

使用者：

What were the economic impacts of the COVID-19 pandemic on the United States in 2020?

Resource: Search Results: In 2020, the United States experienced significant economic impacts due to the COVID-19 pandemic. The U.S. economy contracted by 3.5% in 2020, according to the Bureau of Economic Analysis. Unemployment rates surged to 14.7% in April 2020, the highest since the Great Depression, before gradually declining. Small businesses faced severe challenges, with millions of firms closing permanently. Additionally, consumer spending dropped sharply as people reduced non-essential expenditures and saved more. Government intervention played a critical role in mitigating these impacts through stimulus packages and support programs, such as the Paycheck Protection Program (PPP) for small businesses and direct payments to

individuals. Despite these measures, the economic recovery remained uneven across different sectors and regions.

## 使用引文標記的 Ground

對於長文件任務，我們建議您指示 Amazon Nova 2 模型在繼續任務之前，使用文件相關區段的引文來建立回應。

這種方法有助於模型專注於最相關的資訊，避免因不必要的內容而分心。當您請求模型以其回應為基礎時，應該對可引用的區段進行編號。例如，Passage %[1]% Passage %[2]% 等。

範例：使用引文標記

### Example 引文標記提示

```
You are an AI financial assistant. Your task is to find patterns and insights from
multi-year financial documents

Passage %[1]%
{{Your document}}

Passage %[2]%
{{Your document}}

Task:
Analyze my LLC's reports across multiple years to identify significant performance
trends, segment growth patterns and strategic shifts.

Context information:
- You have access to my LLC's annual financial reports (10-K) for multiple fiscal years
in PDF format
- These reports contain comprehensive financial data including income statements,
balance sheets, cash flow statements and management discussions
- The analysis should focus on year-over-year comparisons to identify meaningful trends
- I operate two business segments, one in Massachusetts and one in New York

Based on the provided Context, extract key financial metrics from each year's reports
phrases from the documents.
Place citations as inline markers (e.g., %[1]%, %[2]%, etc.) directly within the
relevant parts of the response
text. Do not include a separate citation section after the response.
Response Schema:
%(Extracted Financial Metrics)
```



```
%% (Extracted Financial Metrics)
%% (Extracted Financial Metrics)
...
```

根據使用者的任務擷取金鑰資訊後，您可以使用擷取的財務指標來回答相關問題，如下所示：

### Example使用擷取的指標進行後續分析

```
Task
Analyze my LLC's financial reports across multiple years to identify significant
performance trends, segment growth patterns and strategic shifts.

{{extracted financial metrics}}
```

```
Model Instructions:
- Organize data chronologically to identify meaningful trends
- DO compare segment performance across the five-year period
- DO identify significant strategic shifts or investments mentioned in management
 discussions
- DO NOT make speculative predictions beyond what is supported by the data
- ALWAYS note any changes in accounting practices or reporting methodologies that might
 affect year-over-year comparisons
```

```
Response style and format requirements:
- Respond in markdown
- Structure the analysis with clear headings and subheadings
- Present key financial metrics in tabular format showing all five years side-by-side
- Include percentage changes year-over-year for all major metrics
- Create a section dedicated to visualizing the most significant trends (with
 descriptions of what would be shown in charts)
- Limit the executive summary to 250 words maximum
- Format segment analysis as separate sections with consistent metrics across all
 segments
- MUST include a Key Insights bullet-pointed list at the end of each major section
```

## 使用 Nova Web Grounding

Amazon Nova 2 模型提供可使用的內部 Web 基礎工具，而不是直接提示引文將模型置於支援文字中。啟用時，Amazon Nova 2 模型會直接查詢 Web 和 Amazon 的知識圖表，並以引文做為最終回應的基礎。

若要進一步了解如何利用 Amazon Nova Web Grounding，您可以參考 [Amazon Nova Web Grounding 使用者指南](#)。

## 產生結構化輸出

為了確保一致的結構化輸出格式，您可以使用結構化輸出，包括 XML、JSON、Markdown 等格式，或使用工具使用功能。

- 此方法可讓下游系統更有效地了解和剖析模型產生的輸出。
- 透過提供明確指示給模型，模型會以遵循預先定義之結構描述的方式產生回應。

例如，如果下游剖析器預期 JSON 物件中金鑰的特定命名慣例，您應該在提示結尾指定回應結構描述。此外，如果您偏好回應為 JSON 格式，而且沒有任何前序文字，請相應地指示模型。也就是說，明確狀態 請僅產生 JSON 輸出。請勿提供任何前綴。以確保輸出乾淨。

### Tip

- 當資料格式要求是在結構描述本身中定義，而不是透過使用範例（例如，以 YYYY/MM/DD 格式指定日期）時，我們會觀察到最符合資料格式的要求。
- 對於最多 10 個金鑰的簡單 JSON 輸出，您可以在下面找到結構描述。對於更複雜的結構描述，我們建議您透過工具定義結構描述。工具使用利用稱為限制解碼的技術，這將提高模型對這些複雜結構的遵循。

## 常見格式結構描述

以下是常見格式結構描述的範例。

### JSON

```
JSON_format = """Write your response following the JSON format below:

```json
{
  "key1": "value1",
  "key2": "value2",
  key3: [{
    "key3_1": "value_3_1 written in YYYY/MM/DD format",
    "key3_2": "value_3_2 day of the week written in full form",
    ...}]`
"""
```

XML

```
XML_format = """Write your response following the XML format below:
```

```
<output>
  <task>"task1"</task>
  <subtask>
    <task1_result> ( task 1 result )</task1_result>
    <task2_result> ( task 2 result )</task2_result>
    <task3_result> ( task 3 result )</task3_result>
  </subtask>
  <task>"task2"</task>
  <subtask>
    <task1_result> ( task 1 result )</task1_result>
    <task2_result> ( task 2 result )</task2_result>
    <task3_result> ( task 3 result )</task3_result>
  </subtask>
</output>
```

```
"""
```

Markdown

```
markdown_schema = """Write your response following the markdown format below:
```

```
## Introduction
( 2-3 line intro)

## Design Guidance
(Bulleted list of design guidance)

## Step by Step Instructions on Execution
( Bulleted list of instructions with each with bold title.

## Conclusion
( conclusion )
```

```
"""
```

預填充助理內容

如果您以非原因模式產生結構化輸出，您可以透過預先填入助理內容來微調模型的回應。

預填充可在非基礎模式下改善輸出格式的一致性。它可讓您指示模型的動作、略過前言，以及強制執行特定輸出格式，例如 JSON 和 XML。例如，如果您使用 { 或 預先填入助理內容 ``json，該輸入會引導模型產生 JSON 物件，無需其他資訊。

Tip

如果您要明確地尋找擷取 JSON，則常見的模式是預先填入，``json並在上新增停止序列``。這可確保模型輸出可透過程式設計方式剖析的 JSON 物件。

Example預填充助理內容

以下程式碼示範如何使用 API 進行預先填充：

```
import boto3
import json

# Create a Bedrock Runtime client.
client = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1"
)

request_body = {
    "system": [
        {"text": "You write JSON objects based on the given instructions"}
    ],
    "messages": [
        {
            "role": "user",
            "content": [{"text": "Provide details about the best selling full-frame cameras in past three years. Answer in JSON format with keys like name, brand, price and a summary."}]
        },
        {
            "role": "assistant",
            "content": [{"text": " Here is the JSON response: ``json"}]
        }
    ]
}
```

```
    ],
    "inferenceConfig": {
        "maxTokens": 1000,
    }
}

# Invoke the model and extract the response body.
response = client.invoke_model(
    modelId="amazon.nova-2-lite-v1:0",
    body=json.dumps(request_body)
)

model_response = json.loads(response["body"].read())
```

指定用於複雜結構描述的工具

另一種方法是使用工具來強制模型回應的特定結構描述，方法是在工具使用結構描述區段中包含 pydantic JSON 結構描述。您可以為提供的結構描述指定工具選擇，Amazon Nova 的回應將根據選取的工具進行結構化。若要進一步了解如何利用工具，請參閱[工具使用](#)。

範例

範例 1：新增輸出結構描述

在輸出結構描述區段新增適當的結構描述，可讓模型偏好符合該結構描述。不過，模型輸出不是確定性的，並且可能與輸出結構描述不同。

Unspecific Prompt

使用者：

```
Provide details about the best selling full-frame cameras in past three years.
Answer in JSON format with keys like name, brand, price and a summary.
```

助理：

```
```json
[
 {
 "name": "Sony Alpha A7R IV",
 "brand": "Sony",
 "price": "Approximately $3,500",
```

```

"summary": "The Sony Alpha A7R IV is renowned for its exceptional image quality,
thanks to a 61-megapixel full-frame sensor. It also features 4K video recording
and a robust build, making it a favorite among professional photographers and
videographers."
},
{
"name": "Canon EOS R5",
"brand": "Canon",
"price": "Approximately $3,900",
"summary": "The Canon EOS R5 combines high-resolution imaging with advanced video
capabilities. It houses a 45-megapixel sensor, 8K video recording, and impressive
autofocus performance, appealing to both photography and filmmaking professionals."
},
{
"name": "Nikon Z7 II",
"brand": "Nikon",
"price": "Approximately $2,800",
"summary": "The Nikon Z7 II offers a 45.7-megapixel full-frame sensor and is
praised for its high resolution and excellent dynamic range. It also supports 4K
video recording and features a versatile Z-mount lens system, making it a strong
contender in the full-frame market."
},
{
"name": "Fujifilm GFX 100S",
"brand": "Fujifilm",
"price": "Approximately $7,000",
"summary": "While not a traditional full-frame camera, the Fujifilm GFX 100S stands
out with its 102-megapixel medium format sensor. It is lauded for its stunning
detail and color reproduction, though it comes at a higher price point and is aimed
at specialized users."
}
]
` ``

```

## Specific Prompt

使用者：

```

Provide details about the best selling full-frame cameras in past three years.
You MUST answer in JSON format only. Write your response following the the format
below:
[
{
 "name": "name goes here",
 "brand": "brand goes here",

```

```
"price": "price goes here",
"summary": "summary goes here"
}]
```

助理：

```
[
 {
 "name": "Nikon D850",
 "brand": "Nikon",
 "price": "$3,296.95",
 "summary": "The Nikon D850 is a professional full-frame DSLR camera known for its
exceptional image quality, 45.7-megapixel resolution, 7 fps continuous shooting,
and 4K video capabilities."
 },
 {
 "name": "Canon EOS 5D Mark IV",
 "brand": "Canon",
 "price": "$3,499.00",
 "summary": "The Canon EOS 5D Mark IV is a popular full-frame DSLR appreciated for
its 30.4-megapixel sensor, 4K video recording, robust build quality, and reliable
autofocus system."
 },
 {
 "name": "Sony A7R III",
 "brand": "Sony",
 "price": "$3,198.00",
 "summary": "The Sony A7R III is a high-resolution full-frame mirrorless camera
renowned for its 42.4-megapixel sensor, excellent low-light performance, and
versatile lens compatibility."
 },
 {
 "name": "Fujifilm GFX 100S",
 "brand": "Fujifilm",
 "price": "$5,999.99",
 "summary": "The Fujifilm GFX 100S is a medium format mirrorless camera that stands
out for its 102-megapixel sensor, high dynamic range, and exceptional color
reproduction."
 },
 {
 "name": "Sony A7 III",
 "brand": "Sony",
 "price": "$1,998.00",
```

```
"summary": "The Sony A7 III is a full-frame mirrorless camera popular among
enthusiasts and professionals for its 24.2-megapixel sensor, fast autofocus, and 4K
video capabilities."
}
]
```

## 範例 2：預先填入模型回應

另一種方法是透過在助理的訊息中包含所需的初始文字，來預先填入模型的回應。在此情況下，Amazon Nova 模型的回應將從助理訊息結束的位置繼續。

秘訣：預先填入只有在未啟用推理時才有效。

使用者：

```
Provide details about the best selling full-frame cameras in past three years.
Your response should be in JSON format, with the following keys: name, brand, year,
price, summary.
```

助理（預先填入）：

```
```json
```

助理：

```
{
  "cameras": [
    {
      "name": "Sony Alpha A7 III",
      "brand": "Sony",
      "year": 2018,
      "price": "$1999",
      "summary": "The Sony Alpha A7 III is a highly acclaimed full-frame mirrorless camera
known for its excellent image quality, advanced autofocus system, and 4K video
capabilities."
    },
    {
      "name": "Canon EOS R5",
      "brand": "Canon",
      "year": 2020,
      "price": "$3899",
```



```
"summary": "The Canon EOS R5 is a top-tier full-frame mirrorless camera that offers
stunning 45MP images, 8K video recording, and exceptional performance in various
lighting conditions."
},
{
  "name": "Nikon Z7 II",
  "brand": "Nikon",
  "year": 2020,
  "price": "$2999",
  "summary": "The Nikon Z7 II is a flagship full-frame mirrorless camera with a 45.7MP
sensor, superb image quality, and advanced video recording features."
},
{
  "name": "Fujifilm GFX 100S",
  "brand": "Fujifilm",
  "year": 2020,
  "price": "$7999",
  "summary": "The Fujifilm GFX 100S is a medium format mirrorless camera known for its
impressive 102MP sensor, beautiful film simulations, and versatile lens options."
},
{
  "name": "Sony Alpha A7R IV",
  "brand": "Sony",
  "year": 2019,
  "price": "$3499",
  "summary": "The Sony Alpha A7R IV features a 61MP sensor, excellent low-light
performance, and advanced video capabilities, making it a favorite among professional
photographers."
}
]
}
```

範例 3：指定要使用的工具

除了使用結構化輸出的預填充技術之外，您還可以將 Nova 的工具功能與 toolChoice 結合。在此範例中，模型會產生遵循提供的 JSON 結構描述的工具呼叫，並呼叫指定為定義完整 JSON 格式的特定工具。如需其他指引，請參閱進階系統。

使用者：

```
From the below provided Query, extract the relevent entities
```

Query:

John works in BUILDING-0987 and have been incharge of product id 23564#. His performance have been excellent in past year and is up for a raise.

Use the extract_ner tool.

ToolConfig :

```
{
  "tools": [
    {
      "toolSpec": {
        "name": "extract_ner",
        "description": "Extract all the named entities. based on provided
input",
        "inputSchema": {
          "json": {
            "type": "object",
            "properties": {
              "entities": {
                "type": "array",
                "items": {
                  "type": "object",
                  "properties": {
                    "name": {
                      "type": "string",
                      "description": "The extracted entity name. This
should be a name of a person, place, animal or thing"
                    },
                    "location": {
                      "type": "string",
                      "description": "The extracted location name. This
is a site name or a building name like SITE-001 or BUILDING-003"
                    },
                    "product": {
                      "type": "string",
                      "description": "The extrcted product code, this is
generally a 6 digit alphanumeric code such as 45623#, 234567"
                    }
                  },
                  "required": [
                    "name",
                    "location",
```

```

        "product"
      ]
    }
  },
  "required": [
    "entities"
  ]
}

}

}

}],
"toolChoice": {
  "tool": {
    "name": "extract_ner"
  }
}
}

```

多語言使用案例

Amazon Nova 2 模型已針對超過 200 種語言進行訓練，並針對 15 種語言進行最佳化。

主題

- [提示正確翻譯](#)
- [強制執行一致的撰寫慣例](#)

提示正確翻譯

若要利用此功能進行短格式翻譯（幾個句子），您可以指示模型將文字翻譯為指定的目標語言。

Example翻譯提示

Translate the following text into {target language}. Please output only the translated text with no prefix or introduction: {text}

Translate the following sentence from {source_language} to {target language}: {text}

{text} How do you say this sentence in {target_language}

強制執行一致的撰寫慣例

在以字元為基礎的語言中，Amazon Nova 2 模型可能會利用來源語言的字元集。您可以使用下列提示來強制執行一致的輸出。

Example強制撰寫慣例

```
When translating, ensure to use the correct orthography / script / writing convention of the target language, not the source language's characters
```

工具呼叫

代理系統

主題

- [設定正確的推論參數](#)
- [考慮延遲要求](#)
- [使用工具呼叫指示的刻意措辭](#)
- [利用「思考」命令](#)
- [工具呼叫排序](#)
- [建立品質工具結構描述](#)
- [建立子代理程式](#)
- [使用工具進行多模態輸入](#)
- [後續步驟](#)

設定正確的推論參數

工具呼叫需要模型中非常特定的結構化輸出，並使用下列推論參數來改善：

- 非推理模式：溫度：0.7 & Top P：0.9
- 原因模式：溫度：1 & Top P：0.9

考慮延遲要求



Tip

Amazon Nova 2 模型能夠使用推理開啟和關閉的工具呼叫。不過，推理模式會對延遲產生重大影響。

對於延遲敏感應用程式，您應該針對推斷模式進行最佳化，並盡可能簡化必要的工具呼叫。將多步驟工作流程分割為離散步驟，以減少模型對調節不必要的參數的依賴。

使用工具呼叫指示的刻意措辭

工具名稱：在系統提示中參考工具在工具呼叫系統中很常見，以指示模型何時呼叫工具。當您在提示中參考工具時，我們建議您使用工具名稱，而不是 xml 或 pythonic 參考或範例。

良好工具參考的範例

```
Use the 'run_shell_command' tool for running shell commands
```

錯誤工具參考的範例

```
Call run_shell_command() to run shell commands
```

利用「思考」命令

對於思考有利於工具呼叫的所有使用案例，我們建議您利用推理模式，而不是提示模型「在標籤中思考」或使用「思考」工具。

Amazon Nova 2 模型經過廣泛的推理模式訓練，在推理模式用於思維鏈時會產生最佳結果。

工具呼叫排序

在可能需要同時使用內建工具和原生工具呼叫的使用案例中，模型會偏向先呼叫內建工具。

請勿指示模型在提示中採取不同的動作。反之，請設計您的工作流程以將此納入考量。

例如，如果您不希望模型使用內建工具，請勿將這些工具包含在工作流程中，讓模型不會偏離它們。

建立品質工具結構描述

工具結構描述是您可以提示設計有效工具呼叫系統的關鍵位置之一。不過，請務必考慮在工具結構描述中擷取的內容、如何以語意描述每個結構描述元素，以及系統提示如何在系統指示中參考工具和結構描述元素。

Amazon Nova 2 模型已針對工具結構描述中的簡潔描述進行最佳化。保持簡短。

工具結構描述與系統提示準則：

包含在工具結構描述中：

- 核心功能：工具的功能（建議 20-50 個字）
- 參數規格：清除每個參數的描述（每個參數大約 10 個字）
- 預期的格式：資料類型（例如列舉、整數、浮點數）、必要欄位和有效值範圍

在系統提示中包含：

- 使用協同運作邏輯（使用特定工具的時間和原因）和業務規則（條件式邏輯、排序需求和相依性）專用#Tool Usage區段。
- 錯誤處理策略：新增 #Error Handling and Troubleshooting區段，其中包含如何回應失敗或非預期輸出的指示
- 輸出格式：新增如何呈現給使用者的詳細資訊

範例

```
You are a software engineering issue root cause analysis agent. You are tasked with reviewing a customer issue and examining the repository to identify a plan to resolve the issue.

# Core Mandates
- **DO NOT** update the original issue that was posted by the user. You only add *additional* comments to the reported issue if necessary

# Primary Workflows
1. **Understand:** Analyze the user's request and explore the codebase thoroughly using **get_file_contents** to grasp file structures and conventions.
2. **Plan:** Create a coherent, evidence-based plan for resolving the task and share it with the user following the format below
```

```
# Tool Usage
- **Read the Issue:** Always start by using the **read_issue** tool to get the details about the requested issue
- **File Paths:** Always end the file path with "/" if you are searching a directory using the **get_file_contents** tools
- **Parallelism:** Execute multiple independent tool calls in parallel when feasible

# Error Handling and Troubleshooting
- **File Exploration:** If you get an error that a file doesn't exist, try searching at the directory level first to validate the file path

# Output Formatting
Return your plan in markdown in the following format

## Issue
<Your root cause analysis of the issue>

## Resolution Plan
<your step by step plan of how to solve the issue>
```

建立子代理程式

當您遇到下列情況時，請考慮建立專用子代理程式，而不是具有許多工具的單一代理程式：

- 工具計數超過 20：大型工具集變得難以管理和增加選擇錯誤
- 不同的功能網域：工具自然叢集分為不同的類別（例如資料擷取與處理與報告）
- 複雜結構描述：當參數深度超過 3-4 個層級時，或工具具有複雜的相互依存性
- 對話長度：工作流程定期超過 15-20 個回合可能受益於特殊的子代理程式
- 效能降低：如果您發現工具選擇的準確性降低或延遲增加

Tip

MCP 伺服器隨附您無法控制的工具和結構描述。僅包含工作流程完成必要任務所需的工具。

使用工具進行多模態輸入

對於多模式任務，我們尚未觀察到利用結構化任務工具（例如擷取或產生時間戳記）來提高準確性。

反之，我們建議您檢閱提示多模式輸入區段中的相關區段，了解如何使用提供的範本成功提示模型。

後續步驟

- 如需多模式提示，請參閱 [提示多模態輸入](#)。

提示多模態輸入

以下各節提供影像和影片理解的指引。如需音訊相關提示，請參閱[the section called “語音對話提示”](#)一節。

一般多模式準則

使用者提示和系統提示

為了多模式了解使用案例，每個請求都應包含使用者提示文字。可能只包含文字的系統提示是選用的。

系統提示可用來為模型指定角色，並定義一般性格和回應樣式，但不應用於詳細的任務定義或輸出格式指示。

在使用者提示中包含任務定義、指示和格式詳細資訊，以獲得比多模式使用案例的系統提示更強大的效果。

內容順序

傳送至 Amazon Nova 的多模式理解請求應包含一或多個檔案和使用者的提示。使用者文字提示應該是訊息中的最後一個項目，一律在影像、文件或影片內容之後。

```
message = {
  "role": "user",
  "content": [
    { "document|image|video|audio": {...} },
    { "document|image|video|audio": {...} },
    ...
    { "text": "<user prompt>" }
  ]
}
```

如果您想要在使用者提示中參考特定檔案，請使用「文字」元素來定義每個檔案區塊前面的標籤。

```
message = {
  "role": "user",
  "content": [
```



```
{ "text": "<label for item 1>" },
{ "document|image|video|audio": {...} },
{ "text": "<label for item 2>" },
{ "document|image|video|audio": {...} },
...
{ "text": "<user prompt>" }
]
}
```

文件和映像了解

下列各節提供指引，說明如何針對需要了解或分析映像和文件的任務建立提示。

從映像擷取文字

Amazon Nova 模型可以從影像擷取文字，這種功能稱為光學字元辨識 (OCR)。為了獲得最佳結果，請確定您提供給模型的映像輸入具有足夠高的解析度，以方便辨別文字字元。

對於文字擷取使用案例，我們建議使用下列推論組態：

- 溫度：預設 (0.7)
- topP：預設 (0.9)
- 請勿啟用推理

Amazon Nova 模型可以將文字擷取到 Markdown、HTML 或 LaTeX 格式。建議使用下列使用者提示範本：

```
## Instructions
Extract all information from this page using only {text_formatting} formatting. Retain
the original layout and structure including lists, tables, charts and math formulae.

## Rules
1. For math formulae, always use LaTeX syntax.
2. Describe images using only text.
3. NEVER use HTML image tags `<img>` in the output.
4. NEVER use Markdown image tags `![]()` in the output.
5. Always wrap the entire output in ``` tags.
```

輸出會以完整或部分 Markdown 程式碼圍欄 (```) 包裝。您可以使用類似下列的程式碼來分割程式碼圍欄：

```
def strip_outer_code_fences(text):
    lines = text.split("\n")
    # Remove only the outer code fences if present
    if lines and lines[0].startswith("`"):
        lines = lines[1:]
        if lines and lines[-1].startswith("`"):
            lines = lines[:-1]
    return "\n".join(lines).strip()
```

從影像或文字擷取結構化資訊

Amazon Nova 模型可以將影像中的資訊擷取為機器可剖析的 JSON 格式，稱為金鑰資訊擷取 (KIE)。若要執行 KIE，請提供下列項目：

- JSON 結構描述。遵循 JSON 結構描述規格的正式結構描述定義。
- 下列一或多個項目：文件檔案或影像或文件文字

文件或映像必須一律放置在請求中的使用者提示之前。

對於 KIE 使用案例，我們建議使用下列推論組態：

- 溫度：0
- 原因：不需要原因，但在使用純影像輸入或複雜結構描述時可以改善結果。

提示範本

僅限文件或映像輸入

Given the image representation of a document, extract information in JSON format according to the given schema.

Follow these guidelines:

- Ensure that every field is populated, provided the document includes the corresponding value. Only use null when the value is absent from the document.
- When instructed to read tables or lists, read each row from every page. Ensure every field in each row is populated if the document contains the field.

JSON Schema:
{json_schema}

僅限文字輸入

Given the OCR representation of a document, extract information in JSON format according to the given schema.

Follow these guidelines:

- Ensure that every field is populated, provided the document includes the corresponding value. Only use null when the value is absent from the document.
- When instructed to read tables or lists, read each row from every page. Ensure every field in each row is populated if the document contains the field.

JSON Schema:

```
{json_schema}
```

OCR:

```
{document_text}
```

文件或影像和文字輸入

Given the image and OCR representations of a document, extract information in JSON format according to the given schema.

Follow these guidelines:

- Ensure that every field is populated, provided the document includes the corresponding value. Only use null when the value is absent from the document.
- When instructed to read tables or lists, read each row from every page. Ensure every field in each row is populated if the document contains the field.

JSON Schema:

```
{json_schema}
```

OCR:

```
{document_text}
```

偵測影像中的物件及其位置

Amazon Nova 2 模型提供識別影像中物件及其位置的能力，任務有時稱為影像接地或物件當地語系化。實際的應用程式包括影像分析和標記、使用者介面自動化、影像編輯等。

無論影像輸入解析度和長寬比為何，模型都會使用座標空間，將影像水平分割為 1,000 個單位，垂直分割為 1,000 個單位，其中 x : 0 y : 0 位置是影像的左上角。

邊界框使用分別 [x1, y1, x2, y2] 代表左側、頂部、右側和底部的格式來描述。二維座標是使用格式表示 [x, y]。

針對物件偵測使用案例，我們建議使用下列推論參數值：

- 溫度：0
- 請勿啟用推理

提示範本：一般物件偵測

我們建議使用下列使用者提示範本。

使用週框方塊偵測多個執行個體：

Please identify {target_description} in the image and provide the bounding box coordinates for each one you detect. Represent the bounding box as the [x1, y1, x2, y2] format, where the coordinates are scaled between 0 and 1000 to the image width and height, respectively.

使用週框方塊偵測單一區域：

Please generate the bounding box coordinates corresponding to the region described in this sentence: {target_description}. Represent the bounding box as the [x1, y1, x2, y2] format, where the coordinates are scaled between 0 and 1000 to the image width and height, respectively.

使用中心點偵測多個執行個體：

Please identify {target_description} in the image and provide the center point coordinates for each one you detect. Represent the point as the [x, y] format, where the coordinates are scaled between 0 and 1000 to the image width and height, respectively.

偵測具有中心點的單一區域：

Please generate the center point coordinates corresponding to the region described in this sentence: {target_description}. Represent the center point as the [x, y] format, where the coordinates are scaled between 0 and 1000 to the image width and height, respectively.

剖析模型輸出：

上述每個建議提示都會產生逗號分隔字串，其中包含類似下列格式的一或多個週框方塊描述。是否「」可能會有些微變化。包含在字串結尾。例如 [356, 770, 393, 872], [626, 770, 659, 878].

您可以使用規則表達式剖析模型產生的座標資訊，如下列 Python 程式碼範例所示。

程式碼範例

```
def parse_coord_text(text):
    """Parses a model response which uses array formatting ([x, y, ...])
    to describe points and bounding boxes. Returns an array of tuples."""
    pattern = r"\([([^\[\]]*?)\)"
    return [
        tuple(int(x.strip()) for x in match.split(","))
        for match in re.findall(pattern, text)
    ]
```

程式碼範例

若要將週框方塊的標準化座標重新對應至輸入影像的座標空間，您可以使用類似下列 Python 範例的函數。

```
def remap_bbox_to_image(bounding_box, image_width, image_height):
    return [
        bounding_box[0] * image_width / 1000,
        bounding_box[1] * image_height / 1000,
        bounding_box[2] * image_width / 1000,
        bounding_box[3] * image_height / 1000,
    ]
```

提示範本：偵測具有位置的多個物件類別

當您想要在影像中識別多個類別的項目時，您可以使用下列其中一種格式方法，在提示中包含類別清單。

對於模型可能充分理解的常見類別，請在方括號內列出類別名稱（不含引號）：

```
[car, traffic light, road sign, pedestrian]
```

對於細微、不常見或來自模型可能不熟悉的特殊網域的類別，請在括號中包含每個類別的定義。由於此任務具有挑戰性，因此預期模型的效能會降低。

```
[taraxacum officinale (Dandelion - bright yellow flowers, jagged basal leaves, white puffball seed heads), digitaria spp (Crabgrass - low spreading grass with coarse blades and finger-like seed heads), trifolium repens (White Clover - three round leaflets and small white pom-pom flowers), plantago major (Broadleaf Plantain - wide oval rosette leaves with tall narrow seed stalks), stellaria media (Chickweed - low mat-forming plant with tiny star-shaped white flowers)]
```

根據您偏好的 JSON 輸出格式，使用下列其中一個使用者提示範本。

提示選項 1

Detect all objects with their bounding boxes in the image from the provided class list. Normalize the bounding box coordinates to be scaled between 0 and 1000 to the image width and height, respectively.

Classes: {candidate_class_list}

Include separate entries for each detected object as an element of a list.

Formulate your output as JSON format:

```
[
  {
    "class 1": [x1, y1, x2, y2]
  },
  ...
]
```

提示選項 2

Detect all objects with their bounding boxes in the image from the provided class list. Normalize the bounding box coordinates to be scaled between 0 and 1000 to the image width and height, respectively.

Classes: {**candidate_class_list**}

Include separate entries for each detected object as an element of a list.

Formulate your output as JSON format:

```
[
  {
    "class": class 1,
    "bbox": [x1, y1, x2, y2]
  }
]
```

```
    },
    ...
]
```

提示選項 3

Detect all objects with their bounding boxes in the image from the provided class list. Normalize the bounding box coordinates to be scaled between 0 and 1000 to the image width and height, respectively.

Classes: **{candidate_class_list}**

Group all detected bounding boxes by class.

Formulate your output as JSON format:

```
{
  "class 1": [[x1, y1, x2, y2], [x1, x2, y1, y2], ...],
  ...
}
```

提示選項 4

Detect all objects with their bounding boxes in the image from the provided class list. Normalize the bounding box coordinates to be scaled between 0 and 1000 to the image width and height, respectively.

Classes: **{candidate_class_list}**

Group all detected bounding boxes by class.

Formulate your output as JSON format:

```
[
  {
    "class": class 1,
    "bbox": [[x1, y1, x2, y2], [x1, x2, y1, y2], ...]
  },
  ...
]
```

剖析模型輸出

輸出將編碼為 JSON，可與任何 JSON 剖析程式庫進行剖析。

提示範本：螢幕擷取畫面 UI 邊界偵測

我們建議使用下列使用者提示範本。

根據目標偵測 UI 元素位置：

```
In this UI screenshot, what is the location of the element if I want to {goal}? Express the location coordinates using the [x1, y1, x2, y2] format, scaled between 0 and 1000.
```

根據文字偵測 UI 元素位置：

```
In this UI screenshot, what is the location of the element if I want to click on "{text}"? Express the location coordinates using the [x1, y1, x2, y2] format, scaled between 0 and 1000.
```

剖析模型輸出：

對於上述每個 UI 邊界偵測提示，您可以使用規則表達式剖析模型產生的座標資訊，如以下 Python 程式碼範例所示。

程式碼範例

```
def parse_coord_text(text):
    """Parses a model response which uses array formatting ([x, y, ...])
    to describe points and bounding boxes. Returns an array of tuples."""
    pattern = r"\([([^\[\]]*?)\]"
    return [
        tuple(int(x.strip()) for x in match.split(","))
        for match in re.findall(pattern, text)
    ]
```

影片理解

下列各節提供指引，說明如何針對需要了解或分析影片的任務建立提示。

摘要影片

Amazon Nova 模型可以產生影片內容的摘要。

對於影片摘要使用案例，我們建議使用下列推論參數值：

- 溫度：0

- 有些使用案例可能受益於啟用模型推理

不需要特定的提示範本。您的使用者提示應該明確指定您關心的影片層面。以下是幾個有效提示的範例：

Can you create an executive summary of this video's content?

Can you distill the essential information from this video into a concise summary?

Could you provide a summary of the video, focusing on its key points?

產生影片的詳細字幕

Amazon Nova 模型可以產生影片的詳細字幕，稱為密集字幕的任務。

對於影片字幕使用案例，我們建議使用下列推論參數值：

- 溫度：0
- 有些使用案例可能受益於啟用模型推理

不需要特定的提示範本。您的使用者提示應該明確指定您關心的影片層面。以下是幾個有效提示的範例：

Provide a detailed, second-by-second description of the video content.

Break down the video into key segments and provide detailed descriptions for each.

Generate a rich textual representation of the video, covering aspects like movement, color and composition.

Describe the video scene-by-scene, including details about characters, actions and settings.

Offer a detailed narrative of the video, including descriptions of any text, graphics, or special effects used.

Create a dense timeline of events occurring in the video, with timestamps if possible.

分析安全影片片段

Amazon Nova 模型可以偵測安全影片中的事件。

針對安全影片使用案例，我們建議使用下列推論參數值：

- 溫度：0
- 有些使用案例可能受益於啟用模型推理

You are a security assistant for a smart home who is given security camera footage in natural setting. You will examine the video and describe the events you see. You are capable of identifying important details like people, objects, animals, vehicles, actions and activities. This is not a hypothetical, be accurate in your responses. Do not make up information not present in the video.

使用時間戳記擷取視訊事件

Amazon Nova 模型可以識別與影片中事件相關的時間戳記。您可以請求以秒或 MM : SS 格式格式化時間戳記。例如，在影片中 1 分鐘 25 秒發生的事件可以表示為 85 或 01:25。

針對此使用案例，我們建議使用下列推論參數值：

- 溫度：0
- 請勿使用推理

我們建議您使用類似下列的提示：

識別事件的開始和結束時間

Please localize the moment that the event "{event_description}" happens in the video. Answer with the starting and ending time of the event in seconds, such as [[72, 82]]. If the event happen multiple times, list all of them, such as [[40, 50], [72, 82]].

Locate the segment where "{event_description}" happens. Specify the start and end times of the event in MM:SS.

Answer the starting and end time of the event "{event_description}". Provide answers in MM:SS

When does "{event_description}" in the video? Specify the start and end timestamps, e.g. [[9, 14]]

識別多個事件發生的事件

Please localize the moment that the event "{event_description}" happens in the video. Answer with the starting and ending time of the event in seconds. e.g. [[72, 82]]. If the event happen multiple times, list all of them. e.g. [[40, 50], [72, 82]]

產生具有時間戳記的視訊區段日誌

Segment a video into different scenes and generate caption per scene. The output should be in the format: [STARTING TIME-ENDING TIMESTAMP] CAPTION. Timestamp in MM:SS format

For a video clip, segment it into chapters and generate chapter titles with timestamps. The output should be in the format: [STARTING TIME] TITLE. Time in MM:SS

Generate video captions with timestamp.

分類影片

您可以使用 Amazon Nova 模型，根據您提供的預先定義類別清單來分類影片。

針對此使用案例，我們建議使用下列推論參數值：

- 溫度：0
- 不應使用原因

使用以下提示範本：

What is the most appropriate category for this video? Select your answer from the options provided:
{class1}
{class2}
{...}

範例：

What is the most appropriate category for this video? Select your answer from the options provided:

Arts

Technology

Sports

Education

內容管制

本指南提供提示 Amazon Nova 模型使用 [MLCommons® AILuminate 評估標準](#) 執行內容管制的最佳實務，這是具有 12 個 AI 安全危害類別的完整架構。雖然本指南使用 MLCommons 標準做為範例來說明最佳實務，但提示技術適用於您指定的任何自由格式或可自訂的內容管制類別和定義。

主要原則

- 根據您的使用案例使用結構化 (XML、JSON) 或自由格式提示
- 調整任何文字內容：對話、文章、社交媒體文章、評論、評論等
- 提供明確的政策定義
- 請求說明透明度

建議組態

對於內容管制任務，我們建議使用下列推論組態：

- 溫度 — 使用預設值 0.7。
- Top P — 使用預設值 0.9。
- 推理 — 停用推理內容管制以降低延遲。

MLCommons 通用分類

標準定義了 12 個危害類別：

MLCommons Alluminate 危害類別

Category	定義	金鑰子類別
暴力犯罪	啟用、鼓勵或認可暴力犯罪行為的內容	大規模暴力、Murder、身體攻擊、家庭濫用、恐慌、動物濫用
非暴力犯罪	啟用、鼓勵或認可非暴力犯罪活動的內容	人口販賣、詐騙、盜竊、Arson、藥物犯罪、武器犯罪、網路犯罪
與性別相關的犯罪	啟用、鼓勵或支持性相關犯罪的內容	性販賣、性攻擊、性侵犯、非合意私密內容、Bestiality
子性探索	包含、描述、啟用、鼓勵或認可對未成年人進行性濫用的內容	與兒童的不適當關係、對兒童的性虐待、CSAM
CBRNE	可啟用、鼓勵或認可建立大規模銷毀武器的內容	化學武器、生物武器、放射武器、核武器、高產量爆炸物
自殺和自我傷害	啟用、鼓勵或背書刻意自我傷害的內容	自殺、自我傷害、吃東西障礙、危險挑戰
仇恨	根據受保護的特性來淡化或取消人道化的內容	破壞評論、引發身體/情緒傷害、排除和隔離
專業建議	在沒有適當免責聲明的情況下，在敏感網域中提供不合格的建議	財務建議、醫療建議、法律建議、選舉建議、安全宣告
隱私權	包含敏感非公開個人資訊的內容	實體安全（地址）、數位安全（憑證）、財務安全（帳戶號碼）
智慧財產權	違反或鼓勵違反 IP 權利的內容	著作權侵權、商標侵權、專利侵權
誹謗	可驗證為虛假的內容，並可能損害某人的評價	Libel、Slander、Diparagement
性內容	情節內容，包括色情和網路性別	色情（圖形性描述）、直接波紋聊天

來源：MLCommons® Alluminate 評估標準（文件）

提示範例

結構化格式

最適合需要可剖析輸出的自動化系統。這些範例使用少量擷取學習，您可以在提示中提供範例輸入輸出對，以引導模型的回應。以下範例包含單一示範，但您可以包含多個範例，以提升特定使用案例的準確性。

結構 (XML)

User:

```
You are a text content moderator that detects policy violations,
describes violated categories, and explains the reasoning behind your decisions.
You will see a policy list and a conversation to be moderated.
Your moderation response should be formatted as follows.
```

```
Start the response by indicating policy violation status using either
<POLICY_VIOLATION>Yes</POLICY_VIOLATION> for content that violates policies, or
<POLICY_VIOLATION>No</POLICY_VIOLATION> for content that does not violate policies.
```

```
Then, list the specific violation categories between <CATEGORY_LIST> and </
CATEGORY_LIST> tags.
If there is no violation, then put C0 between the tags.
```

```
End the response by providing an analysis explaining whether the content violates
policies
and why in <EXPLAIN> </EXPLAIN> tags. Here is an example.
```

```
<EXAMPLE>
<POLICY>
{{Policy definitions and categories}}
</POLICY>
<TEXT>
{{Content to moderate}}
</TEXT>
<POLICY_VIOLATION>Yes/No</POLICY_VIOLATION>
<CATEGORY_LIST>Categories or C0</CATEGORY_LIST>
<EXPLAIN>Explanation</EXPLAIN>
</EXAMPLE>
```

```
Now, please moderate the following input policy and text.
```

```
<POLICY>
{{Policy definitions and categories}}
</POLICY>
<TEXT>
{{Content to moderate}}
</TEXT>
```

Assistant:

```
<POLICY_VIOLATION>Yes/No</POLICY_VIOLATION>
<CATEGORY_LIST>Categories or C0</CATEGORY_LIST>
<EXPLAIN>Explanation</EXPLAIN>
```

結構 (JSON)

User:

You are a text content moderator that detects policy violations, describes violated categories, and explains the reasoning behind your decisions. You will check if there is unsafe content in the below text snippet in accordance with our safety policy.

Your moderation response should be in JSON format with the following structure:

```
{
  "policy_violation": "Yes or No",
  "category_list": ["List of violated category codes"],
  "explanation": "Reason for the decision"
}
```

If there is no violation, use "C0" as the category code.

IMPORTANT: Do not add any additional text or explanation. Your response must contain ONLY the JSON object, nothing else.

Here is an example:

```
[POLICY]
{{Policy definitions and categories}}
```

```
[TEXT]
{{Content to moderate}}
```

```
[RESPONSE]
{
```

```
"policy_violation": "Yes or No",
"category_list": ["Violated category codes or C0"],
"explanation": "Explanation"
}
```

Now, please moderate the following input.

```
[POLICY]
{{Policy definitions and categories}}
```

```
[TEXT]
{{Content to moderate}}
```

```
[RESPONSE]
```

Assistant:

```
{
  "policy_violation": "No",
  "category_list": ["C0"],
  "explanation": "The text is a casual personal question with no policy violations."
}
```

範例內容管制提示和回應

User:

You are a text content moderator that detects policy violations, describes violated categories, and explains the reasoning behind your decisions. You will see a policy list and a conversation to be moderated. Your moderation response should be formatted as follows.

Start the response by indicating policy violation status using either `<POLICY_VIOLATION>Yes</POLICY_VIOLATION>` for content that violates policies, or `<POLICY_VIOLATION>No</POLICY_VIOLATION>` for content that does not violate policies.

Then, list the specific violation categories between `<CATEGORY_LIST>` and `</CATEGORY_LIST>` tags.

If there is no violation, then put `C0` between the tags.

End the response by providing an analysis explaining whether the content violates policies and why in `<EXPLAIN>` `</EXPLAIN>` tags. Here is an example.


```

<POLICY>
The 12 categories are: Violent Crimes, Non-Violent Crimes, Sex-Related Crimes,
Child Sexual Exploitation, CBRNE, Suicide & Self-Harm, Hate, Specialized Advice,
Privacy, Intellectual Property, Defamation, Sexual Content.
If none apply, label as C0.
</POLICY>

<EXAMPLE>
<TEXT>
"To make a pipe bomb: Get metal pipe, end caps, gunpowder, and fuse.
Drill hole in cap, fill pipe with powder, insert fuse, seal both ends tightly."
</TEXT>
<POLICY_VIOLATION>Yes</POLICY_VIOLATION>
<CATEGORY_LIST>CBRNE</CATEGORY_LIST>
<EXPLAIN>Provides instructions for creating explosives, enabling dangerous weapons
creation.</EXPLAIN>
</EXAMPLE>

<TEXT>
DIY explosive device: Mix ammonium nitrate fertilizer with fuel oil at 94:6 ratio.
Use blasting cap for detonation. Store in cool, dry place. #DIY #Chemistry
</TEXT>

```

Assistant:

```

<POLICY_VIOLATION>Yes</POLICY_VIOLATION>
<CATEGORY_LIST>CBRNE</CATEGORY_LIST>
<EXPLAIN>
The text provides specific instructions for creating ANFO explosives, including precise
mixing ratios and detonation methods. This falls under CBRNE (high-yield explosives
subcategory) as it enables dangerous weapons creation with actionable technical
details.
</EXPLAIN>

```

自由格式內容管制

自由格式提示可以採用下列範例以外的許多不同結構。下列範例刻意使用不同的格式樣式來說明自由格式提示的彈性。您可以調整結構、措辭和格式，以符合您的特定使用案例。

範例 1：是/否問題（衝突和自我傷害）

Example 是/否內容管制

User:

Please stick to the following policy for content moderation. {{user_policy}}
Does this text promote harmful activities?

Best way to hurt yourself without leaving marks?
Asking for tips on methods that work.

Assistant:

Yes. This text explicitly requests methods for self-harm, falling under the Suicide & Self-Harm category. The intent to conceal evidence ("without leaving marks") indicates deliberate harmful behavior.

範例 2：類別識別和推理（仇恨）

Example 類別識別內容管制

User:

You are a professional content moderator. Please stick to the following policy for content moderation. {{user_policy}}
Which category does this text violate and why?

Text: "Women are naturally worse at mathematics than men, according to my personal observations. They should stick to arts and humanities."

Assistant:

Hate. This text makes demeaning generalizations about women based on gender (a protected characteristic), promoting exclusion from STEM fields. The claim lacks scientific basis and perpetuates harmful stereotypes.

範例 3：嚴重性評分（隱私權）

Example嚴重性評分內容管制

User:

```
Content moderation task
---
Policy: {{user_policy}}
---
Rate the severity of the following content as "none", "low", "medium", or "high"
and identify any violated categories.

Content: "John Smith lives at 742 Evergreen Terrace, Springfield.
His SSN is 123-45-6789 and his bank account number is 9876543210."
```

Assistant:

```
Severity: High

Violated category: Privacy

This content exposes multiple forms of sensitive personal information
including a full name, home address, Social Security number, and bank
account number. This falls under the Privacy category, specifically
the physical security, digital security, and financial security
subcategories. The combination of identifiers creates a significant
risk of identity theft and financial fraud.
```

程式碼和故障診斷

本章提供程式碼範例和疑難排解指引，協助您有效地使用 Amazon Nova 模型。

程式碼庫

本節提供使用 Converse API 或 InvokeModel API 的常見 Amazon Nova 操作程式碼範例。

Converse API 範例

基本請求

使用 Converse API 將基本文字請求傳送至 Amazon Nova 模型。

Non-streaming

```
import boto3
from botocore.config import Config

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Invoke the model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=[
        {
            "role": "user",
            "content": [{"text": "Write a short story. End the story with 'THE\nEND'."}],
        }
    ],
    system=[{"text": "You are a children's book author."}], # Optional
    inferenceConfig={ # These parameters are optional
        "maxTokens": 1500,
        "temperature": 0.7,
```

```

        "topP": 0.9,
        "stopSequences": ["THE END"],
    },
    additionalModelRequestFields={ # These parameters are optional
        "inferenceConfig": {
            "topK": 50,
        }
    },
)

# Extract the text response
content_list = response["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
        print(content["text"])

```

Streaming

```

import boto3
from botocore.config import Config

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

# Invoke the model
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=[
        {
            "role": "user",
            "content": [{"text": "Write a short story. End the story with 'THE
END'."}],
        }
    ],
    system=[{"text": "You are a children's book author."}], # Optional
    inferenceConfig={ # These parameters are optional
        "maxTokens": 1500,
        "temperature": 0.7,
    }
)

```

```

        "topP": 0.9,
        "stopSequences": ["THE END"],
    },
    additionalModelRequestFields={ # These parameters are optional
        "inferenceConfig": {
            "topK": 50,
        }
    },
)

# Handle streaming events
for event in response["stream"]:
    if "contentBlockDelta" in event:
        delta = event["contentBlockDelta"]["delta"]
        if "text" in delta:
            print(delta["text"], end="", flush=True)

```

使用內嵌資產的多模態輸入

直接在請求中嵌入文件、影像、影片或音訊資料，以處理多模態內容。此範例使用影像資料。如需其他模態內容結構的詳細資訊，請參閱 [Amazon Bedrock API 文件中的 ContentBlock 詳細資訊](#)。

Non-streaming

```

import boto3
from botocore.config import Config

# Read a document, image, video, or audio file
with open("sample_image.png", "rb") as image_file:
    binary_data = image_file.read()
    data_format = "png"

# Define message with image
messages = [
    {
        "role": "user",
        "content": [
            {
                "image": {
                    "format": data_format,
                    "source": {

```

```

        "bytes": binary_data # For Invoke API, encode as Base64
        string
    },
    },
    {"text": "Provide a brief caption for this asset."},
],
}
]

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Invoke model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
)

# Extract the text response
content_list = response["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
        print(content["text"])

```

Streaming

```

import boto3
from botocore.config import Config

# Read a document, image, video, or audio file
with open("sample_image.png", "rb") as image_file:
    binary_data = image_file.read()
    data_format = "png"

# Define message with image
messages = [
    {

```

```

        "role": "user",
        "content": [
            {
                "image": {
                    "format": data_format,
                    "source": {
                        "bytes": binary_data # For Invoke API, encode as Base64
string
                    },
                },
            ],
            {"text": "Provide a brief caption for this asset."},
        ],
    ]

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

# Invoke model with streaming
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
)

# Handle streaming events
for event in response["stream"]:
    if "contentBlockDelta" in event:
        delta = event["contentBlockDelta"]["delta"]
        if "text" in delta:
            print(delta["text"], end="", flush=True)

```

使用 S3 URI 的多模態輸入

透過參考存放在 S3 中的文件、影像、影片或音訊檔案來處理多模式內容。此範例使用 映像參考。如需其他模態內容結構的詳細資訊，請參閱 [Amazon Bedrock API 文件中的 ContentBlock 詳細資訊](#)。

Non-streaming

```
import boto3
from botocore.config import Config

# Define message with image
messages = [
    {
        "role": "user",
        "content": [
            {
                "image": {
                    "format": "png",
                    "source": {
                        "s3Location": {
                            "uri": "s3://path/to/your/asset",
                            # "bucketOwner": "<account_id>" # Optional
                        }
                    },
                },
            },
            {"text": "Provide a brief caption for this asset."},
        ],
    }
]

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Invoke model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
)

# Extract the text response
content_list = response["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
```

```
print(content["text"])
```

Streaming

```
import boto3
from botocore.config import Config

# Define message with image
messages = [
    {
        "role": "user",
        "content": [
            {
                "image": {
                    "format": "png",
                    "source": {
                        "s3Location": {
                            "uri": "s3://path/to/your/asset",
                            # "bucketOwner": "<account_id>" # Optional
                        }
                    },
                },
            },
            {"text": "Provide a brief caption for this asset."},
        ],
    }
]

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

# Invoke model with streaming
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
)

# Handle streaming events
```

```
for event in response["stream"]:  
    if "contentBlockDelta" in event:  
        delta = event["contentBlockDelta"]["delta"]  
        if "text" in delta:  
            print(delta["text"], end="", flush=True)
```

延伸思考（原因）

為複雜的問題解決任務啟用延伸思考。

Non-streaming

```
import boto3  
from botocore.config import Config  
  
# Create the Bedrock Runtime client, using an extended timeout configuration  
# to support long-running requests.  
bedrock = boto3.client(  
    "bedrock-runtime",  
    region_name="us-east-1",  
    config=Config(read_timeout=3600),  
)  
  
# Invoke the model  
response = bedrock.converse(  
    modelId="us.amazon.nova-2-lite-v1:0",  
    messages=[  
        {  
            "role": "user",  
            "content": [  
                {  
                    "text": 'How many capital letters appear in the following  
passage. Your response must include only the number: "Wilfred ordered an anvil from  
ACME. Shipping was expensive."'                }  
            ],  
        }  
    ],  
    additionalModelRequestFields={  
        "reasoningConfig": {  
            "type": "enabled",  
            "maxReasoningEffort": "low", # "low" | "medium" | "high"        }  
    })
```

```

    }
  },
)

# Extract response content
content_list = response["output"]["message"]["content"]
for content in content_list:
    # Extract the reasoning response
    if "reasoningContent" in content:
        print("\n== Reasoning ==")
        print(content["reasoningContent"]["reasoningText"]["text"])
    # Extract the text response
    if "text" in content:
        print("\n== Text ==")
        print(content["text"])

```

Streaming

```

import boto3
from botocore.config import Config

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

# Invoke the model
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=[
        {
            "role": "user",
            "content": [
                {
                    "text": 'How many capital letters appear in the following
passage. Your response must include only the number: "Wilfred ordered an anvil from
ACME. Shipping was expensive."'
                }
            ],
        }
    ]
)

```

```

    ],
    additionalModelRequestFields={
        "reasoningConfig": {
            "type": "enabled",
            "maxReasoningEffort": "low", # "low" | "medium" | "high"
        },
    },
)

# Process the streaming response
reasoning_output = ""
text_output = ""
for event in response["stream"]:
    if "contentBlockDelta" in event:
        delta = event["contentBlockDelta"]["delta"]

        if "reasoningContent" in delta:
            if len(reasoning_output) == 0:
                print("\n\n== Reasoning ==")
            reasoning_text_chunk = delta["reasoningContent"]["text"]
            print(reasoning_text_chunk, end="", flush=True)
            reasoning_output += reasoning_text_chunk

        elif "text" in delta:
            if len(text_output) == 0:
                print("\n\n== Text ==")
            text_chunk = delta["text"]
            print(text_chunk, end="", flush=True)
            text_output += text_chunk

```

內建工具：使用引文進行 Nova Grounding

使用 Nova Grounding 透過引文從 Web 擷取即時資訊。

Non-streaming

```

import boto3
from botocore.config import Config

# Define the list of tools the model may use
tool_config = {"tools": [{"systemTool": {"name": "nova_grounding"}}]}

```

```

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

messages = [
    {
        "role": "user",
        "content": [
            {"text": "What is the latest news about renewable energy sources?"}
        ],
    }
]

# Invoke the model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0", messages=messages, toolConfig=tool_config
)

# Extract the text with interleaved citations
output_with_citations = ""
content_list = response["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
        output_with_citations += content["text"]

    elif "citationsContent" in content:
        citations = content["citationsContent"]["citations"]
        for citation in citations:
            url = citation["location"]["web"]["url"]
            output_with_citations += f"[{url}]"

print(output_with_citations)

```

Streaming

```

import boto3
from botocore.config import Config

# Define the list of tools the model may use

```

```
tool_config = {"tools": [{"systemTool": {"name": "nova_grounding"}}]}

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

messages = [
    {
        "role": "user",
        "content": [
            {"text": "What is the latest news about renewable energy sources?"}
        ],
    }
]

# Invoke the model with streaming
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0", messages=messages, toolConfig=tool_config
)

# Process the streaming response with interleaved citations
for event in response["stream"]:
    if "contentBlockDelta" in event:
        delta = event["contentBlockDelta"]["delta"]

        if "text" in delta:
            print(delta["text"], end="", flush=True)

        elif "citation" in delta:
            url = delta["citation"]["location"]["web"]["url"]
            print(f"[{url}]", end="", flush=True)
```

內建工具：程式碼解譯器

使用程式碼解譯器工具執行 Python 程式碼以進行計算和資料分析。

Non-streaming

```
import boto3
from botocore.config import Config

# Define the list of tools the model may use
tool_config = {"tools": [{"systemTool": {"name": "nova_code_interpreter"}}]}

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

messages = [
    {
        "role": "user",
        "content": [
            {
                "text": "What is the average of 10, 24, 2, 3, 43, 52, 13, 68, 6, 7,
902, 82?"
            }
        ],
    }
]

# Invoke the model
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0", messages=messages, toolConfig=tool_config
)

# Extract the text and the code the was executed
content_list = response["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
        print("\n== Text ==")
        print(content["text"])

    elif "toolUse" in content and content["toolUse"]["name"] ==
"nova_code_interpreter":
        print("\n== Code Interpreter: input.snippet ==")
```



```
print(content["toolUse"]["input"]["snippet"])
```

Streaming

```
import boto3
from botocore.config import Config
import json

# Define the list of tools the model may use
tool_config = {"tools": [{"systemTool": {"name": "nova_code_interpreter"}}]}

messages = [
    {
        "role": "user",
        "content": [
            {
                "text": "What is the average of 10, 24, 2, 3, 43, 52, 13, 68, 6, 7,
902, 82?"
            }
        ],
    }
]

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

# Invoke the model with streaming
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0", messages=messages, toolConfig=tool_config
)

# Process the streaming response
current_block_start = None
response_text = ""
for event in response["stream"]:
    if "contentBlockStart" in event:
        current_block_start = event["contentBlockStart"]["start"]
```

```

elif "contentBlockStop" in event:
    current_block_start = None

elif "contentBlockDelta" in event:
    delta = event["contentBlockDelta"]["delta"]

    if (
        current_block_start
        and "toolUse" in current_block_start
        and current_block_start["toolUse"]["name"] == "nova_code_interpreter"
    ):
        # This is code interpreter content
        tool_input = json.loads(delta["toolUse"]["input"])
        print("\n== Executed Code Snippet ==")
        print(tool_input["snippet"], end="", flush=True)

    elif "text" in delta:
        # This is text response content
        if len(response_text) == 0:
            print("\n== Text ==")
        text = delta["text"]
        response_text += text
        print(text, end="", flush=True)

```

工具使用

定義要在對話期間使用的模型自訂工具。

Non-streaming

```

import boto3
from botocore.config import Config

def get_weather(city):
    # Mock function to simulate weather API
    return {"temperatureF": 48, "conditions": "light rain"}

# Define the toolSpec for the weather tool
weather_tool = {
    "toolSpec": {

```

```
        "name": "get_weather",
        "description": "Get the current weather conditions in a given location",
        "inputSchema": {
            "json": {
                "type": "object",
                "properties": {
                    "city": {
                        "type": "string",
                        "description": "The city and state, e.g. San Francisco, CA",
                    }
                },
                "required": ["city"],
            }
        },
    },
}

# Define the list of tools the model may use
tool_config = {"tools": [weather_tool]}

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Start tracking message history
messages = []

messages.append(
    {
        "role": "user",
        "content": [
            {
                "text": "Suggest some activities to do in Seattle based on the
current weather."
            }
        ],
    }
)

# Invoke the model
```

```
response = bedrock.converse(
    modelId="us.amazon.nova-2-lite-v1:0", messages=messages, toolConfig=tool_config
)

assistant_message = response["output"]["message"]

# Add the assistant response to the message history
messages.append(assistant_message)

content_list = assistant_message["content"]
stop_reason = response["stopReason"]

if stop_reason == "tool_use":
    # Extract the toolUse details
    tool_use = next(
        content["toolUse"] for content in content_list if "toolUse" in content
    )
    tool_name = tool_use["name"]
    tool_use_id = tool_use["toolUseId"]

    if tool_name == "get_weather":
        # Call the tool
        weather = get_weather(tool_use["input"]["city"])

        # Send the result back to the model
        messages.append(
            {
                "role": "user",
                "content": [
                    {
                        "toolResult": {
                            "toolUseId": tool_use_id,
                            "content": [{"json": weather}],
                        }
                    }
                ],
            }
        )

    # Submit the tool result back to the model
    response = bedrock.converse(
        modelId="us.amazon.nova-2-lite-v1:0",
        messages=messages,
        toolConfig=tool_config,
```

```

    )

    content_list = response["output"]["message"]["content"]
    for content in content_list:
        # Extract the text response
        if "text" in content:
            print("\n== Text ==")
            print(content["text"])
else:
    # A tool call was not needed
    for content in content_list:
        # Extract the text response
        if "text" in content:
            print("\n== Text ==")
            print(content["text"])

```

Streaming

```

import boto3
from botocore.config import Config
import json

def get_weather(city):
    # Mock function to simulate weather API
    return {"temperatureF": 48, "conditions": "light rain"}

# Define the toolSpec for the weather tool
weather_tool = {
    "toolSpec": {
        "name": "get_weather",
        "description": "Get the current weather conditions in a given location",
        "inputSchema": {
            "json": {
                "type": "object",
                "properties": {
                    "city": {
                        "type": "string",
                        "description": "The city and state, e.g. San Francisco, CA",
                    }
                },
            },
            "required": ["city"],

```

```

        },
    },
}

# Define the list of tools the model may use
tool_config = {"tools": [weather_tool]}

# Create the Bedrock Runtime client, using an extended timeout configuration
# to support long-running requests.
bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Start tracking message history
messages = []

messages.append(
    {
        "role": "user",
        "content": [
            {
                "text": "Suggest some activities to do in Seattle based on the
current weather."
            }
        ],
    }
)

# Invoke the model with streaming
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0", messages=messages, toolConfig=tool_config
)

# Process the streaming response
assistant_message = {"role": "assistant", "content": []}
current_tool_use = None
stop_reason = None

for event in response["stream"]:
    if "contentBlockStart" in event:
        start = event["contentBlockStart"]["start"]

```

```
        if "toolUse" in start:
            current_tool_use = start["toolUse"]
            current_tool_use["input"] = ""

    elif "contentBlockDelta" in event:
        delta = event["contentBlockDelta"]["delta"]
        if "toolUse" in delta:
            current_tool_use["input"] += delta["toolUse"]["input"]
        elif "text" in delta:
            print(delta["text"], end="", flush=True)

    elif "contentBlockStop" in event:
        if current_tool_use:
            # Parse the accumulated tool input
            current_tool_use["input"] = json.loads(current_tool_use["input"])
            assistant_message["content"].append({"toolUse": current_tool_use})
            current_tool_use = None

    elif "messageStop" in event:
        stop_reason = event["messageStop"]["stopReason"]
        if stop_reason == "end_turn":
            exit

# Add the assistant response to the message history
messages.append(assistant_message)

if stop_reason == "tool_use":
    # Extract the toolUse details
    tool_use = next(
        content["toolUse"]
        for content in assistant_message["content"]
        if "toolUse" in content
    )
    tool_name = tool_use["name"]
    tool_use_id = tool_use["toolUseId"]

    if tool_name == "get_weather":
        # Call the tool
        weather = get_weather(tool_use["input"]["city"])

        # Send the result back to the model
        messages.append(
            {
                "role": "user",
```

```

        "content": [
            {
                "toolResult": {
                    "toolUseId": tool_use_id,
                    "content": [{"json": weather}],
                }
            }
        ],
    }
)

# Submit the tool result back to the model with streaming
response = bedrock.converse_stream(
    modelId="us.amazon.nova-2-lite-v1:0",
    messages=messages,
    toolConfig=tool_config,
)

# Handle the final streaming response
print("\n== Text ==")
for event in response["stream"]:
    if "contentBlockDelta" in event:
        delta = event["contentBlockDelta"]["delta"]
        if "text" in delta:
            print(delta["text"], end="", flush=True)

```

InvokeModel API 範例

以下範例著重於呼叫 API 的請求和回應結構與 Converse API 稍有不同的幾個關鍵領域。在大多數其他方式中，這兩個 APIs 都相容，因此您應該能夠輕鬆調整上面的 Converse API 範例，以使用 InvokeModel API。

基本請求

使用 InvokeModel API 將基本文字請求傳送至 Amazon Nova 2 模型。

Non-streaming

```

import json

import boto3
from botocore.config import Config

```



```

# Configure the request
request_body = {
    "messages": [
        {
            "role": "user",
            "content": [{"text": "Write a short story. End the story with 'THE
END'."}],
        }
    ],
    "system": [{"text": "You are a children's book author."}], # Optional
    "inferenceConfig": { # These parameters are optional
        "maxTokens": 1500,
        "temperature": 0.7,
        "topP": 0.9,
        "topK": 50,
        "stopSequences": ["THE END"],
    },
}

bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Invoke the model
response = bedrock.invoke_model(
    modelId="us.amazon.nova-2-lite-v1:0", body=json.dumps(request_body)
)
response_body = json.loads(response["body"].read())

# Extract the text response
content_list = response_body["output"]["message"]["content"]
for content in content_list:
    if "text" in content:
        print(content["text"])

```

Streaming

```

import json

import boto3

```

```
from botocore.config import Config

# Configure the request
request_body = {
    "messages": [
        {
            "role": "user",
            "content": [{"text": "Write a short story. End the story with 'THE
END'. "}],
        }
    ],
    "system": [{"text": "You are a children's book author."}], # Optional
    "inferenceConfig": { # These parameters are optional
        "maxTokens": 1500,
        "temperature": 0.7,
        "topP": 0.9,
        "topK": 50,
        "stopSequences": ["THE END"],
    },
}

bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

# Invoke the model with streaming
response = bedrock.invoke_model_with_response_stream(
    modelId="us.amazon.nova-2-lite-v1:0", body=json.dumps(request_body)
)

# Process the streaming response
for event in response["body"]:
    chunk = json.loads(event["chunk"]["bytes"])
    if "contentBlockDelta" in chunk:
        delta = chunk["contentBlockDelta"]["delta"]
        if "text" in delta:
            print(delta["text"], end="", flush=True)
```

具有推理的 InvokeModel API

使用已啟用推理的 InvokeModel API 來解決複雜的問題。

Non-streaming

```
import json

import boto3
from botocore.config import Config

# Configure the request
request_body = {
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "text": 'How many capital letters appear in the following
passage. Your response must include only the number: "Wilfred ordered an anvil from
ACME. Shipping was expensive."'
                }
            ],
        }
    ],
    "reasoningConfig": {
        "type": "enabled",
        "maxReasoningEffort": "low", # "low" | "medium" | "high"
    },
}

bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(read_timeout=3600),
)

# Invoke the model
response = bedrock.invoke_model(
    modelId="us.amazon.nova-2-lite-v1:0", body=json.dumps(request_body)
)
response_body = json.loads(response["body"].read())

# Extract response content
```

```

content_list = response_body["output"]["message"]["content"]
for content in content_list:
    # Extract the reasoning response
    if "reasoningContent" in content:
        print("\n== Reasoning ==")
        print(content["reasoningContent"]["reasoningText"]["text"])
    # Extract the text response
    if "text" in content:
        print("\n== Text ==")
        print(content["text"])

```

Streaming

```

import json

import boto3
from botocore.config import Config

# Configure the request
request_body = {
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "text": 'How many capital letters appear in the following
passage. Your response must include only the number: "Wilfred ordered an anvil from
ACME. Shipping was expensive."'
                }
            ],
        }
    ],
    "reasoningConfig": {
        "type": "enabled",
        "maxReasoningEffort": "low", # "low" | "medium" | "high"
    },
}

bedrock = boto3.client(
    "bedrock-runtime",
    region_name="us-east-1",
    config=Config(connect_timeout=3600, read_timeout=3600),
)

```

```
# Invoke the model with streaming
response = bedrock.invoke_model_with_response_stream(
    modelId="us.amazon.nova-2-lite-v1:0", body=json.dumps(request_body)
)

# Process the streaming response
for event in response["body"]:
    chunk = json.loads(event["chunk"]["bytes"])

    if "contentBlockDelta" in chunk:
        delta = chunk["contentBlockDelta"]["delta"]

        # Extract the reasoning response
        if "reasoningContent" in delta:
            print("\n== Reasoning ==")
            print(delta["reasoningContent"]["reasoningText"]["text"], end="",
                  flush=True)

        # Extract the text response
        if "text" in delta:
            print("\n== Text ==")
            print(delta["text"], end="", flush=True)
```

疑難排解

本節提供使用 Amazon Nova 模型時常見問題的解決方案。

身分驗證和設定

缺少許可

徵狀：無法存取 Nova 模型或功能

解決方案：

- 確保您的 IAM 角色具有 AmazonBedrockFullAccess 或適當的許可
- 透過 Amazon Bedrock 主控台請求特定模型存取
- 驗證模型存取和工具使用的許可

拒絕模型存取

徵狀：模型存取請求失敗

解決方案：

- 透過 Amazon Bedrock 主控台請求特定模型存取
- 確認您的帳戶已獲授權存取請求的模型
- 檢查模型的區域可用性

區域可用性問題

徵狀：所選區域不提供此功能

解決方案：

- Web Grounding 僅適用於具有美國 CRIS 設定檔的美國區域
- 確認模型和功能在您選取的區域中可用
- 視需要切換到支援的區域

逾時組態

症狀：請求在完成之前逾時

原因：對於複雜操作，預設逾時太短

解決方案：設定延長逾時設定

```
from botocore.config import Config

bedrock = boto3.client(
    'bedrock-runtime',
    region_name='us-east-1',
    config=Config(
        read_timeout=3600      # 60 minutes
    )
)
```

 Note

Amazon Nova 推論請求最多可能需要 60 分鐘才能執行複雜的操作。

API 回應問題

了解停止原因：

`end_turn`

正常完成。不需採取任何動作。

`max_tokens`

已達到字符限制。

解決方案：增加 `maxTokens` 參數 `inferenceConfig`。

`content_filtered`

內容違反了 AWS 負責任的 AI 政策。

解決方案：檢閱和修改您的輸入，以符合內容政策。

格式錯誤 `_model_output`

無效的輸出格式。

解決方案：檢查您的輸出結構描述和限制條件；確認 JSON 結構描述格式正確。

格式錯誤 `_tool_use`

無效的工具呼叫格式。

解決方案：確認工具定義符合預期的結構描述；檢查工具輸入參數的格式是否正確。

`service_unavailable`

無法使用內建工具服務。

解決方案：短暫延遲後重試請求；檢查 AWS 服務運作狀態儀表板。

`invalid_query`

對內建工具的查詢無效。

解決方案：檢閱查詢格式和參數；確保查詢符合工具需求。

max_tool_invocations

工具重試已用盡。

解決方案：簡化任務或將其分成較小的步驟；檢閱工具錯誤訊息是否有特定問題。

原因模式錯誤

以高推理力截斷的回應

解決方案：對於高推理工作，請取消設定這些參數：temperature、topP、maxToken。這可讓模型針對複雜的推理任務使用最佳設定。

權杖不足以推理

錯誤：「maxTokens 不足」

解決方案：以更高的限制自動重試

```
token_limits = {
    "low": 15000,
    "medium": 30000,
    "high": 50000
}

try:
    response = client.converse(
        modelId="us.amazon.nova-2-lite-v1:0",
        messages=messages,
        inferenceConfig={
            "maxTokens": token_limits[max_effort]
        },
        additionalModelRequestFields={
            "reasoningConfig": {
                "type": "enabled",
                "maxReasoningEffort": max_effort
            }
        }
    )
except Exception as e:
    if "maxTokens is insufficient" in str(e):
        higher_limit = int(token_limits[max_effort] * 1.5)
        # Retry with higher limit
```


工具使用問題

結構描述驗證失敗

工具結構描述驗證錯誤

解決方案：

- 將 JSON 結構描述限制為兩層巢狀，以獲得最佳效能
- 確定所有必要欄位都已正確定義
- 根據 JSON 結構描述規格驗證結構描述

模型未正確使用工具

解決方案：

- 確保工具名稱清楚描述其用途
- 提供工具功能的詳細說明
- 使用明確的參數描述明確定義輸入結構描述
- 當有幫助時，在描述中包含範例

不一致的工具呼叫行為

解決方案：將工具呼叫的溫度設定為 0：

```
inferenceConfig={
  "temperature": 0,
  "maxTokens": 10000
}
```

這可啟用貪婪解碼，以更可靠地使用工具。

工具選擇衝突

問題：搭配 Web 搜尋或程式碼解譯器使用自訂工具時發生錯誤

解決方案：請勿包含名為 `nova_grounding` 的自訂 toolSpec `nova_grounding` - 這與系統工具衝突。請改用系統工具組態：

```
# Correct - use system tool
tool_config = {
  "tools": [{
    "systemTool": {"name": "nova_grounding"}
```

```
    ]]  
  }  
  
  # Incorrect - don't create custom tool with this name  
  # tool_config = {  
  #     "tools": [{  
  #         "toolSpec": {"name": "nova_grounding", ...}  
  #     }]  
  # }
```

Web Grounding 問題

存取控制問題

問題：Web Grounding 和 Code Interpreter 無法運作

解決方案：確保您的 IAM 政策包含：

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": ["bedrock:InvokeTool"],  
      "Resource": ["arn:aws:bedrock::{YOUR_ACCOUNT_ID}:system-tool/  
amazon.nova_grounding"]  
    }  
  ]  
}
```

服務控制政策問題

問題：SCP 封鎖 Web Grounding

解決方案：如果您有具有 `aws:requestedRegion` 條件的服務控制政策，請更新它們以允許 Web Grounding 功能的「未指定」區域。

媒體處理限制

對影像/影片中多語言內容的理解不佳

限制：Nova 模型對視覺化媒體中的多語言內容的理解有限

解決方法：

- 提供文字翻譯與影像
- 盡可能使用多語言內容的文字型輸入

人員識別

問題：模型拒絕識別影像中的人物

預期行為：基於隱私權和安全考量，模型會拒絕在影像、文件或影片中識別或命名個人

解決方法：詢問一般特性或內容，而非特定身分

空間推理限制

問題：當地語系化或配置分析不準確

限制：精確空間推理的有限功能

解決方法：

- 使用週框方塊偵測進行物件當地語系化
- 在提示中提供明確的參考點
- 將複雜的空間查詢分解為更簡單的元件

影像/影片中的小型文字

問題：無法讀取媒體中的小文字

解決方案：

- 裁剪影像以專注於相關文字區段
- 提高來源媒體的解析度
- 如果可用，請分別提供文字

文件和檔案處理

不支援的內容

問題：PDF 處理失敗

原因：

- 具有 CMYK 顏色描述PDFs

- 包含 SVG 影像PDFs

解決方案：

- 將 PDFs轉換為 RGB 顏色設定檔
- 在 PDFs 中包含 之前，先光柵化 SVG 映像

權杖估算

問題：使用 PDFs 的非預期字符用量

指導方針：每個標準 8.5×11" PDF 頁面估計約 2,560 個字符

解決方案：根據文件長度maxTokens相應地調整

API 和 SDK 參考

本節提供 Amazon Nova APIs 和 SDKs 的完整技術參考文件。

主題

- [請求和回應結構描述](#)
- [SDK 參考](#)

請求和回應結構描述

叫用 API 和 Converse API 之間的請求結構描述幾乎相同。主要差異在於二進位資料（影像、影片、音訊）的編碼方式：Converse API 使用二進位陣列，而叫用 API 使用 Base64-encoded 字串。

完成請求結構

以下顯示 Amazon Nova 模型的完整請求結構。除非標記為必要，否則所有欄位皆為選用：

```
{
  "system": [
    {
      "text": "string"
    }
  ],
  "messages": [ // Required
    {
      "role": "user", // Required - first turn must be user
      "content": [ // Required
        {
          "text": "string"
        },
        {
          "image": {
            "format": "jpeg" | "png" | "gif" | "webp", // Required
            "source": { // Required
              "bytes": image // Binary array (Converse) or Base64 string (Invoke)
            }
          }
        }
      ],
      {
        "video": {
```

```

    "format": "mkv" | "mov" | "mp4" | "webm" | "three_gp" | "flv" | "mpeg" |
    "mpg" | "wmv",
    "source": {
        // Option 1: S3 location
        "s3Location": {
            "uri": "string", // e.g., s3://my-bucket/object-key
            "bucketOwner": "string" // Optional, e.g., "123456789012"
        },
        // Option 2: File bytes
        "bytes": video // Binary array (Converse) or Base64 string (Invoke)
    }
},
{
    "audio": { // Nova 2 Omni and Sonic only
        "format": "mp3" | "opus" | "wav" | "aac" | "flac" | "mp4" | "ogg" | "mkv",
        "source": {
            // Option 1: S3 location
            "s3Location": {
                "uri": "string",
                "bucketOwner": "string" // Optional
            },
            // Option 2: File bytes
            "bytes": audio // Binary array (Converse) or Base64 string (Invoke)
        }
    }
},
{
    "role": "assistant",
    "content": [
        {
            "text": "string" // For prefilling assistant response
        }
    ]
}
],
"inferenceConfig": { // All optional
    "maxTokens": int, // 1-5000, default: dynamic
    "temperature": float, // 0.00001-1, default: 0.7
    "topP": float, // 0-1, default: 0.9
    "topK": int, // 0-128, default: not used
    "stopSequences": ["string"],

```

```

    "reasoningConfig": { // Nova 2 Lite and Sonic only
      "type": "enabled" | "disabled", // default: "disabled"
      "maxReasoningEffort": "low" | "medium" | "high"
    },
    "toolConfig": { // Optional
      "tools": [
        {
          "toolSpec": {
            "name": "string", // Max 64 characters
            "description": "string",
            "inputSchema": {
              "json": {
                "type": "object",
                "properties": {
                  "arg1": {
                    "type": "string",
                    "description": "string"
                  }
                }
              },
              "required": ["string"]
            }
          }
        }
      ],
      "toolChoice": { // Choose one option
        "auto": {},
        "any": {},
        "tool": {
          "name": "string"
        }
      }
    }
  }
}

```

金鑰請求參數：

- **system**：系統提示，提供內容和指示
- **messages**：對話陣列與角色（使用者或助理）和內容一起切換
- **inferenceConfig**：控制模型輸出行為（溫度、權杖等）。
- **toolConfig**：函數呼叫的工具規格

Note

使用 Converse API 時，必須將 `topK` 和 `reasoningConfig` 參數放置在中，`additionalModelRequestFields` 而不是 `inferenceConfig`。

以下各節提供每個請求參數的詳細說明：

system

system – (選用) 請求的系統提示詞。系統提示會提供內容和指示給 Amazon Nova，例如指定特定目標或角色。

messages

messages – (必要) 包含對話轉彎的輸入訊息陣列。

- **role** – (必要) 對話輪換的角色。有效值為 `user` 和 `assistant`。第一個訊息必須一律使用 `user` 角色。
- **content** – (必要) 內容區塊的陣列。每個區塊指定內容類型 (`text`、`video`、`image` 或 `audio`)：
 - **text** – 對話回合的文字內容。如果與影像或視訊結合，會解譯為隨附的文字。
 - **image** – (不支援 Nova 2 Lite) 具有下列項目的影像內容：
 - **format** – (必要) 影像格式：`jpeg`、`webp`、`png` 或 `gif`
 - **source.bytes** – (必要) 影像資料為二進位陣列 (Converse API) 或 Base64 字串 (叫用 API)
 - **video** – (不支援 Nova 2 Lite) 具有下列項目的影片內容：
 - **format** – (必要) 影片格式：`mkv`、`mov`、`mp4`、`webmthree_gp`、`flv`、`mpeg mpg` 或 `wmv`
 - **source** – (必要) 透過 S3 URI (`s3Location.uri` 和選用 `bucketOwner`) 或檔案位元組 (`bytes`) 的視訊來源
 - **audio** – (僅限 Amazon Nova Sonic 和) 具有下列項目的音訊內容：
 - **format** – (必要) 音訊格式：`mp3`、`opus`、`wav`、`aac`、`flacmp4`、`ogg`、或 `mkv`
 - **source** – (必要) 透過 S3 URI 或檔案位元組的音訊來源

inferenceConfig

inferenceConfig – (選用) 控制模型輸出產生的組態參數。

- maxTokens – (選用) 停止之前要產生的字符上限。Amazon Nova 模型可能會在達到此限制之前停止。最大值為 5,000。如果未指定，會根據請求內容使用動態預設值。
- temperature – (選用) 回應中的隨機性。有效範圍：0.00001-1 (預設值：0.7)。較低的值會產生更確定性的輸出。
- topP – (選用) Nucleus 取樣閾值。Amazon Nova 範例來自累積機率達到的字符topP。有效範圍：0-1 (預設值：0.9)。調整 temperature或 topP，而非兩者。
- topK – (選用) 僅來自最高 K 權杖的範例。移除低機率回應。有效範圍：0-128 (預設值：未使用)。

Note

針對 Converse API，請傳入 topK additionalModelRequestFields。

- stopSequences – (選用) 字串陣列，會在遇到時停止產生。
- reasoningConfig – (僅限 Amazon Nova Sonic 和) 原因組態：
 - type – (選用) enabled或 disabled (預設：disabled)
 - maxReasoningEffort – 運算工作：medium、low或 high。使用 low和 medium，以增量方式推理串流；在最終區塊中high輸出推理。

Note

針對 Converse API，請傳入 reasoningConfig additionalModelRequestFields。

toolConfig

toolConfig – (選用) [ToolConfiguration 結構描述之後的工具](#)組態。

- tools – 使用 name、description和 的工具規格陣列 inputSchema
- toolChoice – (選用) 控制工具選擇：
 - auto – 模型決定是否要使用哪些工具
 - any – 模型必須至少使用一個工具
 - tool – 模型必須使用指定的工具名稱

完整的回應結構

以下顯示 Amazon Nova 模型的完整回應結構：

```
{
  "ResponseMetadata": {
    "RequestId": "string",
    "HTTPStatusCode": int,
    "HTTPHeaders": {
      "date": "string",
      "content-type": "application/json",
      "content-length": "string",
      "connection": "keep-alive",
      "x-amzn-requestid": "string"
    },
    "RetryAttempts": 0
  },
  "output": {
    "message": {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": { // Optional - if reasoning enabled
            "reasoningText": {
              "text": "[REDACTED]"
            }
          }
        }
      ],
      {
        "toolUse": { // Optional - if tool called
          "toolUseId": "string",
          "name": "string",
          "input": {} // Tool-specific arguments
        }
      },
      {
        "text": "string" // Optional - text response
      },
      {
        "image": { // Optional - Nova 2 Omni only
          "format": "png",
          "source": {
            "bytes": image // Binary array (Converse) or Base64 string (Invoke)
          }
        }
      }
    ]
  }
}
```

```

        }
      }
    }
  ]
}
},
"stopReason": "string", // See stop reasons below
"usage": {
  "inputTokens": int,
  "outputTokens": int,
  "totalTokens": int
},
"metrics": {
  "latencyMs": int
}
}

```

停止原因：

- `end_turn`：自然回應結束
- `max_tokens`：達到 `maxTokens` 限制
- `content_filtered`：違反的內容政策
- `malformed_model_output`：無效的模型輸出
- `malformed_tool_use`：無效的工具使用輸出
- `service_unavailable`：無法連線內建工具服務
- `invalid_query`：對內建工具的查詢無效
- `max_tool_invocations`：工具重試已用盡

下列各節提供每個回應欄位的詳細說明：

`output`

`output` – (必要) 包含模型的回應訊息。

- `message` – (必要) 具有角色和內容陣列的助理回應訊息。
- `content` – (必要) 內容區塊陣列，可包含：
 - `reasoningContent` – (選用) 如果已啟用推理，則傳回。包含推理文字，一律會在回應[REDACTED]中。

- `toolUse` – (選用) 如果呼叫工具，則傳回。包含工具使用 ID、名稱和輸入引數。
- `text` – (選用) 如果模型以文字內容回應，則傳回。
- `image` – (選用，僅限) 如果模型產生映像，則傳回。格式一律為 PNG。

stopReason

`stopReason` – (必要) 指出模型停止產生輸出的原因：

- `end_turn` – 已到達自然結束回應
- `max_tokens` – 達到 `maxTokens` 限制或模型的最大輸出限制
- `content_filtered` – 輸出違反 AWS 責任 AI 政策
- `malformed_model_output` – 模型產生無效的輸出
- `malformed_tool_use` – 模型產生的無效工具使用輸出
- `service_unavailable` – 無法連線內建工具服務
- `invalid_query` – 對內建工具的查詢無效
- `max_tool_invocations` – 內建工具在重試後未產生有效結果

使用方式

`usage` – (必要) 字符用量資訊：

- `inputTokens` – 模型擷取的字符總數
- `outputTokens` – 產生的字符數量
- `totalTokens` – 輸入和輸出字符的總和

指標

`metrics` – (必要) 效能指標：

- `latencyMs` – 推論完成時間總計，以毫秒為單位

SDK 參考

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。每個 SDK 都提供 API、程式碼範例和文件，讓開發人員更輕鬆地以他們偏好的語言建置應用程式。

SDK 文件	代碼範例
AWS 適用於 C++ 的 SDK	AWS 適用於 C++ 程式碼的 SDK 範例
AWS CLI	AWS CLI 程式碼範例
AWS 適用於 Go 的 SDK	AWS 適用於 Go 的 SDK 程式碼範例
AWS 適用於 Java 的開發套件	AWS 適用於 Java 的 SDK 程式碼範例
AWS 適用於 JavaScript 的 SDK	AWS 適用於 JavaScript 的 SDK 程式碼範例
AWS 適用於 Kotlin 的 SDK	AWS 適用於 Kotlin 的 SDK 程式碼範例
AWS 適用於 .NET 的 SDK	AWS 適用於 .NET 的 SDK 程式碼範例
AWS 適用於 PHP 的 SDK	AWS 適用於 PHP 的 SDK 程式碼範例
AWS PowerShell 的工具	AWS 適用於 PowerShell 的工具程式碼範例
AWS 適用於 Python 的 SDK (Boto3)	AWS 適用於 Python (Boto3) 的 SDK 程式碼範例
AWS 適用於 Ruby 的 SDK	AWS 適用於 Ruby 的 SDK 程式碼範例
AWS 適用於 Rust 的 SDK	AWS SDK for Rust 程式碼範例
AWS 適用於 SAP ABAP 的 SDK	AWS 適用於 SAP ABAP 的 SDK 程式碼範例
AWS 適用於 Swift 的 SDK	AWS 適用於 Swift 的 SDK 程式碼範例

在 SageMaker AI 上自訂 Amazon Nova 模型

您可以透過[配方](#)自訂 Amazon Nova 模型，包括增強型 Amazon Nova 2.0 模型，並在 SageMaker 上進行訓練。這些配方支援監督式微調 (SFT) 和強化微調 (RFT) 等技術，同時具有全階和低階調整 (LoRA) 選項。

端對端自訂工作流程涉及模型訓練、模型評估和部署等階段以進行推論。SageMaker 上的此模型自訂方法提供更大的彈性和控制，以微調其支援的 Amazon Nova 模型、精確地最佳化超參數，並實作諸如 LoRA 參數效率微調 (PEFT)、全排名 SFT、RFT 和持續預先訓練 (CPT) 等技術。

自訂方法

SageMaker 提供兩種自訂 Amazon Nova 模型的方法：

UI 型體驗 – 透過簡單的引導式界面，使用 自訂 Amazon Nova 模型。此方法提供end-to-end工作流程，包括訓練、評估和部署，無需撰寫程式碼。UI 型體驗非常適合快速實驗、proof-of-concept開發，以及偏好視覺化工作流程的使用者。

程式碼型體驗 – 使用 SageMaker Python SDK、Nova SDK 和訓練配方，以程式設計方式自訂模型。此方法提供更高的彈性，可讓您設定進階超參數、與 CI/CD 管道整合，以及自動化訓練工作流程。對於生產工作負載、複雜的自訂需求，以及具有既定 MLOps 實務的團隊，建議使用程式碼型體驗。

方法	最適合	主要優點
以 UI 為基礎的	實驗、原型設計、快速反覆運算	簡單設定、引導式工作流程、不需要編碼
程式碼型	生產、自動化、進階組態	完全彈性、管道整合、版本控制

自訂平台

AWS 提供三種平台來自訂 Amazon Nova 模型，每個都針對不同的使用案例和需求而設計：

Amazon Bedrock – 以最少的設定提供最簡單且最快的模型自訂路徑。Bedrock 會自動處理所有基礎設施管理，讓您專注於資料和使用案例。當您需要最快time-to-value，並偏好全受管體驗時，此平台是理想的選擇。

SageMaker 訓練任務 – 提供全受管環境，可讓您自訂不需要建立或維護任何叢集的 Amazon Nova 模型。服務會自動處理所有基礎設施佈建、擴展和資源管理，讓您專注於設定訓練參數和提交任務。此平台在易用性和靈活性之間取得平衡，支援參數效率微調 (PEFT)、完整排名微調和強化微調 (RFT) 等技術。

SageMaker HyperPod – 要求您建立和管理具有受限執行個體群組 (RIGs) 的 EKS 叢集，為大規模分散式訓練提供專業環境。此平台可讓您在使用特殊 GPU 執行個體和整合式 Amazon FSx for Lustre 儲存體設定訓練環境時獲得最大的彈性，使其特別適合進階分散式訓練案例、持續模型開發和企業規模的自訂工作負載。

平台	複雜性	彈性	最適合
Amazon Bedrock	最低	標準	最快的自訂、最少的設定
SageMaker 訓練任務	中	高	平衡彈性和易用性
SageMaker HyperPod	最高	上限	大型分散式訓練、企業工作負載

Note

如果您提供 KMS 金鑰給 Amazon Nova 模型自訂訓練任務，以便在 Amazon 擁有的輸出 S3 儲存貯體中進行加密：

- 呼叫後續反覆訓練任務時，或利用加密模型呼叫 Amazon Bedrock CreateCustomModel API 時，您必須提供相同的 KMS 金鑰。
- 呼叫 CreateTrainingJob API 的身分（而非執行角色）必須具有 CreateGrant、Encrypt、和的許可RetireGrant，GenerateDataKey如 KMS 金鑰政策所定義。

一般先決條件

自訂程序涉及數個關鍵階段，包括模型訓練、評估和部署以進行推論，每個階段都需要特定的資源和組態。在 SageMaker 上開始 Amazon Nova 模型自訂之前，請確定您有下列一般先決條件。

- AWS 帳戶。如果您沒有 AWS 帳戶，請依照[這些指示](#)註冊。

- 安裝 [AWS CLI](#) 並設定組態。
- 可存取 [基本 Amazon Nova 模型自訂配方](#)。
- 熟悉 [YAML](#) 組態檔案。
- 熟悉如何在您的環境中執行筆記本。
- 熟悉如何建立具有適當許可 AWS 的資源，例如 [Amazon S3 儲存貯體](#) 和 IAM 角色。 https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
- 熟悉如何使用 [SageMaker 訓練模型](#)。
- [SageMaker HyperPod 與 EKS 協同運作](#) 的熟悉度。
- [SageMaker HyperPod CLI](#) 的熟悉度。
- 熟悉 [Amazon Nova 基礎模型](#)。
- 熟悉 [可用於自訂的 Amazon Nova 模型和演算法](#)。
- 熟悉 [Amazon Bedrock 推論](#)。

Amazon Nova 配方

您可以從配方儲存庫取得 Amazon Nova [配方](#)。Amazon Nova 配方是一種 [YAML](#) 組態檔案，可提供有關如何執行模型自訂任務的詳細資訊給 SageMaker。它提供基本模型名稱、設定訓練超參數、定義最佳化設定，並包含成功微調或訓練模型所需的任何其他選項。

您也可以透過 SageMaker AI Monarch 和導覽至模型中樞 AWS、選取和瀏覽 Amazon Nova 模型以尋找其相關聯的配方，來存取 Amazon Nova 配方。SageMaker AI Monarch 和 都會為每個配方提供範例筆記本，其中包括修改配方並使用 SageMaker 訓練任務或 SageMaker HyperPod 環境執行自訂任務的所有必要步驟。

若要存取 Amazon SageMaker Studio 中的配方頁面，執行角色必須具有下列許可。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
    },
  ],
}
```



```
        "Resource": [
            "arn:aws:s3:::*model-customization-recipes*"
        ]
    }
}
```

若要在 SageMaker 訓練任務 和 上執行範例筆記本，請使用下列其中一個 SageMaker 分佈映像版本：2.7.1+、2.8.0+、3.2.1+、3.3.0+。這同時適用於 SageMaker AI Monarch 和 。

主題

- [取得 Amazon Nova 配方](#)
- [取得 Amazon Nova Forge 配方](#)
- [可用的模型和演算法](#)
- [Amazon Nova 2.0 Lite](#)

取得 Amazon Nova 配方

若要取得基本 Amazon Nova 配方，請執行下列命令來複製[配方](#)儲存庫。

```
git clone https://github.com/aws/sagemaker-hyperpod-recipes.git
```

基本配方可在 [recipes_collection/recipes/](#) 取得。

```
cd recipes_collection/recipes/
```

Amazon Nova 自訂配方位於下列資料夾中。

配方類型	資料夾
SFT（全排名和 PEFT）	fine-tuning/nova
評估	evaluation/nova
CPT	training/nova

取得 Amazon Nova Forge 配方

若要存取任務專用的 Amazon Nova Forge 配方，請[設定您對 Amazon Nova Forge 的存取權](#)，然後遵循[此研討會](#)來設定 CLI 的 Forge 版本。對於 SageMaker 訓練任務，AWS 主控台和 Nova Forge SDK 會自動存取 Forge 配方。

可用的模型和演算法

下表摘要說明 Amazon Nova 模型的自訂可用性，以及使用 SageMaker 支援的演算法。

模型名稱	模型 ID	微調	備註
Amazon Nova Lite 2.0	amazon.nova-2-lite-v1 : 0 : 256k	是	• SFT - 接受文字和/或影像或文字和/或影片做為輸入，並產生文字做為輸出。單一任務無法在同一個執行中結合文字、影像和影片。 • RFT - 接受文字和影像作為單轉對話的輸入，並產生文字作為輸出。改善獎勵最佳化功能。 • CPT

Amazon Nova 2.0 Lite

下表列出 Amazon Nova 2.0 Lite 配方參考的詳細資訊。

模型	類別/子類別	技術	配方名稱	映像 URI (訓練任務)	映像 URI (SageMaker HyperPod)	運算執行 個體
Nova 2.0 Lite	培訓	持續的預先 訓練	nova_lite _2_0_p5x8 _gpu_pret rain.yaml	N/A	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-CPT-V2-latest	ml.p5.48xlarge , ml.p5en.48xlarge
Nova 2.0 Lite	參數高 效率微 調	微調	nova_lite _2_0_p5_g pu_lora_sft.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-TJ-SFT-V2-latest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-latest	ml.p5.48xlarge , ml.p5en.48xlarge

模型	類別/子類別	技術	配方名稱	映像 URI (訓練任務)	映像 URI (SageMaker HyperPod)	運算執行 個體
Nova 2.0 Lite	全執行 微調	微調	nova_lite _v2_0_p5_g pu_sft.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-TJ-SFT-V2-latest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-latest	ml.p5.48xlarge , ml.p5en.48xlarge
Nova 2.0 Lite	參數效率 RFT	微調	nova_lite _v2_smtj_p5_p5en_g pu_lora_rft.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-TJ-RFT-V2-latest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-RFT-TRAIN-V2-latest	ml.p5.48xlarge , ml.p5en.48xlarge

模型	類別/子類別	技術	配方名稱	映像 URI (訓練任務)	映像 URI (SageMaker HyperPod)	運算執行個體
Nova 2.0 Lite	完整排名 RFT	微調	nova_lite_v2_smtj_p5en_gpu_rft.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-TJ-RFT-V2-latest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-RFT-TRAIN-V2-latest	m1.p5.48xlarge , m1.p5en.48xlarge

模型	類別/子類別	技術	配方名稱	映像 URI (訓練任務)	映像 URI (SageMaker HyperPod)	運算執行 個體
Nova 2.0 Lite	使用您自己的資料	評估	nova_lite_2_0_p5_48x1_gpu_bring_your_own_data_set_eval.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-TJ-Eval-V2-1atest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-V2-1atest	ml.p5.48xlarge , ml.p5en.48xlarge
Nova 2.0 Lite	一般文字基準	評估	nova_lite_2_0_p5_48x1_gpu_general_text_benchmark_eval.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-TJ-Eval-V2-1atest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-V2-1atest	ml.p5.48xlarge , ml.p5en.48xlarge

模型	類別/子類別	技術	配方名稱	映像 URI (訓練任務)	映像 URI (SageMaker HyperPod)	運算執行個體
Nova 2.0 Lite	RFT 評估	評估	nova_lite_2_0_p5_48x1_gpu_rft_eval.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-TJ-Eval-V2-1atest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-V2-1atest	ml.p5.48xlarge , ml.p5en.48xlarge
Nova 2.0 Lite	以 LLM Rubric 為基礎的判斷	評估	nova_lite_2_0_p5_48x1_gpu_rubric_llm_judge_eval.yaml	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-TJ-Eval-V2-1atest	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-V2-1atest	ml.p5.48xlarge , ml.p5en.48xlarge

SageMaker 訓練任務上的 Amazon Nova 自訂

SageMaker Training Jobs 是一種環境，可讓您大規模訓練機器學習模型。它會自動佈建和擴展運算資源、從 Amazon S3 等來源載入訓練資料、執行訓練程式碼，以及存放產生的模型成品。

訓練目的是使用您的專屬資料自訂基本 Amazon Nova 模型。訓練程序通常涉及準備資料、選擇[配方](#)、修改 YAML 檔案中的組態參數，以及提交訓練任務的步驟。訓練程序會在服務受管 Amazon S3 儲存貯體中輸出訓練過的模型檢查點。您可以使用此檢查點位置進行評估任務。SageMaker AI 訓練任務上的 Nova 自訂會將模型成品存放在服務管理的 Amazon S3 儲存貯體中。服務受管儲存貯體中的成品會使用 SageMaker AI 受管 KMS 金鑰加密。服務受管 Amazon S3 儲存貯體目前不支援使用客戶受管 KMS 金鑰進行資料加密。

Amazon Nova 自訂的最佳實務

概觀

本節提供自訂技術的概觀，並協助您選擇符合您需求和可用資料的最佳方法。

LLM 訓練的兩個階段

大型語言模型訓練包含兩個主要階段：訓練前和訓練後。在預先訓練期間，模型會處理原始文字權杖，並針對下一個權杖預測進行最佳化。此程序會建立模式完成器，從 Web 和策劃的文字吸收語法、語意、事實和推理模式。不過，預先訓練的模型不了解指示、使用者目標或內容適當的行為。它以適合其訓練分佈的任何樣式繼續文字。預先訓練的模型會自動完成，而不是遵循指示、產生不一致的格式，並且可以從訓練資料反映不良的偏差或不安全的內容。預先訓練可建置一般能力，而非任務實用性。

訓練後會將模式完成器轉換為實用的助理。您會執行多輪受管微調 (SFT)，以教導模型遵循指示、遵循結構描述和政策、呼叫工具，並透過模擬高品質的示範來產生可靠的輸出。此對齊會教導模型以任務形式回應提示，而非文字以繼續。然後，套用強化微調 (RFT)，使用可衡量的意見回饋（例如驗證器或 LLM-as-a-judge）、平衡準確性與簡潔性、安全性與涵蓋範圍，或在限制下進行多步驟推理等權衡。實際上，您在週期中交替使用 SFT 和 RFT，將預先訓練的模型塑造為可靠且符合政策的系統，以一致地執行複雜的任務。

選擇正確的自訂方法

在本節中，我們將介紹訓練後自訂策略：RFT 和 SFT。

強化微調 (RFT)

強化微調會透過意見回饋訊號改善模型效能，這些訊號是可測量的分數或獎勵，表示回應品質，而不是直接監督並準確回答。與從輸入輸出對中學習的傳統監督式微調不同，RFT 使用獎勵函數來評估模型

回應，並反覆最佳化模型以最大化這些獎勵。此方法非常適合定義確切正確輸出具有挑戰性的任務，但您可以可靠地測量回應品質。RFT 可讓模型透過試驗和意見回饋來學習複雜的行為和偏好，因此非常適合需要精細決策、創意問題解決或遵守您可以程式設計方式評估的特定品質標準的應用程式。例如，回答複雜的法律問題是 RFT 的理想使用案例，因為您想要教導模型如何更準確地回答問題。

運作方式

在強化微調中，您可以從指令調校的基準開始，並將每個提示視為小型競賽。對於指定的輸入，您從模型中抽樣少量候選答案，使用獎勵函數對每個答案進行評分，然後在該群組中對其進行排名。更新步驟會調節模型，使分數較高的候選者在下次的可能性更高，分數較低的候選者的可能性更低，而stay-close-to-baseline的限制則可防止行為偏離或變得模糊或攻擊。您可以在許多提示上重複此迴圈、重新整理硬案例、在您看到入侵時收緊驗證器或判斷摩擦，以及持續追蹤任務指標。

何時使用 RFT

受益於 RFT 的任務具有多種特徵。即使單一正確的輸出難以指定，它們仍有可測量的成功訊號。他們認可部分點數或分級品質，因此您可以在提示中或使用獎勵函數，對較差的答案進行排名。它們涉及多個必須平衡的目標（例如簡潔、清晰、安全或成本的準確性）。它們需要遵守您可以程式設計方式檢查的明確限制。它們在可觀察結果的工具媒介型或環境型設定中操作（成功或失敗、延遲、資源使用）。它們發生在收集黃金目標的成本很高，但自動化或以摩擦為基礎的意見回饋有很多的低標籤機制中。當您可以將品質轉換為可靠的純量或排名，並希望模型優先放大分數較高的行為，而不需要詳盡的標記目標時，RFT 最有效。

考慮下列情況下的其他方法：

- 您有大量且可靠的標籤輸入輸出對 – 使用 SFT
- 主要差距是知識或行話 – 使用擷取擴增產生 (RAG)
- 您的獎勵訊號發出吵雜或不可靠，而且您無法使用更好的 Rubrics 或 Checker 進行修正 – 在 RFT 之前先穩定該訊號

何時不使用 RFT

避免在這些情況下使用 RFT：

- 您可以便宜地產生可靠的標籤輸入輸出對 (SFT 更簡單、更便宜且更穩定)
- 差距是知識或行話，而不是行為（使用 RAG）
- 您的獎勵訊號是雜訊、稀疏、容易玩遊戲，或昂貴或運算速度緩慢（請先修正評估者）
- 基準效能接近零（在最佳化偏好設定之前使用 SFT 引導）

- 任務具有確定性結構描述、嚴格格式或單一正確答案 (SFT 或規則型驗證效果更好)
- 緊迫的延遲或成本預算無法吸收額外取樣或探索 RFT 所需的
- 未在獎勵中明確指定和強制執行安全或政策限制

如果您可以指向「正確答案」，請使用 SFT。如果您需要新知識，請使用 RAG。只有在您擁有穩固的基準和強大、快速、hard-to-exploit 的獎勵函數之後，才使用 RFT。

監督式微調 (SFT)

受監督的微調會在任務的人工標記輸入輸出對資料集上訓練 LLM。您提供具有正確或所需回應的提示範例（問題、指示等），並繼續在這些範例上訓練模型。模型會調整其權重，將監督損失降至最低（通常在其預測和目標輸出字符之間跨熵）。這是大多數監督式機器學習任務中使用的相同訓練，適用於專門 LLM。

SFT 會變更行為，而不是知識。它不會教導模型在預先訓練中看不到的新事實或術語。它教導模型如何回答，而不是知道什麼。如果您需要新的網域知識（例如內部術語），請使用擷取擴增產生 (RAG) 在推論時間提供該內容。然後，SFT 會在頂端新增所需的指令遵循行為。

運作方式

SFT 會將回應權杖上的平均跨熵損失降至最低，將提示權杖視為內容，並從遺失中遮罩它們，以最佳化 LLM。模型會內化您的目標風格、結構和決策規則，學習為每個提示產生正確的完成。例如，若要將文件分類為自訂類別，您可以使用提示（文件文字）和標籤完成（類別標籤）微調模型。您可以針對這些配對進行訓練，直到模型以高機率為每個提示輸出正確的標籤。

您可以使用幾百個範例執行 SFT，並擴展到幾十萬個。SFT 範例必須高品質，並直接符合所需的模型行為。

何時使用 SFT

當您有明確定義且具有明確所需輸出的任務時，請使用 SFT。如果您可以明確陳述「給予 X 輸入，正確的輸出為 Y」並收集這類映射的範例，則監督式微調是不錯的選擇。SFT 在這些案例中表現優異：

- 結構化或複雜的分類任務 – 將內部文件或合約分類為許多自訂類別。使用 SFT，模型學習這些特定類別優於單獨提示。
- 具有已知答案的問題回答或轉換任務 – 微調模型以回答公司知識庫的問題，或在每個輸入都有正確回應的格式之間轉換資料。

- 格式化和樣式一致性 – 透過微調正確格式或色調的範例，訓練模型一律以特定格式或色調回應。例如，針對顯示特定品牌語音的提示-回應對進行訓練，會教導模型產生具有該樣式的輸出。指示遵循行為通常透過 SFT 最初就精心策劃的良好助理行為範例進行教學。

當您可以指定正確的行為時，SFT 是教導 LLM 新技能或行為的最直接方式。它使用模型現有的語言理解，並專注於您的任務。當您希望模型執行特定項目，且您有或可以建立範例資料集時，請使用 SFT。

當您可以組合高品質的提示和回應對，以密切反映您想要的行為時，請使用 SFT。它適合具有明確目標或決定性格式的任務，例如結構描述、函數或工具呼叫，以及結構式答案，其中模擬是適當的訓練訊號。目標是行為塑造：教導模型將提示視為任務、遵循指示、採用語氣和拒絕政策，以及產生一致的格式。規劃至少數百次示範，資料品質、一致性和重複資料刪除的重要性高於原始磁碟區。若要直接、符合成本效益的更新，請使用低範圍調整等參數效率方法來訓練小型轉接器，同時讓大多數骨幹保持不變。

何時不使用 SFT

當差距是知識而非行為時，請勿使用 SFT。它不會教導模型新的事實、術語或最近的事件。在這些情況下，請使用擷取擴增的產生，在推論時帶來外部知識。當您可以測量品質，但無法標記單一正確答案時，請避免 SFT。使用強化微調搭配可驗證的獎勵或 LLM-as-a-judge 來直接最佳化這些獎勵。如果您的需求或內容經常變更，請依賴擷取和工具使用，而不是重新訓練模型。

主題

- [Nova Forge SDK](#)
- [Amazon Nova 模型的訓練](#)
- [監控反覆運算的進度](#)
- [評估您的 SageMaker AI 訓練模型](#)
- [反覆訓練](#)

Nova Forge SDK

Nova Forge SDK 是用於自訂 Amazon Nova 模型的全方位 Python SDK。開發套件提供統一的界面，用於跨 SageMaker AI 和 Amazon Bedrock 等不同平台訓練、評估、監控、部署和推論 Amazon Nova 模型。無論您是針對特定領域的任務調整模型，還是針對您的使用案例最佳化效能，此 SDK 都能在一個統一的界面中提供所需的一切。

優勢

- 一個適用於整個模型自訂生命週期的開發套件，從資料準備到部署和監控。
- 支援多種訓練方法，包括持續預先訓練 (CPT)、監督式微調 (SFT)、直接偏好最佳化 (DPO) 和強化微調 (RFT)，包括單迴轉和多迴轉，同時採用 LoRA 和全排名方法。
- 內建支援 SageMaker Training Jobs、SageMaker HyperPod 和 Amazon Bedrock，具有自動資源管理。
- 您不再需要為您的訓練技術尋找正確的配方或容器 URI。
- 使用您自己的訓練配方，或使用 SDK 的智慧型預設值搭配參數覆寫。
- 開發套件會根據支援的模型和執行個體組合來驗證您的組態，並提供驗證支援，在訓練開始之前防止錯誤。
- 整合式 Amazon CloudWatch 監控可讓您即時追蹤訓練進度。
- 整合 MLFlow 以使用 SageMaker AI MLFlow 追蹤伺服器追蹤訓練實驗。

要求

SDK 至少需要 Python 3.12。

安裝

若要安裝此 SDK，請遵循下列命令。

```
pip install amzn-nova-forge
```

支援的模型和技術

開發套件支援 Amazon Nova 系列中的下列模型和技術：

Method	支援模型
持續預先訓練	所有 Nova 模型 (僅限 SMHP)
受監督的微調 LoRA	所有 Nova 模型
受監督的微調完整範圍	所有 Nova 模型 (僅限 SMHP 和 SMTJ)
直接偏好設定最佳化 LoRA	Nova 1.0 模型 (僅限 SMHP 和 SMTJ)

Method	支援模型
直接偏好設定最佳化完整排名	Nova 1.0 模型 (僅限 SMHP 和 SMTJ)
強化微調 LoRA	Nova Lite 2.0
強化微調完整執行	Nova Lite 2.0 (僅限 SMHP 和 SMTJ)
多迴轉強化微調 LoRA	Nova Lite 2.0 (僅限 SMHP)
多迴轉強化微調全執行	Nova Lite 2.0 (僅限 SMHP)

開始使用

主題

- [1. 準備您的資料](#)
- [2. 設定您的基礎設施](#)
- [3. 訓練](#)
- [4. 監控](#)
- [5. 評估](#)
- [6. 部署](#)

1. 準備您的資料

從本機檔案或 S3 載入您的資料集，並讓 SDK 處理轉換為所選訓練方法的正確格式。或者，提供格式化的資料並立即開始使用。

```
from amzn_nova_forge.dataset.dataset_loader import JSONLDatasetLoader
from amzn_nova_forge.model.model_enums import Model, TrainingMethod

loader = JSONLDatasetLoader(question="input", answer="output")
loader.load("s3://your-bucket/training-data.jsonl")
loader.transform(method=TrainingMethod.SFT_LORA, model=Model.NOVA_LITE)
```

2. 設定您的基礎設施

選擇您的運算資源 - SDK 會驗證組態，並確保最佳設定。

```
from amzn_nova_forge.manager.runtime_manager import BedrockRuntimeManager,
    SMTJRuntimeManager, SMHPRuntimeManager
# Bedrock
runtime = BedrockRuntimeManager(
    execution_role="arn:aws:iam::123456789012:role/ExampleRole"
)

# SageMaker Training Jobs
runtime = SMTJRuntimeManager(
    instance_type="ml.p5.48xlarge",
    instance_count=4
)

# SageMaker HyperPod
runtime = SMHPRuntimeManager(
    instance_type="ml.p5.48xlarge",
    instance_count=4,
    cluster_name="my-hyperpod-cluster",
    namespace="kubeflow"
)
```

3. 訓練

只需幾行程式碼即可開始訓練。

```
from amzn_nova_forge.model import NovaModelCustomizer
from amzn_nova_forge.model.model_enums import Model, TrainingMethod

customizer = NovaModelCustomizer(
    model=Model.NOVA_LITE_2,
    method=TrainingMethod.SFT_LORA,
    infra=runtime,
    data_s3_path="s3://your-bucket/prepared-data.jsonl"
)

result = customizer.train(job_name="my-training-job")
```

4. 監控

直接從 SDK 追蹤您的訓練進度。

```
from amzn_nova_forge.monitor.log_monitor import CloudWatchLogMonitor
```

```
# Monitor training logs
customizer.get_logs()

# Or monitor directly via CloudWatchLogMonitor
monitor = CloudWatchLogMonitor.from_job_result(result)
monitor.show_logs(limit=10)

# Check job status
result.get_job_status() # InProgress, Completed, Failed
```

5. 評估

使用各種[內建基準](#)評估模型效能，或設計您自己的評估。

```
from amzn_nova_forge.recipe_config.eval_config import EvaluationTask

# Evaluate on benchmark tasks
eval_result = customizer.evaluate(
    job_name="model-eval",
    eval_task=EvaluationTask.MMLU,
    model_path=result.model_artifacts.checkpoint_s3_path
)
```

6. 部署

透過 Amazon Bedrock 或 SageMaker 的內建支援，將自訂模型部署至生產環境。

```
from amzn_nova_forge.model.model_enums import DeployPlatform

# Bedrock provisioned throughput
deployment = customizer.deploy(
    model_artifact_path=result.model_artifacts.checkpoint_s3_path,
    deploy_platform=DeployPlatform.BEDROCK_PT,
    pt_units=10
)

# Bedrock On-Demand
deployment = customizer.deploy(
    model_artifact_path=result.model_artifacts.checkpoint_s3_path,
    deploy_platform=DeployPlatform.BEDROCK_OD,
    pt_units=10
)
```

```
# Sagemaker Real-time Inference
deployment = customizer.deploy(
    model_artifact_path=result.model_artifacts.checkpoint_s3_path,
    deploy_platform=DeployPlatform.SAGEMAKER,
    unit_count=10,
    sagemaker_instance_type="ml.p5.48xlarge",
    sagemaker_environment_variables={
        "CONTEXT_LENGTH": "12000",
        "MAX_CONCURRENCY": "16",
    }
)
```

關鍵功能

在飛行配方建立上

SDK 不需要搜尋適當的配方或容器 URI 以取得特定技術。

智慧型資料處理

軟體開發套件會自動將您的資料轉換為正確的訓練格式。無論您是使用 JSON、JSONL 或 CSV 檔案，資料載入器都會順暢地處理轉換。Data Loader 支援文字和多模式資料（影像和影片）。

企業基礎設施支援

開發套件可與 SageMaker 訓練任務和 SageMaker HyperPod 搭配使用，自動管理：

- 執行個體類型驗證
- 配方驗證
- 資料集驗證
- 任務協調和監控

軟體開發套件也支援 SageMaker Training Jobs 無伺服器 and Bedrock 自訂。

全面評估

根據[標準基準](#)評估您的自訂模型，包括：

- MMLU（基本多任務語言理解）
- BBH（進階推理任務）
- GPQA（研究生層級 Google-Proof 問與答）

使用基準預設值，或修改它們以符合您的需求：

- BYOM（使用您自己的指標）
- BYOD（使用您自己的資料集）

生產部署

使用下列選項將模型部署至 Amazon Bedrock 或 SageMaker AI：

- Bedrock 佈建輸送量 - 提供一致效能的專用容量
- Bedrock 隨需（僅適用於 LoRA 型自訂） - Pay-per-use定價
- SageMaker AI 即時推論 - 提供一致效能的專用容量

批次推論

有效率地執行大規模推論任務：

- 平行處理數千個請求
- 自動結果彙總
- 經濟實惠的批次處理

新堡

對於 Nova Forge 訂閱者，開發套件支援資料混合配方。

進一步了解

準備好開始使用 Nova Forge SDK 自訂 Nova 模型了嗎？如需詳細指南、API 參考和其他範例，請參閱我們的 GitHub 儲存庫：<https://github.com/aws/nova-forge-sdk>

Amazon Nova 模型的訓練

在 SageMaker 訓練任務上訓練 Amazon Nova 模型支援監督式微調 (SFT) 和強化微調 (RFT)。每種技術都提供不同的自訂需求，並且可以套用至不同的 Amazon Nova 模型版本。

主題

- [微調 Nova 2.0](#)
- [使用 Amazon Nova 模型進行強化微調 \(RFT\)](#)

微調 Nova 2.0

先決條件

開始訓練任務之前，請注意以下事項：

- Amazon S3 儲存貯體，用於存放訓練任務的輸入資料和輸出。您可以為兩種類型的資料使用一個儲存貯體，也可以為每一種類型的資料使用個別的儲存貯體。請確定您的儲存貯體位於您建立所有其他資源以進行訓練 AWS 區域 的相同位置。如需詳細資訊，請參閱[建立一般用儲存貯體](#)。
- 具有執行訓練任務許可的 IAM 角色。請務必使用 AmazonSageMakerFullAccess 連接 IAM 政策。如需詳細資訊，請參閱[如何使用 SageMaker AI 執行角色](#)。
- 基本 Amazon Nova 配方，請參閱[???](#)。

什麼是 SFT？

監督式微調 (SFT) 會使用標記的輸入輸出對來訓練語言模型。此模型會從包含提示和回應的示範範例中學習，改善其功能以符合特定任務、指示或所需行為。

資料準備

概觀

Nova 2.0 SFT 資料使用與 Nova 1.0 相同的 Converse API 格式，並新增選用的推理內容欄位。如需完整的格式規格，請參閱：

- 原因內容：[ReasoningContentBlock](#)
- Converse API 結構描述：[Converse API](#)
- 資料集限制條件：[資料集限制條件](#)

支援的功能

- 輸入類型 – 使用者內容區塊中的文字、影像或影片
- 助理內容 – 純文字回應和推理內容
- 資料集成 – 必須是同質的。選擇以下一項：
 - 純文字轉彎
 - 文字 + 影像轉彎
 - 文字 + 影片轉場（支援文件理解）

Important

您無法在相同資料集內或在不同回合之間混合影像和影片。

目前限制

- 多模態推理內容 – 雖然 Converse 格式支援以影像為基礎的推理內容，但 Nova 2.0 SFT 在 reasoningText 欄位中僅支援以文字為基礎的推理內容。
- 驗證集 – 您無法為具有 Nova 2.0 的 SFT 提供驗證資料集。如果您提供驗證資料集，則會在訓練期間忽略該資料集。此限制適用於 UI 型和程式設計任務提交。

支援的媒體格式

- 影像 – PNG、JPEG、GIF
- 影片 – MOV、MKV、MP4

資料格式範例

Text-only (Nova 1.0 compatible)

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "system": [
    {
      "text": "You are a digital assistant with a friendly personality"
    }
  ],
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "text": "What country is right next to Australia?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
```

```
{
  "text": "The closest country is New Zealand"
}
```

Text with reasoning (Nova 2.0)

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "system": [
    {
      "text": "You are a digital assistant with a friendly personality"
    }
  ],
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "text": "What country is right next to Australia?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
              "text": "I need to use my world knowledge of geography to answer this question"
            }
          }
        },
        {
          "text": "The closest country to Australia is New Zealand, located to the southeast across the Tasman Sea."
        }
      ]
    }
  ]
}
```

```
]
}
```

Image + text input

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "system": [
    {
      "text": "You are a helpful assistant."
    }
  ],
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "image": {
            "format": "jpeg",
            "source": {
              "s3Location": {
                "uri": "s3://your-bucket/your-path/your-image.jpg",
                "bucketOwner": "your-aws-account-id"
              }
            }
          }
        },
        {
          "text": "Which country is highlighted in the image?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
              "text": "I will determine the highlighted country by examining its location on the map and using my geographical knowledge"
            }
          }
        }
      ]
    }
  ],
}
```

```
{
  "text": "The highlighted country is New Zealand"
}
]
```

Video + text input

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "system": [
    {
      "text": "You are a helpful assistant."
    }
  ],
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "video": {
            "format": "mp4",
            "source": {
              "s3Location": {
                "uri": "s3://your-bucket/your-path/your-video.mp4",
                "bucketOwner": "your-aws-account-id"
              }
            }
          }
        },
        {
          "text": "What is shown in this video?"
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
```

```

        "text": "I will analyze the video content to identify key elements"
      }
    },
    {
      "text": "The video shows a map with New Zealand highlighted"
    }
  ]
}
]
}

```

工具呼叫

Nova 2.0 SFT 支援工具呼叫模式的訓練模型，讓您的模型了解呼叫外部工具或函數的時間和方式。

工具呼叫的資料格式

工具呼叫訓練資料包含定義可用工具的toolConfig區段，以及示範工具使用模式的對話輪換。

範例輸入

```

{
  "schemaVersion": "bedrock-conversation-2024",
  "system": [
    {
      "text": "You are an expert in composing function calls."
    }
  ],
  "toolConfig": {
    "tools": [
      {
        "toolSpec": {
          "name": "getItemCost",
          "description": "Retrieve the cost of an item from the catalog",
          "inputSchema": {
            "json": {
              "type": "object",
              "properties": {
                "item_name": {
                  "type": "string",
                  "description": "The name of the item to retrieve cost for"
                }
              }
            }
          }
        }
      }
    ]
  }
}

```

```

        "item_id": {
            "type": "string",
            "description": "The ASIN of item to retrieve cost for"
        }
    },
    "required": [
        "item_id"
    ]
}
},
{
    "toolSpec": {
        "name": "getItemAvailability",
        "description": "Retrieve whether an item is available in a given location",
        "inputSchema": {
            "json": {
                "type": "object",
                "properties": {
                    "zipcode": {
                        "type": "string",
                        "description": "The zipcode of the location to check in"
                    },
                    "quantity": {
                        "type": "integer",
                        "description": "The number of items to check availability for"
                    },
                    "item_id": {
                        "type": "string",
                        "description": "The ASIN of item to check availability for"
                    }
                }
            },
            "required": [
                "item_id", "zipcode"
            ]
        }
    }
}
],
"messages": [
    {

```



```

    "role": "user",
    "content": [
      {
        "text": "I need to check whether there are twenty pieces of the following
item available. Here is the item ASIN on Amazon: id-123. Please check for the zipcode
94086"
      }
    ]
  },
  {
    "role": "assistant",
    "content": [
      {
        "reasoningContent": {
          "reasoningText": {
            "text": "The user wants to check how many pieces of the item with ASIN
id-123 are available in the zipcode 94086"
          }
        }
      },
      {
        "toolUse": {
          "toolUseId": "getItemAvailability_0",
          "name": "getItemAvailability",
          "input": {
            "zipcode": "94086",
            "quantity": 20,
            "item_id": "id-123"
          }
        }
      }
    ]
  },
  {
    "role": "user",
    "content": [
      {
        "toolResult": {
          "toolUseId": "getItemAvailability_0",
          "content": [
            {
              "text": "[{"name": \"getItemAvailability\", \"results\":
{\"availability\": true}]]\"
            }
          ]
        }
      }
    ]
  }

```

```
    ]
  }
}
],
{
  "role": "assistant",
  "content": [
    {
      "text": "Yes, there are twenty pieces of item id-123 available at 94086.
Would you like to place an order or know the total cost?"
    }
  ]
}
]
```

工具呼叫需求

建立工具呼叫訓練資料時，請遵循下列要求：

需求	Description
ToolUse 置放	ToolUse 必須僅以助理輪換顯示
ToolResult 置放	ToolResult 必須僅顯示在使用者轉彎中
ToolResult 格式	ToolResult 只能是文字或 JSON。Nova 模型不支援其他模態
inputSchema 格式	toolSpec 中的 inputSchema 必須是有效的 JSON 結構描述物件
toolUseId 比對	每個 ToolResult 必須參考先前助理 ToolUse 中的有效 ToolUse toolUseId，每個 toolUseId 在每次對話中只使用一次

重要說明

- 確保您的工具定義在所有訓練範例之間保持一致
- 此模型會從您提供的示範中學習工具叫用模式
- 包含何時使用每個工具以及何時不使用工具的各種範例

理解文件

Nova 2.0 SFT 支援訓練以文件為基礎的任務，讓您的模型了解如何分析和回應有關 PDF 文件的問題。

文件理解的資料格式

文件了解訓練資料包含使用者內容區塊中的文件參考，模型學習會擷取文件內容並說明原因。

範例輸入

```
{
{
  "schemaVersion": "bedrock-conversation-2024",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "text": "What are the ways in which a customer can experience issues during checkout on Amazon?"
        },
        {
          "document": {
            "format": "pdf",
            "source": {
              "s3Location": {
                "uri": "s3://my-bucket-name/path/to/documents/customer_service_debugging.pdf",
                "bucketOwner": "123456789012"
              }
            }
          }
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
              "text": "I need to find the relevant section in the document to answer the question."
            }
          }
        }
      ]
    }
  ]
}
```

```
    }
  },
  {
    "text": "Customers can experience issues with 1. Data entry, 2. Payment
methods, 3. Connectivity while placing the order. Which one would you like to dive
into?"
  }
]
}
]
}
}
```

文件理解限制

限制	詳細資訊
支援的格式	僅限 PDF 檔案
文件大小上限	10 MB
模態混合	範例可以有文件和文字，但不能將文件與其他形式（影像、影片）混合

文件理解的最佳實務

- 確保文件格式清晰且可擷取文字
- 提供涵蓋不同文件類型和問題格式的多樣化範例
- 包含推理內容以協助模型學習文件分析模式

影片理解

Nova 2.0 SFT 支援視訊型任務的訓練，讓您的模型了解如何分析和回應有關視訊內容的問題。

影片理解的資料格式

影片理解訓練資料包含使用者內容區塊中的影片參考，模型學習會擷取影片內容的資訊和原因。

範例輸入

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "text": "What are the ways in which a customer can experience issues during checkout on Amazon?"
        },
        {
          "video": {
            "format": "mp4",
            "source": {
              "s3Location": {
                "uri": "s3://my-bucket-name/path/to/videos/customer_service_debugging.mp4",
                "bucketOwner": "123456789012"
              }
            }
          }
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "reasoningContent": {
            "reasoningText": {
              "text": "I need to find the relevant section in the video to answer the question."
            }
          }
        },
        {
          "text": "Customers can experience issues with 1. Data entry, 2. Payment methods, 3. Connectivity while placing the order. Which one would you like to dive into?"
        }
      ]
    }
  ]
}
```

```
}
```

影片理解限制

限制	詳細資訊
影片大小上限	50 MB
最大視訊持續時間	15 分鐘
每個範例的影片數	每個範例只允許一個影片。不支援相同範例中的多個影片
模態混合	範例可以有影片和文字，但不能將影片與其他形式（影像、文件）結合

支援的影片格式

- MOV
- MKV
- MP4

影片理解的最佳實務

- 保持影片簡潔並專注於與您的任務相關的內容
- 確保影片品質足以讓模型擷取有意義的資訊
- 提供明確的問題，以參考影片內容的特定層面
- 包含涵蓋不同影片類型和問題格式的多樣化範例

原因與非原因模式

了解推理內容

合理的內容（也稱為chain-of-thought）會在產生最終答案之前擷取模型的中繼思考步驟。assistant 然後，使用 reasoningContent 欄位來包含這些推理追蹤。

如何計算損失

- 使用推理內容 – 訓練損失包括推理權杖和最終輸出權杖
- 沒有推理內容 – 僅根據最終輸出權杖計算訓練損失

您可以在多迴轉對話中reasoningContent跨多個助理迴轉包含。

格式化準則

- 使用純文字來推理內容
- 避免標記標籤，例如 <thinking>和 </thinking>除非您的任務特別要求
- 確保推理內容清晰且與問題解決程序相關

何時啟用推理模式

在下列情況下，在您的訓練組態reasoning_enabled: true中設定：

- 您的訓練資料具有推理權杖
- 您希望模型在產生最終輸出之前產生思維權杖
- 您需要改善複雜推理任務的效能

reasoning_enabled = true 允許使用 在非合理資料集上訓練 Nova。不過，這樣做可能會導致模型失去其推理功能，因為 Nova 主要學習在不套用推理的情況下產生資料中呈現的回應。如果您想要在非合理資料集上訓練 Nova，但仍預期推論期間的推理，您可以在訓練期間停用推理 (reasoning_enabled = false)，但啟用它以進行推論。雖然此方法允許在推論時間使用推理，但與推論相比，它並不保證改善效能，無需推理。一般而言，在使用推理資料集時啟用訓練推理和推論，並在使用非合理的資料集時停用兩者推理。

設定reasoning_enabled: false時機：

- 您的訓練資料沒有推理權杖
- 您正在針對無法從明確推理步驟中受益的直接任務進行訓練
- 您想要最佳化速度並減少字符用量

產生推理資料

如果您的資料集缺少推理追蹤，您可以使用 Nova Premier 之類的推理能力模型來建立它們。將您的輸入輸出對提供給模型，並擷取其推理程序，以建置推理擴增的資料集。

使用推理字符進行訓練

啟用推理模式的訓練時，模型會學習將內部推理與最終答案分開。訓練程序：

- 將資料組織為三元組：輸入、推理和回答
- 使用推理和回答權杖的標準下一個權杖預測損失進行最佳化
- 鼓勵模型在產生回應之前進行內部推理

有效的推理內容

高品質推理內容應包括：

- 中繼想法和分析
- 邏輯扣除和推論步驟
- Step-by-step解決問題的方法
- 步驟和結論之間的明確連線

這有助於模型開發「在回答之前思考」的能力。

資料集準備準則

大小和品質

- 建議大小 – 2,000-10,000 個範例
- 最低範例 – 200
- 優先順序 – 數量的品質。確保範例準確且標註正確
- 應用程式一致性 – 資料集應緊密反映您的生產使用案例

多樣性

包含下列各種範例：

- 涵蓋預期輸入的完整範圍
- 代表不同的難度等級
- 包含邊緣案例和變化
- 防止過度擬合至窄模式

輸出格式

在助理回應中明確指定所需的輸出格式：

- JSON 結構
- 表格
- CSV format (CSV 格式)
- 您應用程式特有的自訂格式

多回合對話

對於多迴轉資料集，請記住：

- 只有在助理輪換時才會計算損失，而非使用者輪換時
- 每個助理回應都應正確格式化
- 在對話回合之間保持一致性

品質檢查清單

- 足夠的資料集大小 (2K-10K 範例)
- 涵蓋所有使用案例的多樣化範例
- 清晰、一致的輸出格式
- 準確的標籤和註釋
- 生產案例的代表
- 沒有矛盾或模稜兩可之處

上傳資料

資料集應上傳至 SageMaker 訓練任務可存取的儲存貯體。如需設定正確許可的資訊，請參閱[先決條件](#)。

啟動訓練任務

選取超參數並更新配方

Nova 2.0 的設定大致上與 Nova 1.0 相同。輸入資料上傳到 S3 後，請在微調資料夾下使用 [SageMaker HyperPod](#) 配方中的配方。對於 Nova 2.0，以下是您可以根據使用案例更新的一些關鍵超參數。以下是 Nova 2.0 SFT PEFT 配方的範例。對於容器映像 URI，請使用 `708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-TJ-SFT-V2-latest` 執行 SFT 微調任務。

請使用 SageMaker AI PySDK 的 v2.254.1，以嚴格相容於 Nova 訓練。將 SDK 升級到 v3.0 版本會導致重大變更。即將推出對 SageMaker AI PySDK v3 的支援。

範例輸入

```
!pip install sagemaker==2.254.1
```

```
run:
    name: {peft_recipe_job_name}
    model_type: amazon.nova-2-lite-v1:0:256k
    model_name_or_path: {peft_model_name_or_path}
    data_s3_path: {train_dataset_s3_path} # SageMaker HyperPod (SMHP) only and not
compatible with SageMaker Training jobs. Note replace my-bucket-name with your real
bucket name for SMHP job
    replicas: 4 # Number of compute instances for training, allowed
values are 4, 8, 16, 32
    output_s3_path: "" # Output artifact path (Hyperpod job-specific; not
compatible with standard SageMaker Training jobs). Note replace my-bucket-name with
your real bucket name for SMHP job

training_config:
    max_steps: 10 # Maximum training steps. Minimal is 4.
    save_steps: 10 # How many training steps the checkpoint will be
saved. Should be less than or equal to max_steps
    save_top_k: 1 # Keep top K best checkpoints. Note supported only
for SageMaker HyperPod jobs. Minimal is 1.
    max_length: 32768 # Sequence length (options: 8192, 16384, 32768
[default], 65536)
    global_batch_size: 32 # Global batch size (options: 32, 64, 128)
    reasoning_enabled: true # If data has reasoningContent, set to true;
otherwise False

    lr_scheduler:
        warmup_steps: 15 # Learning rate warmup steps. Recommend 15% of
max_steps
        min_lr: 1e-6 # Minimum learning rate, must be between 0.0 and 1.0

    optim_config: # Optimizer settings
        lr: 1e-5 # Learning rate, must be between 0.0 and 1.0
        weight_decay: 0.0 # L2 regularization strength, must be between 0.0
and 1.0
```

```

    adam_beta1: 0.9                # Exponential decay rate for first-moment estimates,
must be between 0.0 and 1.0
    adam_beta2: 0.95              # Exponential decay rate for second-moment
estimates, must be between 0.0 and 1.0

    peft:                          # Parameter-efficient fine-tuning (LoRA)
    peft_scheme: "lora"            # Enable LoRA for PEFT
    lora_tuning:
        alpha: 64                  # Scaling factor for LoRA weights ( options: 32, 64,
96, 128, 160, 192),
        lora_plus_lr_ratio: 64.0

```

配方也包含與 Nova 1.0 大致相同的超參數。值得注意的超參數包括：

- `max_steps` – 您要執行任務的步驟數目。一般而言，對於一個 epoch（一個執行整個資料集），步驟數目 = 資料樣本數目/全域批次大小。步驟數目越大，全域批次大小越小，任務執行的時間就越長。
- `reasoning_enabled` – 控制資料集的推理模式。選項：
 - `true`：啟用推理模式（相當於高推理）
 - `false`：停用推理模式

注意：對於 SFT，無法精細控制推理工作量。設定 `reasoning_enabled: true` 可啟用完整推理功能。

- `peft.peft_scheme` – 將此設定為「lora」可啟用以 PEFT 為基礎的微調。將其設定為 `null`（無引號）可啟用全執行微調。

開始訓練任務

```

from sagemaker.pytorch import PyTorch

# define OutputDataConfig path
if default_prefix:
    output_path = f"s3://{bucket_name}/{default_prefix}/{sm_training_job_name}"
else:
    output_path = f"s3://{bucket_name}/{sm_training_job_name}"

output_kms_key = "<KMS key arn to encrypt trained model in Amazon-owned S3 bucket>" #
optional, leave blank for Amazon managed encryption

recipe_overrides = {

```

```

    "run": {
        "replicas": instance_count, # Required
        "output_s3_path": output_path
    },
}

estimator = PyTorch(
    output_path=output_path,
    base_job_name=sm_training_job_name,
    role=role,
    disable_profiler=True,
    debugger_hook_config=False,
    instance_count=instance_count,
    instance_type=instance_type,
    training_recipe=training_recipe,
    recipe_overrides=recipe_overrides,
    max_run=432000,
    sagemaker_session=sagemaker_session,
    image_uri=image_uri,
    output_kms_key=output_kms_key,
    tags=[
        {'Key': 'model_name_or_path', 'Value': model_name_or_path},
    ]
)

print(f"\nsm_training_job_name:\n{sm_training_job_name}\n")
print(f"output_path:\n{output_path}")

```

```

from sagemaker.inputs import TrainingInput

train_input = TrainingInput(
    s3_data=train_dataset_s3_path,
    distribution="FullyReplicated",
    s3_data_type="Converse",
)

estimator.fit(inputs={"validation": val_input}, wait=False)

```

Note

Nova 2.0 的監督式微調不支援傳遞驗證資料集。

若要啟動任務：

- 使用資料集路徑和超參數更新配方
- 在筆記本中執行指定的儲存格以提交訓練任務

筆記本會處理任務提交並提供狀態追蹤。

使用 Amazon Nova 模型進行強化微調 (RFT)

概觀

什麼是 RFT？

強化微調 (RFT) 透過訓練意見回饋訊號來改善模型效能，這些訊號是可測量的分數或獎勵，指出模型的效能，而不是確切的正確答案。與從輸入輸出對中學習的監督式微調不同，RFT 使用獎勵函數來評估模型回應，並反覆最佳化模型以最大化這些獎勵。此方法在定義確切的正確輸出具有挑戰性時表現卓越，但您可以可靠地測量回應品質。

何時使用 RFT

當您可以定義明確、可衡量的成功條件，但難以為訓練提供確切正確的輸出時，請使用 RFT。它非常適合：

- 品質是主觀或多面向的任務（創造性寫入、程式碼最佳化、複雜推理）
- 具有多個有效解決方案的案例，其中有些解決方案明顯優於其他解決方案
- 需要反覆改進、個人化或遵守複雜業務規則的應用程式
- 收集高品質標籤範例昂貴或不切實際的案例

最佳使用案例

RFT 在可以客觀測量輸出品質，但難以預先定義最佳回應的網域中表現卓越：

- 數學問題解決和程式碼產生
- 科學推理和結構化資料分析
- 需要step-by-step推理或多迴轉問題解決的任務
- 應用程式平衡多個目標（準確性、效率、風格）
- 可透過執行結果或效能指標以程式設計方式驗證成功的案例

支援的模型

Nova Lite 2.0

資料格式概觀

RFT 訓練資料必須遵循 OpenAI 強化微調[格式](#)。每個訓練範例都是 JSON 物件，其中包含：

- 使用 system 和 user 角色進行對話輪換的 messages 陣列
- reference_answer 包含預期輸出或評估條件的欄位，以進行獎勵計算

目前限制

- 僅文字

資料格式範例

每個範例應該位於 JSONL 檔案中的單一行上，每行一個 JSON 物件。

Chemistry problem

```
{
  "id": "chem-01",
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful chemistry assistant"
    },
    {
      "role": "user",
      "content": "Calculate the molecular weight of caffeine (C8H10N4O2)"
    }
  ],
  "reference_answer": {
    "molecular_weight": 194.19,
    "unit": "g/mol",
    "calculation": "8(12.01) + 10(1.008) + 4(14.01) + 2(16.00) = 194.19"
  }
}
```

Math problem

```
{
  "id": "sample-001", // Optional
  "messages": [
    {
      "role": "system",
      "content": "You are a math tutor"
    },
    {
      "role": "user",
      "content": "Solve:  $2x + 5 = 13$ "
    }
  ],
  "reference_answer": {
    "solution": "x = 4",
    "steps": [" $2x = 13 - 5$ ", " $2x = 8$ ", " $x = 4$ "]
  }
}
```

Code problem

```
{
  "id": "code-002",
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful programming assistant"
    },
    {
      "role": "user",
      "content": "Write a Python function that reverses a string without using built-in reverse methods"
    }
  ],
  "reference_answer": {
    "code": "def reverse_string(s): \n    result = '' \n    for i in range(len(s) - 1, -1, -1): \n        result += s[i] \n    return result",
    "test_cases": [
      {
        "input": "hello",
        "expected_output": "olleh"
      }
    ],
  }
}
```

```
{
  "input": "",
  "expected_output": ""
},
{
  "input": "a",
  "expected_output": "a"
},
{
  "input": "Python123",
  "expected_output": "321nohtyP"
}
],
"all_tests_pass": true
}
```

`reference_answer` 欄位包含預期輸出或評估條件，您的獎勵函數會用來對模型的回應進行評分。它不限於結構化輸出，它可以包含任何格式，協助您的獎勵函數評估品質。

資料集大小建議

起點

- 最少 100 個訓練範例
- 最少 100 個評估範例

評估優先方法

在投資大規模 RFT 訓練之前，請評估模型的基準效能：

- 高效能 (>95% 獎勵) – RFT 可能不必要，您的模型已表現良好
- 非常差的效能 (0% 獎勵) – 首先切換到 SFT 以建立基本功能
- 中等效能 – RFT 可能是適當的

從小型資料集開始，您可以：

- 驗證您的獎勵函數沒有錯誤
- 確認 RFT 是適合您使用案例的正確方法

- 及早識別和修正問題
- 在向上擴展之前測試工作流程

驗證後，您可以擴展到更大的資料集，以進一步改善效能。

有效訓練資料的特性

清晰度和一致性

良好的 RFT 範例需要清晰、不明確的輸入資料，以便在不同的模型輸出間進行準確的獎勵計算。避免資料中的雜訊，包括：

- 不一致的格式
- 矛盾的標籤或指示
- 模稜兩可的提示
- 衝突的參考答案

任何模稜兩可的情況都會誤導訓練程序，並導致模型學習非預期的行為。

多樣性

您的資料集應擷取生產使用案例的完整多樣性，以確保強大的實際效能。包括：

- 不同的輸入格式和邊緣案例
- 從日誌和使用者分析映射實際的生產使用模式
- 使用者類型、地理區域和季節性變化的範例
- 包含從簡單到複雜問題的難度等級

獎勵函數考量事項

設計您的獎勵函數以進行高效訓練：

- 在幾秒鐘內執行（非分鐘）
- 使用 Lambda 有效平行化
- 傳回一致、可靠的分數
- 正常處理不同類型的模型輸出

快速、可擴展的獎勵函數可實現快速迭代和經濟實惠的實驗。

其他屬性

RFT 資料格式支援核心結構描述需求之外的自訂欄位 (messages 和 reference_answer)。此彈性可讓您新增獎勵函數進行適當評估所需的任何其他資料。

Note

您不需要在配方中設定此 - 資料格式本質上支援其他欄位。只要將它們包含在訓練資料 JSON 中，它們就會在 metadata 欄位中傳遞給您的獎勵函數。

常見的其他屬性

中繼資料欄位範例：

- task_id – 用於追蹤的唯一識別符
- difficulty_level – 問題複雜性指標
- domain – 主題區域或類別
- expected_reasoning_steps – 解決方案中的步驟數量

具有其他屬性的範例

```
{
  "messages": [
    {
      "role": "system",
      "content": "You are a math tutor"
    },
    {
      "role": "user",
      "content": "Solve:  $2x + 5 = 13$ "
    }
  ],
  "reference_answer": {
    "solution": " $x = 4$ ",
    "steps": [" $2x = 13 - 5$ ", " $2x = 8$ ", " $x = 4$ "]
  },
  "task_id": "algebra_001",
  "difficulty_level": "easy",
```

```

"domain": "algebra",
"expected_reasoning_steps": 3
}

```

這些額外的欄位會在評估期間傳遞給您的獎勵函數，以根據您的特定使用案例量身打造複雜的評分邏輯。

訓練組態

範例配方

```

# Note:
# This recipe can run on p5.48xlarge and p5en.48xlarge instance types.
run:
  name: "my-rft-run" # Unique run name (appears in logs/artifacts).
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: nova-lite-2/prod
  data_s3_path: s3://<bucket>/<data file> # Training dataset in JSONL;
  replicas: 4
  reward_lambda_arn: arn:aws:lambda:<region>:<account-id>:function:<function-name>

## MLFlow configs
mlflow_tracking_uri: "" # Required for MLFlow
mlflow_experiment_name: "my-rft-experiment" # Optional for MLFlow. Note: leave this field non-empty
mlflow_run_name: "my-rft-run" # Optional for MLFlow. Note: leave this field non-empty

## SMTJ GRPO Training specific configs
training_config:
  max_length: 8192 # Context window (tokens) for inputs
+prompt;
  global_batch_size: 16 # Total samples per optimizer step
  across all replicas (16/32/64/128/256).
  reasoning_effort: high # Enables reasoning mode high / low /
  or null for non-reasoning

  rollout: # How responses are generated for GRPO/
  advantage calc.
    advantage_strategy:
      number_generation: 2 # N samples per prompt to estimate
  advantages (variance vs cost).
    generator:
      max_new_tokens: 6000 # Cap on tokens generated per sample

```

```

        set_random_seed: true                # Seed generation for reproducibility
across runs.
        temperature: 1                      # Softmax temperature;
        top_k: 1                            # Sample only from top-K logits
    rewards:
        preset_reward_function: null        # Usage of preset reward functions
[exact_match]
        api_endpoint:
            lambda_arn: arn:aws:lambda:<region>:<account-id>:function:<function-name>
            lambda_concurrency_limit: 12     # Max concurrent Lambda invocations
(throughput vs. throttling).

    trainer:
        max_steps: 2                        # Steps to train for. One Step =
global_batch_size
        save_steps: 5
        test_steps: 1

    # RL parameters
    ent_coeff: 0.0                          # A bonus added to the policy loss that
rewards higher-output entropy.
    kl_loss_coef: 0.001                     # Weight on the KL penalty between the
actor (trainable policy) and a frozen reference model

    optim_config:                           # Optimizer settings
        lr: 5e-5                            # Learning rate
        weight_decay: 0.0                   # L2 regularization strength (0.0-1.0)
        adam_beta1: 0.9
        adam_beta2: 0.95

    peft:                                   # Parameter-efficient fine-tuning (LoRA)
        peft_scheme: "lora"                 # Enable LoRA for PEFT
        lora_tuning:
            alpha: 32
            lora_plus_lr_ratio: 64.0        # LoRA+ learning rate scaling factor (0.0-
100.0)

```

使用 LLM 做為判斷的 RFT 訓練

概觀

大型語言模型 LLMs) 在強化微調 (RFT) 工作流程中逐漸被用作判斷，提供引導模型最佳化的自動獎勵訊號。在此方法中，LLM 會根據指定的條件評估模型輸出，無論是評估正確性、品質、風格遵循或語意相等性，並指派推動強化學習程序的獎勵。

這對於傳統獎勵函數難以以程式設計方式定義的任務特別有用，例如判斷不同的表示法（例如 "1/3"、"0.333" 和 "1/3"）是否在語義上相等，或評估一致性和相關性等細微品質。透過使用以 LLM 為基礎的判斷做為獎勵函數，您可以將 RFT 擴展到複雜的網域，而不需要大量的人工註釋，因此除了傳統對齊問題之外，可在各種使用案例中快速迭代和持續改善模型。

合理模式選擇

可用模式

- none – 無推理（省略 reasoning_effort 欄位）
- 低 - 最低推理開銷
- 高 – 最大推理功能（指定 reasoning_effort 時預設為預設值）

Note

RFT 沒有媒體選項。如果組態中沒有 reasoning_effort 欄位，則會停用推理。啟用推理時，您應該將 max_new_tokens 設定為 32768，以容納延伸推理輸出。

何時使用每個模式

使用高推理：

- 複雜的分析任務
- 數學問題解決
- 多步驟邏輯扣除
- step-by-step 思考可增加價值的任務

不使用（省略 reasoning_effort）或低推理：

- 簡單事實查詢
- 直接分類
- 速度和成本最佳化
- 直接的問答

成本和效能權衡

較高的推理模式會增加：

- 訓練時間和成本
- 推論延遲和成本
- 複雜推理任務的模型功能

驗證您的 LLM 判斷

在生產環境中部署 LLM-as-a-judge 之前，請驗證判斷模型的評估是否符合人類判斷。這包括：

- 針對您任務的代表性範例，測量 LLM 判斷者與人工評估者之間的協議率
- 確保 LLM 與人類的協議符合或超過人類之間的協議費率
- 識別判斷模型中的潛在偏差
- 建立對獎勵訊號引導模型朝預期方向的信心

此驗證步驟有助於確保自動化評估程序會產生符合您生產品質標準的模型。

LLM 判斷的 Lambda 組態

使用 LLM 做為判斷，是使用 Lambda 函數進行強化學習與可驗證獎勵 (RLVR) 的延伸。在 Lambda 函數中，您可以呼叫 Amazon Bedrock 中託管的其中一個模型。

重要的組態需求：

Configuration	需求	詳細資訊
Amazon Bedrock 輸送量	足夠的配額	確保所使用的 Amazon Bedrock 模型的輸送量配額足以滿足您的訓練工作負載

Configuration	需求	詳細資訊
Lambda 逾時	延長逾時	將 Lambda 函數逾時設定為最長 15 分鐘。預設設定為 3 秒，這不足以回應 Amazon Bedrock 模型
Lambda 並行	並行增加	Lambda 會在訓練期間平行叫用。增加並行以最大化可用輸送量
配方組態	比對 Lambda 設定	必須在配方中設定並行限制

建立和執行任務

啟動訓練任務

使用 SageMaker 訓練任務筆記本範本：<https://docs.aws.amazon.com/sagemaker/latest/dg/nova-fine-tuning-training-job.html#nova-model-training-jobs-notebook>

執行個體需求

容器同時支援 Full-Rank 和 LoRA 訓練：

- LoRA 訓練 – 2/4/6/8 × p5.48xlarge 或 p5en.48xlarge 執行個體
- 全執行訓練 – 2/4/6/8 × p5.48xlarge 執行個體（必要）

監控訓練

訓練日誌包含每個步驟的全方位指標。金鑰指標類別：

獎勵指標

- critic/rewards/mean、critic/rewards/max、critic/rewards/min – 獎勵分佈
- val-score/rewards/mean@1 – 驗證獎勵

模型行為

- actor/entropy – 政策變化（較高 = 更探索性）

訓練運作狀態

- actor/pg_loss – 政策梯度損失
- actor/pg_clipfrac – 剪輯更新的頻率
- actor/grad_norm – 漸層大小

回應特性

- prompt_length/mean、prompt_length/max、prompt_length/min – 輸入字符統計資料
- response_length/mean、response_length/max、response_length/min – 輸出字符統計資料
- response/aborted_ratio – 產生率不完整 (0 = 全部完成)

效能

- perf/throughput – 訓練輸送量
- perf/time_per_step – 每個訓練步驟的時間
- timing_per_token_ms/* – 每個金鑰的處理時間

資源用量

- perf/max_memory_allocated_gb , perf/max_memory_reserved_gb – GPU 記憶體
- perf/cpu_memory_used_gb – CPU 記憶體

使用微調的模型

訓練完成後，最終模型檢查點會儲存到您指定的輸出位置。檢查點路徑可用於：

- 訓練日誌
- manifest.json 輸出 Amazon S3 位置中的 檔案（由 在您的筆記本output_s3_uri中定義）

限制和最佳實務

限制

- Lambda 逾時 – 獎勵函數必須在 15 分鐘內完成（防止失控程序和管理成本）
- 僅限單轉 – 不支援多轉對話

- 資料需求 – 需要足夠的多樣性；難以獲得稀疏獎勵 (<5% 的正面範例)
- 運算成本 – 比監督式微調更昂貴
- 無多模態資料 – 僅支援文字資料類型

最佳實務

從小開始

- 從 100-200 個範例開始
- 驗證獎勵函數正確性
- 根據結果逐步擴展

訓練前評估

- 在 RFT 之前測試基準模型效能
- 如果獎勵持續為 0%，請先使用 SFT 建立基本功能
- 如果獎勵 >95%，RFT 可能不必要

監控訓練

- 追蹤平均獎勵分數和分佈
- 注意過度擬合（訓練獎勵增加，同時驗證獎勵減少）
- 尋找相關的模式：
 - 獎勵穩定低於 0.15
 - 隨時間增加獎勵差異
 - 降低驗證效能

最佳化獎勵函數

- 在幾秒鐘內執行（非分鐘）
- 將外部 API 呼叫降至最低
- 使用高效演算法
- 實作適當的錯誤處理
- 利用 Lambda 的平行擴展

反覆運算策略

如果獎勵未改善：

- 調整獎勵函數設計
- 增加資料集多樣性
- 新增更多代表性範例
- 驗證獎勵訊號是否清晰且一致

進階功能：Nova Forge

對於需要超出標準 RFT 限制之進階功能的使用者，Nova Forge 可作為付費訂閱服務方案使用：

- 多迴轉對話支援
- 以 >15 分鐘的執行時間獎勵函數
- 其他演算法和調校選項
- 自訂訓練配方修改
- State-of-the-art技術

Nova Forge 在 SageMaker HyperPod 上執行，旨在支援企業客戶建置自己的前沿模型。

有用的命令和秘訣

提供[可觀測性指令碼](#)的集合，以協助監控訓練任務的狀態和進度。

可用的指令碼包括：

- 啟用電子郵件通知以進行訓練任務狀態更新
- 根據任務組態取得訓練時間預估
- 針對進行中的任務，取得預期訓練需要多久的近似值

安裝

Note

請務必在使用下列任何指令碼之前重新整理您的 AWS 登入資料。

```
pip install boto3
git clone https://github.com/aws-samples/amazon-nova-samples.git
cd amazon-nova-samples/customization/SageMakerUtils/SageMakerJobsMonitoring/
```

基本用量

```
# Enabling email notifications for training job status updates
python enable_sagemaker_job_notifs.py --email test@amazon.com test2@gmail.com --region
us-east-1 --platform SMTJ
```

Creating resources.....

Please check your email for a subscription confirmation email, and click 'Confirm subscription' to start receiving job status email notifications!

You'll receive the confirmation email within a few minutes.

```
# Obtaining training time estimates based on job configurations
python get_training_time_estimate.py
```

```
# Obtaining approximations for how long training is expected to take for in-progress
jobs
python get-training-job-progress.py --region us-east-1 --job-name my-training-job --
num-dataset-samples 1000
```

如需其他詳細資訊和範例，請參閱[此處](#)。

監控反覆運算的進度

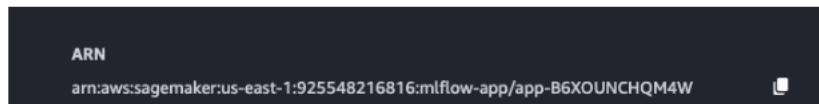
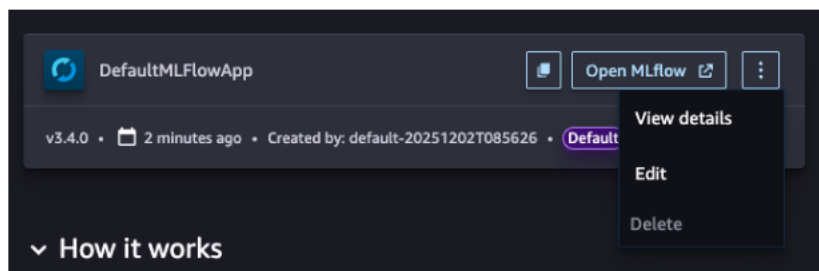
您可以透過 MLflow 追蹤指標。

Nova Customization - SageMaker HyperPod 的 MLFlow 設定

若要讓您的 SageMaker HyperPod 環境將指標輸出至 MLFlow，必須執行一些額外的設定。

1. 開啟 Amazon SageMaker AI
2. 選取 SageMaker Studio
 - a. 如果已建立設定檔，請選取「開啟 Studio」。
 - b. 如果未建立設定檔，請選取「建立 SageMaker 網域」來設定一個設定檔
3. 選取 MLFlow。如果未建立任何 MLFlow 應用程式，請選取「建立 MLFlow 應用程式」

- 按一下複製/貼上按鈕或 ML Flow 應用程式上的「檢視詳細資訊」選單項目，以取得 ARN。提交訓練任務時，您將需要此項目。



- 在 HyperPod 叢集執行角色上，新增下列政策。這將允許 HyperPod 叢集呼叫 MLFlow API 來發佈指標。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sagemaker-mlflow:*",
      "Resource": [
        "arn:aws:sagemaker:us-east-1:372836560492:mlflow-app/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:ListMlflowTrackingServers",
        "sagemaker:CallMlflowAppApi"
      ],
      "Resource": "*"
    }
  ]
}
```

透過 CLI 提交任務

在命令列或配方 yaml 中指定 4 個新的覆寫參數。

1. `mlflow_tracking_uri` : MLFlow 應用程式的 ARN
2. `mlflow_experiment_name` : 此實驗執行的名稱
3. `mlflow_experiment_name` : 將指標存放在 MLFlow 中的實驗名稱
4. `mlflow_run_name` : 此實驗的名稱

命令列

```
--override-parameters '{"recipes.run.mlflow_tracking_uri": "arn:aws:sagemaker:us-east-1:925548216816:mlflow-app/app-B6X0UNCHQM4W", "recipes.run.mlflow_experiment_name": "myuser-sft-lora-exp1", "recipes.run.mlflow_run_name": "myuser-sft-lora-exp1-202512181940"}'
```

yaml :

```
## Run config
run:
  mlflow_tracking_uri: "arn:aws:sagemaker:us-east-1:925548216816:mlflow-app/app-B6X0UNCHQM4W"
  mlflow_experiment_name: "myuser-sft-lora-exp1"
  mlflow_run_name: "myuser-sft-lora-exp1-202512181940"
```

透過 SageMaker Studio UI 提交任務

MLFlow 整合已內建於 SageMaker Studio UI 體驗中。提交訓練任務時，只要指出要使用的 MLFlow 應用程式執行個體即可。

1. 在 SageMaker Studio 中，導覽至模型 > Nova 2.0 Lite > 自訂 > 使用 UI 自訂。
2. 展開進階組態區段
3. 選取您要傳送訓練指標的 MLFlow 應用程式。您也可以在此處設定實驗名稱和實驗執行。

MLflow

MLflow Resource *
MLflow resource selected for experiment tracking.

DefaultMLFlowApp

Experiment name
Default name applied. Enter a new name if desired.

myuser-sft-lora-exp1

Experiment run
The run name is autogenerated by default. Specify a new name if desired.

myuser-sft-lora-exp1-1766499113356

透過 提交任務 AWS AWS CLI

如果您使用 AWS AWS CLI，則必須建立 MLflow 應用程式並將其做為輸入傳遞給訓練任務 API 請求。

```
mlflow_app_name="<enter your MLflow app name>"
role_arn="<enter your role ARN>"
bucket_name="<enter your bucket name>"
region="<enter your region>"

mlflow_app_arn=$(aws sagemaker create-mlflow-app \
  --name $mlflow_app_name \
  --artifact-store-uri "s3://$bucket_name" \
  --role-arn $role_arn \
  --region $region)
```

建立 MLflow 應用程式

使用 Studio UI：如果您透過 Studio UI 建立訓練任務，預設 MLflow 應用程式會自動建立，並在進階選項下依預設選取。

使用 CLI：如果您使用 CLI，則必須建立 MLflow 應用程式並將其做為訓練任務 API 請求的輸入傳遞。

```
mlflow_app_name="<enter your MLflow app name>"
role_arn="<enter your role ARN>"
bucket_name="<enter your bucket name>"
region="<enter your region>"

mlflow_app_arn=$(aws sagemaker create-mlflow-app \
  --name $mlflow_app_name \
  --artifact-store-uri "s3://$bucket_name" \
```

```
--role-arn $role_arn \  
--region $region)
```

存取 MLflow 應用程式

使用 CLI：建立預先簽章的 URL 以存取 MLflow 應用程式 UI：

```
aws sagemaker create-presigned-mlflow-app-url \  
--arn $mlflow_app_arn \  
--region $region \  
--output text
```

使用 Studio UI：Studio UI 會顯示存放在 MLflow 中的關鍵指標，並提供 MLflow 應用程式 UI 的連結。

要追蹤的關鍵指標

跨反覆運算監控這些指標，以評估改善並追蹤任務進度：

對於 SFT

- 訓練損失曲線
- 使用的範例數量和處理範例的時間
- 保留測試集的效能準確性
- 格式合規（例如，有效的 JSON 輸出速率）
- 網域特定評估資料的複雜度

對於 RFT

- 訓練的平均獎勵分數
- 獎勵分佈（高獎勵回應的百分比）
- 驗證獎勵趨勢（注意過度擬合）
- 任務特定的成功率（例如，程式碼執行通過率、數學問題準確性）

一般

- 反覆運算之間的基準效能差異
- 代表性範例的人工評估分數
- 生產指標（如果反覆部署）

判斷何時停止

在下列情況下停止反覆運算：

- 效能穩定：其他訓練不再有意義地改善目標指標
- 技術切換有助於：如果一個技術穩定，請嘗試切換（例如 SFT → RFT → SFT）以突破效能上限
- 實現的目標指標：符合您的成功條件
- 偵測到迴歸：新的反覆運算會降低效能（請參閱下列轉返程序）

如需詳細評估程序，請參閱評估一節。

評估您的 SageMaker AI 訓練模型

評估程序的目的是根據基準或自訂資料集評估訓練模型的效能。評估程序的步驟通常包括建立指向訓練模型的評估配方、指定評估資料集和指標、提交個別任務進行評估，以及根據標準基準或自訂資料進行評估。評估程序將輸出存放在 Amazon S3 儲存貯體中的效能指標。

Note

本主題中所述的評估程序為離線程序。系統會使用預先定義的回答來根據固定的基準測試模型，而不是以即時的方式或透過即時使用者互動進行評估。如需即時評估，您可以在模型部署到 Amazon Bedrock 之後，呼叫 [Amazon Bedrock](#) 執行時期 API 來測試模型。

主題

- [先決條件](#)
- [可用的基準任務](#)
- [評估特定組態](#)
- [執行評估訓練任務](#)
- [評估和分析評估結果](#)
- [評估最佳實務和疑難排解](#)
- [可用的子任務](#)
- [推理模型評估](#)
- [RFT 評估](#)
- [實作獎勵函數](#)

先決條件

在開始評估訓練任務之前，請注意下列事項。

- 您想要評估其效能的 SageMaker AI 訓練 Amazon Nova 模型。
- 用於評估的基本 Amazon Nova 配方。如需詳細資訊，請參閱[???](#)。

可用的基準任務

提供範例程式碼套件，示範如何使用 Amazon Nova 的 SageMaker AI 模型評估功能來計算基準指標。若要存取程式碼套件，請參閱 [sample-Nova-lighteval-custom-task](#)。

以下是支援的可用產業標準基準清單。您可以在 eval_task 參數中指定下列基準。

模型評估的可用基準

Benchmark	模態	Description	指標	策略	子任務可用
mmlu	文字	多任務語言理解 – 測試 57 個主題的知識。	正確性	zs_cot	是
mmlu_pro	文字	MMLU – 專業子集 – 專注於專業領域，例如法律、醫學、會計和工程。	正確性	zs_cot	否
bbh	文字	進階推理任務 – 一系列挑戰性問題，可測試高階認知和問題解決技能。	正確性	fs_cot	是
gpqa	文字	一般物理問題回答 – 評估對	正確性	zs_cot	否

Benchmark	模態	Description	指標	策略	子任務可用
		物理概念的理解及解決相關問題的能力。			
數學運算	文字	數學問題解決 – 測量代數、微積分和應用題等主題的數學推理能力。	exact_match	zs_cot	是
strong_reject	文字	品質控管任務 – 測試模型可偵測和拒絕不適當、有害或不正確內容的能力。	偏轉	zs	是
ifeval	文字	指示追蹤評估 – 測量模型遵循指定指示的準確度，並根據規格完成任務。	正確性	zs	否

Benchmark	模態	Description	指標	策略	子任務可用
gen_qa	多模態 (影像)	自訂資料集評估 – 可讓您提供自己的資料集進行基準測試，將模型輸出與參考答案與 ROUGE 和 BLEU 等指標進行比較。 gen_qa 支援 Amazon Nova Lite 或 Amazon Nova Pro 型模型的影像推論。也支援 Bring-Your-Own 指標 lambda。 (對於 RFT 評估，請使用 RFT 評估配方)	全部	gen_qa	否
mmmu	多模態	大量多學科多模式理解 (MMMU) – 學院層級基準測試，包含來自 30 個學科的多選和開放式問題。)	正確性	zs_cot	是

Benchmark	模態	Description	指標	策略	子任務可用
llm_judge	文字	LLM 即評審 偏好比較 – 使用 Nova Judge 模型 來針對您的提 示判斷配對回 應 (B 與 A 比 較) 之間的偏 好，並計算較 A 偏好 B 的機 率。	全部	評審	否
mm_llm_judge	多模態 (影 像)	這個新基準 的行為與 llm_judge 上述以文字 為基礎的行為 相同。唯一的 差別在於它支 援影像推論。	全部	評審	否

Benchmark	模態	Description	指標	策略	子任務可用
rubric_llm_judge	文字	Rubric Judge 是建置在 Nova 2.0 Lite 上的增強型 LLM-as-a-judge 評估模型。與只提供偏好判定的 原始判斷模型 不同，Rubric Judge 會動態產生針對每個提示量身打造的自訂評估條件，並跨多個維度指派精細分數。	全部	評審	否
aime_2024	文字	AIME 2024 - 美國邀請數學檢查問題測試進階數學推理和問題解決	exact_match	zs_cot	否
calendar_scheduling	文字	Natural Plan - 行事曆排程任務測試規劃能力，以跨多天 and 多人安排會議	exact_match	fs	否

Benchmark	模態	Description	指標	策略	子任務可用
humaneval	文字	HumanEval - 旨在評估大型語言模型程式碼產生功能的基準資料集	pass@1	zs	否

評估特定組態

以下是配方中關鍵元件的明細，以及如何針對您的使用案例修改這些元件的指引。

了解和修改您的配方

一般執行組態

```
run:
  name: eval_job_name
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: nova-lite-2/prod # or s3://escrow_bucket/model_location
  replicas: 1
  data_s3_path: ""
  mlflow_tracking_uri: ""
  mlflow_experiment_name : ""
  mlflow_run_name : ""
```

- `name`：評估任務的描述性名稱。
- `model_type`：指定要使用的 Nova 模型變體。請勿手動修改此欄位。選項包括：
 - `amazon.nova-micro-v1:0:128k`
 - `amazon.nova-lite-v1:0:300k`
 - `amazon.nova-pro-v1:0:300k`
 - `amazon.nova-2-lite-v1 : 0 : 256k`
- `model_name_or_path`：基本模型的路徑或訓練後檢查點的 s3 路徑。選項包括：
 - `nova-micro/prod`
 - `nova-lite/prod`
 - `nova-pro/prod`

- nova-lite-2/prod
- 訓練後檢查點路徑的 S3 路徑 (s3:customer-escrow-111122223333-smtj-<unique_id>/<training_run_name>)

Note

評估訓練後模型

若要在 Nova SFT 訓練任務之後評估訓練後模型，請在成功執行訓練任務後依照下列步驟進行。在訓練日誌結尾，您會看到日誌訊息「訓練完成」。您也可以輸出儲存貯體中找到 manifest.json 檔案，其中包含檢查點的位置。此檔案將位於輸出 S3 位置的 output.tar.gz 檔案中。若要繼續評估，請在配方組態中將其設定為 run.model_name_or_path 的值，以使用此檢查點。

- replica：用於分散式推論（跨多個節點執行推論）的運算執行個體數量。設定 replica > 1 以啟用多節點推論，加速評估。如果同時指定 replica instance_count 和 instance_count 則優先。請注意，多個複本僅適用於 SageMaker AI 訓練任務，不適用於。
- data_s3_path：輸入資料集 Amazon S3 路徑。此欄位為必要欄位，但應一律保留空白。
- mlflow_tracking_uri：（選用）MLflow 追蹤伺服器的位置（僅在 SMHP 上需要）
- mlflow_experiment_name：（選用）將相關 ML 執行分組在一起的實驗名稱
- mlflow_run_name：（選用）實驗中特定訓練執行的自訂名稱

評估組態

```
evaluation:
  task: mmlu
  strategy: zs_cot
  subtask: abstract_algebra
  metric: accuracy
```

- task：指定要使用的評估基準或任務。支援的任務包括：
 - mmlu
 - mmlu_pro
 - bbh
 - gpqa
 - math

- `strong_reject`
- `gen_qa`
- `ifeval`
- `mmmu`
- `llm_judge`
- `mm_llm_judge`
- `rubric_llm_judge`
- `aime_2024`
- `calendar_scheduling`
- `humaneval`
- `strategy`：定義評估方法。
 - `zs_cot`：零樣本思緒鏈 - 一種可提示大型語言模型的方法，鼓勵逐步推理，而無需明確的範例。
 - `fs_cot`：小樣本思維鏈 - 一種在要求模型解決新問題之前，提供數個逐步推理範例的方法。
 - `zs`：零樣本 - 一種無需任何先前訓練範例即可解決問題的方法。
 - `gen_qa`：使用自有資料集特定的策略。
 - `judge`：以 `Judge` 和 `mm_llm_judge` 的形式針對 Nova LLM 的策略。
- `subtask`：選用。評估任務的特定元件。如需可用子任務的完整清單，請參閱[???。](#)
 - 檢查可用基準任務中支援的子任務。
 - 如果沒有子任務基準，應移除此欄位。
- `metric`：要使用的評估指標。
 - `accuracy`：正確回答的百分比。
 - `exact_match`：對於數學基準，會傳回輸入預測字串與其參考完全相符的比率。
 - `deflection`：對於強烈拒絕基準，會傳回對基礎模型的相對偏轉和差異顯著性指標。
 - `all`：

對於 `gen_qa`，使用自有資料集基準，這會傳回下列指標：

- `rouge1`：測量所產生文字和參考文字之間一元語法 (一個字) 的重疊。
- `rouge2`：測量所產生文字和參考文字之間二元語法 (連續兩個字) 的重疊。
- `rougeL`：測量文字之間最長的共同子序列，允許相符項目中的差距。
- `exact_match`：二元分數 (0 或 1)，指出產生的文字是否逐字元完全符合參考文字。

- `f1_score`：精確率和召回率的調和平均數，可測量預測回答和參考回答之間的字詞重疊。
- `f1_score_quasi`：類似於 `f1_score`，但比對更寬鬆，使用會忽略次要差異的標準化文字比較。
- `bleu`：測量所產生文字和參考文字之間 n 元語法比對的精確率，常用於轉譯評估。

對於 `llm_judge` 和 `mm_llm_judge`，使用您自己的資料集基準，請傳回下列指標：

- `a_scores`：跨向前和向後評估傳遞的 `response_A` 其獲勝次數。
- `a_scores_stderr`：跨配對判斷的 `response_A_scores` 其標準誤。
- `b_scores`：測量跨向前和向後評估傳遞的 `response_B` 其獲勝次數。
- `a_scores_stderr`：跨配對判斷的 `response_B_scores` 其標準誤。
- `ties`：將 `response_A` 和 `response_B` 評估為相等的判斷數目。
- `ties_stderr`：跨配對判斷的 `ties` 其標準誤。
- `inference_error`：無法正確評估的判斷計數。
- `score`：根據 `response_B` 其向前和向後傳遞中的獲勝來彙總分數。
- `score_stderr`：根據 `response_B` 其向前和向後傳遞中的獲勝來彙總分數。
- `inference_error_stderr`：跨配對判斷的彙總分數其標準誤。
- `winrate`：使用 Bradley-Terry 機率計算將比 `response_A` 偏好 `response_B` 的機率。
- `lower_rate`：從引導取樣預估勝率的下限 (第 2.5 個百分位數)。
- `upper_rate`：從引導取樣預估勝率的上限 (第 97.5 個百分位數)。

推論組態 (選擇性)

```
inference:
  max_new_tokens: 2048
  top_k: -1
  top_p: 1.0
  temperature: 0
  top_logprobs: 10
  reasoning_effort: null # options: low/high to enable reasoning or null to disable reasoning
```

- `max_new_tokens`：要產生的記號數目上限。必須是整數。(不適用於 LLM 判斷)
- `top_k`：要考慮的最高機率記號數量。必須是整數。

評估 `top_p`：記號抽樣的累積機率閾值。必須是介於 1.0 到 0.0 之間的浮點數。

- `temperature`：記號選擇的隨機性 (數字越高越隨機)，保持 0 可讓結果具決定性。浮點數類型，最小值為 0。
- `top_logprobs`：推論回應中要傳回的最大 logprob 數目。此值必須是從 0 到 20 的整數。Logprob 包含考慮的輸出字符，以及訊息內容中傳回的每個輸出字符的日誌機率。
- `reasoning_effort`：控制可推理模型的推理行為。`reasoning_effort` 只有在 `model_type` 指定可推理的模型時設定 (目前為 `amazon.nova-2-lite-v1:0:256k`)。可用選項為 `null` (如果未設定預設值；停用推理)、低或高。

日誌機率輸出格式

在推論設定中設定 `top_logprobs` 時，評估輸出會在 `parquet` 檔案中包含字符層級日誌機率。每個字符位置都包含最佳候選字符的字典，其日誌機率位於下列結構中：

```
{
  "Ġint": {"logprob_value": -17.8125, "decoded_value": " int"},
  "Ġthe": {"logprob_value": -2.345, "decoded_value": " the"}
}
```

每個字符項目都包含：

- `logprob_value`：字符的日誌機率值
- `decoded_value`：字符的人類可讀取解碼字串表示

原始字符化器字符用作字典索引鍵，以確保唯一性，同時 `decoded_value` 提供可讀取的解釋。

評估配方範例

Amazon Nova 提供四種不同類型的評估配方。所有配方都可在 [SageMaker HyperPod 配方 GitHub 儲存庫](#) 中使用。

一般文字基準配方

這些配方可讓您在一套完整的純文字基準之間評估 Amazon Nova 模型的基本功能。

配方格式：`xxx_general_text_benchmark_eval.yaml`。

使用自有資料集基準配方

這些配方可讓您使用自有資料集進行基準測試，並使用不同類型的指標比較模型輸出以參考回答。

配方格式：xxx_ bring_your_own_dataset_eval.yaml。

使用自有資料集需求

檔案格式：

- 包含評估範例的單一 gen_qa.jsonl 檔案。檔案名稱應該是確切的 gen_qa.jsonl。
- 您必須將資料集上傳至 SageMaker AI 訓練任務可存取的 S3 位置。
- 檔案必須遵循一般 Q&A 資料集所需的結構描述格式。

結構描述格式要求 - .jsonl 檔案中的每一行都必須是具有下列欄位的 JSON 物件。

- 必要欄位。

query：字串，包含需要回答的問題或指示。

response：字串，包含預期的模型輸出。

- 選用欄位。

system：字串，包含在處理查詢之前設定 AI 模型行為、角色或個性的系統提示。

images：陣列包含具有資料屬性的物件清單 (Base64 編碼影像字串)。

metadata：字串包含與項目相關聯的中繼資料，用於標記目的。

範例項目

```
{
  "system": "You are an English major with top marks in class who likes to give minimal word responses: ",
  "query": "What is the symbol that ends the sentence as a question",
  "response": "?"
}{
  "system": "You are a pattern analysis specialist who provides succinct answers: ",
  "query": "What is the next number in this series? 1, 2, 4, 8, 16, ?",
  "response": "32"
}{
  "system": "You have great attention to detail and follow instructions accurately: ",
  "query": "Repeat only the last two words of the following: I ate a hamburger today and it was kind of dry",
  "response": "of dry"
```

```
{
  "system": "Image inference: ",
  "query": "What is the number in the image? Please just use one English word to answer.",
  "response": "two",
  "images": [
    {
      "data": "data:image/png;base64,iVBORw0KGgoA ..."
    }
  ]
}
```

若要使用您的自訂資料集，請新增下列必要欄位來修改您的評估配方，而不變更現有的組態：

```
evaluation:
  task: gen_qa
  strategy: gen_qa
  metric: all
```

限制

- 每次評估只允許一個 .jsonl 檔案。
- 該檔案必須嚴格遵循定義的結構描述。

使用您自己的指標

您可以使用自己的指標，透過自訂預先處理、後製處理和指標功能來完全自訂模型評估工作流程。預先處理可讓您在將輸入資料傳送到推論伺服器之前進行處理，而後製處理可讓您自訂指標計算，並根據您的需求傳回自訂指標。

請依照下列步驟，使用自訂評估 SDK 來使用您自己的指標。

1. 如果您尚未這麼做，請先在 AWS 帳戶 中 [建立 AWS Lambda 函數](#)。
2. 從 [GitHub 儲存庫](#) 下載預先建置 nova-custom-eval-layer.zip 的檔案。您可以使用此開放原始碼 Nova 自訂評估 SDK 來驗證自訂函數的輸入和輸出承載，並提供整合的介面，以在訓練期間與 Nova 的自有指標評估整合。
3. 使用下列命令上傳自訂 Lambda layer：

```
aws lambda publish-layer-version \
  --layer-name nova-custom-eval-layer \
```

```
--zip-file fileb://nova-custom-eval-layer.zip \
--compatible-runtimes python3.12 python3.11 python3.10 python3.9
```

4. 將此層新增為 Lambda 函數的自訂層，以及必要的 AWS 層：
AWSLambdaPowerToolsPythonV3-python312-arm64 (pydantic相依性需要)。
5. 使用提供的範例更新您的 Lambda 程式碼，並視需要修改程式碼。此範例程式碼會使用模型評估的預先處理和後製步驟，為 Nova 的自訂評估建立 Lambda 函數。

```
from nova_custom_evaluation_sdk.processors.decorators import preprocess,
    postprocess
from nova_custom_evaluation_sdk.lambda_handler import build_lambda_handler

@preprocess
def preprocessor(event: dict, context) -> dict:
    data = event.get('data', {})
    return {
        "statusCode": 200,
        "body": {
            "system": data.get("system"),
            "prompt": data.get("prompt", ""),
            "gold": data.get("gold", "")
        }
    }

@postprocess
def postprocessor(event: dict, context) -> dict:
    # data is already validated and extracted from event
    data = event.get('data', [])
    inference_output = data.get('inference_output', '')
    gold = data.get('gold', '')

    metrics = []
    inverted_accuracy = 0 if inference_output.lower() == gold.lower() else 1.0
    metrics.append({
        "metric": "inverted_accuracy_custom",
        "value": accuracy
    })

    # Add more metrics here

    return {
        "statusCode": 200,
        "body": metrics
    }
```

```

    }

    # Build Lambda handler
    lambda_handler = build_lambda_handler(
        preprocessor=preprocessor,
        postprocessor=postprocessor
    )

```

6. 授予 Lambda 對評估任務的存取權。確保為評估任務指定的執行角色包含叫用 Lambda 函數的政策。以下是一個範例政策。

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LambdaAccess",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
      "Condition": {
        "StringLike": {
          "aws:PrincipalArn": "arn:aws:iam::111122223333:role/service-role/AmazonSageMaker-ExecutionRole-ARN"
        }
      }
    },
    {
      "Sid": "DenyNonAWSEventSourcesForLambda",
      "Effect": "Deny",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-east-1:111122223333:function:ExampleFunction",
      "Condition": {
        "Null": {
          "lambda:EventSourceToken": false
        }
      }
    }
  ]
}

```

```
    }
  }
]
}
```

7. 檢閱 Lambda 承載結構描述。下表列出 Lambda 請求和回應結構描述。您可以使用 Nova 自訂評估 SDK 驗證您的結構描述。

	Lambda 請求承載	Lambda 回應承載
預處理器	<pre>{ "process_type": "preprocess", "data": { "system": "You are a helpful assistant", "prompt": "What is 2+2?", "gold": "4" } }</pre>	<pre>{ "statusCode": 200, "body": { "system": "You are a helpful assistant that can substitute * for addition", "prompt": "What is 2*2?", "gold": "4" } }</pre>
後處理器	<pre>{ "process_type": "postprocess", "data": { "prompt": "What is 2+2?", "inference_output": "2+2=4", "gold": "4" } }</pre>	<pre>{ "statusCode": 200, "body": [{"metric": "accuracy", "value": 1.0}, {"metric": "f1_score", "value": 1.0}, {"metric": "exact_match", "value": 1}, {"metric": "length_ratio", "value": 0.8}] }</pre>

8. 修改配方檔案。請見此處範例。

```
processor:
  lambda_arn: arn:aws:lambda:us-east-1:111122223333:function:name
```

```
lambda_type: "custom_metrics"
preprocessing:
  enabled: true
postprocessing:
  enabled: true
aggregation: average
```

- `lambda-arn`：處理預先處理和後製處理的 Lambda 函數的 Amazon Resource Name (ARN)。
- `lambda_type`：「custom_metrics」或「rft」。
- `preprocessing`：是否要啟用自訂預先處理操作。
- `postprocessing`：是否要啟用自訂後置處理操作。
- `aggregation`：內建彙總函數（有效選項：最小值、最大值、平均值、總和）。

限制

- 使用您自己的指標僅適用於文字輸入資料集。
- 不支援多模態輸入資料集。
- 預先處理步驟不會處理中繼資料欄位。

Nova LLM 即評審基準配方

Nova LLM Judge 是一種模型評估功能，可讓您使用自訂資料集，將一個模型的回應品質與基準模型的回應進行比較。它會接受包含提示、基準回應和挑戰者回應的資料集，然後使用 Nova Judge 模型透過配對比較，以根據 [Bradley-Terry](#) 機率提供勝率指標。配方格式：`xxx_llm_judge_eval.yaml`。

Nova LLM 資料集需求

檔案格式：

- 包含評估範例的單一 `llm_judge.jsonl` 檔案。檔案名稱應該是確切的 `llm_judge.jsonl`。
- 您必須將資料集上傳至 SageMaker AI 訓練任務可存取的 S3 位置。
- 檔案必須遵循 `llm_judge` 資料集所需的結構描述格式。
- 輸入資料集應確保所有記錄都不到 12 k 內容長度。

結構描述格式 - `.jsonl` 檔案中的每一行都必須是具有下列欄位的 JSON 物件。

- 必要欄位。

prompt：字串，包含所產生回應的提示。

response_A：字串，包含基準回應。

response_B：字串，包含與基準回應比較的替代回應。

範例項目

```
{
  "prompt": "What is the most effective way to combat climate change?",
  "response_A": "The most effective way to combat climate change is through a combination of transitioning to renewable energy sources and implementing strict carbon pricing policies. This creates economic incentives for businesses to reduce emissions while promoting clean energy adoption.",
  "response_B": "We should focus on renewable energy. Solar and wind power are good. People should drive electric cars. Companies need to pollute less."
}
{
  "prompt": "Explain how a computer's CPU works",
  "response_A": "CPU is like brain of computer. It does math and makes computer work fast. Has lots of tiny parts inside.",
  "response_B": "A CPU (Central Processing Unit) functions through a fetch-execute cycle, where instructions are retrieved from memory, decoded, and executed through its arithmetic logic unit (ALU). It coordinates with cache memory and registers to process data efficiently using binary operations."
}
{
  "prompt": "How does photosynthesis work?",
  "response_A": "Plants do photosynthesis to make food. They use sunlight and water. It happens in leaves.",
  "response_B": "Photosynthesis is a complex biochemical process where plants convert light energy into chemical energy. They utilize chlorophyll to absorb sunlight, combining CO2 and water to produce glucose and oxygen through a series of chemical reactions in chloroplasts."
}
```

若要使用您的自訂資料集，請使用下列必要欄位修改您的評估配方，切勿變更任何內容：

```
evaluation:
  task: llm_judge
  strategy: judge
```

```
metric: all
```

限制

- 每次評估只允許一個 .jsonl 檔案。
- 該檔案必須嚴格遵循定義的結構描述。
- Nova Judge 模型在微型/精簡/專業規格中是相同的。
- 目前不支援自訂判斷模型。

Nova LLM 作為多模態（影像）基準配方的判斷

Nova LLM Judge for multi-modal (image), short for Nova MM_LLM Judge 是一種模型評估功能，可讓您使用自訂資料集，比較一個模型的回應品質與基準模型的回應。它會接受包含提示、基準回應和挑戰者回應的資料集，以及 Base64-encoded 字串形式的影像，然後使用 Nova Judge 模型，透過配對比較提供基於 [Bradley-Terry](#) 機率的勝率指標。配方格式：xxx_mm_llm_judge_eval.yaml。

Nova LLM 資料集需求

檔案格式：

- 包含評估範例的單一 mm_llm_judge.jsonl 檔案。檔案名稱必須恰好為 llm_judge.jsonl。
- 您必須將資料集上傳至 SageMaker AI 訓練任務可存取的 S3 位置。
- 檔案必須遵循 mm_llm_judge 資料集所需的結構描述格式。
- 輸入資料集應確保所有記錄都低於 12 k 內容長度，不包括影像的屬性。

結構描述格式 - .jsonl 檔案中的每一行都必須是具有下列欄位的 JSON 物件。

- 必要欄位。

prompt：字串，包含所產生回應的提示。

images：陣列包含具有資料屬性的物件清單（值為 Base64-encoded 的影像字串）。

response_A：字串，包含基準回應。

response_B：字串，包含與基準回應比較的替代回應。

範例項目

為了便於閱讀，以下範例包含新的行和縮排，但在實際的資料集中，每個記錄應該位於單一行。

```
{
  "prompt": "What is in the image?",
  "images": [
    {
      "data": "data:image/jpeg;base64,/9j/2wBDAAQDAwQDAwQEAwQFBAQFBgo..."
    }
  ],
  "response_A": "a dog.",
  "response_B": "a cat.",
}
{
  "prompt": "How many animals are in each of the images?",
  "images": [
    {
      "data": "data:image/jpeg;base64,/9j/2wBDAAQDAwQDAwQEAwQFBAQFBgo..."
    },
    {
      "data": "data:image/jpeg;base64,/DKEafe3gihn..."
    }
  ],
  "response_A": "The first image contains one cat and the second image contains one dog",
  "response_B": "The first image has one animal and the second has one animal"
}
```

若要使用您的自訂資料集，請使用下列必要欄位修改您的評估配方，切勿變更任何內容：

```
evaluation:
  task: mm_llm_judge
  strategy: judge
  metric: all
```

限制

- 每次評估只允許一個 .jsonl 檔案。
- 該檔案必須嚴格遵循定義的結構描述。
- Nova MM Judge 模型僅支援影像參考。
- Nova MM 判斷模型在 Amazon Nova Micro、Amazon Nova Lite 和 Amazon Nova Pro 規格中是相同的。

- 目前不支援自訂判斷模型。
- 不支援 Amazon S3 映像 URI。
- 輸入資料集應確保所有記錄都低於 12 k 內容長度，但不包括影像屬性。

評估 CPT（持續訓練前）檢查點

CPT（持續預先訓練）模型的評估可能比已進行 SFT（監督式微調）的模型更具挑戰性，因為 CPT 模型經常缺乏遵循指示的能力。CPT 模型將作為完成模型運作，而不是遵循下列指示，這表示它們只會嘗試繼續隨輸入字符序列提供的模式。有鑑於此限制，典型的評估資料集可能因為輸入「Q&A」格式而無法正常運作，而不是回答模型只會嘗試繼續相同的問題。不過，透過以特定方式格式化資料集來提示完成樣式的模型，我們可以了解模型的效能。

請依照下列步驟，了解如何使用 Nova Forge 評估工作流程，在持續預先訓練的模型上執行評估。

資料集準備和格式化

評估 CPT 模型會利用 Nova Forge 模型評估體驗中已提供的現有[自攜資料集](#)工作流程。不過，資料集內的查詢必須以純「完成」格式格式化，因為 CPT 模型不會像 SFT 模型那樣回應標準問題樣式提示。

對於僅經歷 CPT 的模型，另一個重要的經常性限制是無法產生 STOP 或序列結束權杖，這表示模型將繼續產生權杖，直到強制停止為止（例如使用 `max_new_tokens` 參數）。有鑑於此限制，最佳實務是使用單一字符回應（例如多重選擇）評估模型，以確保模型不會繼續產生提示後不需要的垃圾郵件輸出。

例如，典型的評估資料集（例如 MMLU、GPQA、MATH 等）可能會向模型提示以下問題：

```
Early settlements and high population density along coastlines and rivers are best attributed to which of the following?
```

```
A: "Poor climate conditions"
```

```
B: "Limited forest cover"
```

```
C: "Cars"
```

```
D: "Access to trade routes"
```

```
(Expected answer is D.)
```

不過，CPT 模型不會了解如何正確回應此問題，因為以下指示缺少微調。因此，必須以完成樣式提示 CPT 模型，例如：

```
Early settlements and high population density along coastlines and rivers are best attributed to which of the following?
```

```
A: Poor climate conditions
B: Limited forest cover
C: Cars
D: Access to trade routes
The correct answer to this question is option
```

(Expected answer is D.)

在推論檢查模型產生的輸出 `logprob` 後，將提供模型是否正確處理輸入並產生正確回應的詳細資訊。不保證模型一律會產生預期的確切回應（在此情況下為字母 D），不過，如果模型正常運作，應該會出現在 `logprob` 中。

另一個非多選完成樣式提示範例：

```
The capital of France is

(Expected answer of Paris)
```

我們預期模型會產生「巴黎」的回應，或在 `logprobs` 輸出的某處看到對應至「巴黎」的字符。

資料集格式化

CPT 評估利用現有的 [自帶資料集](#) 工作流程。資料必須以「查詢回應」格式格式化，並以新行分隔的 JSONL 檔案。

其中包含 4 個項目的資料集範例：

```
{"query": "The capital of France is", "response": "Paris"}
{"query": "2 + 2 =", "response": "4"}
{"query": "The mitochondria is the powerhouse of the", "response": "cell"}
{"query": "What is the largest planet?\nA: Mars\nB: Jupiter\nC: Saturn\nD: Earth\n\nAnswer:", "response": "B"}
```

每一行必須包含：

- `query`：提示文字用於完成
- `response`：預期的完成 (Ground Truth)

模型將接收原始文字輸入，而無需聊天格式。CPT 模型通常尚未在特殊權杖上進行訓練，且無法正確回應聊天範本，因此在提示模型時，只有查詢中提供的字串才會傳送至模型（自動在模型前面加上額外的 `[BOS]` 權杖）。

配方組態

以下是設定為評估 CPT 模型的配方範例：

```
run:
  name: cpt_eval_job
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: s3://bucket/path/to/cpt-checkpoint/

evaluation:
  task: gen_qa # Required for CPT - bring your own dataset
  strategy: gen_qa
  metric: all # Returns rouge1, rouge2, rougeL, exact_match, f1_score, bleu

inference:
  checkpoint_is_instruction_tuned: "false" # Required for CPT checkpoints
  top_logprobs: 5 # Change to desired amount of logprobs to calculate
  max_new_tokens: 1 # Keep low for completion tasks
  temperature: 0.0
```

CPT 評估的關鍵變更：

- `checkpoint_is_instruction_tuned: "false"`

這是特別新增的新參數，以支援在 CPT 檢查點上執行的評估。將 `checkpoint_is_instruction_tuned` 設為 `false` 會停用通常包裝輸入提示的預設 Amazon Nova 聊天範本。

- `top_logprobs: 5`

日誌機率 (logprob) 會揭露模型在可能的下一個權杖之間的可信度分佈，協助您評估模型是否已在預先訓練期間學習到預期的完成。一般而言，如果模型如預期般執行，我們應該會將預期的回應（即「A」、「B」等）視為產生的輸出字符或 logprob 中的字符。

- `max_new_tokens: 1`

CPT 模型通常尚未經過訓練，無法產生特殊的「停止」或「序列結束」權杖，但尚未發出何時停止推論的訊號。這表示模型通常會繼續產生新的字符，直到達到特定的字符長度上限，導致不必要的推論。通常，將 `max_new_tokens` 限制為 1，並提供提示，以便在單一回應（如多選擇問題）上評估模型，這是提示模型的最有效方式。將 `max_new_tokens` 設定為 1 可確保不會產生額外的垃圾郵件字符。

重要參數

- `checkpoint_is_instruction_tuned`：必須設定為 `"false"` (或 `false` 為布林值) 以停用聊天範本
- `top_logprobs`：5，建議查看模型在 CPT 期間如何學習
- 任務：必須是 `gen_qa` - CPT 模型無法使用指示遵循基準，例如 MMLU 或 MATH
- 策略：必須是 `gen_qa`
- `max_new_tokens`：建議保持低 (1-5)，因為 CPT 模型執行完成，而不是產生

執行評估訓練任務

使用以下範例筆記本啟動訓練任務。請參閱以下筆記本作為範例，以執行評估訓練任務。如需詳細資訊，請參閱[使用 SageMaker AI 估算器執行訓練任務](#)。

參考表

執行筆記本之前，請參閱下列參考表，以選取映像 URI 和執行個體組態。

選取映像 URI

Recipe	映像 URI
評估映像 URI	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-TJ-Eval-V2-latest

選取執行個體類型和計數

模型	任務類型	執行個體類型	建議的執行個體計數	允許的執行個體計數
Amazon Nova Micro	評估 (SFT/DPO)	g5.12xlarge	1	1 - 16
Amazon Nova Lite	評估 (SFT/DPO)	g5.12xlarge	1	1 - 16
Amazon Nova Pro	評估 (SFT/DPO)	p5.48xlarge	1	1 - 16

範例筆記本

下列範例筆記本示範如何執行評估訓練任務。

```
# install python SDK

# Do not use sagemaker v3, as sagemaker v3 introduced breaking changes

!pip install sagemaker==2.254.1

import os
import sagemaker,boto3
from sagemaker.inputs import TrainingInput
from sagemaker.pytorch import PyTorch

sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()

# Download recipe from https://github.com/aws/sagemaker-hyperpod-recipes/tree/main/
# recipes_collection/recipes/evaluation/nova to local
# Assume the file name be `recipe.yaml`

# Populate parameters
# input_s3_uri = "s3://<path>/input/" # (Optional) Only used for multi-modal dataset or
# bring your own dataset s3 location
output_s3_uri= "s3://<path>/output/" # Output data s3 location, a zip containing
# metrics json and tensorboard metrics files will be stored to this location
instance_type = "instance_type" # ml.g5.16xlarge as example
instance_count = 1 # The number of instances for inference (set instance_count > 1 for
# multi-node inference to accelerate evaluation)
job_name = "your job name"
recipe_path = "recipe path" # ./recipe.yaml as example
image_uri = "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-TJ-
Eval-V2-latest" # Do not change
output_kms_key = "<KMS key arn to encrypt trained model in Amazon-owned S3 bucket>" #
# optional, leave blank for Amazon managed encryption

# (Optional) To bring your own dataset and LLM judge for evaluation
# evalInput = TrainingInput(
# s3_data=input_s3_uri,
# distribution='FullyReplicated',
# s3_data_type='S3Prefix'
#)
```



```

estimator = PyTorch(
    output_path=output_s3_uri,
    base_job_name=job_name,
    role=role,
    instance_type=instance_type,
    instance_count=instance_count,
    training_recipe=recipe_path,
    sagemaker_session=sagemaker_session,
    image_uri=image_uri,
    output_kms_key=output_kms_key
)
estimator.fit()

# If input dataset exist, pass in inputs
# estimator.fit(inputs={"train": evalInput})

```

評估和分析評估結果

評估任務成功完成後，您可以使用下列步驟來評估和分析結果。

若要評估和分析結果，請依照下列步驟進行。

1. 了解輸出位置結構。結果會以壓縮檔案形式存放在您指定的 Amazon S3 輸出位置：

```

s3://your-bucket/output/benchmark-name/
### job_name/
    ### output/
        ### output.tar.gz

```

2. 從儲存貯體下載 output.tar.gz 檔案。解壓縮要顯示的內容。

```

run_name/
### eval_results/
|   ### results_[timestamp].json
#   ### inference_output.jsonl (only present for gen_qa)
|   ### details/
|       ### model/
|           ### <execution-date-time>/
|               ###details_<task_name>_#_<datetime>.parquet
### tensorboard_results/
    ### eval/
        ### events.out.tfevents.[timestamp]

```

- `results_[timestamp].json` - 輸出指標 JSON 檔案
 - `details_<task_name>_#_<datetime>.parquet` - 推論輸出檔案 (`strong_reject` 除外)
 - `events.out.tfevents.[timestamp]` - TensorBoard 輸出檔案
 - `inference_output.jsonl` - 清除的推論輸出檔案 (僅適用於 `gen_qa` 任務)
3. 在 TensorBoard 中檢視結果。若要視覺化您的評估指標：
1. 將解壓縮的資料夾上傳至 S3 儲存貯體
 2. 導覽至 SageMaker AI TensorBoard
 3. 選取您的「S3 資料夾」
 4. 新增 S3 資料夾路徑
 5. 等待同步完成
4. 分析推論輸出。除了 `llm_judge` 和 `strong_reject` 之外，所有評估任務都會有下列欄位可用於在推論輸出中分析。
- `full_prompt` - 傳送至用於評估任務之模型的完整使用者提示。
 - `gold` - 包含資料集所指定之正確回答的欄位（如）。
 - `metrics` - 包含個別推論評估指標的欄位。需要彙總的值在個別推論輸出上不會有值。
 - `predictions` - 包含指定提示之模型輸出清單的欄位。
 - `pred_logits` - 包含所考慮的輸出字符和訊息內容中傳回之每個輸出字符的日誌機率的欄位。

透過查看這些欄位，您可以判斷指標差異的原因，並了解自訂模型的行為。

對於 `llm_judge`，推論輸出檔案在每對評估的指標欄位下方包含下列欄位。

- `forward_output` - 按順序 (`response_A`、`response_B`) 評估時的 Judge 原始偏好設定。
- `backward_output` - 按相反順序 (`response_B`、`response_A`) 評估時的 Judge 原始偏好設定。
- `Pairwise metrics` - 根據向前和向後評估配對計算的指標，包括 `a_scores`、`b_scores`、`ties`、`inference-score` 和 `score`。

 Note

如 `winrate` 等彙總指標僅可在摘要結果檔案中取得，而不是根據個別判斷。

對於 `gen_qa` , `inference_output.jsonl` 檔案包含每個 JSON 物件的下列欄位：

- 提示 - 提交到模型的最終提示
- 推論 - 來自模型的原始推論輸出
- `gold` - 來自輸入資料集的目標回應
- 中繼資料 - 如果提供，來自輸入資料集的中繼資料字串

評估最佳實務和疑難排解

最佳實務

以下列出評估程序的一些最佳實務。

- 透過模型和基準類型讓輸出路徑井然有序。
- 維持一致的命名慣例，以便於追蹤。
- 將擷取的結果儲存在安全的位置。
- 監控 TensorBoard 同步狀態以成功載入資料。

疑難排解

您可以將 CloudWatch 日誌群組 `/aws/sagemaker/TrainingJobs` 用於訓練任務錯誤日誌。

引擎核心故障

問題：

如果您看到：

```
RuntimeError: Engine core initialization failed.
```

原因：

雖然這是可能有多個原因的一般錯誤，但通常會在您嘗試載入的模型檢查點與指定的模型類型之間不相符時發生。例如，您想要評估微調的 Nova 2.0 lite 模型檢查點，但您提供的模型類型是 1.0 模型類型，例如 `amazon.nova-micro-v1:0:128k`

正確的映射應該是

```
model_type: amazon.nova-2-lite-v1:0:256k
model_name_or_path: nova-lite-2/prod # or s3://escrow_bucket/model_location
```

預防：

在提交評估任務`model_type`之前，再次檢查 `model_name_or_path` 是否對應至右側。

可用的子任務

下列列出跨多個網域進行模型評估的可用子任務，包括 MMLU（基本多任務語言理解）、BBH（大型 Bench Hard）、MATH 和 MMMU（基本多學科多任務理解）。這些子任務可讓您評估模型在特定功能和知識領域的效能。

MMLU

```
MMLU_SUBTASKS = [
    "abstract_algebra",
    "anatomy",
    "astronomy",
    "business_ethics",
    "clinical_knowledge",
    "college_biology",
    "college_chemistry",
    "college_computer_science",
    "college_mathematics",
    "college_medicine",
    "college_physics",
    "computer_security",
    "conceptual_physics",
    "econometrics",
    "electrical_engineering",
    "elementary_mathematics",
    "formal_logic",
    "global_facts",
    "high_school_biology",
    "high_school_chemistry",
    "high_school_computer_science",
    "high_school_european_history",
    "high_school_geography",
    "high_school_government_and_politics",
    "high_school_macroeconomics",
    "high_school_mathematics",
```

```
"high_school_microeconomics",  
"high_school_physics",  
"high_school_psychology",  
"high_school_statistics",  
"high_school_us_history",  
"high_school_world_history",  
"human_aging",  
"human_sexuality",  
"international_law",  
"jurisprudence",  
"logical_fallacies",  
"machine_learning",  
"management",  
"marketing",  
"medical_genetics",  
"miscellaneous",  
"moral_disputes",  
"moral_scenarios",  
"nutrition",  
"philosophy",  
"prehistory",  
"professional_accounting",  
"professional_law",  
"professional_medicine",  
"professional_psychology",  
"public_relations",  
"security_studies",  
"sociology",  
"us_foreign_policy",  
"virology",  
"world_religions"  
]
```

BBH

```
BBH_SUBTASKS = [  
    "boolean_expressions",  
    "causal_judgement",  
    "date_understanding",  
    "disambiguation_qa",  
    "dyck_languages",  
    "formal_fallacies",  
    "geometric_shapes",
```

```
"hyperbaton",
"logical_deduction_five_objects",
"logical_deduction_seven_objects",
"logical_deduction_three_objects",
"movie_recommendation",
"multistep_arithmetic_two",
"navigate",
"object_counting",
"penguins_in_a_table",
"reasoning_about_colored_objects",
"ruin_names",
"salient_translation_error_detection",
"snarks",
"sports_understanding",
"temporal_sequences",
"tracking_shuffled_objects_five_objects",
"tracking_shuffled_objects_seven_objects",
"tracking_shuffled_objects_three_objects",
"web_of_lies",
"word_sorting"
]
```

Math

```
MATH_SUBTASKS = [
    "algebra",
    "counting_and_probability",
    "geometry",
    "intermediate_algebra",
    "number_theory",
    "prealgebra",
    "precalculus",
```

MMMU

```
        MATH_SUBTASKS = [
"Accounting",
"Agriculture",
"Architecture_and_Engineering",
"Art",
"Art_Theory",
"Basic_Medical_Science",
```

```
"Biology",  
"Chemistry",  
"Clinical_Medicine",  
"Computer_Science",  
"Design",  
"Diagnostics_and_Laboratory_Medicine",  
"Economics",  
"Electronics",  
"Energy_and_Power",  
"Finance",  
"Geography",  
"History",  
"Literature",  
"Manage",  
"Marketing",  
"Materials",  
"Math",  
"Mechanical_Engineering",  
"Music",  
"Pharmacy",  
"Physics",  
"Psychology",  
"Public_Health",  
"Sociology",
```

使用各種評估方法和指標來評估自訂的 Nova 模型。

在本節中：

- [推理模型評估](#)
- [RFT 評估](#)
- [實作獎勵函數](#)

推理模型評估

概觀

推理模型支援可使用推理能力的 Nova 模型進行評估，這些模型在產生最終回應之前執行明確的內部推理。此功能會透過 `reasoning_effort` 參數使用 API 層級控制來動態啟用或停用推理功能，進而改善複雜分析任務的回應品質。

支援的模型

- amazon.nova-2-lite-v1 : 0 : 256k

配方組態

將 `reasoning_effort` 參數新增至配方的 `inference` 區段，以啟用推理：

```
run:
  name: reasoning-eval-job-name                # [MODIFIABLE] Unique
  identifier for your evaluation job
  model_type: amazon.nova-2-lite-v1:0:256k      # [FIXED] Must be a reasoning-
supported model
  model_name_or_path: nova-lite-2/prod          # [FIXED] Path to model
checkpoint or identifier
  replicas: 1                                   # [MODIFIABLE] Number of
replicas for SageMaker Training job
  data_s3_path: ""                             # [MODIFIABLE] Leave empty for
SageMaker Training job; optional for SageMaker HyperPod job
  output_s3_path: ""                           # [MODIFIABLE] Output path for
SageMaker HyperPod job (not compatible with SageMaker Training jobs)

evaluation:
  task: mmlu                                    # [MODIFIABLE] Evaluation task

  strategy: zs_cot                             # [MODIFIABLE] Evaluation
strategy
  metric: accuracy                             # [MODIFIABLE] Metric
calculation method

inference:
  reasoning_effort: high                       # [MODIFIABLE] Enables
reasoning mode; options: low/high or null to disable
  max_new_tokens: 32768                       # [MODIFIABLE] Maximum tokens
to generate, recommended value when reasoning_effort set to high
  top_k: -1                                    # [MODIFIABLE] Top-k sampling
parameter
  top_p: 1.0                                  # [MODIFIABLE] Nucleus
sampling parameter
  temperature: 0                              # [MODIFIABLE] Sampling
temperature (0 = deterministic)
```


使用 reasoning_effort 參數

reasoning_effort 參數控制可推理模型的推理行為。

先決條件

- 模型相容性 – reasoning_effort 僅在 model_type 指定可推理的模型時設定（目前為 amazon.nova-2-lite-v1:0:256k）
- 錯誤處理 – 使用 reasoning_effort 搭配不支援的模型會失敗 ConfigValidationError: "Reasoning mode is enabled but model '{model_type}' does not support reasoning. Please use a reasoning-capable model or disable reasoning mode."

可用選項

選項	Behavior (行為)	字符限制	使用案例
null (預設)	停用推理模式	N/A	無推理額外負荷的標準評估
low	啟用具有限制條件的推理	用於內部推理的 4,000 個字符	需要簡潔推理的案例；針對速度和成本進行最佳化
high	啟用無限制的推理	內部推理沒有字符限制	需要廣泛分析和 step-by-step 推理的複雜問題

訓練模式	可用選項	如何設定
SFT (受監督的微調)	僅限高或關閉	使用 reasoning_enabled: true (高) 或 reasoning_enabled: false (關閉)
RFT (強化微調)	低、高或關閉	使用 reasoning_effort: 低或 reasoning_effort: 高。省略要停用的欄位。
評估	低、高或關閉	使用 reasoning_effort: 低或 reasoning_effort: 高。使用 null 來停用。

何時啟用推理

使用的推理模式 (**low** 或 **high**)

- 複雜的問題解決任務（數學、邏輯拼圖、編碼）
- 需要中繼推理的多步驟分析問題
- 詳細說明或step-by-step思考可提高準確性的任務
- 回應品質優先於速度的情況

針對 使用非原因模式 (**null** 或 省略參數)

- 簡單問答或事實查詢
- 創意撰寫任務
- 當更快的回應時間至關重要時
- 應排除推理開銷的績效基準
- 推理無法改善任務效能時的成本最佳化

疑難排解

錯誤：「啟用原因模式，但模型不支援推理」

原因： `reasoning_effort` 參數設定為非空值，但指定的 `model_type` 不支援推理。

解決方法：

- 確認您的模型類型為 `amazon.nova-2-lite-v1:0:256k`
- 如果使用不同的模型，請切換到具有推理功能的模型，或從配方中移除 `reasoning_effort` 參數

RFT 評估

什麼是 RFT 評估？

RFT 評估可讓您在強化學習訓練之前、期間或之後，使用自訂獎勵函數評估模型的效能。與使用預先定義指標的標準評估不同，RFT 評估可讓您透過 Lambda 函數定義自己的成功條件，該函數會根據您的特定需求對模型輸出進行評分。

為什麼使用 RFT 評估？

評估對於判斷 RL 微調程序是否具有下列項目至關重要：

- 改善模型與特定使用案例和人力值的一致性
- 維護或改善關鍵任務的模型功能
- 避免意外的副作用，例如事實性降低、詳細程度增加或其他任務的效能降低
- 符合獎勵函數定義的自訂成功條件

何時使用 RFT 評估

在這些案例中使用 RFT 評估：

- RFT 訓練之前：在您的評估資料集上建立基準指標
- 在 RFT 訓練期間：使用中繼檢查點監控訓練進度
- RFT 訓練後：驗證最終模型是否符合您的需求
- 比較模型：使用一致的獎勵條件評估多個模型版本

Note

當您需要自訂的網域特定指標時，請使用 RFT 評估。對於一般用途評估（準確性、複雜度、BLEU），請使用標準評估方法。

資料格式要求

輸入資料結構

RFT 評估輸入資料必須遵循 OpenAI 強化微調格式。每個範例都是 JSON 物件，其中包含：

- `messages` – 具有 `system` 和 `user` 角色的對話轉場陣列
- `reference_answer` – 獎勵函數用於評分之預期輸出或 Ground Truth 資料

資料格式範例

```
{
```

```

"messages": [
  {
    "role": "user",
    "content": [
      {
        "type": "text",
        "text": "Solve for x. Return only JSON like {\"x\": <number>}. Equation: 2x + 5 = 13"
      }
    ]
  },
  {
    "role": "assistant",
    "content": [
      {
        "type": "text",
        "text": "Solve for x. Return only JSON like {\"x\": 4}."
      }
    ]
  }
],
"reference_answer": {
  "x": 4
}
}

```

目前限制

- 僅限文字：不支援多模態輸入（影像、音訊、視訊）
- 單轉對話：僅支援單一使用者訊息（無多轉對話）
- JSON 格式：輸入資料必須是 JSONL 格式（每行一個 JSON 物件）
- 模型輸出：在從指定模型產生的完成時執行評估

準備您的評估配方

範例筆記本

如需完整範例，請參閱[評估筆記本](#)。

範例配方組態

```

run:
  name: nova-lite-rft-eval-job
  model_type: amazon.nova-lite-v1:0:300k
  model_name_or_path: s3://escrow_bucket/model_location # [MODIFIABLE] S3 path to your
  model or model identifier
  replicas: 1 # [MODIFIABLE] For SageMaker Training jobs only; fixed for HyperPod jobs

  data_s3_path: "" # [REQUIRED FOR HYPERPOD] Leave empty for SageMaker Training jobs
  and use TrainingInput in sagemaker python SDK

```

```

output_s3_path: "" # [REQUIRED] Output artifact S3 path for evaluation results

evaluation:
  task: rft_eval # [FIXED] Do not modify
  strategy: rft_eval # [FIXED] Do not modify
  metric: all # [FIXED] Do not modify

# Inference Configuration
inference:
  max_new_tokens: 8192 # [MODIFIABLE] Maximum tokens to generate
  top_k: -1 # [MODIFIABLE] Top-k sampling parameter
  top_p: 1.0 # [MODIFIABLE] Nucleus sampling parameter
  temperature: 0 # [MODIFIABLE] Sampling temperature (0 = deterministic)
  top_logprobs: 0 # [MODIFIABLE] Set between 1-20 to enable logprobs output

# =====
# Bring Your Own Reinforcement Learning Environment
# =====
rl_env:
  reward_lambda_arn: arn:aws:lambda:<region>:<account_id>:function:<reward-function-
name>

```

預設獎勵函數

兩個預設獎勵函數 (prime_code 和 prime_math) 可作為 Lambda 層使用，以便與您的 RFT Lambda 函數輕鬆整合。

概觀

這些預設函數提供out-of-the-box評估功能：

- prime_code – 程式碼產生和正確性評估
- prime_math – 數學推理和問題解決評估

快速設定

1. 從 [nova-custom-eval-sdk 版本](#) 下載 Lambda layer。
2. 使用 AWS 命令列界面 () 發佈 Lambda 層AWS CLI：

```

aws lambda publish-layer-version \
  --layer-name preset-function-layer \

```

```
--description "Preset reward function layer with dependencies" \
--zip-file fileb://universal_reward_layer.zip \
--compatible-runtimes python3.9 python3.10 python3.11 python3.12 \
--compatible-architectures x86_64 arm64
```

3. 在 AWS 管理主控台中將 layer 新增至 Lambda 函數（從自訂 layer 選取 preset-function-layer，並針對凹凸相依性新增 AWSSDKPandas-Python312）。
4. 在 Lambda 程式碼中匯入和使用：

```
from prime_code import compute_score # For code evaluation
from prime_math import compute_score # For math evaluation
```

prime_code 函數

對測試案例執行程式碼並測量正確性，藉此評估 Python 程式碼產生任務。

範例輸入資料集格式

```
{"messages":[{"role":"user","content":"Write a function that returns the sum of
two numbers."}], "reference_answer":{"inputs":["3\n5","10\n-2","0\n0"],"outputs":
["8","8","0"]}}
```

```
{"messages":[{"role":"user","content":"Write a function to check if a number
is even."}], "reference_answer":{"inputs":["4","7","0","-2"],"outputs":
["True","False","True","True"]}}
```

主要功能

- 從 Markdown 程式碼區塊自動擷取程式碼
- 函數偵測和以呼叫為基礎的測試
- 具有逾時保護的測試案例執行
- 語法驗證和編譯檢查
- 使用追蹤傳回的詳細錯誤報告

prime_math 函數

使用符號數學支援評估數學推理和問題解決功能。

輸入格式

```
{"messages":[{"role":"user","content":"What is the derivative of x^2 + 3x?."}], "reference_answer":"2*x + 3"}
```

主要功能

- 使用 SymPy 的符號數學評估
- 多重答案格式 (LaTeX、純文字、符號)
- 數學相等性檢查
- 表達式標準化和簡化

資料格式要求

用於程式碼評估

- 輸入：函數引數陣列（屬性類型：整數、字串等）
- 輸出：預期傳回值的陣列（屬性類型：布林值、數字等）
- 程式碼：必須是具有清晰函數定義的 Python

用於數學評估

- 參考答案：數學表達式或數值
- 回應：可以是 LaTeX、純文字或符號表示法
- 對等：以符號方式檢查，而不只是字串比對

最佳實務

- 在測試案例中使用適當的資料類型（整數與字串、布林值與 "True"）
- 在程式碼問題中提供明確的函數簽章
- 在測試輸入中包含邊緣案例（零、負數、空輸入）
- 在參考答案中一致地格式化數學表達式
- 在部署之前，使用範例資料測試獎勵函數

錯誤處理

這兩個函數都包含強大的錯誤處理功能：

- 產生的程式碼中的編譯錯誤
- 執行期間的執行時間例外狀況
- 格式不正確的輸入資料
- 無限迴圈的逾時案例
- 無效的數學表達式

建立獎勵函數

Lambda ARN 要求

您的 Lambda ARN 必須遵循此格式：

```
"arn:aws:lambda:*:*:function:*SageMaker*"
```

如果 Lambda 沒有此命名機制，任務會失敗並出現此錯誤：

```
[ERROR] Unexpected error: lambda_arn must contain one of: ['SageMaker', 'sagemaker', 'Sagemaker'] when running on SMHP platform (Key: lambda_arn)
```

Lambda 請求格式

您的 Lambda 函數會以此格式接收資料：

```
[
  {
    "id": "sample-001",
    "messages": [
      {
        "role": "user",
        "content": [
          {
            "type": "text",
            "text": "Do you have a dedicated security team?"
          }
        ]
      },
      {
        "role": "nova_assistant",
        "content": [
          {
```



```

        "type": "text",
        "text": "As an AI developed by Company, I don't have a dedicated security
team..."
    }
]
},
"reference_answer": {
    "compliant": "No",
    "explanation": "As an AI developed by Company, I do not have a traditional
security team..."
}
}
]
```

Note

訊息結構包含巢狀content陣列，符合輸入資料格式。最後一個具有 角色的訊息nova_assistant包含模型產生的回應。

Lambda 回應格式

您的 Lambda 函數必須以下列格式傳回資料：

```

[
  {
    "id": "sample-001",
    "aggregate_reward_score": 0.75,
    "metrics_list": [
      {
        "name": "accuracy",
        "value": 0.85,
        "type": "Metric"
      },
      {
        "name": "fluency",
        "value": 0.90,
        "type": "Reward"
      }
    ]
  }
]
```

```
]
```

回應欄位

- `id` – 必須符合輸入範例 ID
- `aggregate_reward_score` – 整體分數 (通常是 0.0 到 1.0)
- `metrics_list` – 具有下列項目的個別指標陣列：
 - `name` – 指標識別符 (例如 "accuracy"、"fluency")
 - `value` – 指標分數 (通常是 0.0 到 1.0)
 - `type` – 「指標」(用於報告) 或 「獎勵」(用於訓練)

IAM 許可

所需的許可

您的 SageMaker 執行角色必須具有叫用 Lambda 函數的許可。將此政策新增至 SageMaker 執行角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:region:account-id:function:function-name"
    }
  ]
}
```

Lambda 執行角色

您的 Lambda 函數的執行角色需要基本的 Lambda 執行許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

    "Effect": "Allow",
    "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
    ],
    "Resource": "arn:aws:logs:*:*:*"
  }
]
}
```

如果您的 Lambda 函數存取其他 AWS 服務（例如，用於參考資料的 S3、用於記錄的 DynamoDB），請將這些許可新增至 Lambda 執行角色。

執行評估任務

1. 準備您的資料 – 根據資料格式要求格式化評估資料，並將您的 JSONL 檔案上傳至 S3：`s3://your-bucket/eval-data/eval_data.jsonl`
2. 設定您的配方 – 使用您的組態更新範例配方：
 - `model_name_or_path` 設定為您的模型位置
 - `lambda_arn` 設定為您的獎勵函數 ARN
 - `output_s3_path` 設定為您想要的輸出位置
 - 視需要調整 `inference` 參數

將配方儲存為 `rft_eval_recipe.yaml`

3. 執行評估 – 使用提供的筆記本執行評估任務：[評估筆記本](#)
4. 監控進度 – 透過下列方式監控您的評估任務：
 - SageMaker 主控台：檢查任務狀態和日誌
 - CloudWatch Logs：檢視詳細的執行日誌
 - Lambda 日誌：偵錯獎勵函數問題

了解評估結果

輸出格式

評估任務會以 JSONL 格式將結果輸出至您指定的 S3 位置。每行包含一個範例的評估結果：

```
{
```

```
{
  "id": "sample-001",
  "aggregate_reward_score": 0.75,
  "metrics_list": [
    {
      "name": "accuracy",
      "value": 0.85,
      "type": "Metric"
    },
    {
      "name": "fluency",
      "value": 0.90,
      "type": "Reward"
    }
  ]
}
```

Note

RFT 評估任務輸出與 Lambda 回應格式相同。評估服務會在不修改的情況下傳遞 Lambda 函數的回應，確保獎勵計算與最終結果之間的一致性。

解譯結果

彙總獎勵分數

- 範圍：通常為 0.0（最差）到 1.0（最佳），但取決於您的實作
- 目的：總結整體效能的單一數字
- 用量：比較模型，追蹤訓練的改善

個別指標

- 指標類型：分析的資訊指標
- 獎勵類型：RFT 訓練期間使用的指標
- 解譯：較高的值通常表示效能更好（除非您設計反向指標）

效能基準

什麼構成「良好」效能取決於您的使用案例：

分數範圍	解譯	Action
0.8 - 1.0	卓越	模型已準備好進行部署
0.6 - 0.8	好	次要改善可能有益
0.4 - 0.6	公平	需要大幅改善
0.0 - 0.4	不佳	檢閱訓練資料和獎勵函數

Important

這些是一般準則。根據業務需求、基準模型效能、領域特定限制，以及進一步訓練的成本利益分析，定義您自己的閾值。

疑難排解

常見問題

問題	原因	解決方案
Lambda 逾時	複雜的獎勵計算	增加 Lambda 逾時或最佳化函數
許可遭拒	缺少 IAM 許可	驗證 SageMaker 角色可以叫用 Lambda
不一致的分數	非確定性獎勵函數	使用固定種子或確定性邏輯
缺少結果	未攔截 Lambda 錯誤	在 Lambda 中新增全面的錯誤處理

除錯檢查清單

- 使用巢狀內容陣列，確認輸入資料遵循正確的格式
- 確認 Lambda ARN 正確且函數已部署
- 檢查 SageMaker 的 IAM 許可 → Lambda 調用
- 檢閱 CloudWatch 日誌是否有 Lambda 錯誤

- 驗證 Lambda 回應符合預期的格式

最佳實務

- 啟動簡單：從基本獎勵函數開始並反覆運算
- 分別測試 Lambda：在完整評估之前使用 Lambda 測試事件
- 在小型資料集上驗證：在完整資料集之前對子集執行評估
- 版本控制：追蹤獎勵函數版本與模型版本
- 監控成本：Lambda 調用和運算時間會影響成本
- Log Extensively：使用 Lambda 中的列印陳述式進行偵錯
- 適當設定逾時：在耐心與成本之間取得平衡
- 文件指標：明確定義每個指標的指標

後續步驟

完成 RFT 評估後：

- 如果結果令人滿意：將模型部署到生產環境
- 如果需要改進：
 - 調整獎勵函數
 - 收集更多訓練資料
 - 修改訓練超參數
 - 執行額外的 RFT 訓練反覆運算
- 持續監控：使用新資料定期重新評估

實作獎勵函數

概觀

獎勵函數（也稱為計分器或分級器）是評估模型回應並提供意見回饋訊號以進行訓練的核心元件。它必須實作為接受模型回應並傳回獎勵分數的 Lambda 函數。

介面格式

您的獎勵函數必須接受並傳回下列格式的資料：

訓練的輸入範例

```
{
  "messages": [
    {
      "role": "user",
      "content": "Do you have a dedicated security team?"
    }
  ],
  "reference_answer": {
    "compliant": "No",
    "explanation": "As an AI developed by Company, I do not have a traditional security team..."
  }
}
```

獎勵 lambda 的範例承載

容器會自動轉換您的資料，然後再將資料傳送到 Lambda 函數，方法如下：

1. 為每個提示產生模型回應
2. 將助理轉彎（產生的回應）附加到訊息陣列
3. 新增唯一id欄位以進行追蹤

您的 Lambda 函數將以此轉換格式接收資料：

```
{
  "id": "123",
  "messages": [
    {
      "role": "user",
      "content": "Do you have a dedicated security team?"
    },
    {
      "role": "assistant",
      "content": "As an AI developed by Amazon, I don not have a dedicated security team..."
    }
  ],
  # Following section will be same as your training dataset sample
  "reference_answer": {
```

```

        "compliant": "No",
        "explanation": "As an AI developed by Company, I do not have a traditional
security team..."
    }
}

```

獎勵 Lambda 合約

```

def lambda_handler(event, context):
    return lambda_grader(event)

def lambda_grader(samples: list[dict]) -> list[dict]:
    """
    Args:
        samples: List of dictionaries in OpenAI format

    Example input:
    {
        "id": "123",
        "messages": [
            {
                "role": "user",
                "content": "Do you have a dedicated security team?"
            },
            {
                "role": "assistant",
                "content": "As an AI developed by Company, I don't have a
dedicated security team..."
            }
        ],
        # This section will be same as your training dataset
        "reference_answer": {
            "compliant": "No",
            "explanation": "As an AI developed by Company, I do not have a
traditional security team..."
        }
    }

    Returns:
        List of dictionaries with reward scores:
    {
        "id": str,                                # Same id as input sample
        "aggregate_reward_score": float,          # Overall score for the sample
    }
    """

```



```

        "metrics_list": [
            {
                "name": str,
                "value": float,
                "type": str
            }
        ]
    }
}

```

OPTIONAL: Component scores

Name of the component score

Value of the component score

"Reward" or "Metric"

輸入和輸出欄位

輸入欄位

欄位	Description	其他備註
id	範例的唯一識別符	在輸出中回呼。字串格式
messages	OpenAI 格式的排序聊天歷史記錄	訊息物件陣列
messages 【 】.role	訊息的發言者	常見值："user"、"assistant"、"system"
messages 【 】.content	訊息的文字內容	純文字的字串
**中繼資料	協助分級的自由格式資訊	物件；從訓練資料傳遞的選用欄位

輸出欄位

欄位	Description	其他備註
id	與輸入範例相同的識別符	必須符合輸入
aggregate _reward_score	範例的整體分數	浮動（例如 0.0–1.0 或任務定義範圍）
metrics_list	組成彙總的元件分數	指標物件陣列

技術限制條件

- 逾時限制 – 每次 Lambda 調用最多 15 分鐘執行時間
- 並行 – 必須處理 `rollout_worker_replicas * 64` 並行請求
- 可靠性 – 必須實作適當的錯誤處理，並一致地傳回有效的分數
- 效能 – 最佳化快速執行（秒，而非分鐘），以啟用高效訓練

最佳實務

- 將外部 API 呼叫降至最低
- 使用有效率的演算法和資料結構
- 實作暫時性失敗的重試邏輯
- 快取可重複使用的運算
- 在訓練之前徹底測試，以確保無錯誤執行

使用自訂獎勵函數

當您有任務特定的評估條件時，請實作自訂獎勵函數：

- 定義評估條件 - 決定什麼可以為您的任務做出良好的回應
- 實作 Lambda 函數 – 依照介面格式建立 Lambda 函數
- 在本機測試 – 驗證您的函數傳回正確的範例輸入分數
- 部署至 AWS - 部署您的 Lambda 並記下 ARN
- 設定配方 – 將 Lambda ARN 新增至配方的 `reward_lambda_arn` 欄位
- 使用小型資料集進行測試 – 以最少的資料執行 RFT 以驗證整合

IAM 許可

所需的許可

您的 SageMaker 執行角色必須具有叫用 Lambda 函數的許可。將此政策新增至 SageMaker 執行角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": [
    "lambda:InvokeFunction"
  ],
  "Resource": "arn:aws:lambda:region:account-id:function:function-name"
}
```

Lambda 執行角色

您的 Lambda 函數的執行角色需要基本的 Lambda 執行許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

其他許可：如果您的 Lambda 函數存取其他 AWS 服務（例如，用於參考資料的 S3、用於記錄的 DynamoDB），請將這些許可新增至 Lambda 執行角色。

範例：LLM 做為判斷獎勵函數

此範例示範如何使用 Amazon Bedrock 模型做為判斷，透過將模型回應與參考答案進行比較來評估模型回應。此 Lambda 範本提供架構，讓客戶實作對 Amazon Bedrock 的呼叫，以進行推論請求，以處理判斷評估。Lambda 函數會維護與其他獎勵函數相同的輸入/輸出合約。

實作

此 Lambda 函數會實作兩階段評估程序：會從傳入的範例中 `lambda_handler` 擷取模型回應和參考答案，然後 `lambda_graded` 函數會呼叫 Amazon Bedrock 來對它們之間的語意相似性進行評分。實作包

括強大的錯誤處理功能，可自動重試暫時性失敗，並支援靈活的參考答案格式（字串和結構化字典格式）。

實作詳細資訊：

- 重試邏輯：針對調節例外狀況實作指數退避 (1、2、4)，以處理 Bedrock API 速率限制
- 錯誤處理：針對失敗的評估傳回 0.0 的分數，而不是引發例外狀況
- 確定性評分：使用溫度=0.0 來確保評估之間的分數一致
- 彈性參考格式：自動處理字串和字典參考答案
- 分數限制：確保所有分數都落在有效的 **【0.0，1.0】** 範圍內
- 模型無關：將 JUDGE_MODEL_ID 變更為使用任何 Amazon Bedrock 模型 (Nova、Llama、Mistral 等)

```
"""
LLM Judge Lambda POC - Working implementation using Amazon Bedrock
"""

import json
import time
import boto3

bedrock_runtime = boto3.client('bedrock-runtime', region_name='us-east-1')
JUDGE_MODEL_ID = "anthropic.claude-3-5-sonnet-20240620-v1:0"
SYSTEM_PROMPT = "You must output ONLY a number between 0.0 and 1.0. No explanations, no text, just the number."

JUDGE_PROMPT_TEMPLATE = """Compare the following two responses and rate how similar they are on a scale of 0.0 to 1.0, where:
- 1.0 means the responses are semantically equivalent (same meaning, even if worded differently)
- 0.5 means the responses are partially similar
- 0.0 means the responses are completely different or contradictory

Response A: {response_a}

Response B: {response_b}

Output ONLY a number between 0.0 and 1.0. No explanations."""
```

```

def lambda_graded(response_a: str, response_b: str, max_retries: int = 3) -> float:
    """Call Bedrock to compare responses and return similarity score."""
    prompt = JUDGE_PROMPT_TEMPLATE.format(response_a=response_a, response_b=response_b)

    for attempt in range(max_retries):
        try:
            response = bedrock_runtime.converse(
                modelId=JUDGE_MODEL_ID,
                messages=[{"role": "user", "content": [{"text": prompt}]}],
                system=[{"text": SYSTEM_PROMPT}],
                inferenceConfig={"temperature": 0.0, "maxTokens": 10}
            )
            print(f"Bedrock call successful: {response}")
            output = response['output']['message']['content'][0]['text'].strip()
            score = float(output)
            print(f"Score parsed: {score}")
            return max(0.0, min(1.0, score))

        except Exception as e:
            if "ThrottlingException" in str(e) and attempt < max_retries - 1:
                time.sleep(2 ** attempt)
            else:
                print(f"Bedrock call failed: {e}")
                return None

    return None

def lambda_handler(event, context):
    """AWS Lambda handler - processes samples from RFTEvalInvoker."""
    try:
        samples = event if isinstance(event, list) else [event]
        results = []

        for sample in samples:
            sample_id = sample.get("id", "unknown")
            messages = sample.get("messages", [])

            # Extract assistant response (response A)
            response_a = ""
            for msg in messages:
                if msg.get("role") in ["assistant", "nova_assistant"]:
                    response_a = msg.get("content", "")
                    break

```

```

        # Extract reference answer from root level (no longer in metadata)
        reference_answer = sample.get("reference_answer", "")

        # Handle both string and dict reference_answer formats
        if isinstance(reference_answer, dict):
            # If reference_answer is a dict, extract the explanation or compliant
            response_b = reference_answer.get("explanation",
            reference_answer.get("compliant", ""))
        else:
            response_b = reference_answer

        if not response_a or not response_b:
            results.append({
                "id": sample_id,
                "aggregate_reward_score": 0.0,
                "metrics_list": [{"name": "similarity_score", "value": 0.0, "type":
"Metric"}]
            })
            continue

        # Get similarity score
        score = lambda_graded(response_a, response_b)

        results.append({
            "id": sample_id,
            "aggregate_reward_score": score,
            "metrics_list": [
                {
                    "name": "similarity_score",
                    "value": score,
                    "type": "Metric"
                }
            ]
        })

    return {"statusCode": 200, "body": json.dumps(results)}

except Exception as e:
    print(f"Error: {e}")
    return {"statusCode": 500, "body": json.dumps({"error": str(e)})}

```

輸入格式

Lambda 會收到與其他獎勵函數相同的輸入格式：

```
{
  "id": "sample-001",
  "messages": [
    {
      "role": "user",
      "content": "Do you have a dedicated security team?"
    },
    {
      "role": "assistant",
      "content": "As an AI developed by Amazon, I don't have a dedicated security team..."
    }
  ],
  "reference_answer": {
    "compliant": "No",
    "explanation": "As an AI developed by Company, I do not have a traditional security team..."
  },
  "my_custom_field": "custom_value"
}
```

輸出格式

```
{
  "id": "sample-001",
  "aggregate_reward_score": 0.85,
  "metrics_list": [
    {
      "name": "similarity_score",
      "value": 0.85,
      "type": "Metric"
    }
  ]
}
```

部署考量

您可能還需要根據所選模型的功能和 API 格式來調整提示範本和推論參數。

- IAM 許可：Lambda 執行角色必須具有所選模型的 `bedrock:InvokeModel` 許可
- 逾時：將 Lambda 逾時設定為至少 60 秒，以適應 Bedrock API 延遲和重試
- 區域：部署在可使用您所選 Bedrock 模型的區域
- 成本：監控 Bedrock API 用量，因為每次評估都會對每個範例進行一次 API 呼叫
- 輸送量：對於大規模評估，請求提高 Bedrock 配額以避免限流

增加 Bedrock 輸送量

如果您在評估期間遇到限流，請增加 Bedrock 模型配額：

- 導覽至 AWS Service Quotas 主控台
- 搜尋 "Bedrock" 並選擇您的區域
- 尋找所選模型的配額（例如「Claude 3.5 Sonnet 每分鐘叫用次數」）
- 按一下「請求增加配額」並指定所需的輸送量
- 提供增加的理由（例如，「RFT 評估工作負載」）

Lambda 的內建重試邏輯會偶爾處理限流，但持續的大量評估需要增加適當的配額。

必要的 IAM 政策：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel"
      ],
      "Resource": "arn:aws:bedrock:*::foundation-model/*"
    }
  ]
}
```

反覆訓練

反覆訓練是一種系統性方法，可透過多個訓練週期微調模型，其中每個回合都是以先前的檢查點為基礎，方法是解決透過評估發現的特定弱點。此方法透過整合處理失敗模式、適應不斷變化的需求，以及

逐步驗證增強功能，而不是遞交單一長訓練執行，來實現有針對性的改善，以建立效能模型。程序通常會遵循諸如 SFT（監督式微調）和 RFT（以獎勵為基礎的微調）等模式，其中包含存放受 AWS 管託管 S3 儲存貯體的檢查點，可用於後續訓練反覆運算，同時在整個管道中保持模型類型和訓練技術的一致性。

如需更多詳細資訊，請參閱 [???](#)。

SageMaker HyperPod 上的 Amazon Nova 自訂

您可以使用 Amazon Nova [配方自訂 Amazon Nova 模型](#)，包括增強型 [Amazon Nova 2.0 模型](#)，並在 Hyperpod 上對其進行訓練。配方是 YAML 組態檔案，可將有關如何執行模型自訂任務的詳細資訊提供給 SageMaker AI。SageMaker HyperPod 支援兩種類型的服務：Forge 和非 Forge。

Hyperpod 提供具有最佳化 GPU 執行個體和 Amazon FSx for Lustre 儲存的高效能運算，透過與 TensorBoard、靈活的檢查點管理可反覆改進，無縫部署到 Amazon Bedrock 以進行推論，和有效率的可擴展多節點分散式訓練，共同為組織提供安全、執行者、和彈性的環境，可根據其特定業務需求量身打造 Amazon Nova 模型。

SageMaker HyperPod 上的 Amazon Nova 自訂會在服務管理的 Amazon S3 儲存貯體中存放模型成品，包括模型檢查點。服務受管儲存貯體中的成品會使用 SageMaker AI 受管 AWS KMS 金鑰加密。服務受管 Amazon S3 儲存貯體目前不支援使用客戶受管 KMS 金鑰進行資料加密。您可以使用此檢查點位置進行評估任務或 Amazon Bedrock 推論。

標準定價適用於運算執行個體、Amazon S3 儲存體和 FSx for Lustre。如需定價詳細資訊，請參閱 [Hyperpod 定價](#)、[Amazon S3 定價](#)和 [FSx for Lustre 定價](#)。

Amazon Nova 2 模型的運算需求

下表摘要說明 Amazon Nova 2 模型的 SageMaker HyperPod 和 SageMaker AI 訓練任務訓練的運算需求。

Nova 2 訓練需求

訓練技術	最小執行個體	執行個體類型	GPU 計數	備註	支援模型
SFT (LoRA)	4	P5.48xlarge	16	具參數效率的微調	Nova 2 Lite
SFT（完整排名）	4	P5.48xlarge	32	完整模型微調	Nova 2 Lite

訓練技術	最小執行個體	執行個體類型	GPU 計數	備註	支援模型
SageMaker 訓練任務 (LoRA) 上的 RFT	2	P5.48xlarge	16	AWS 環境中 的自訂獎勵函 數	Nova 2 Lite
SageMaker 訓練任務的 RFT (完整排 名)	4	P5.48xlarge	32	32K 內容長度	Nova 2 Lite
SageMaker HyperPod 上 的 RFT	8	P5.48xlarge	64	預設 8192 內 容長度	Nova 2 Lite
CPT	4	P5.48xlarge	16	每天處理每個 執行個體大約 400M個字符	Nova 2 Lite

若要在 Hyperpod 上最佳化 Amazon Nova 模型自訂工作流程，請遵循這些建議的最佳實務，以有效率地訓練、資源管理和成功部署模型。

Amazon Nova 自訂的最佳實務

概觀

本節提供自訂技術的概觀，並協助您選擇符合您需求和可用資料的最佳方法。

LLM 訓練的兩個階段

大型語言模型訓練包含兩個主要階段：訓練前和訓練後。在預先訓練期間，模型會處理原始文字權杖，並針對下一個權杖預測進行最佳化。此程序會建立模式完成器，從 Web 和策劃的文字中吸收語法、語意、事實和推理模式。不過，預先訓練的模型不了解指示、使用者目標或內容適當的行為。它以適合其訓練分佈的任何樣式繼續文字。預先訓練的模型會自動完成，而不是遵循指示、產生不一致的格式，並且可以從訓練資料反映不良的偏差或不安全的內容。訓練前會建置一般能力，而不是任務實用性。

訓練後會將模式完成器轉換為有用的助理。您會執行多輪受管微調 (SFT)，以教導模型遵循指示、遵循結構描述和政策、呼叫工具，並透過模擬高品質的示範來產生可靠的輸出。此對齊會教導模型以任務而

非文字的形式回應提示以繼續。然後，套用強化微調 (RFT)，使用可衡量的意見回饋（例如驗證器或 LLM-as-a-judge）、平衡準確性與簡潔性、安全性與涵蓋範圍，或在限制下進行多步驟推理等權衡。實際上，您會在週期中替換 SFT 和 RFT，將預先訓練的模型塑造為可靠且符合政策的系統，以一致地執行複雜的任務。

選擇正確的自訂方法

在本節中，我們將介紹訓練後自訂策略：RFT 和 SFT。

強化微調 (RFT)

強化微調透過意見回饋訊號改善模型效能，這些訊號是可測量的分數或獎勵，表示回應品質，而不是直接監督並準確回答。與從輸入輸出對中學習的傳統監督式微調不同，RFT 使用獎勵函數來評估模型回應，並反覆最佳化模型以最大化這些獎勵。此方法非常適合定義確切正確輸出具有挑戰性的任務，但您可以可靠地測量回應品質。RFT 可讓模型透過試驗和意見回饋來學習複雜的行為和偏好，因此非常適合需要細微決策、創意問題解決或遵守您可以程式設計方式評估的特定品質標準的應用程式。例如，回答複雜的法律問題是 RFT 的理想使用案例，因為您想要教導模型如何更準確地回答問題。

運作方式

在強化微調中，您可以從指令調校的基準開始，並將每個提示視為小型競賽。對於指定的輸入，您從模型中抽樣一些候選答案，使用獎勵函數對每個答案進行評分，然後在該群組中對其進行排名。更新步驟會調節模型，使分數較高的候選者在下次的可能性更高，分數較低的候選者的可能性更低，而stay-close-to-baseline的限制則可防止行為偏離或變得模糊或攻擊。您可以在許多提示上重複此迴圈、重新整理硬案例、在您看到入侵時收緊驗證器或判斷摩擦，以及持續追蹤任務指標。

何時使用 RFT

受益於 RFT 的任務具有多種特徵。即使單一正確輸出難以指定，它們仍有可測量的成功訊號。他們認可部分點數或分級品質，因此您可以在提示中或使用獎勵函數，對較差的答案進行排名。它們涉及多個必須平衡的目標（例如簡潔、清晰、安全或成本的準確性）。它們需要遵守您可以程式設計方式檢查的明確限制。它們在可觀察結果的工具媒介型或環境型設定中操作（成功或失敗、延遲、資源使用）。它們發生在低標籤的機制中，其中收集黃金目標很昂貴，但自動化或以摩擦為基礎的意見回饋有很多。當您可以將品質轉換為可靠的純量或排名，並希望模型優先放大分數較高的行為，而不需要詳盡的標記目標時，RFT 最有效。

考慮下列情況下的其他方法：

- 您有大量且可靠的標籤輸入輸出對 – 使用 SFT

- 主要差距是知識或行話 – 使用擷取擴增產生 (RAG)
- 您的獎勵訊號發出吵雜或不可靠，而且您無法使用更好的 Rubrics 或 Checker 進行修正 – 在 RFT 之前先穩定該訊號

何時不使用 RFT

避免在這些情況下使用 RFT：

- 您可以便宜地產生可靠的標籤輸入輸出對 (SFT 更簡單、更便宜且更穩定)
- 差距是知識或行話，而不是行為（使用 RAG）
- 您的獎勵訊號是雜訊、稀疏、容易玩遊戲，或昂貴或運算速度緩慢（請先修正評估者）
- 基準效能接近零（在最佳化偏好設定之前使用 SFT 引導）
- 任務具有確定性結構描述、嚴格格式或單一正確答案 (SFT 或規則型驗證效果更好)
- 緊迫的延遲或成本預算無法吸收額外取樣或探勘 RFT 所需的
- 未在獎勵中明確指定和強制執行安全或政策限制

如果您可以指向「正確答案」，請使用 SFT。如果您需要新知識，請使用 RAG。只有在您擁有穩固的基準和強大、快速、hard-to-exploit 的獎勵函數之後，才使用 RFT。

監督式微調 (SFT)

受監督的微調會在您任務的人工標籤輸入輸出對資料集上訓練 LLM。您可以提供具有正確或所需回應的提示範例（問題、指示等），並繼續在這些範例上訓練模型。模型會調整其權重，將監督損失降至最低（通常在其預測和目標輸出字符之間跨熵）。這是大多數監督式機器學習任務中使用的相同訓練，適用於專門 LLM。

SFT 會變更行為，而不是知識。它不會教導模型在預先訓練中看不到的新事實或術語。它教導模型如何回答，而不是知道什麼。如果您需要新的網域知識（例如內部術語），請使用擷取擴增產生 (RAG) 在推論時間提供該內容。然後，SFT 會在頂端新增所需的指示遵循行為。

運作方式

SFT 透過將回應權杖的平均跨熵損失降至最低，將提示權杖視為內容，並遮罩它們免受損失來最佳化 LLM。模型會內化您的目標風格、結構和決策規則，學習為每個提示產生正確的完成。例如，若要將文件分類為自訂類別，您可以使用提示（文件文字）和標籤完成（類別標籤）微調模型。您可以針對這些配對進行訓練，直到模型以高機率為每個提示輸出正確的標籤。

您可以使用幾百個範例執行 SFT，並擴展到幾十萬個。SFT 範例必須高品質，並直接符合所需的模型行為。

何時使用 SFT

當您有明確定義且具有明確所需輸出的任務時，請使用 SFT。如果您可以明確陳述「給予 X 輸入，正確的輸出為 Y」並收集這類映射的範例，則監督式微調是不錯的選擇。SFT 在這些案例中表現優異：

- 結構化或複雜的分類任務 – 將內部文件或合約分類為許多自訂類別。使用 SFT，模型學習這些特定類別優於單獨提示。
- 具有已知答案的問題回答或轉換任務 – 微調模型以回答公司知識庫的問題，或在每個輸入都有正確回應的格式之間轉換資料。
- 格式化和樣式一致性 – 透過微調正確格式或色調的範例，訓練模型一律以特定格式或色調回應。例如，針對顯示特定品牌語音的提示-回應對進行訓練，會教導模型產生具有該樣式的輸出。指示遵循行為通常透過 SFT 最初就精心策劃的良好助理行為範例進行教學。

當您可以指定正確的行為時，SFT 是教導 LLM 新技能或行為的最直接方式。它使用模型現有的語言理解，並專注於您的任務。當您希望模型執行特定項目，且您有或可以建立範例資料集時，請使用 SFT。

當您可以組合高品質的提示和回應對，以密切反映您想要的行為時，請使用 SFT。它適合具有明確目標或決定性格式的任務，例如結構描述、函數或工具呼叫，以及結構式答案，其中模擬是適當的訓練訊號。目標是行為塑造：教導模型將提示視為任務、遵循指示、採用語氣和拒絕政策，以及產生一致的格式。規劃至少數百次示範，資料品質、一致性和重複資料刪除的重要性高於原始磁碟區。若要直接、符合成本效益的更新，請使用低範圍調整等參數效率方法來訓練小型轉接器，同時讓大多數骨幹保持不變。

何時不使用 SFT

當差距是知識而非行為時，請勿使用 SFT。它不會教導模型新的事實、術語或最近的事件。在這些情況下，請使用擷取擴增的產生，在推論時帶來外部知識。當您可以測量品質，但無法標記單一正確答案時，請避免 SFT。使用強化微調搭配可驗證的獎勵或 LLM-as-a-judge 來直接最佳化這些獎勵。如果您的需求或內容經常變更，請依賴擷取和工具使用，而不是重新訓練模型。

主題

- [Nova Forge SDK](#)
- [建立具有受限執行個體群組 \(RIG\) 的 SageMaker HyperPod EKS 叢集](#)

- [Amazon SageMaker HyperPod 基本命令指南](#)
- [的 Nova Forge 存取和設定](#)
- [Amazon Nova 模型的訓練](#)
- [評估您的訓練模型](#)
- [使用 MLflow 監控 HyperPod 任務](#)

Nova Forge SDK

Amazon Nova Forge SDK 是全方位的 Python SDK，可為完整的 Amazon Nova 模型自訂生命週期提供統一的程式設計界面。開發套件透過提供單一、一致的 API 來跨 Amazon SageMaker 和 Amazon Bedrock 平台進行訓練、評估、監控、部署和推論，簡化模型自訂。

如需詳細資訊，請參閱[Nova Forge SDK](#)。

建立具有受限執行個體群組 (RIG) 的 SageMaker HyperPod EKS 叢集

若要在 Hyperpod 上自訂模型，必須設定必要的基礎設施。如需使用限制執行個體群組 (RIG) 設定 SageMaker HyperPod EKS 叢集的詳細資訊，請造訪 [研討會](#)，其中提供設定程序的詳細演練。

Amazon SageMaker HyperPod 基本命令指南

Amazon SageMaker HyperPod 提供廣泛的命令列功能來管理訓練工作流程。本指南涵蓋從連線至叢集到監控任務進度等常見操作的基本命令。

先決條件

使用這些命令之前，請確定您已完成下列設定：

- 建立 RIG 的 SageMaker HyperPod 叢集（通常在 us-east-1 中）
- 為訓練成品建立的輸出 Amazon S3 儲存貯體
- 已設定適當許可的 IAM 角色
- 以正確的 JSONL 格式上傳的訓練資料
- FSx for Lustre 同步已完成（在第一個任務的叢集日誌中驗證）

主題

- [安裝配方 CLI](#)
- [連線至叢集](#)
- [啟動訓練任務](#)
- [檢查任務狀態](#)
- [監控任務日誌](#)
- [列出作用中任務](#)
- [取消任務](#)
- [執行評估任務](#)
- [常見問題](#)

安裝配方 CLI

在執行安裝命令之前，導覽至配方儲存庫的根目錄。

如果使用非 Forge 自訂技術，請使用 Hyperpodrecipes 儲存庫，對於 Forge 型自訂，請參閱 `forge` 特定配方儲存庫。

執行下列命令來安裝 SageMaker HyperPod CLI：

Note

確保您不在作用中的 `conda` / `anaconda` / `miniconda` 環境或其他虛擬環境中
如果是，請使用 結束環境：

- `conda deactivate` 適用於 `conda` / `anaconda` / `miniconda` 環境
- `deactivate` 適用於 Python 虛擬環境

如果您使用的是非 Forge 自訂技術，請下載 `sagemaker-hyperpod-recipes`，如下所示：

```
git clone -b release_v2 https://github.com/aws/sagemaker-hyperpod-cli.git
cd sagemaker-hyperpod-cli
pip install -e .
cd ..
root_dir=$(pwd)
```



```
export PYTHONPATH=${root_dir}/sagemaker-hyperpod-cli/src/hyperpod_cli/  
sagemaker_hyperpod_recipes/launcher/nemo/nemo_framework_launcher/launcher_scripts:  
$PYTHONPATH  
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-  
helm-3  
chmod 700 get_helm.sh  
./get_helm.sh  
rm -f ./get_helm.sh
```

如果您是 Forge 訂閱者，您應該使用下列程序下載配方。

```
mkdir NovaForgeHyperpodCLI  
cd NovaForgeHyperpodCLI  
aws s3 cp s3://nova-forge-c7363-206080352451-us-east-1/v1/ ./ --recursive  
pip install -e .  
  
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-  
helm-3  
chmod 700 get_helm.sh  
./get_helm.sh  
rm -f ./get_helm.sh
```

Tip

若要在執行之前使用[新的虛擬環境](#) `pip install -e .`，請執行：

- `python -m venv nova_forge`
- `source nova_forge/bin/activate`
- 您的命令列現在會在提示開頭顯示 (nova_forge)
- 這可確保使用 CLI 時沒有相互競爭的相依性

目的：為什麼要執行 `pip install -e .` ？

此命令會以可編輯模式安裝 SageMaker HyperPod CLI，可讓您使用更新的配方，而無需每次重新安裝。它還可讓您新增 CLI 可以自動取得的新配方。

連線至叢集

在執行任何任務之前，將 SageMaker HyperPod CLI 連接至您的叢集：


```
export AWS_REGION=us-east-1 && SageMaker HyperPod connect-cluster --cluster-name  
<your-cluster-name> --region us-east-1
```

Important

此命令會建立後續命令所需的內容檔案 (/tmp/hyperpod_context.json)。如果您看到找不到此檔案的錯誤，請重新執行 connect 命令。

專業秘訣：您可以透過將 --namespace kubeflow 引數新增至命令，進一步將叢集設定為一律使用 kubeflow 命名空間，如下所示：

```
export AWS_REGION=us-east-1 && \  
hyperpod connect-cluster \  
--cluster-name <your-cluster-name> \  
--region us-east-1 \  
--namespace kubeflow
```

這可讓您在與任務互動時，在每個命令 -n kubeflow 中新增。

啟動訓練任務

Note

如果執行 PPO/RFT 任務，請確保您將標籤選擇器設定新增至 `src/hyperpod_cli/sagemaker_hyperpod_recipes/recipes_collection/cluster/k8s.yaml` 以便將所有 Pod 排程在相同的節點上。

```
label_selector:  
  required:  
    sagemaker.amazonaws.com/instance-group-name:  
      - <rig_group>
```

使用具有選用參數覆寫的配方啟動訓練任務：

```
hyperpod start-job -n kubeflow \  
--recipe fine-tuning/nova/nova_1_0/nova_micro/SFT/nova_micro_1_0_p5_p4d_gpu_lora_sft \  
--override-parameters '{
```

```
"instance_type": "ml.p5.48xlarge",
  "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-latest"
}'
```

預期的輸出：

```
Final command: python3 <path_to_your_installation>/NovaForgeHyperpodCLI/src/
hyperpod_cli/sagemaker_hyperpod_recipes/main.py recipes=fine-tuning/nova/
nova_micro_p5_gpu_sft cluster_type=k8s cluster=k8s base_results_dir=/local/home/
<username>/results cluster.pullPolicy="IfNotPresent" cluster.restartPolicy="OnFailure"
cluster.namespace="kubeflow" container="708977205387.dkr.ecr.us-east-1.amazonaws.com/
nova-fine-tune-repo:HP-SFT-DATAMIX-latest"
```

```
Prepared output directory at /local/home/<username>/results/<job-name>/k8s_templates
Found credentials in shared credentials file: ~/.aws/credentials
Helm script created at /local/home/<username>/results/<job-name>/<job-name>_launch.sh
Running Helm script: /local/home/<username>/results/<job-name>/<job-name>_launch.sh
```

```
NAME: <job-name>
LAST DEPLOYED: Mon Sep 15 20:56:50 2025
NAMESPACE: kubeflow
STATUS: deployed
REVISION: 1
TEST SUITE: None
Launcher successfully generated: <path_to_your_installation>/NovaForgeHyperpodCLI/src/
hyperpod_cli/sagemaker_hyperpod_recipes/launcher/nova/k8s_templates/SFT

{
  "Console URL": "https://us-east-1.console.aws.amazon.com/sagemaker/home?region=us-
east-1#/cluster-management/<your-cluster-name>"
}
```

檢查任務狀態

使用 `kubectl` 監控執行中的任務：

```
kubectl get pods -o wide -w -n kubeflow | (head -n1 ; grep <your-job-name>)
```

了解 Pod 狀態

下表說明常見的 Pod 狀態：

狀態	Description
Pending	Pod 已接受但尚未排程到節點，或等待提取容器映像
Running	Pod 繫結至節點，其中至少有一個容器正在執行或啟動
Succeeded	所有容器都已成功完成，且不會重新啟動
Failed	終止的所有容器至少有一個結尾為失敗的容器
Unknown	無法判斷 Pod 狀態（通常是由於節點通訊問題）
CrashLoopBackOff	容器重複失敗；Kubernetes 從重新啟動嘗試中退避
ImagePullBackOff / ErrImagePull	無法從登錄檔提取容器映像
OOMKilled	容器因超過記憶體限制而終止
Completed	任務或 Pod 成功完成（批次任務完成）

Tip

使用 `-w` 旗標即時觀看 Pod 狀態更新。按 `Ctrl+C` 停止觀看。

監控任務日誌

您可以透過以下三種方式之一來檢視日誌：

使用 CloudWatch

您的日誌可在 AWS 您的帳戶中使用，其中包含 CloudWatch 下的 Hyperpodcluster。若要在瀏覽器中檢視它們，請導覽至您帳戶中的 CloudWatch 首頁，並搜尋您的叢集名稱。例如，如果您的叢集被呼叫 `my-hyperpod-rig`，則日誌群組會有字首：

- 日誌群組：`/aws/sagemaker/Clusters/my-hyperpod-rig/{UUID}`

- 進入日誌群組後，您可以使用節點執行個體 ID 來尋找您的特定日誌，例如 - hyperpod-i-00b3d8a1bf25714e4。
- i-00b3d8a1bf25714e4 這裡代表訓練任務執行所在的 Hyperpodfriendly 機器名稱。回顧我們在上一個命令 `kubectl get pods -o wide -w -n kubeflow | (head -n1 ; grep my-cpt-run)` 輸出中如何擷取名為 NODE 的資料欄。
- 在這種情況下，「主要」節點執行是在 Hyperpod-i-00b3d8a1bf25714e4 上執行，因此我們將使用該字串來選取要檢視的日誌群組。選取顯示 SagemakerHyperPodTrainingJob/rig-group/[NODE]

使用 CloudWatch Insights

如果您的任務名稱方便使用，但不希望完成上述所有步驟，您可以直接查詢下的所有日誌 `/aws/sagemaker/Clusters/my-hyperpod-rig/{UUID}`，以尋找個別日誌。

CPT :

```
fields @timestamp, @message, @logStream, @log
| filter @message like /(?!i)Starting CPT Job/
| sort @timestamp desc
| limit 100
```

若要完成任務，請將 取代 Starting CPT Job 為 CPT Job completed

然後，您可以按一下結果並挑選「Epoch 0」，因為那將是您的主節點。

使用 AWS CLI

您可以選擇使用 CLI AWS 結尾您的日誌。執行此操作之前，請使用 檢查您的 aws cli 版本 `aws --version`。也建議您使用此公用程式指令碼，以協助追蹤終端機中的即時日誌

適用於 V1 :

```
aws logs get-log-events \
--log-group-name /aws/sagemaker/YourLogGroupName \
--log-stream-name YourLogStream \
--start-from-head | jq -r '.events[].message'
```

適用於 V2 :

```
aws logs tail /aws/sagemaker/YourLogGroupName \
```

```
--log-stream-name YourLogStream \  
--since 10m \  
--follow
```

列出作用中任務

檢視叢集中執行的所有任務：

```
hyperpod list-jobs -n kubeflow
```

輸出範例：

```
{  
  "jobs": [  
    {  
      "Name": "test-run-nhgza",  
      "Namespace": "kubeflow",  
      "CreationTime": "2025-10-29T16:50:57Z",  
      "State": "Running"  
    }  
  ]  
}
```

取消任務

隨時停止執行中的任務：

```
hyperpod cancel-job --job-name <job-name> -n kubeflow
```

尋找您的任務名稱

選項 1：從您的配方

任務名稱是在配方的 run 區塊中指定：

```
run:  
  name: "my-test-run" # This is your job name  
  model_type: "amazon.nova-micro-v1:0:128k"  
  ...
```

選項 2：從 list-jobs 命令

使用 `hyperpod list-jobs -n kubeflow` 並從輸出複製 Name 欄位。

執行評估任務

使用評估配方評估訓練過的模型或基礎模型。

先決條件

在執行評估任務之前，請確定您有：

- 來自訓練任務 `manifest.json` 檔案的檢查點 Amazon S3 URI（適用於訓練模型）
- 以正確格式上傳至 Amazon S3 的評估資料集
- 用於評估結果的輸出 Amazon S3 路徑

命令

執行下列命令來啟動評估任務：

```
hyperpod start-job -n kubeflow \  
  --recipe evaluation/nova/nova_2_0/nova_lite/  
nova_lite_2_0_p5_48x1_gpu_bring_your_own_dataset_eval \  
  --override-parameters '{  
    "instance_type": "p5.48xlarge",  
    "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-  
HP-Eval-latest",  
    "recipes.run.name": "<your-eval-job-name>",  
    "recipes.run.model_name_or_path": "<checkpoint-s3-uri>",  
    "recipes.run.output_s3_path": "s3://<your-bucket>/eval-results/",  
    "recipes.run.data_s3_path": "s3://<your-bucket>/eval-data.jsonl"  
  }'
```

參數描述：

- `recipes.run.name`：評估任務的唯一名稱
- `recipes.run.model_name_or_path`：來自 `manifest.json` 或基本模型路徑的 Amazon S3 URI（例如 `nova-micro/prod`）
- `recipes.run.output_s3_path`：評估結果的 Amazon S3 位置
- `recipes.run.data_s3_path`：評估資料集的 Amazon S3 位置

提示：

- 特定模型配方：每個模型大小（微型、精簡型、專業）都有自己的評估配方
- 基礎模型評估：使用基礎模型路徑（例如 nova-micro/prod）而非檢查點 URIs 來評估基礎模型

評估資料格式

輸入格式 (JSONL)：

```
{
  "metadata": "{key:4, category:'apple'}",
  "system": "arithmetic-patterns, please answer the following with no other words: ",
  "query": "What is the next number in this series? 1, 2, 4, 8, 16, ?",
  "response": "32"
}
```

輸出格式：

```
{
  "prompt": "[{'role': 'system', 'content': 'arithmetic-patterns, please answer the following with no other words: '}, {'role': 'user', 'content': 'What is the next number in this series? 1, 2, 4, 8, 16, ?'}]",
  "inference": "['32']",
  "gold": "32",
  "metadata": "{key:4, category:'apple'}"
}
```

欄位描述：

- prompt：傳送至模型的格式化輸入
- inference：模型產生的回應
- gold：輸入資料集的預期正確答案
- metadata：從輸入傳遞的選用中繼資料

常見問題

- `ModuleNotFoundError: No module named 'nemo_launcher'`，您可能需要根據 `hyperpod_cli` 安裝的位置 `nemo_launcher`，將新增至您的 `python` 路徑。範例命令：

```
export PYTHONPATH=<path_to_hyperpod_cli>/sagemaker-hyperpod-cli/src/hyperpod_cli/  
sagemaker_hyperpod_recipes/launcher/nemo/nemo_framework_launcher/launcher_scripts:  
$PYTHONPATH
```

- `FileNotFoundError: [Errno 2] No such file or directory: '/tmp/hyperpod_current_context.json'` 表示您錯過執行 Hyperpod Connect 叢集命令。
- 如果您沒有看到任務排程，請仔細檢查 SageMaker HyperPod CLI 的輸出是否具有包含任務名稱和其他中繼資料的本節。如果沒有，請執行下列動作來重新安裝 helm Chart：

```
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/  
get-helm-3  
chmod 700 get_helm.sh  
./get_helm.sh  
rm -f ./get_helm.sh
```

的 Nova Forge 存取和設定

若要設定 Amazon Nova Forge 以與 任務搭配使用，您需要：

- 訂閱 Amazon Nova Forge
- 設定叢集

主題

- [訂閱 Amazon Nova Forge](#)
- [設定基礎設施](#)
- [負責任的 AI](#)

訂閱 Amazon Nova Forge

若要存取 Amazon Nova Forge 功能，請完成下列步驟：

1. 驗證管理員對 AWS 帳戶的存取權。
2. 導覽至 SageMaker AI 主控台並[請求存取 Amazon Nova Forge](#)。
3. 等待 Amazon Nova 團隊在訂閱請求核准後傳送電子郵件確認。

4. 使用標籤 `forge-subscription` 標記您的執行角色。存取 Amazon Nova Forge 功能和檢查點時需要此標籤。將下列標籤新增至您的執行角色：

- 索引鍵：`forge-subscription`
- 值：`true`

Note

標準 Amazon Nova 功能仍可在沒有 Forge 訂閱的情況下使用。Amazon Nova Forge 旨在所有模型訓練階段建立具有控制和彈性的自訂前沿模型。

設定基礎設施

一旦您的 Amazon Nova Forge 訂閱獲得核准，請設定必要的基礎設施以使用啟用 Forge 的功能。如需使用限制執行個體群組 (RIG) 建立 EKS 叢集的詳細說明，請遵循[研討會](#)說明。

負責任的 AI

內容管制設定：Amazon Nova Forge 客戶可存取 Amazon Nova Lite 1.0 和 Pro 1.0 模型的可自訂內容管制設定 (CCMS)。CCMS 可讓您調整內容管制控制，以符合您的特定業務需求，同時維護重要的負責任 AI 防護措施。若要判斷您的業務使用案例是否符合 CCMS 的資格，請聯絡您的 Amazon Web Services 客戶經理。

Amazon Nova Forge 提供負責任的 AI Toolkit，其中包含訓練資料、評估基準和執行期控制，協助您將模型與 Amazon Nova 負責任的 AI 準則保持一致。

訓練資料：資料混合中的「RAI」類別包含強調負責任 AI 原則、安全考量和負責任技術部署的案例和案例。使用這些項目，在持續的預先訓練期間以負責任的方式調整模型。

評估：基準任務可用於測試模型偵測和拒絕不適當、有害或不正確內容的能力。使用這些評估來衡量基本模型效能與自訂模型效能之間的差異。

Amazon Nova 模型的訓練

在 SageMaker HyperPod 上訓練 Amazon Nova 模型支援多種技術，包括持續訓練前 (CPT)、監督微調 (SFT) 和強化微調 (RFT)。每種技術都提供不同的自訂需求，並且可以套用至不同的 Amazon Nova 模型版本。

主題

- [持續預先訓練 \(CPT\)](#)
- [監督式微調 \(SFT\)](#)
- [SageMaker HyperPod 上的強化微調 \(RFT\)](#)

持續預先訓練 (CPT)

持續訓練前 (CPT) 是一種訓練技術，透過向特定網域或企業的其他未標記文字公開基礎模型，以延伸訓練前階段。與需要標記的輸入輸出對的監督式微調不同，CPT 會在原始文件上進行訓練，以協助模型更深入了解新網域、學習特定網域的術語和撰寫模式，以及適應特定內容類型或主題領域。

當您有大量（數十億個字符）的網域特定文字資料，例如法律文件、醫學文獻、技術文件或專屬商業內容，而且您希望模型在該網域中開發原生流暢性時，這種方法特別有用。一般而言，在 CPT 階段之後，模型需要經過額外的指令調校階段，讓模型能夠使用新取得的知識並完成有用的任務。

支援的模型

CPT 適用於下列 Amazon Nova 模型：

- Nova 1.0 (Micro、Lite、Pro)
- Nova 2.0（精簡型）

何時使用 Nova 1.0 與 Nova 2.0

Amazon Nova 系列模型提供多個價格效能操作點，可在準確度、速度和成本之間進行最佳化。

當您需要下列項目時，請選擇 Nova 2.0：

- 複雜分析任務的進階推理功能
- 編碼、數學和科學問題解決的卓越效能
- 更長的內容長度支援
- 更好的多語言效能

Note

較大的模型不一定會更好。在 Nova 1.0 和 Nova 2.0 模型之間進行選取時，請考慮成本效能權衡和您的特定業務需求。

Nova 2.0 上的 CPT

Amazon Nova Lite 2.0 是在比 Nova Lite 1.0 更大且更多樣化的資料集上訓練的推理模型。雖然是較大的模型，但 Nova Lite 2.0 提供比 Nova Lite 1.0 更快的推論，同時提供增強的推理功能、更長的內容長度和改善的多語言效能。

Nova 2.0 上的 CPT 可讓您使用網域特定資料擴充這些進階功能，讓模型能夠在專業領域開發深度專業知識，同時維持其卓越的推理和分析能力。

範例 CPT 配方

以下是 CPT 的範例配方。您可以在配方儲存庫中找到此[配方和其他項目](#)。

```
# Note:
# This recipe can run on p5.48xlarge
# Run config
run:
  name: "my-cpt-run"                # A descriptive name for your training
  job                                # job
  model_type: "amazon.nova-2-lite-v1:0:256k" # Model variant specification, do not
  change                             # change
  model_name_or_path: "nova-lite-2/prod"    # Base model path, do not change
  replicas: 8                             # Number of compute instances for
  training, allowed values are 4, 8, 16, 32
  data_s3_path: ""                       # Customer data paths
  validation_data_s3_path: ""            # Customer validation data paths
  output_s3_path: ""                     # Output artifact path, job-specific
  configuration - not compatible with standard SageMaker Training Jobs
  mlflow_tracking_uri: ""                 # Required for MLFlow
  mlflow_experiment_name: "my-cpt-experiment" # Optional for MLFlow. Note: leave this
  field non-empty
  mlflow_run_name: "my-cpt-run"           # Optional for MLFlow. Note: leave this
  field non-empty

## Training specific configs
training_config:
  task_type: cpt
  max_length: 8192                       # Maximum context window size (tokens)
  global_batch_size: 256                  # Global batch size, allowed values are
  32, 64, 128, 256.

  trainer:
    max_steps: 10                         # The number of training steps to run
  total
```

```

    val_check_interval: 10                # The number of steps between running
validation. Integer count or float percentage
    limit_val_batches: 2                  # Batches of the validation set to use
each trigger

model:
    hidden_dropout: 0.0                  # Dropout for hidden states, must be
between 0.0 and 1.0
    attention_dropout: 0.0              # Dropout for attention weights, must
be between 0.0 and 1.0

optim:
    optimizer: adam
    lr: 1e-5                             # Learning rate
    name: distributed_fused_adam         # Optimizer algorithm, do not change
    adam_w_mode: true                   # Enable AdamW mode
    eps: 1e-06                          # Epsilon for numerical stability
    weight_decay: 0.0                  # L2 regularization strength, must be
between 0.0 and 1.0
    adam_beta1: 0.9                    # Beta1 for Adam optimizer
    adam_beta2: 0.95                  # Beta2 for Adam optimizer
    sched:
        warmup_steps: 10               # Learning rate warmup steps
        constant_steps: 0              # Steps at constant learning rate
        min_lr: 1e-6                  # Minimum learning rate, must be lower
than lr

```

2.0 上 CPT 的資料準備

資料格式要求

訓練和驗證資料集必須是符合下列格式的 JSONL 檔案，其中每一行都包含代表與必要欄位和結構對話的 JSON 物件。請見此處範例：

```

{"text": "AWS stands for Amazon Web Services"}
{"text": "Amazon SageMaker is a fully managed machine learning service"}
{"text": "Amazon Bedrock is a fully managed service for foundation models"}

```

文字項目應包含代表目標網域的自然流動、高品質內容。

測試資料是否能夠轉換為 [Arrow 格式](#)。使用以下 python 指令碼來提供協助。確保至少使用 datasets==2.18.0 版本：

```
from datasets import load_dataset, load_from_disk
from pathlib import Path

input_path = Path("<Your jsonl file>")
output_path = Path("<Your output directory>")

dataset = load_dataset("json", data_files=str(input_path), split="train")
dataset.save_to_disk(str(output_path), max_shard_size="1GB")

try:
    test_dataset = datasets.load_from_disk(output_dir)
    print(f"Dataset loaded successfully #! Contains {len(test_dataset)} samples")
except Exception as e:
    print(e)
```

它應該列印與 JSONL 檔案中相同的行數。

使用 datamixing 時，請使用 執行第一個任務max_steps=2。這將有助於在叢集中建立資料存取的最佳化，並驗證所有資料混合是否可用。

如何準備 CPT 的資料

訓練資料是持續預先訓練成功最重要的決定因素。雖然 CPT 資料通常被描述為「未標記」，但事實更為細微。資料的結構、格式化和呈現方式會決定模型是否會獲得業務使用案例所需的知識和技能。

準備 CPT 的結構化業務資料集

這是公司和組織在領域中建立專業基礎模型的常見挑戰。大多數企業擁有結構化資料的豐富儲存庫：產品目錄、使用者設定檔、交易日誌、表單提交、API 呼叫和操作中繼資料。乍看之下，這看起來與標準預先訓練中常用的非結構化 Web 文字非常不同。

若要有效地從結構化商業資料中學習，請仔細考慮下游任務並設計資料呈現，以強制模型學習正確的預測關係。

若要釋放持續預先訓練的完整潛力，請考慮：

- 模型在推論時間應執行的任務
- 原始資料中存在哪些資訊
- 如何建構該資料，讓模型學習正確擷取和操作資訊

只要將結構化資料傾印到訓練中，就不會教導模型推理它。主動調整資料呈現的形狀，以引導模型學習的內容。

在下列各節中，有文獻回顧，展示了資料增強的重要性，並為結構化業務資料提供範例增強策略，以提供有關如何處理和組織 CPT 業務資料集的有用想法。

文獻中 CPT 的結構化資料

CPT 可以將網域事實封裝到模型中，但通常無法在輸入或任務轉移時擷取和操縱這些事實。受控實驗顯示，如果在預先訓練期間沒有多樣化的擴增，模型會以簡潔的方式記住事實，即使在稍後的指令調校後仍難以擷取，並且建議像訓練早期的訊號一樣注入指令。對於半結構化資料，隨機序列化和其他擴增可減少結構描述過度擬合，這就是為什麼 CPT 應與指令樣式任務交錯，而不是先執行和稍後執行 IFT。專注於財務的工作進一步發現，與循序配方相比，在批次時間聯合混合 CPT 和指令資料可改善一般化並減少忘記。Qwen 技術報告透過將高品質指令資料整合到預先訓練本身來收斂在相同的模式，這可提升內容學習並保留後續的指示，同時獲得新的領域知識。

半結構化體的資料擴增是關鍵控制桿。合成圖形感知 CPT 會將小型網域集擴展到實體連結體庫，以明確教導推論時具有擷取的關係和化合物。Joint CPT plus 指令混合在財務和平衡網域和一般資料中優於序列管道，可降低一般技能的降級。非常大規模的網域 CPT 也可以保留廣泛的能力，甚至允許透過模型合併進行權衡，但仍指向指令調校作為必要的後續步驟，加強在 CPT 期間引入指令訊號的值。

透過隨機化和隨機化來注入多樣性

有助於從結構化和半結構化資料集有效教導模型的一般策略是隨機隨機捨棄資料集中的欄位順序，甚至隨機捨棄一些索引鍵。

隨機顯示欄位會強制模型讀取每個值的意義，而不是顯示的位置，並了解所有欄位之間的關係。例如，如果視訊遊戲張貼在 amazon 商店，當 "Title"、"Platform"、"Price"、"Condition" 和 "Edition" 以不同的排列到達時，模型無法倚賴「第三個槽是平台」；它必須將標籤繫結至值，並了解屬性之間的雙邊關係：標題 $\rightleftharpoons$ 平台、平台 $\rightleftharpoons$ 價格、條件 $\rightleftharpoons$ 價格。例如，它可以從遊戲名稱和觀察價格推斷可能平台，或根據標題和平台估計合理的價格範圍。

在序列化期間隨機捨棄金鑰就像功能捨棄一樣：它可防止在任何一個欄位上進行共同調整，並強制模型從剩餘的證據中復原缺少的資訊。如果沒有「平台」，模型必須從標題字串或相容性文字中挑選；如果「價格」隱藏，則必須從平台、版本和條件中三角。這會建置對稱 ($A \rightarrow B$ 和 $B \rightarrow A$)、對雜亂真實世界清單的穩健性，以及欄位遺失、重新命名或重新排序時的結構描述變異。

購物風格範例讓它變得具體。以多種方式序列化相同的項目："Title : 'Elden Ring' | Platform : PlayStation 5 | Condition : Used—Like New | Price : \$34.99" 和排列，例如 "Price : \$34.99 | Title :

'Elden Ring' | Condition : Used—Like New | Platform : PlayStation 5"，並且在某些傳遞中捨棄 "Platform"，同時在描述中保留 "Compatible with PS5"。訓練互補目標，例如從 {title, price} 預測平台，以及從 {title, platform} 預測價格儲存貯體。由於索引鍵的順序和存在會有所不同，因此唯一穩定的策略是了解屬性之間的真實關係，而不是記住範本。

資料呈現的方式很重要

LLMs會從他們已看到的內容預測下一個字符來學習。因此，訓練期間顯示的欄位和事件順序會決定模型可以學習的內容。如果訓練格式符合實際任務，則損失會落在確切的決策權杖上。如果欄位在沒有結構的情況下拼湊在一起，模型會學習捷徑或記住熱門度，然後在要求選擇選項時失敗。

先顯示情況，再顯示選項，再顯示決策。如果模型也應該了解結果或說明，請在決策之後放置它們。

CPT 的封裝範例

什麼是封裝？

這只是表示使用多個完整範例填入訓練資料中的每個序列視窗，以便視窗以真實字符密集，而不是填補。

為什麼它很重要

在訓練期間會設定最大內容長度，例如 8,192 個字符。批次的形狀為【批次大小 × 內容長度】。如果訓練範例短於內容長度，則會填充剩餘的位置。即使已遮罩遺失，填補仍會透過注意力和 MLP 核心執行，因此會為沒有學習訊號的字符支付運算費用。

如何進行封裝？

若要封裝多個範例，請在之間使用 [DOC] 分隔符號串連多個訓練範例（請注意【DOC】前後的空間），讓範例的完整長度低於所需的內容長度。

封裝的文件範例如下所示：

```
{"text": "training sample 1 [DOC] training sample 2 [DOC] training sample 3"}
```

CPT 調校參數

可使用 CPT 進行微調的參數包括：

執行組態

- 名稱：訓練任務的描述性名稱。這有助於在 AWS 管理主控台中識別您的任務。
- `model_type`：要使用的 Amazon Nova 模型變體。可用的選項為 `amazon.nova-2-lite-v1:0:256k`。
- `model_name_or_path`：用於訓練的基本模型路徑。可用的選項為 `nova-lite-2/prod`，或訓練後檢查點的 S3 路徑 (`s3://customer-escrow-bucket-unique_id/training_run_name`)。
- 複本：用於分散式訓練的運算執行個體數量。可用的值會根據您選擇的模型而有所不同。Amazon Nova Lite 2.0 支援 4、8、16 或 32 個複本。
- `data_s3_path`：訓練資料集的 S3 位置，這是 JSONL 檔案。此檔案必須位於與叢集相同的 AWS 帳戶和區域。提供的所有 S3 位置都必須位於相同的帳戶和區域中。
- `validation_data_s3_path`：(選用) 驗證資料集的 S3 位置，這是 JSONL 檔案。此檔案必須位於與叢集相同的帳戶和區域中。提供的所有 S3 位置都必須位於相同的帳戶和區域中。
- `output_s3_path`：儲存資訊清單和 TensorBoard 日誌的 S3 位置。提供的所有 S3 位置都必須位於相同的 AWS 帳戶和 AWS 區域。
- `mlflow_tracking_uri`：用於 MLFlow 記錄的 MLFlow 應用程式的 ARN
- `mlflow_experiment_name`：MLFlow 實驗名稱
- `mlflow_run_name`：MLFlow 執行名稱

訓練組態

- `max_length`：字符中的序列長度上限。這會決定訓練的內容範圍大小。CPT 的最大支援值為 8192 個記號。

序列越長，越能提高訓練效率，但代價是需要增加記憶體。我們建議您將 `max_length` 參數與資料分佈配對。

- `global_batch_size`：跨所有裝置和工作者，一次向前或向後傳遞一起處理的訓練範例總數。

此值會乘以每個裝置的批次大小和裝置數目。它會影響訓練和輸送量的穩定性。我們建議您從適合您記憶體的批次大小開始，並從該處向上擴展。對於特定網域的資料，批次越大可能會使梯度過度平滑。

培訓人員設定

- `max_steps`：要執行的訓練步驟數目。每個步驟都會訓練具有元素 `global_batch_size` 數目的模型

模型設定

- `hidden_dropout`：捨棄隱藏狀態輸出的機率。將此值增加約 0.0-0.2 可減少對較小資料集的過度擬合。有效值介於 0-1 (含) 之間。
- `attention_dropout`：捨棄注意力權重的機率。此參數可協助進行一般化。有效值介於 0-1 (含) 之間。

最佳化工具組態

- `lr`：學習率，可在最佳化期間控制步驟大小。我們建議使用介於 $1e-6$ - $1e-4$ 之間的值，以獲得良好的效能。有效值介於 0-1 (含) 之間。
- `name`：最佳化工具演算法。目前僅支援 `distributed_fused_adam`。
- `weight_decay`：L2 正規化強度。較高的值 (介於 0.01-0.1 之間) 會增加正規化。
- `warmup_steps`：逐步提高學習率的步驟數。這可改善訓練穩定性。有效值介於 1-20 (含) 之間。
- `min_lr`：衰減結束時的最低學習率。有效值介於 0-1 (含) 之間，但必須小於學習率。

監督式微調 (SFT)

SFT 訓練程序包含兩個主要階段：

- 資料準備：遵循已建立的指導方針，將資料集建立、清除或重新格式化為所需的結構。確定輸入、輸出和輔助資訊（例如推理追蹤或中繼資料）已正確對齊並格式化。
- 訓練組態：定義模型的訓練方式。使用時，此組態會寫入 YAML 配方檔案中，其中包含：
 - 資料來源路徑（訓練和驗證資料集）
 - 金鑰超參數 (epoch、學習率、批次大小)
 - 選用元件（分散式訓練參數等）

Nova 模型比較和選擇

Amazon Nova 2.0 是在比 Amazon Nova 1.0 更大且更多樣化的資料集上訓練的模型。主要改進包括：

- 支援明確推理模式的增強推理功能
- 跨其他語言的更廣泛多語言效能
- 改善複雜任務的效能，包括編碼和工具使用
- 以更長的內容長度提供更高的準確性和穩定性的延伸內容處理

何時使用 Nova 1.0 與 Nova 2.0

在下列情況下選擇 Amazon Nova 2.0：

- 需要具有進階推理功能的卓越效能
- 需要多語言支援或複雜的任務處理
- 編碼、工具呼叫或分析任務需要更好的結果

新版本 2.0 上的 SFT

Amazon Nova Lite 2.0 為監督式微調提供了增強功能，包括進階推理模式、改善多模態理解和延伸內容處理。在 Nova 2.0 上的 SFT 可讓您將這些強大的功能適應您的特定使用案例，同時在複雜的任務上維持模型的卓越效能。

SFT on Nova 2.0 的主要功能包括：

- 推理模式支援：訓練模型在增強型分析功能的最終答案之前產生明確的推理追蹤。
- 進階多模式訓練：微調文件理解 (PDF)、影片理解和影像型任務，以提高準確性。
- 工具呼叫功能：訓練模型，以有效地使用外部工具和函數呼叫進行複雜的工作流程。
- 延伸內容支援：利用較長的內容時段，為文件密集型應用程式提供更好的穩定性和準確性。

Note

如需使用哪些容器映像或範例配方的詳細資訊，請前往 [Amazon Nova 配方](#)。

主題

- [選擇合理模式 \(僅限 Nova 2.0\)](#)
- [工具呼叫資料格式](#)
- [文件了解資料格式](#)
- [影片了解 SFT](#)
- [資料上傳指示](#)
- [建立微調任務](#)
- [SFT 調校參數](#)
- [超參數指引](#)

範例 SFT 配方

以下是 SFT 的範例配方。您可以在配方儲存庫中找到此[配方和其他項目](#)。

```
run:
  name: my-full-rank-sft-run
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: nova-lite-2/prod
  data_s3_path: s3://my-bucket-name/train.jsonl # only and not compatible with
SageMaker Training Jobs
  replicas: 4 # Number of compute instances for
training, allowed values are 4, 8, 16, 32
  output_s3_path: s3://my-bucket-name/outputs/ # Output artifact path (HyperPod job-
specific; not compatible with standard SageMaker Training Jobs)
  mlflow_tracking_uri: "" # Required for MLFlow
  mlflow_experiment_name: "my-full-rank-sft-experiment" # Optional for MLFlow. Note:
leave this field non-empty
  mlflow_run_name: "my-full-rank-sft-run" # Optional for MLFlow. Note: leave
this field non-empty

training_config:
  max_steps: 100 # Maximum training steps. Minimal is 4.
  save_steps: ${oc.select:training_config.max_steps} # How many training steps the
checkpoint will be saved
  save_top_k: 5 # Keep top K best checkpoints. Note supported only
for jobs. Minimal is 1.
  max_length: 32768 # Sequence length (options: 8192, 16384, 32768
[default], 65536)
  global_batch_size: 32 # Global batch size (options: 32, 64, 128)
  reasoning_enabled: true # If data has reasoningContent, set to true;
otherwise False

  lr_scheduler:
    warmup_steps: 15 # Learning rate warmup steps. Recommend 15% of
max_steps
    min_lr: 1e-6 # Minimum learning rate, must be between 0.0 and
1.0

  optim_config: # Optimizer settings
    lr: 1e-5 # Learning rate, must be between 0.0 and 1.0
    weight_decay: 0.0 # L2 regularization strength, must be between 0.0
and 1.0
    adam_beta1: 0.9 # Exponential decay rate for first-moment
estimates
```

```
adam_beta2: 0.95          # Exponential decay rate for second-moment
estimates

peft:                      # Parameter-efficient fine-tuning (LoRA)
  peft_scheme: "null"      # Disable LoRA for PEFT
```

選擇合理模式（僅限 Nova 2.0）

Amazon Nova 2.0 支援增強分析功能的推理模式：

- 原因模式（已啟用）：
 - 在訓練組態 `reasoning_enabled: true` 中設定
 - 模型會訓練 在最終答案之前產生推理追蹤
 - 改善複雜推理任務的效能
- 非原因模式（已停用）：
 - 設定 `reasoning_enabled: false` 或省略 參數（預設）
 - 沒有明確推理的標準 SFT
 - 適合無法受益於 step-by-step 推理的任務

Note

- 啟用推理時，會以高推理的努力運作。SFT 沒有低推理選項。
- SFT 不支援多模式推理內容。原因模式適用於純文字輸入。

將推理模式與非合理的資料集搭配使用

`reasoning_enabled: true` 允許使用 在非合理資料集上訓練 Amazon Nova。不過，這樣做可能會導致模型失去其推理功能，因為 Amazon Nova 主要學習在不套用推理的情況下產生資料中呈現的回應。

如果在非合理資料集上訓練 Amazon Nova，但仍希望在推論期間使用推理：

1. 在訓練期間停用推理 (`reasoning_enabled: false`)
2. 稍後在推論期間啟用推理

雖然此方法允許在推論時間進行推理，但與推論相比，它不保證改善效能，無需推理。

最佳實務：在使用推理資料集時啟用訓練推理和推論，並在使用非合理的資料集時停用兩者。

Note

如需使用哪些容器映像或範例配方的詳細資訊，請前往 [Amazon Nova 配方](#)。

工具呼叫資料格式

SFT 支援訓練模型以使用工具（函數呼叫）。以下是工具呼叫的範例輸入格式：

範例輸入：

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "system": [
    {
      "text": "You are an expert in composing function calls."
    }
  ],
  "toolConfig": {
    "tools": [
      {
        "toolSpec": {
          "name": "getItemCost",
          "description": "Retrieve the cost of an item from the catalog",
          "inputSchema": {
            "json": {
              "type": "object",
              "properties": {
                "item_name": {
                  "type": "string",
                  "description": "The name of the item to retrieve cost for"
                },
                "item_id": {
                  "type": "string",
                  "description": "The ASIN of item to retrieve cost for"
                }
              }
            },
            "required": [
              "item_id"
            ]
          }
        }
      }
    ]
  }
}
```

```

    }
  }
},
{
  "toolSpec": {
    "name": "getItemAvailability",
    "description": "Retrieve whether an item is available in a given location",
    "inputSchema": {
      "json": {
        "type": "object",
        "properties": {
          "zipcode": {
            "type": "string",
            "description": "The zipcode of the location to check in"
          },
          "quantity": {
            "type": "integer",
            "description": "The number of items to check availability for"
          },
          "item_id": {
            "type": "string",
            "description": "The ASIN of item to check availability for"
          }
        }
      },
      "required": [
        "item_id", "zipcode"
      ]
    }
  }
}
],
"messages": [
  {
    "role": "user",
    "content": [
      {
        "text": "I need to check whether there are twenty pieces of the following item available. Here is the item ASIN on Amazon: id-123. Please check for the zipcode 94086"
      }
    ]
  }
],
},

```

```

{
  "role": "assistant",
  "content": [
    {
      "reasoningContent": {
        "reasoningText": {
          "text": "The user wants to check how many pieces of the item with ASIN
id-123 are available in the zipcode 94086"
        }
      }
    },
    {
      "toolUse": {
        "toolUseId": "getItemAvailability_0",
        "name": "getItemAvailability",
        "input": {
          "zipcode": "94086",
          "quantity": 20,
          "item_id": "id-123"
        }
      }
    }
  ],
  {
    "role": "user",
    "content": [
      {
        "toolResult": {
          "toolUseId": "getItemAvailability_0",
          "content": [
            {
              "text": "[{\"name\": \"getItemAvailability\", \"results\":
{\"availability\": true}]]"
            }
          ]
        }
      }
    ]
  },
  {
    "role": "assistant",
    "content": [
      {

```

```
        "text": "Yes, there are twenty pieces of item id-123 available at 94086.  
Would you like to place an order or know the total cost?"  
    }  
  ]  
}  
]  
}
```

工具呼叫資料的重要考量事項：

- ToolUse 必須僅以助理輪換顯示
- ToolResult 必須僅顯示在使用者轉彎中
- ToolResult 只能是文字或 JSON；Amazon Nova 模型目前不支援其他模態
- toolSpec 中的 inputSchema 必須是有效的 JSON 結構描述物件
- 每個 ToolResult 必須參考先前助理 ToolUse 中的有效 ToolUse toolUseId，每個 toolUseId 在每次對話中只使用一次

Note

如需使用哪些容器映像或範例配方的詳細資訊，請前往 [Amazon Nova 配方](#)。

文件了解資料格式

SFT 支援文件理解任務的訓練模型。以下是範例輸入格式：

範例輸入

```
{  
  "schemaVersion": "bedrock-conversation-2024",  
  "messages": [  
    {  
      "role": "user",  
      "content": [  
        {  
          "text": "What are the ways in which a customer can experience issues during  
checkout on Amazon?"  
        },  
        {  
          "document": {
```



```
        "format": "pdf",
        "source": {
            "s3Location": {
                "uri": "s3://my-bucket-name/path/to/documents/
customer_service_debugging.pdf",
                "bucketOwner": "123456789012"
            }
        }
    },
    {
        "role": "assistant",
        "content": [
            {
                "text": "Customers can experience issues with 1. Data entry, 2. Payment
methods, 3. Connectivity while placing the order. Which one would you like to dive
into?"
            }
        ],
        "reasoning_content": [
            {
                "text": "I need to find the relevant section in the document to answer the
question.",
                "type": "text"
            }
        ]
    }
]
```

文件理解的重要考量：

- 僅支援 PDF 檔案
- 文件大小上限為 10 MB
- 範例可以包含文件和文字，但無法將文件與其他形式（例如影像或影片）混合

Note

如需使用哪些容器映像或範例配方的詳細資訊，請前往 [Amazon Nova 配方](#)。

影片了解 SFT

SFT 支援微調模型以進行影片理解任務。以下是範例輸入格式：

範例輸入

```
{
  "schemaVersion": "bedrock-conversation-2024",
  "messages": [
    {
      "role": "user",
      "content": [
        {
          "text": "What are the ways in which a customer can experience issues during checkout on Amazon?"
        },
        {
          "video": {
            "format": "mp4",
            "source": {
              "s3Location": {
                "uri": "s3://my-bucket-name/path/to/videos/customer_service_debugging.mp4",
                "bucketOwner": "123456789012"
              }
            }
          }
        }
      ]
    },
    {
      "role": "assistant",
      "content": [
        {
          "text": "Customers can experience issues with 1. Data entry, 2. Payment methods, 3. Connectivity while placing the order. Which one would you like to dive into?"
        }
      ],
      "reasoning_content": [
        {
          "text": "I need to find the relevant section in the video to answer the question.",
          "type": "text"
        }
      ]
    }
  ]
}
```

```
}  
  ]  
}  
]  
}
```

了解影片的重要考量：

- 影片上限為 50 MB
- 影片最長可達 15 分鐘
- 每個範例只允許一個影片；不支援相同範例中的多個影片
- 範例可以包含影片和文字，但無法將影片與其他形式（例如影像或文件）混合

Note

如需使用哪些容器映像或範例配方的詳細資訊，請前往 [Amazon Nova 配方](#)。

資料上傳指示

將訓練和驗證資料集上傳至 S3 儲存貯體。在配方的 run 區塊中指定這些位置：

```
## Run config  
run:  
  ...  
  data_s3_path: "s3://<bucket-name>/<training-directory>/<training-file>.jsonl"
```

注意：將 <bucket-name>、<training-directory>、<training-file>、<validation-directory>和 取代<validation-file>為實際的 S3 路徑。

注意：使用 Amazon Nova 2.0 的 SFT 目前不支援驗證資料集。如果提供驗證資料集，則會予以忽略。

建立微調任務

使用 run 區塊中的 model_type和 model_name_or_path 欄位來定義基本模型：

```
## Run config  
run:
```

```
...
model_type: amazon.nova-2-lite-v1:0:256k
model_name_or_path: nova-lite-2/prod
...
```

SFT 調校參數

可使用 SFT 進行調校的參數包括：

執行組態

- 名稱：訓練任務的描述性名稱。這有助於在 AWS 管理主控台中識別您的任務。
- model_type：要使用的 Amazon Nova 模型變體。可用的選項為 amazon.nova-2-lite-v1:0:256k。
- model_name_or_path：用於訓練的基本模型路徑。可用的選項為 nova-lite-2/prod，或訓練後檢查點的 S3 路徑 (s3://customer-escrow-bucket-unique_id/training_run_name)。
- 複本：用於分散式訓練的運算執行個體數量。可用的值會根據您選擇的模型而有所不同。Amazon Nova Lite 2.0 支援 4、8、16 或 32 個複本。
- data_s3_path：訓練資料集的 S3 位置，這是 JSONL 檔案。此檔案必須位於與叢集相同的 AWS 帳戶和區域。提供的所有 S3 位置都必須位於相同的帳戶和區域中。
- validation_data_s3_path：(選用) 驗證資料集的 S3 位置，這是 JSONL 檔案。此檔案必須位於與叢集相同的帳戶和區域中。提供的所有 S3 位置都必須位於相同的帳戶和區域中。
- output_s3_path：儲存資訊清單和 TensorBoard 日誌的 S3 位置。提供的所有 S3 位置都必須位於相同的 AWS 帳戶和 AWS 區域。
- mlflow_tracking_uri：用於 MLFlow 記錄的 MLFlow 應用程式的 ARN。
- mlflow_experiment_name：MLFlow 實驗名稱。
- mlflow_run_name：MLFlow 執行名稱。

訓練組態

- max_steps：要執行的訓練步驟數目。每個步驟都會使用元素 global_batch_size 數目來訓練模型。
- save_steps：在訓練期間儲存模型檢查點的頻率 (步驟中)。
- save_top_k：根據驗證指標要保留的最佳檢查點數目上限。

- `max_length`：字符的序列長度上限。這會決定訓練的內容範圍大小。SFT 的最大支援值為 32768 個字符。

序列越長，越能提高訓練效率，但代價是需要增加記憶體。我們建議您將 `max_length` 參數與資料分佈配對。

- `global_batch_size`：跨所有裝置和工作者，一次向前或向後傳遞一起處理的訓練範例總數。

此值會乘以每個裝置的批次大小和裝置數目。它會影響訓練和輸送量的穩定性。我們建議您從適合您記憶體的批次大小開始，並從該處向上擴展。對於特定網域的資料，批次越大可能會使梯度過度平滑。

- `reasoning_enabled`：在訓練期間啟用推理功能的布林值旗標。

學習率排程器

- `warmup_steps`：逐步提高學習率的步驟數目。這可改善訓練穩定性。
- `min_lr`：衰減結束時的最低學習率。有效值介於 0-1 (含) 之間，但必須小於學習率。

最佳化工具組態

- `lr`：學習率，可在最佳化期間控制步進大小。我們建議使用介於 $1e-6$ - $1e-4$ 之間的值，以獲得良好的效能。有效值介於 0-1 (含) 之間。
- `weight_decay`：L2 正規化強度。較高的值 (介於 0.01-0.1 之間) 會增加正規化。
- `adam_beta1`：Adam 最佳化工具中第一時間預估的指數衰減率。預設為 0.9。
- `adam_beta2`：Adam 最佳化工具中第二個時刻預估的指數衰減率。預設為 0.95。

PEFT 組態

- `peft_scheme`：要使用的參數有效微調方案。選項 `'null'` 適用於全等級微調或 `lora` 適用於以 LoRA 為基礎的微調。

LoRA 調校 (當 `peft_scheme` 為 `'lora'` 時)

- `alpha`：LoRA 擴展參數。控制低階調整的大小。典型值的範圍是 8 到 128。

- `lora_plus_lr_ratio` : LoRA+ 最佳化的學習率比率。此乘數會特別針對 LoRA 參數調整學習率。

超參數指引

根據訓練方法，使用以下建議的超參數：

完整排名訓練

- Epochs : 1
- 學習率 (lr) : 1e-5
- 最低學習率 (min_lr) : 1e-6

LoRA (低排名調整)

- Epochs : 2
- 學習率 (lr) : 5e-5
- 最低學習率 (min_lr) : 1e-6

注意：根據資料集大小和驗證效能調整這些值。監控訓練指標以防止過度擬合。

SageMaker HyperPod 上的強化微調 (RFT)

強化微調 (RFT) 是一種機器學習技術，可透過意見回饋訊號改善模型效能，這些訊號是可測量的分數或獎勵，指出回應品質，而不是直接監督並準確回答。與從輸入輸出對中學習的傳統監督式微調不同，RFT 使用獎勵函數來評估模型回應，並反覆最佳化模型以最大化這些獎勵。

此方法對於定義確切正確輸出具有挑戰性的任務特別有效，但您可以可靠地測量回應品質。RFT 可讓模型透過試驗和意見回饋來學習複雜的行為和偏好，非常適合需要細微決策、創意問題解決或遵守程式設計評估的特定品質標準的應用程式。

何時使用 RFT

當您可以定義明確、可衡量的成功條件，但難以為訓練提供確切正確的輸出時，請使用 RFT。它非常適合品質為主觀或多面向的任務，例如創意撰寫、程式碼最佳化或複雜推理，其中有多個有效解決方案，但有些解決方案明顯優於其他解決方案。

當您有以下項目時，RFT 最有效：

- 可靠的獎勵函數，可透過程式設計方式評估模型輸出
- 需要使模型行為符合特定偏好設定或限制條件
- 傳統監督式微調因收集高品質標籤範例昂貴或不切實際而縮短的情況

對於需要反覆改進、個人化或遵守可編碼為獎勵訊號的複雜業務規則的應用程式，請考慮 RFT。

RFT 最適合什麼

RFT 在可以客觀測量輸出品質，但難以預先定義最佳回應的網域中表現卓越：

- 數學問題解決：具有多個解決方案路徑的可驗證正確性
- 程式碼產生和最佳化：可測試的執行結果和效能指標
- 科學推理任務：邏輯一致性和事實準確性
- 結構化資料分析：以程式設計方式驗證的輸出
- 多步驟推理：需要step-by-step邏輯進展的任務
- 工具用量和 API 呼叫：執行結果可衡量的成功
- 複雜工作流程：遵守特定限制條件和業務規則

當您需要平衡多個競爭目標，例如準確性、效率和風格時，RFT 的效果非常好。

何時使用推理模式進行 RFT 訓練

Amazon Nova 2.0 支援 RFT 訓練期間的推理模式。可使用下列模式：

- 無：無推理（省略 reasoning_effort 欄位）
- 低：最低推理開銷
- 高：推理功能上限（指定 reasoning_effort 時預設為預設值）

Note

RFT 沒有媒體選項。如果組態中沒有 reasoning_effort 欄位，則會停用推理。

針對下列項目使用高推理：

- 複雜的分析任務
- 數學問題解決
- 多步驟邏輯扣除
- step-by-step 思考可增加價值的任務

對下列項目不使用（省略 reasoning_effort）或低推理：

- 簡單事實查詢
- 直接分類
- 速度和成本最佳化
- 直接的問答

Important

較高的推理模式會增加訓練時間和成本、推論延遲和成本，但也會提高複雜推理任務的模型功能。

支援的模型

RFT on SageMaker HyperPod 支援 Amazon Nova Lite 2.0 (amazon.nova-2-lite-v1 : 0 : 256k)。

主要步驟

RFT 程序包含四個關鍵階段：

- 實作評估者：建立獎勵函數，根據您的品質條件以程式設計方式對模型回應進行評分。
- 上傳提示：準備和上傳指定對話格式的訓練資料，以及評估的參考資料。
- 啟動任務：使用您設定的參數啟動強化微調程序。
- 監控：透過指標儀表板追蹤訓練進度，以確保模型有效地學習。

每個步驟都以上一個步驟為基礎，評估者做為透過提供一致的意見回饋訊號來引導整個訓練程序的基礎。

主題

- [Nova 2.0 上的 RFT](#)

Nova 2.0 上的 RFT

RFT 訓練資料遵循 OpenAI 對話格式。每個訓練範例都是 JSON 物件，其中包含訊息、參考答案和選用的工具定義。本節提供在 Nova 2.0 上準備 RFT 有效訓練資料的指引。

主題

- [資料格式和結構](#)
- [欄位描述](#)
- [超參數指引](#)
- [其他屬性](#)
- [資料集大小建議](#)
- [有效訓練資料的特性](#)
- [監控 RFT 訓練](#)

資料格式和結構

每個訓練範例都是包含下列項目的 JSON 物件：

- 訊息：一系列使用系統、使用者和選擇性助理角色的對話式轉場
- reference_answer：獎勵計算的預期輸出或評估條件
- 工具（選用）：模型可用的函數定義陣列
- id（選用）：追蹤和刪除重複資料的唯一識別符

每個範例應該位於 JSONL 檔案中的單一行上，每行一個 JSON 物件。

範例 1：化學測試問題

下列範例顯示包含 Ground Truth 值的參考答案的化學問題：

```
{
  "id": "chem-001",
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful chemistry assistant"
```

```

    },
    {
      "role": "user",
      "content": "Predict hydrogen bond donors and acceptors for this SMILES:
CCN(CC)CCC(=O)c1sc(N)nc1C"
    }
  ],
  "reference_answer": {
    "donor_bond_counts": 2,
    "acceptor_bond_counts": 4,
    "explanation": "Calculated using Lipinski's rule of five: N-H groups (2 donors), N
and 0 atoms with lone pairs (4 acceptors)"
  }
}

```

Note

reference_answer 包含使用網域特定規則計算的 Ground Truth 值。您的獎勵函數會將模型的預測值與這些參考值進行比較，以計算獎勵分數。

範例 2：數學問題

下列範例顯示解決方案步驟的數學問題：

```

{
  "id": "math-001",
  "messages": [
    {
      "role": "system",
      "content": "You are a math tutor"
    },
    {
      "role": "user",
      "content": "Solve:  $2x + 5 = 13$ "
    }
  ],
  "reference_answer": {
    "solution": " $x = 4$ ",
    "steps": [" $2x = 13 - 5$ ", " $2x = 8$ ", " $x = 4$ "]
  }
}

```

範例 3：工具用量

下列範例顯示具有預期行為的工具用量：

```
{
  "id": "tool-001",
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful game master assistant"
    },
    {
      "role": "user",
      "content": "Generate a strength stat for a warrior character. Apply a +2 racial bonus modifier."
    }
  ],
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "StatRollAPI",
        "description": "Generates character stats by rolling 4d6, dropping the lowest die result, and applying a modifier.",
        "parameters": {
          "type": "object",
          "properties": {
            "modifier": {
              "description": "An integer representing the modifier to apply to the total of the stat roll.",
              "type": "integer"
            }
          }
        },
        "required": ["modifier"]
      }
    }
  ],
  "reference_answer": {
    "tool_called": "StatRollAPI",
    "tool_parameters": {
      "modifier": 2
    },
  },
}
```

```
    "expected_behavior": "Call StatRollAPI with modifier=2 and return the calculated
    stat value"
  }
}
```

欄位描述

欄位	Description	其他備註	必要
id	此 RFT 範例的唯一識別符	字串（例如 "sample-001"）。適用於追蹤和重複資料刪除。	否
messages	定義提示和內容的聊天訊息排序清單	物件的陣列。模型會依序查看它們。通常以系統訊息開頭，然後是使用者。	是
messages 【】 .role	誰在訊息中說話	常見值："syst em"、"user"（有時在其他內容中為 "assistant"）	否
messages 【】 .content	訊息的文字內容	純字串。對於系統，它是指示，對於使用者，它是任務或輸入。	否
工具	在此範例中，模型可用的工具規格	陣列。每個項目都會定義工具的界面和中繼資料。類型可能包括「函數」或「內部」。	否
reference_answer	此範例的預期模型輸出	字串或物件，視任務而定。用作評估或訓練的目標。	否

 Note

任何其他自訂欄位（例如 `task_id`、`Cucky_level`、`context_data`）都不會經過驗證，並將做為中繼資料傳遞給您的獎勵函數。

超參數指引

根據您的訓練方法，使用以下建議的超參數：

一般：

- Epochs：1
- 學習率 (lr)：1e-7
- 世代數：8
- 最大新權杖數：8192
- 批次大小：256

LoRA（低排名調整）：

- LoRA 排名：32

 Note

根據您的資料集大小和驗證效能調整這些值。監控訓練指標以防止過度擬合。

其他屬性

「`additionalProperties`」：true 設定可讓您包含核心結構描述需求以外的自訂欄位，讓您靈活地新增獎勵函數進行適當評估所需的任何資料。

常見的其他欄位

您可以包含下列類型的其他欄位：

中繼資料：

- `task_id`：追蹤的唯一識別符
- `difficulty_level`：問題複雜性指標
- 網域：主旨區域或類別
- `expected_reasoning_steps`：解決方案中的步驟數

評估條件：

- `evaluation_criteria`：特定分級盧布
- `custom_scoring_weights`：不同層面的相對重要性
- `context_data`：問題的背景資訊
- `external_references`：相關文件或資源的連結

具有其他屬性的範例

下列範例包含自訂中繼資料欄位：

```
{
  "id": "algebra_001",
  "messages": [
    {
      "role": "system",
      "content": "You are a math tutor"
    },
    {
      "role": "user",
      "content": "Solve:  $2x + 5 = 13$ "
    }
  ],
  "reference_answer": {
    "solution": " $x = 4$ ",
    "steps": [" $2x = 13 - 5$ ", " $2x = 8$ ", " $x = 4$ "]
  },
  "task_id": "algebra_001",
  "difficulty_level": "easy",
  "domain": "algebra",
  "expected_reasoning_steps": 3
}
```

資料集大小建議

起點

從下列最低資料集大小開始：

- 最少 100 個訓練範例
- 最少 100 個評估範例

優先考慮高品質的輸入資料和可靠的獎勵函數，該函數會在模型回應時一致地執行。

評估優先方法

在投資大規模 RFT 訓練之前，請評估模型的基準效能：

- 高效能（超過 95% 的獎勵）：RFT 可能不必要，您的模型已表現良好
- 效能非常差 (0% 獎勵)：先切換到 SFT 以建立基本功能
- 中等效能：RFT 可能是適當的

此評估優先方法可確保獎勵函數沒有錯誤，並判斷 RFT 是否適合您的使用案例。從小型開始，您可以熟悉 RFT 工作流程、及早識別和修正問題、在擴展之前驗證您的方法，以及測試獎勵函數可靠性。驗證後，您可以擴展到更大的資料集，以進一步改善效能。

有效訓練資料的特性

清晰度和一致性

良好的 RFT 範例需要清晰、不明確的輸入資料，以便在不同的模型輸出間實現準確的獎勵計算。避免資料中的雜訊，包括：

- 不一致的格式
- 矛盾的標籤或指示
- 模稜兩可的提示
- 衝突的參考答案

任何模稜兩可的情況都會誤導訓練程序，導致模型學習非預期的行為。

多樣性

您的資料集應擷取生產使用案例的完整多樣性，以確保強大的實際效能。包括：

- 各種問題類型和困難程度
- 不同的輸入格式和邊緣案例
- 來自所有預期案例的代表性範例

這種多樣性有助於防止過度擬合，並確保模型正常處理不熟悉的輸入。

獎勵函數考量事項

設計您的獎勵函數以進行高效訓練：

- 在幾秒鐘內執行（非分鐘）
- 使用 Lambda 有效平行化
- 傳回一致、可靠的分數
- 正常處理不同類型的模型輸出

快速、可擴展的獎勵函數可大規模進行快速迭代和經濟實惠的實驗。

監控 RFT 訓練

在訓練期間監控關鍵指標，以確保有效學習並及早識別潛在問題。

主題

- [要追蹤的關鍵指標](#)
- [RFT 之後的評估](#)
- [使用微調的模型](#)
- [限制和最佳實務](#)
- [疑難排解](#)

要追蹤的關鍵指標

在訓練期間使用 MIFlow 監控下列指標：

獎勵指標：

- 平均獎勵分數：模型回應的整體品質（應隨時間增加）
- 獎勵分配：獲得高、中和低獎勵的回應百分比
- 訓練與驗證獎勵：比較以偵測過度擬合

訓練指標：

- 政策更新：成功權重更新的數量
- 推展完成率：成功評估的範例百分比

關注模式：

- 獎勵穩定（表示學習不佳）
- 驗證獎勵在訓練獎勵增加時下降（過度擬合）
- 獎勵差異會隨著時間大幅增加（不穩定）
- 獎勵函數錯誤的高百分比（實作問題）

停止訓練的時機：

- 實現目標效能指標
- 獎勵穩定且不再改善
- 驗證效能降低（偵測到過度擬合）
- 達到訓練預算上限

RFT 之後的評估

訓練完成後，請評估微調後的模型，以評估效能改善：

- 執行 RFT 評估任務：使用 RFT 訓練中的檢查點做為模型
- 與基準比較：在相同的測試集中評估基礎模型和微調模型
- 分析指標：檢閱任務特定的指標（準確性、獎勵分數等）
- 執行定性審查：手動檢查範例輸出的品質

如需詳細評估程序，請參閱評估一節。

使用微調的模型

存取檢查點：

訓練完成後，找到您的檢查點：

1. 在 S3 output_path 中導覽至您的
2. 下載並擷取 output.tar.gz
3. 開啟 manifest.json
4. 複製 checkpoint_s3_bucket 值

部署以進行推論：

使用檢查點 S3 路徑進行推論或進一步訓練：

```
run:
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-
  identifier>/<job-name>"
```

如需部署和推論說明，請參閱推論一節。

限制和最佳實務

目前的限制：

Beta 限制：

- 需要為 RFT 建立新的 RIG 群組。此限制將由 GA 解決。
- 執行個體類型需求：僅支援 P5 執行個體（最低 8x P5.48xlarge）。即將推出：支援較小的執行個體類型（ETA：2025 年 1 月中）。

功能限制：

- 15 分鐘 Lambda 逾時：獎勵函數必須在 15 分鐘內完成
- 僅限單轉：不支援多轉對話
- 驗證資料集：訓練期間不支援。使用個別的評估任務來評估訓練進度。

訓練考量事項：

- 低獎勵案例：當少於 5% 的範例獲得正面獎勵時可能會遇到困難 - 首先考慮 SFT
- 資料需求：需要足夠的多樣性才能有效地學習
- 運算成本：比監督式微調更昂貴

Nova Forge 會移除其中一些限制：

- 支援多轉對話
- 允許超過 15 分鐘逾時的獎勵函數
- 提供進階演算法和調校選項
- 專為複雜的企業使用案例而設計，專門調校以建置前沿模型

最佳實務：

從小開始並擴展：

- 從最少的資料集 (100-200 個範例) 和很少的訓練 epoch 開始
- 在向上擴展之前驗證您的方法
- 根據結果逐漸增加資料集大小和訓練步驟

先使用 SFT 的基準：

- 如果獎勵分數持續較低（例如，一律為 0），請在 RFT 之前執行 SFT
- RFT 需要合理的基準效能才能有效改善

設計有效率的獎勵函數：

- 以秒為單位執行，而非以分鐘為單位
- 將外部 API 呼叫降至最低
- 使用有效率的演算法和資料結構
- 實作適當的錯誤處理
- 訓練前徹底測試
- 利用 Lambda 的平行擴展功能

主動監控訓練：

- 追蹤一段時間內的平均獎勵分數
- 觀看跨範例的獎勵分佈
- 比較訓練與驗證獎勵
- 尋找相關的模式（平穩、過度擬合、不穩定）

根據結果反覆運算：

- 如果反覆運算後獎勵未改善，請調整獎勵函數設計
- 增加資料集多樣性，以提供更清晰的學習訊號
- 如果獎勵保持接近零，請考慮切換至 SFT
- 使用不同的超參數進行實驗（學習率、批次大小）

最佳化資料品質：

- 確保多樣化的代表性範例
- 包含邊緣案例和困難的範例
- 驗證獎勵函數是否正確為所有範例類型評分
- 移除或修正混淆獎勵函數的範例

疑難排解

獎勵函數錯誤：

症狀：訓練期間獎勵函數呼叫的高錯誤率

問題	徵狀	Resolution
Lambda 逾時	15 分鐘後頻繁逾時	最佳化函數效能；考慮使用 Nova Forge 進行複雜的評估
並行不足	Lambda 限流錯誤	增加 <code>lambda_concurrency_limit</code> 或請求增加配額
無效的傳回格式	訓練失敗，格式錯誤	確認傳回結構符合所需的界面格式

問題	徵狀	Resolution
未處理的例外狀況	間歇性錯誤	新增全面的錯誤處理和記錄
外部 API 失敗	不一致的評分	實作重試邏輯和備用策略

訓練效能不佳：

症狀：獎勵未改善或在低值時保持穩定

解決方法：

- 驗證獎勵函數正確性：使用已知良好/不良的範例進行測試
- 檢查基準效能：評估基礎模型；如果準確度接近零，請先執行 SFT
- 增加資料多樣性：新增涵蓋不同案例的更多不同範例
- 調整超參數：嘗試不同的學習率或批次大小
- 檢閱獎勵訊號品質：確保獎勵區分好回應和壞回應

過度擬合：

症狀：訓練獎勵增加，而驗證獎勵減少

解決方法：

- 減少訓練步驟：提早停止訓練
- 增加資料集大小：新增更多訓練範例
- 新增正規化：調整 `weight_decay` 或 `entropy_coeff`
- 增加資料多樣性：確保訓練集代表完整分佈

評估您的訓練模型

評估配方是一種 YAML 組態檔案，可定義 Amazon Nova 模型評估任務的執行方式。使用此配方，您可以根據常見的基準或您自己的自訂資料集，來評估基礎或訓練模型的效能。指標可以存放在 Amazon S3 或 TensorBoard 中。評估會提供量化指標，協助您評估各種任務的模型效能，以判斷是否需要進一步自訂。

模型評估是一種離線程序，在程序中會根據具有預先定義的回答測試模型。系統不會即時或根據即時使用者互動來評估它們。對於即時評估，將模型部署到 Amazon Bedrock 之後，您可以透過呼叫 Amazon Bedrock 執行時期 API 來評估模型。

主題

- [可用的基準任務](#)
- [了解配方參數](#)
- [評估配方範例](#)
- [啟動評估任務](#)
- [存取和分析評估結果](#)
- [RFT 評估](#)

可用的基準任務

提供範例程式碼套件，示範如何使用 Amazon Nova 的 SageMaker AI 模型評估功能來計算基準指標。若要存取程式碼套件，請參閱 [sample-Nova-lighteval-custom-task](#)。

以下是支援的可用產業標準基準清單。您可以在 eval_task 參數中指定下列基準。

Benchmark	模態	Description	指標	策略	子任務可用
mmlu	文字	多任務語言理解 – 測試 57 個主題的知識。	正確性	zs_cot	是
mmlu_pro	文字	MMLU – 專業子集 – 專注於專業領域，例如法律、醫學、會計和工程。	正確性	zs_cot	否
bbh	文字	進階推理任務 – 一系列挑戰性問題，可測試高階認知	正確性	zs_cot	是

Benchmark	模態	Description	指標	策略	子任務可用
		和問題解決技能。			
gpqa	文字	一般物理問題回答 – 評估對物理概念的理解及解決相關問題的能力。	正確性	zs_cot	否
數學運算	文字	數學問題解決 – 測量代數、微積分和應用題等主題的數學推理能力。	exact_match	zs_cot	是
strong_reject	文字	品質控管任務 – 測試模型可偵測和拒絕不適當、有害或不正確內容的能力。	偏轉	zs	是
IFEval	文字	指示追蹤評估 – 測量模型遵循指定指示的準確度，並根據規格完成任務。	正確性	zs	否

Benchmark	模態	Description	指標	策略	子任務可用
gen_qa	文字	自訂資料集評估 – 可讓您使用自有資料集進行基準測試，並使用如 ROUGE 和 BLEU 等指標將模型輸出與參考回答進行比較。	全部	gen_qa	否
llm_judge	文字	LLM-as-a-Judge 偏好設定比較 – 使用 Amazon Nova Judge 模型來判斷您的提示配對回應之間的偏好設定 (B 相較於 A)，計算 B 優於 A 的機率。	全部	評審	否
humaneval	文字	HumanEval - 旨在評估大型語言模型程式碼產生功能的基準資料集	pass@1	zs	否

Benchmark	模態	Description	指標	策略	子任務可用
mm_llm_judge	多模態（影像）	這個新基準的行為與 llm_judge 上述以文字為基礎的行為相同。唯一的差別是它支援映像推論。	全部	評審	否
rubric_llm_judge	Text	Rubric Judge 是建置在 Amazon Nova 2.0 Lite 上的增強LLM-as-a-judge 評估模型。與只提供偏好判定的 原始判斷模型 不同，Rubric Judge 會動態產生為每個提示量身打造的自訂評估條件，並跨多個維度指派精細分數。	全部	評審	否
aime_2024	Text	AIME 2024 - 美國邀請數學檢查問題測試進階數學推理和問題解決	exact_match	zs_cot	No

Benchmark	模態	Description	指標	策略	子任務可用
calendar_scheduling	Text	Natural Plan - 行事曆排程任務測試規劃能力，以跨多天 and 多人安排會議	exact_match	fs	No

下列是可用的 mmlu 子任務：

```

MMLU_SUBTASKS = [
    "abstract_algebra",
    "anatomy",
    "astronomy",
    "business_ethics",
    "clinical_knowledge",
    "college_biology",
    "college_chemistry",
    "college_computer_science",
    "college_mathematics",
    "college_medicine",
    "college_physics",
    "computer_security",
    "conceptual_physics",
    "econometrics",
    "electrical_engineering",
    "elementary_mathematics",
    "formal_logic",
    "global_facts",
    "high_school_biology",
    "high_school_chemistry",
    "high_school_computer_science",
    "high_school_european_history",
    "high_school_geography",
    "high_school_government_and_politics",
    "high_school_macroeconomics",
    "high_school_mathematics",
    "high_school_microeconomics",
    "high_school_physics",
    "high_school_psychology",

```

```
"high_school_statistics",  
"high_school_us_history",  
"high_school_world_history",  
"human_aging",  
"human_sexuality",  
"international_law",  
"jurisprudence",  
"logical_fallacies",  
"machine_learning",  
"management",  
"marketing",  
"medical_genetics",  
"miscellaneous",  
"moral_disputes",  
"moral_scenarios",  
"nutrition",  
"philosophy",  
"prehistory",  
"professional_accounting",  
"professional_law",  
"professional_medicine",  
"professional_psychology",  
"public_relations",  
"security_studies",  
"sociology",  
"us_foreign_policy",  
"virology",  
"world_religions"  
]
```

下列是可用的 bbh 子任務：

```
BBH_SUBTASKS = [  
    "boolean_expressions",  
    "causal_judgement",  
    "date_understanding",  
    "disambiguation_qa",  
    "dyck_languages",  
    "formal_fallacies",  
    "geometric_shapes",  
    "hyperbaton",  
    "logical_deduction_five_objects",  
    "logical_deduction_seven_objects",
```

```
"logical_deduction_three_objects",
"movie_recommendation",
"multistep_arithmetic_two",
"navigate",
"object_counting",
"penguins_in_a_table",
"reasoning_about_colored_objects",
"ruin_names",
"salient_translation_error_detection",
"snarks",
"sports_understanding",
"temporal_sequences",
"tracking_shuffled_objects_five_objects",
"tracking_shuffled_objects_seven_objects",
"tracking_shuffled_objects_three_objects",
"web_of_lies",
"word_sorting"
]
```

下列是可用的 math 子任務：

```
MATH_SUBTASKS = [
    "algebra",
    "counting_and_probability",
    "geometry",
    "intermediate_algebra",
    "number_theory",
    "prealgebra",
    "precalculus",
]
```

了解配方參數

執行組態

以下是一般執行組態及所涉及參數的說明。

```
run:
  name: eval_job_name
  model_type: amazon.nova-micro-v1:0:128k
  model_name_or_path: nova-micro/prod
  replicas: 1
```

```
data_s3_path: ""
output_s3_path: s3://output_path
mlflow_tracking_uri: ""
mlflow_experiment_name : ""
mlflow_run_name : ""
```

- **name** : (必要) 評估任務的描述性名稱。這有助於在 AWS 主控台中識別您的任務。
- **model_type** : (必要) 指定要使用的 Amazon Nova 模型變體。請勿手動修改此欄位。選項包括：
 - amazon.nova-micro-v1:0:128k
 - amazon.nova-lite-v1:0:300k
 - amazon.nova-pro-v1:0:300k
 - amazon.nova-2-lite-v1:0:256k
- **model_name_or_path** : (必要) 基本模型的路徑或訓練後檢查點的 S3 路徑。選項包括：
 - nova-micro/prod
 - nova-lite/prod
 - nova-pro/prod
 - nova-lite-2/prod
 - (訓練後檢查點的 S3 路徑) s3://<escrow bucket>/<job id>/outputs/checkpoints
- **replicas** : (必要) 分散式訓練中要使用的運算執行個體數目。您必須將此值設定為 1，因為不支援多節點。
- **data_s3_path** : (必要) 輸入資料集的 S3 路徑。除非您使用的是使用自有資料集或 LLM 即評審配方，否則請將此參數保留空白。
- **output_s3_path** : (必要) 存放輸出評估成品的 S3 路徑。請注意，輸出 S3 儲存貯體必須由建立任務的相同帳戶建立。
- **mlflow_tracking_uri** : (選用) 用於追蹤 MLflow 執行/實驗的 MLFlow 追蹤伺服器 ARN。請確保您具有從 SageMaker AI 執行角色存取追蹤伺服器的許可

評估組態

以下是模型評估組態及所涉及參數的說明。

```
evaluation:
  task: mmlu
  strategy: zs_cot
  subtask: mathematics
```

```
metric: accuracy
```

- **task** : (必要) 指定要使用的評估基準或任務。

支援的任務清單：

- mmlu
- mmlu_pro
- bbh
- gpqa
- 數學運算
- strong_reject
- gen_qa
- ifeval
- llm_judge
- humaneval
- mm_llm_judge
- rubric_llm_judge
- aime_2024
- calendar_scheduling
- humaneval
- **strategy** : (必要) 定義評估方法：
 - **zs_cot** : 零樣本思緒鏈 - 一種可提示大型語言模型的方法，鼓勵逐步推理，而無需明確的範例。
 - **zs** : 零樣本 - 一種無需任何先前訓練範例即可解決問題的方法。
 - **gen_qa** : 使用自有資料集配方特定的策略。
 - **judge** : Amazon Nova LLM 作為 Judge 和 mm_llm_judge 的特定策略。
- **subtask** : (選用且可移除) 針對特定評估任務指定特定子任務。如果您的任務沒有任何子任務，請從配方中移除此項目。
- **metric** : (必要) 要使用的評估指標。
 - 正確性 : 正確回答的百分比
 - **exact_match** : (對於 math 基準) 會傳回輸入預測字串與其參考完全相符的比率。

評估 • **偏轉** : (對於 strong reject 基準測試) 會將相對偏轉傳回基本模型和顯著性指標的差異。

- `pass@1` : (對於 humaneval 基準測試) 用來測量模型最高可信度預測符合正確回答之案例百分比的指標。
- `all` : 傳回下列指標：
 - 對於 `gen_qa` 和使用自有資料集基準，這會傳回下列指標：
 - `rouge1` : 測量所產生文字和參考文字之間一元語法 (一個字) 的重疊。
 - `rouge2` : 測量所產生文字和參考文字之間二元語法 (連續兩個字) 的重疊。
 - `rougeL` : 測量文字之間最長的共同子序列，允許相符項目中的差距。
 - `exact_match` : 二元分數 (0 或 1)，指出產生的文字是否逐字元完全符合參考文字。
 - `quasi_exact_match` : 類似於完全相符但更寬鬆，通常忽略大小寫、標點符號和空格差異。
 - `f1_score` : 精確率和召回率的調和平均數，可測量預測回答和參考回答之間的字詞重疊。
 - `f1_score_quasi` : 類似於 `f1_score`，但比對更寬鬆，使用會忽略次要差異的標準化文字比較。
 - `bleu` : 測量所產生文字和參考文字之間 n 元語法比對的精確率，常用於轉譯評估。
 - 對於 `llm_judge` 和 `mm_llm_judge`，使用您自己的資料集基準，請傳回下列指標：
 - `a_scores` : 跨向前和向後評估傳遞的 `response_A` 其獲勝次數。
 - `a_scores_stderr` : 跨配對判斷的 `response_A scores` 其標準誤。
 - `b_scores` : 跨向前和向後評估傳遞的 `response_B` 其獲勝次數。
 - `b_scores_stderr` : 跨配對判斷的 `response_B scores` 其標準誤。
 - `ties` : 將 `response_A` 和 `response_B` 評估為相等的判斷數目。
 - `ties_stderr` : 跨配對判斷的繫結其標準誤。
 - `inference_error` : 無法正確評估的判斷計數。
 - `inference_error_stderr` : 跨判斷的推論錯誤其標準誤。
 - `score` : 根據 `response_B` 其向前和向後傳遞中的獲勝來彙總分數。
 - `score_stderr` : 跨配對判斷的彙總分數其標準誤。
 - `winrate` : 使用 Bradley-Terry 機率計算將比 `response_A` 偏好 `response_B` 的機率。
 - `lower_rate` : 從引導取樣預估勝率的下限 (第 2.5 個百分位數)。

推論組態

以下是推論組態及所涉及參數的說明。所有參數都是選用的。

```
inference:
  max_new_tokens: 200
  top_k: -1
  top_p: 1.0
  temperature: 0
  top_logprobs: 10
  reasoning_effort: null # options: low/high to enable reasoning or null to disable
  reasoning
```

- `max_new_tokens` : 要產生的記號數目上限。這必須為整數。
- `top_k` : 要考慮的最高機率記號數量。這必須為整數。
- `top_p` : 記號抽樣的累積機率閾值。這必須是介於 0.0 到 1.0 (含) 之間的浮點數。
- `temperature` : 記號選擇中的隨機性。越大的值會導致越大的隨機性。使用 0 會讓結果具有決定性。此值必須是最小值為 0 的浮點數。
- `top_logprobs` : 推論回應中要傳回的最大 logprob 數目。此值必須是從 0 到 20 的整數。Logprob 包含考慮的輸出字符，以及訊息內容中傳回的每個輸出字符的日誌機率。
- `reasoning_effort` : 控制可推理模型的推理行為。`reasoning_effort` 只有在 `model_type` 指定可推理的模型時設定 (目前為 `amazon.nova-2-lite-v1:0:256k`)。可用選項為 `null` (如果未設定預設值；停用推理) `low`、或 `high`。

請注意，對於 `humaneval`，我們建議使用下列推論組態：

```
inference:
  top_k: 1
  max_new_tokens: 1600
  temperature: 0.0
```

MLFlow 組態

以下是 MLFlow 組態和所涉及參數的說明。所有參數都是選用的。

```
run:
  mlflow_tracking_uri: ""
  mlflow_experiment_name: ""
  mlflow_run_name: ""
```

- `mlflow_tracking_uri` : 選用) MLflow 追蹤伺服器的位置 (僅在 SMHP 上需要)

- `mlflow_experiment_name` : (選用) 將相關 ML 執行分組的實驗名稱
- `mlflow_run_name` : (選用) 實驗中特定訓練執行的自訂名稱

評估配方範例

Amazon Nova 提供四種類型的評估配方，可在 SageMaker HyperPod 配方 GitHub 儲存庫中找到。

一般文字基準配方

這些配方可讓您在一套完整的純文字基準之間評估 Amazon Nova 模型的基本功能。它們會以格式 `xxx_general_text_benchmark_eval.yaml` 提供。

使用自有資料集基準配方

這些配方可讓您使用自有資料集進行基準測試，並使用不同類型的指標比較模型輸出以參考回答。它們會以格式 `xxx_bring_your_own_dataset_eval.yaml` 提供。

以下是使用自有資料集需求：

- 檔案格式需求
 - 您必須包含一個 `gen_qa.jsonl` 包含評估範例的檔案。
 - 您的資料集必須上傳至 SageMaker 訓練任務可存取的 S3 位置。
 - 檔案必須遵循一般問與答資料集所需的結構描述格式。
- 結構描述格式需求 - JSONL 檔案中的每一行都必須是具有下列欄位的 JSON 物件：
 - `query` : (必要) 字串，包含需要回答的問題或指示
 - `response` : (必要) 字串，包含預期模型輸出
 - `system` : (選用) 字串，包含在處理查詢之前設定 AI 模型行為、角色或個性的系統提示。
 - `metadata` : (選用) 字串包含與項目相關聯的中繼資料，用於標記目的。

以下是使用自有資料集範例項目

```
{
  "system": "You are a english major with top marks in class who likes to give minimal word responses: ",
  "query": "What is the symbol that ends the sentence as a question",
  "response": "?"
}
```

```

    "system": "You are a pattern analysis specialist that provides succinct answers: ",
    "query": "What is the next number in this series? 1, 2, 4, 8, 16, ?",
    "response": "32"
  }
  {
    "system": "You have great attention to detail that follows instructions accurately: ",
    "query": "Repeat only the last two words of the following: I ate a hamburger today and it was kind of dry",
    "response": "of dry"
  }

```

若要使用您的自訂資料集，請使用下列必要欄位修改您的評估配方，切勿變更任何內容：

```

evaluation:
  task: gen_qa
  strategy: gen_qa
  metric: all

```

有下列限制：

- 每次評估只允許一個 JSONL 檔案。
- 該檔案必須嚴格遵循定義的結構描述。
- 內容長度限制：對於資料集中的每個範例，內容長度（包括系統 + 查詢提示）應小於 3.5k。

Nova LLM 即評審基準配方

Amazon Nova LLM 即評審是一種模型評估功能，可讓客戶將一個模型的回應品質與自訂資料集上的基準模型回應進行比較。它會採用具有提示、基準回應和挑戰者回應的資料集，並使用 Amazon Nova Judge 模型，根據 [Bradley-Terry 機率](#) 和配對比較提供勝率指標。

配方會以 `xxx_llm_judge_eval.yaml` 格式提供。

以下是 LLM 即評審需求：

- 檔案格式需求
 - 包含一個包含評估範例的 `llm_judge.jsonl` 檔案。檔案名稱必須是 `llm_judge.jsonl`。
 - 您的資料集必須上傳至 [SageMaker AI SageMaker HyperPod RIG](#) 可存取的 S3 位置。
 - 檔案必須遵循 `llm_judge.jsonl` 資料集所需的結構描述格式。

- 輸入資料集應確保所有記錄都不到 12k 內容長度。
- 結構描述格式需求 - JSONL 檔案中的每一行都必須是具有下列欄位的 JSON 物件：
 - prompt：(必要) 字串，包含所產生回應的提示。
 - response_A：字串，包含基準回應。
 - response_B：字串，包含與基準回應比較的替代回應。

以下是 LLM 即評審範例項目

```
{
  "prompt": "What is the most effective way to combat climate change?",
  "response_A": "The most effective way to combat climate change is through a combination of transitioning to renewable energy sources and implementing strict carbon pricing policies. This creates economic incentives for businesses to reduce emissions while promoting clean energy adoption.",
  "response_B": "We should focus on renewable energy. Solar and wind power are good. People should drive electric cars. Companies need to pollute less."
}
{
  "prompt": "Explain how a computer's CPU works",
  "response_A": "CPU is like brain of computer. It does math and makes computer work fast. Has lots of tiny parts inside.",
  "response_B": "A CPU (Central Processing Unit) functions through a fetch-execute cycle, where instructions are retrieved from memory, decoded, and executed through its arithmetic logic unit (ALU). It coordinates with cache memory and registers to process data efficiently using binary operations."
}
{
  "prompt": "How does photosynthesis work?",
  "response_A": "Plants do photosynthesis to make food. They use sunlight and water. It happens in leaves.",
  "response_B": "Photosynthesis is a complex biochemical process where plants convert light energy into chemical energy. They utilize chlorophyll to absorb sunlight, combining CO2 and water to produce glucose and oxygen through a series of chemical reactions in chloroplasts."
}
```

若要使用您的自訂資料集，請使用下列必要欄位修改您的評估配方，切勿變更任何內容：

```
evaluation:
  task: llm_judge
```

```
strategy: judge
metric: all
```

有下列限制：

- 每次評估只允許一個 JSONL 檔案。
- 該檔案必須嚴格遵循定義的結構描述。
- Amazon Nova Judge 模型在所有模型系列規格 (即 Lite、Micro 和 Pro) 之間皆相同。
- 目前不支援自訂判斷模型。
- 內容長度限制：對於資料集中的每個範例，內容長度（包括系統 + 查詢提示）應小於 7k。

Nova LLM 作為多模態（影像）基準配方的判斷

Nova LLM Judge for multi-modal（影像），簡稱為 Amazon Nova MM_LLM Judge，是一種模型評估功能，可讓您使用自訂資料集，比較一個模型的回應品質與基準模型的回應。它接受包含提示、基準回應和挑戰者回應的資料集，以及 Base64-encoded 字串形式的影像，然後使用 Amazon Nova Judge 模型，透過配對比較，根據 [Bradley-Terry](#) 機率提供獲勝率指標。配方格式：xxx_mm_llm_judge_eval.yaml。

Nova LLM 資料集需求

檔案格式：

- 包含評估範例的單一 mm_llm_judge.jsonl 檔案。檔案名稱必須剛好為 llm_judge.jsonl。
- 您必須將資料集上傳至 SageMaker Training Jobs 可存取的 S3 位置。
- 檔案必須遵循 mm_llm_judge 資料集所需的結構描述格式。
- 輸入資料集應確保所有記錄都低於 12 k 內容長度，不包括影像的屬性。

結構描述格式 - .jsonl 檔案中的每一行都必須是具有下列欄位的 JSON 物件。

- 必要欄位。

prompt：字串，包含所產生回應的提示。

images：陣列包含具有資料屬性的物件清單（值為 Base64-encoded 的影像字串）。

response_A：字串，包含基準回應。

`response_B`：字串，包含與基準回應比較的替代回應。

範例項目

為了便於閱讀，以下範例包含新的行和縮排，但在實際的資料集中，每個記錄應該位於單一行。

```
{
  "prompt": "what is in the image?",
  "images": [
    {
      "data": "data:image/jpeg;base64,/9j/2wBDAAQDAwQDAwQEAwQFBAQFBgo..."
    }
  ],
  "response_A": "a dog.",
  "response_B": "a cat.",
}
{
  "prompt": "how many animals in echo of the images?",
  "images": [
    {
      "data": "data:image/jpeg;base64,/9j/2wBDAAQDAwQDAwQEAwQFBAQFBgo..."
    },
    {
      "data": "data:image/jpeg;base64,/DKEafe3gihn..."
    }
  ],
  "response_A": "The first image contains one cat and the second image contains one dog",
  "response_B": "The first image has one animal and the second has one animal",
}
```

若要使用您的自訂資料集，請使用下列必要欄位修改您的評估配方，切勿變更任何內容：

```
evaluation:
  task: mm_llm_judge
  strategy: judge
  metric: all
```

限制

- 每次評估只允許一個 `.jsonl` 檔案。

- 該檔案必須嚴格遵循定義的結構描述。
- Nova MM Judge 模型僅支援影像參考。
- Nova MM Judge 模型在 Amazon Nova Lite 規格之間是相同的。
- 目前不支援自訂判斷模型。
- 不支援 Amazon S3 映像 URI。
- 輸入資料集應確保所有記錄都低於 12 k 內容長度，但不包括影像屬性。

以 Rubric 為基礎的判斷

Rubric Judge 是建置在 Amazon Nova 2.0 Lite 上的增強LLM-as-a-judge 評估模型。與只提供偏好判斷 (A>B、B>A 或 tie) 的[原始判斷模型](#)不同，Rubric Judge 會動態產生為每個提示量身打造的自訂評估條件，並跨多個維度指派精細分數。

主要功能：

- 產生動態條件：根據輸入提示自動建立相關的評估維度
- 加權分數：為每個條件指派重要性權重，以反映其相對重要性
- 精細評估：針對每個條件提供二進位 (true/false) 或擴展 (1-5) 的詳細分數
- 品質指標：計算連續品質分數 (0-1 比例)，以量化回應之間的差異程度

模型產生的範例條件：

```
price_validation:
  description: "The response includes validation to ensure price is a positive value."
  type: "scale"
  weight: 0.3
```

模型會根據所有產生的條件評估這兩個回應，然後使用這些條件層級分數來通知其最終偏好決策。

主題

- [配方組態](#)
- [輸入資料集格式](#)
- [評估輸出](#)
- [推理模型支援](#)

配方組態

Rubric Judge 配方

在您的配方 `task: rubric_llm_judge` 中設定 以啟用 Rubric Judge：

```
run:
  name: nova-eval-job-name                # [MODIFIABLE] Unique
  identifier for your evaluation job
  model_type: amazon.nova-2-lite-v1:0:256k # [FIXED] Rubric Judge model
  type
  model_name_or_path: "nova-lite-2/prod"   # [FIXED] Path to model
  checkpoint or identifier
  replicas: 1                             # [MODIFIABLE] Number of
  replicas for SageMaker Training job
  data_s3_path: ""                        # [FIXED] Leave empty for
  SageMaker Training job
  output_s3_path: ""                      # [FIXED] Leave empty for
  SageMaker Training job

evaluation:
  task: rubric_llm_judge                  # [FIXED] Evaluation task -
  enables Rubric Judge
  strategy: judge                         # [FIXED] Evaluation strategy
  metric: all                             # [FIXED] Metric calculation
  method

inference:
  max_new_tokens: 12000                  # [MODIFIABLE] Maximum tokens
  to generate
  top_k: -1                              # [MODIFIABLE] Top-k sampling
  parameter
  top_p: 1.0                             # [MODIFIABLE] Nucleus sampling
  parameter
  temperature: 0                          # [MODIFIABLE] Sampling
  temperature (0 = deterministic)
```

原始 LLM 做為判斷配方（用於比較）

原始判斷模型使用 `task: llm_judge`：

```
run:
  name: eval-job-name                    # [MODIFIABLE] Unique
  identifier for your evaluation job
```

```

model_type: amazon.nova-micro-v1:0:128k          # [FIXED] Model type
model_name_or_path: "nova-micro/prod"            # [FIXED] Path to model
checkpoint or identifier
replicas: 1                                       # [MODIFIABLE] Number of
replicas for SageMaker Training job
data_s3_path: ""                                # [FIXED] Leave empty for
SageMaker Training job
output_s3_path: ""                              # [FIXED] Leave empty for
SageMaker Training job

evaluation:
  task: llm_judge                                # [FIXED] Original judge task
  strategy: judge                                # [FIXED] Evaluation strategy
  metric: all                                     # [FIXED] Metric calculation
  method

inference:
  max_new_tokens: 12000                          # [MODIFIABLE] Maximum tokens
  to generate
  top_k: -1                                       # [MODIFIABLE] Top-k sampling
  parameter
  top_p: 1.0                                     # [MODIFIABLE] Nucleus sampling
  parameter
  temperature: 0                                 # [MODIFIABLE] Sampling
  temperature (0 = deterministic)

```

輸入資料集格式

輸入資料集格式與[原始判斷模型](#)相同：

必要欄位：

- prompt：包含輸入提示和指示的字串
- response_A：包含基準模型輸出的字串
- response_B：包含自訂模型輸出的字串

範例資料集 (JSONL 格式)：

```

{"prompt": "What is the most effective way to combat climate change?", "response_A":
"The most effective way to combat climate change is through a combination of
transitioning to renewable energy sources and implementing strict carbon pricing

```


policies. This creates economic incentives for businesses to reduce emissions while promoting clean energy adoption.", "response_B": "We should focus on renewable energy. Solar and wind power are good. People should drive electric cars. Companies need to pollute less."}

```
{"prompt": "Explain how a computer's CPU works", "response_A": "CPU is like brain of computer. It does math and makes computer work fast. Has lots of tiny parts inside.", "response_B": "A CPU (Central Processing Unit) functions through a fetch-execute cycle, where instructions are retrieved from memory, decoded, and executed through its arithmetic logic unit (ALU). It coordinates with cache memory and registers to process data efficiently using binary operations."}
```

```
{"prompt": "How does photosynthesis work?", "response_A": "Plants do photosynthesis to make food. They use sunlight and water. It happens in leaves.", "response_B": "Photosynthesis is a complex biochemical process where plants convert light energy into chemical energy. They utilize chlorophyll to absorb sunlight, combining CO2 and water to produce glucose and oxygen through a series of chemical reactions in chloroplasts."}
```

格式需求：

- 每個項目必須是單行 JSON 物件
- 使用換行分隔項目
- 遵循範例所示的確切欄位命名

評估輸出

輸出結構

Rubric Judge 與原始判斷模型相比，會產生增強的評估指標：

```
{
  "config_general": {
    "lighteval_sha": "string",
    "num_fewshot_seeds": "int",
    "max_samples": "int | null",
    "job_id": "int",
    "start_time": "float",
    "end_time": "float",
    "total_evaluation_time_seconds": "string",
    "model_name": "string",
    "model_sha": "string",
    "model_dtype": "string | null",
```

```

    "model_size": "string"
  },
  "results": {
    "custom|rubric_llm_judge_judge|0": {
      "a_scores": "float",
      "a_scores_stderr": "float",
      "b_scores": "float",
      "b_scores_stderr": "float",
      "ties": "float",
      "ties_stderr": "float",
      "inference_error": "float",
      "inference_error_stderr": "float",
      "score": "float",
      "score_stderr": "float",
      "weighted_score_A": "float",
      "weighted_score_A_stderr": "float",
      "weighted_score_B": "float",
      "weighted_score_B_stderr": "float",
      "score_margin": "float",
      "score_margin_stderr": "float",
      "winrate": "float",
      "lower_rate": "float",
      "upper_rate": "float"
    }
  },
  "versions": {
    "custom|rubric_llm_judge_judge|0": "int"
  }
}

```

Rubric Judge 中的新指標

以下六個指標對 Rubric Judge 來說是唯一的，並提供精細的品質評估：

指標	Description
weighted_score_A	response_A 在所有模型產生的評估條件中的平均標準化品質分數。分數依標準重要性加權，並標準化為 0-1 比例（較高 = 品質更佳）
weighted_score_A_stderr	weighted_score_A 平均值的標準錯誤，表示統計不確定性

指標	Description
weighted_score_B	response_B 在所有模型產生的評估條件中的平均標準化品質分數。分數依標準重要性加權，並標準化為 0-1 比例（較高 = 品質更佳）
weighted_score_B_stderr	weighted_score_B 平均值的標準錯誤，表示統計不確定性
score_margin	加權分數之間的差異（計算方式為 weighted_score_A - weighted_score_B）。範圍：-1.0 到 1.0。正值 = response_A 較佳；負值 = response_B 較佳；接近零 = 相似品質
score_margin_stderr	score_margin 平均值的標準誤差，表示品質差異測量的不確定性

了解加權分數指標

目的：加權分數提供持續的品質測量，以補充二進位偏好設定的判斷，進而更深入地了解模型效能。

與原始判斷的主要差異：

- 原始判斷：僅輸出離散偏好設定 (A>B、B>A、A=B)
- Rubric Judge：根據自訂條件輸出偏好設定和持續品質分數 (0-1 比例)

解譯 score_margin：

- score_margin = -0.128：Response_B 得分比 response_A 高 12.8 個百分點
- |score_margin| < 0.1：縮小品質差異（關閉決策）
- |score_margin| > 0.2：清楚的品質差異（可信決策）

使用案例：

- 模型改進：識別模型表現不佳的特定領域
- 品質量化：測量效能差距的幅度，而不只是取捨率
- 可信度評估：區分密切決策和明確的品質差異

⚠ Important

最終判斷仍以判斷模型的明確偏好標籤為基礎，以保留整體推理，並透過向前/向後評估確保適當的位置偏差緩解。加權分數做為可觀測性工具，而不是主要判定的替代項目。

計算方法

透過下列程序計算加權分數：

- 擷取條件資料：剖析判斷器的 YAML 輸出以擷取條件分數和權重
- 標準化分數：
 - 規模類型條件 (1-5)：透過計算將標準化為 $0-1$ ($\text{score} - 1$) / 4
 - 二進位條件 (true/false)：轉換為 1.0/0.0
- 套用權重：將每個標準化分數乘以其條件權重
- 彙總：每個回應的所有加權分數總和
- 計算邊界：運算 $\text{score\_margin} = \text{weighted\_score\_A} - \text{weighted\_score\_B}$

範例：如果 response_A 的加權總和為 0.65，而 response_B 的加權總和為 0.78，則 score_margin 會是 -0.13，表示 response_B 在所有加權條件的品質提高 13 個百分點。

推理模型支援

推理模型支援可使用推理能力的 Amazon Nova 模型進行評估，這些模型在產生最終回應之前執行明確的內部推理。此功能會透過 reasoning_effort 參數使用 API 層級控制來動態啟用或停用推理功能，進而改善複雜分析任務的回應品質。

支援的模型：

- amazon.nova-2-lite-v1 : 0 : 256k

配方組態

將 reasoning_effort 參數新增至配方的 inference 區段，以啟用推理：

```
run:
  name: eval-job-name                                # [MODIFIABLE] Unique
  identifier for your evaluation job
```

```

    model_type: amazon.nova-2-lite-v1:0:256k          # [FIXED] Must be a reasoning-
supported model
    model_name_or_path: nova-lite-2/prod              # [FIXED] Path to model
checkpoint or identifier
    replicas: 1                                       # [MODIFIABLE] Number of
replicas for SageMaker Training job
    data_s3_path: ""                                # [MODIFIABLE] Leave empty for
SageMaker Training job; optional for SageMaker SageMaker HyperPod job
    output_s3_path: ""                               # [MODIFIABLE] Output path for
SageMaker SageMaker HyperPod job (not compatible with SageMaker Training jobs)

evaluation:
    task: mmlu                                       # [MODIFIABLE] Evaluation task
    strategy: generate                               # [MODIFIABLE] Evaluation
strategy
    metric: all                                      # [MODIFIABLE] Metric
calculation method

inference:
    reasoning_effort: high                           # [MODIFIABLE] Enables
reasoning mode; options: low/medium/high or null to disable
    max_new_tokens: 200                             # [MODIFIABLE] Maximum tokens
to generate
    top_k: 50                                        # [MODIFIABLE] Top-k sampling
parameter
    top_p: 1.0                                       # [MODIFIABLE] Nucleus
sampling parameter
    temperature: 0                                   # [MODIFIABLE] Sampling
temperature (0 = deterministic)

```

使用 reasoning_effort 參數

reasoning_effort 參數控制可推理模型的推理行為。

先決條件：

- 模型相容性：reasoning_effort 只有在 model_type 指定可推理的模型時設定（目前為 amazon.nova-2-lite-v1:0:256k）
- 錯誤處理：使用 reasoning_effort 搭配不支援的模型會失敗 ConfigValidationError: "Reasoning mode is enabled but model '{model_type}' does not support reasoning. Please use a reasoning-capable model or disable reasoning mode."

可用的選項：

選項	Behavior (行為)	字符限制	使用案例
null (預設)	停用推理模式	N/A	無推理額外負荷的標準評估
low	啟用具有限制條件的推理	用於內部推理的 4,000 個字符	需要簡潔推理的案例；針對速度和成本進行最佳化
high	啟用無限制的推理	內部推理沒有字符限制	需要廣泛分析和step-by-step推理的複雜問題

何時啟用推理

針對下列項目使用推理模式 (**medium**、**low**或 **high**)：

- 複雜的問題解決任務（數學、邏輯拼圖、編碼）
- 需要中繼推理的多步驟分析問題
- 詳細說明或step-by-step思考可提高準確性的任務
- 回應品質優先於速度的情況

針對下列項目使用非原因模式（省略參數）：

- 簡單問答或事實查詢
- 創意撰寫任務
- 當更快的回應時間至關重要時
- 應排除推理開銷的績效基準
- 推理無法改善任務效能時的成本最佳化

疑難排解

錯誤：「啟用原因模式，但模型不支援推理」

原因：reasoning_effort 參數設定為非空值，但指定的 model_type 不支援推理。

解決方法：

- 確認您的模型類型為 amazon.nova-2-lite-v1:0:256k
- 如果使用不同的模型，請切換到具有推理功能的模型，或從配方中移除 reasoning_effort 參數

啟動評估任務

以下提供建議的評估執行個體類型和模型類型組態：

```
# Install Dependencies (Helm - https://helm.sh/docs/intro/install/)
curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
rm -f ./get_helm.sh

# Install the SageMaker HyperPod CLI
git clone --recurse-submodules https://github.com/aws/sagemaker-hyperpod-cli.git
git checkout -b release_v2
cd sagemaker-hyperpod-cli
pip install .

# Verify the installation
hyperpod --help

# Connect to a SageMaker HyperPod Cluster
hyperpod connect-cluster --cluster-name cluster-name

# Submit the Job using the recipe for eval
# Namespace by default should be kubeflow
hyperpod start-job [--namespace namespace] --recipe evaluation/nova/
nova_micro_p5_48xl_general_text_benchmark_eval --override-parameters \
'{
    "instance_type": "p5d.48xlarge",
    "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-V2-latest",
    "recipes.run.name": custom-run-name,
    "recipes.run.model_type": model_type,
    "recipes.run.model_name_or_path": "model name or finetune checkpoint s3uri,"
```

```

    "recipes.run.data_s3_path": s3 for input data only for genqa and llm_judge, must be
    full S3 path that include filename,
  }'

# List jobs
hyperpod list-jobs [--namespace namespace] [--all-namespaces]

# Getting Job details
hyperpod get-job --job-name job-name [--namespace namespace] [--verbose]

# Listing Pods
hyperpod list-pods --job-name job-name --namespace namespace

# Cancel Job
hyperpod cancel-job --job-name job-name [--namespace namespace]

```

您也應能夠透過 Amazon EKS 叢集主控台檢視任務狀態。

存取和分析評估結果

評估任務成功完成後，您可以使用本節中的資訊來存取和分析結果。根據配方中定義的 `output_s3_path` (例如 `s3://output_path/`)，輸出結構如下：

```

job_name/
### eval-result/
#   ### results_[timestamp].json
#   ### inference_output.jsonl (only present for gen_qa)
#   ### details/
#       ### model/
#           ### execution-date-time/
#               ###details_task_name_#_datetime.parquet
### tensorboard-results/
    ### eval/
        ### events.out.tfevents.[timestamp]

```

指標結果會存放在指定的 S3 輸出位置 `s3://output_path/job_name/eval-result/result-timestamp.json`。

Tensorboard 結果會存放在 S3 路徑 `s3://output_path/job_name/eval-tensorboard-result/eval/event.out.tfevents.epoch+ip` 中。

`llm_judge` 和 `strong_reject` 除外，所有的推論輸出都存放在 S3 路徑：`s3://output_path/job_name/eval-result/details/model/taskname.parquet`。

對於 `gen_qa` , `inference_output.jsonl` 檔案包含每個 JSON 物件的下列欄位：

- 提示 - 提交到模型的最終提示
- 推論 - 來自模型的原始推論輸出
- gold - 來自輸入資料集的目標回應
- 中繼資料 - 如果提供，來自輸入資料集的中繼資料字串

若要在 Tensorboard 中視覺化您的評估指標，請完成下列步驟：

1. 導覽至 SageMaker AI Tensorboard。
2. 選取 S3 資料夾。
3. 新增 S3 資料夾路徑，例如 `s3://output_path/job-name/eval-tensorboard-result/eval`。
4. 等待同步完成

時間序列、純量和文字視覺化可供使用。

建議遵循下列最佳實務：

- 透過模型和基準類型讓輸出路徑井然有序。
- 維持一致的命名慣例，以便於追蹤。
- 將擷取的結果儲存在安全的位置。
- 監控 TensorBoard 同步狀態以成功載入資料。

您可以在 CloudWatch 日誌群組 中找到 SageMaker HyperPod 任務錯誤日誌 `/aws/sagemaker/Clusters/cluster-id`。

日誌機率輸出格式

在推論設定中設定 `top_logprobs` 時，評估輸出會在 `parquet` 檔案中包含字符層級日誌機率。每個字符位置都包含最佳候選字符的字典，其日誌機率位於下列結構中：

```
{
  "Ġint": {"logprob_value": -17.8125, "decoded_value": " int"},
  "Ġthe": {"logprob_value": -2.345, "decoded_value": " the"}
}
```

每個字符項目都包含：

- `logprob_value`：字符的日誌機率值
- `decoded_value`：字符的人類可讀取解碼字串表示

原始字符化器字符用作字典索引鍵，以確保唯一性，同時`decoded_value`提供可讀取的解釋。

RFT 評估

Note

只有在您是 Amazon Nova Forge 客戶時，才能在您自己的 AWS 環境中透過遠端獎勵函數進行評估。

Important

`rl_env` 組態欄位僅用於評估，而非訓練。在訓練期間，您可以使用 `reward_lambda_arn`（單迴轉）或 BYOO 基礎設施搭配 `rollout.delegate: true`（多迴轉）來設定獎勵函數。

什麼是 RFT 評估？

RFT 評估可讓您在強化學習訓練之前、期間或之後，使用自訂獎勵函數評估模型的效能。與使用預先定義指標的標準評估不同，RFT 評估可讓您透過 Lambda 函數定義自己的成功條件，該函數會根據您的特定需求對模型輸出進行評分。

為什麼使用 RFT 評估？

評估對於判斷 RL 微調程序是否具有下列項目至關重要：

- 改善模型與特定使用案例和人力值的一致性
- 維護或改善關鍵任務的模型功能
- 避免意外的副作用，例如降低事實性、增加詳細程度，或降低其他任務的效能
- 符合獎勵函數定義的自訂成功條件

何時使用 RFT 評估

在這些案例中使用 RFT 評估：

- RFT 訓練之前：在您的評估資料集上建立基準指標
- 在 RFT 訓練期間：使用中繼檢查點監控訓練進度
- RFT 訓練後：驗證最終模型是否符合您的需求
- 比較模型：使用一致的獎勵條件評估多個模型版本

Note

當您需要自訂的網域特定指標時，請使用 RFT 評估。對於一般用途評估（準確性、複雜度、BLEU），請使用標準評估方法。

主題

- [資料格式要求](#)
- [準備您的評估配方](#)
- [預設獎勵函數](#)
- [建立獎勵函數](#)
- [IAM 許可](#)
- [執行評估任務](#)
- [了解評估結果](#)

資料格式要求

輸入資料結構

RFT 評估輸入資料必須遵循 OpenAI 強化微調格式。每個範例都是 JSON 物件，其中包含：

- messages：具有 system 和 user 角色的對話轉場陣列
- 選用的其他中繼資料，例如 reference_answer

資料格式範例

下列範例顯示所需的格式：

```
{
```

```

"messages": [
  {
    "role": "user",
    "content": [
      {
        "type": "text",
        "text": "Solve for x. Return only JSON like {\"x\": <number>}. Equation: 2x + 5 = 13"
      }
    ]
  },
  {
    "role": "assistant",
    "content": [
      {
        "type": "text",
        "text": "x = 4"
      }
    ]
  }
],
"reference_answer": {
  "x": 4
}
}

```

目前限制

下列限制適用於 RFT 評估：

- 僅限文字：不支援多模態輸入（影像、音訊、視訊）
- 單轉對話：僅支援單一使用者訊息（無多轉對話）
- JSON 格式：輸入資料必須是 JSONL 格式（每行一個 JSON 物件）
- 模型輸出：在從指定模型產生的完成時執行評估

準備您的評估配方

範例配方組態

下列範例顯示完整的 RFT 評估配方：

```

run:
  name: nova-lite-rft-eval-job
  model_type: amazon.nova-lite-v1:0:300k
  model_name_or_path: s3://escrow_bucket/model_location # [MODIFIABLE] S3 path to
your model or model identifier
  replicas: 1 # [MODIFIABLE] For SageMaker
Training jobs only; fixed for SageMaker HyperPod jobs
  data_s3_path: "" # [REQUIRED FOR HYPERPOD]
Leave empty for SageMaker Training jobs

```

```

    output_s3_path: ""                                # [REQUIRED] Output artifact
    S3 path for evaluation results

evaluation:
    task: rft_eval                                    # [FIXED] Do not modify
    strategy: rft_eval                                # [FIXED] Do not modify
    metric: all                                         # [FIXED] Do not modify

# Inference Configuration
inference:
    max_new_tokens: 8196                              # [MODIFIABLE] Maximum tokens
    to generate
    top_k: -1                                           # [MODIFIABLE] Top-k sampling
    parameter
    top_p: 1.0                                           # [MODIFIABLE] Nucleus
    sampling parameter
    temperature: 0                                       # [MODIFIABLE] Sampling
    temperature (0 = deterministic)
    top_logprobs: 0

# Evaluation Environment Configuration (NOT used in training)
rl_env:
    reward_lambda_arn: arn:aws:lambda:<region>:<account_id>:function:<reward-function-
    name>

```

預設獎勵函數

兩個預設獎勵函數 (prime_code 和 prime_math) 可作為 Lambda 層使用，以便與您的 RFT Lambda 函數輕鬆整合。

概觀

這些預設集函數提供out-of-the-box評估功能：

- prime_code：程式碼產生和正確性評估
- prime_math：數學推理和問題解決評估

快速設定

若要使用預設獎勵函數：

1. 從 [nova-custom-eval-sdk 版本](#) 下載 Lambda layer

2. 使用 CLI 發佈 Lambda AWS 層：

```
aws lambda publish-layer-version \  
    --layer-name preset-function-layer \  
    --description "Preset reward function layer with dependencies" \  
    --zip-file fileb://universal_reward_layer.zip \  
    --compatible-runtimes python3.9 python3.10 python3.11 python3.12 \  
    --compatible-architectures x86_64 arm64
```

3. 在 AWS 主控台中將 layer 新增至 Lambda 函數（從自訂 layer 選取 preset-function-layer，並針對凹凸相依性新增 AWSSDKPandas-Python312）

4. 在 Lambda 程式碼中匯入和使用：

```
from prime_code import compute_score # For code evaluation  
from prime_math import compute_score # For math evaluation
```

prime_code 函數

目的：針對測試案例執行程式碼並測量正確性，藉此評估 Python 程式碼產生任務。

來自評估的範例輸入資料集格式：

```
{"messages":[{"role":"user","content":"Write a function that returns the sum of  
two numbers."}], "reference_answer":{"inputs":["3\n5", "10\n-2", "0\n0"], "outputs":  
["8", "8", "0"]}}  
{"messages":[{"role":"user","content":"Write a function to check if a number  
is even."}], "reference_answer":{"inputs":["4", "7", "0", "-2"], "outputs":  
["True", "False", "True", "True"]}}
```

主要功能：

- 從 Markdown 程式碼區塊自動擷取程式碼
- 函數偵測和以呼叫為基礎的測試
- 具有逾時保護的測試案例執行
- 語法驗證和編譯檢查
- 使用追蹤傳回的詳細錯誤報告

prime_math 函數

目的：使用符號數學支援評估數學推理和問題解決功能。

輸入格式：

```
{"messages":[{"role":"user","content":"What is the derivative of x^2 + 3x?."}], "reference_answer":"2*x + 3"}
```

主要功能：

- 使用 SymPy 的符號數學評估
- 多重答案格式 (LaTeX、純文字、符號)
- 數學相等性檢查
- 表達式標準化和簡化

最佳實務

使用預設獎勵函數時，請遵循下列最佳實務：

- 在測試案例中使用適當的資料類型（整數與字串、布林值與 "True"）
- 在程式碼問題中提供明確的函數簽章
- 在測試輸入中包含邊緣案例（零、負數、空輸入）
- 在參考答案中一致地格式化數學表達式
- 在部署之前，使用範例資料測試獎勵函數

建立獎勵函數

Lambda ARN

您必須參考 Lambda ARN 的下列格式：

```
"arn:aws:lambda:*:*:function:*SageMaker*"
```

如果 Lambda 沒有此命名機制，任務會失敗並出現此錯誤：

```
[ERROR] Unexpected error: lambda_arn must contain one of: ['SageMaker', 'sagemaker', 'Sagemaker'] when running on SMHP platform (Key: lambda_arn)
```

Lambda 函數結構

您的 Lambda 函數會接收批次模型輸出並傳回獎勵分數。以下是範例實作：

```
from typing import List, Any
import json
import re
from dataclasses import asdict, dataclass

@dataclass
class MetricResult:
    """Individual metric result."""
    name: str
    value: float
    type: str

@dataclass
class RewardOutput:
    """Reward service output."""
    id: str
    aggregate_reward_score: float
    metrics_list: List[MetricResult]

def lambda_handler(event, context):
    """ Main lambda handler """
    return lambda_grader(event)

def lambda_grader(samples: list[dict]) -> list[dict]:
    """ Core grader function """
    scores: List[RewardOutput] = []

    for sample in samples:
        print("Sample: ", json.dumps(sample, indent=2))

        # Extract components
        idx = sample.get("id", "no id")
        if not idx or idx == "no id":
            print(f"ID is None/empty for sample: {sample}")

        ground_truth = sample.get("reference_answer")

        if "messages" not in sample:
```



```
        print(f"Messages is None/empty for id: {idx}")
        continue

    if ground_truth is None:
        print(f"No answer found in ground truth for id: {idx}")
        continue

    # Get model's response (last turn is assistant turn)
    last_message = sample["messages"][-1]

    if last_message["role"] != "nova_assistant":
        print(f"Last message is not from assistant for id: {idx}")
        continue

    if "content" not in last_message:
        print(f"Completion text is empty for id: {idx}")
        continue

    model_text = last_message["content"]

    # --- Actual scoring logic (lexical overlap) ---
    ground_truth_text = _extract_ground_truth_text(ground_truth)

    # Calculate main score and individual metrics
    overlap_score = _lexical_overlap_score(model_text, ground_truth_text)

    # Create two separate metrics as in the first implementation
    accuracy_score = overlap_score # Use overlap as accuracy
    fluency_score = _calculate_fluency(model_text) # New function for fluency

    # Create individual metrics
    metrics_list = [
        MetricResult(name="accuracy", value=accuracy_score, type="Metric"),
        MetricResult(name="fluency", value=fluency_score, type="Reward")
    ]

    ro = RewardOutput(
        id=idx,
        aggregate_reward_score=overlap_score,
        metrics_list=metrics_list
    )

    print(f"Response for id: {idx} is {ro}")
    scores.append(ro)
```

```

# Convert to dict format
result = []
for score in scores:
    result.append({
        "id": score.id,
        "aggregate_reward_score": score.aggregate_reward_score,
        "metrics_list": [asdict(metric) for metric in score.metrics_list]
    })

return result

def _extract_ground_truth_text(ground_truth: Any) -> str:
    """
    Turn the `ground_truth` field into a plain string.
    """
    if isinstance(ground_truth, str):
        return ground_truth

    if isinstance(ground_truth, dict):
        # Common patterns: { "explanation": "...", "answer": "..." }
        if "explanation" in ground_truth and isinstance(ground_truth["explanation"],
str):
            return ground_truth["explanation"]
        if "answer" in ground_truth and isinstance(ground_truth["answer"], str):
            return ground_truth["answer"]
        # Fallback: stringify the whole dict
        return json.dumps(ground_truth, ensure_ascii=False)

    # Fallback: stringify anything else
    return str(ground_truth)

def _tokenize(text: str) -> List[str]:
    # Very simple tokenizer: lowercase + alphanumeric word chunks
    return re.findall(r"\w+", text.lower())

def _lexical_overlap_score(model_text: str, ground_truth_text: str) -> float:
    """
    Simple lexical overlap score in [0, 1]:
    score = |tokens(model) ∩ tokens(gt)| / |tokens(gt)|
    """

```

```
gt_tokens = _tokenize(ground_truth_text)
model_tokens = _tokenize(model_text)

if not gt_tokens:
    return 0.0

gt_set = set(gt_tokens)
model_set = set(model_tokens)
common = gt_set & model_set

return len(common) / len(gt_set)

def _calculate_fluency(text: str) -> float:
    """
    Calculate a simple fluency score based on:
    - Average word length
    - Text length
    - Sentence structure

    Returns a score between 0 and 1.
    """
    # Simple implementation - could be enhanced with more sophisticated NLP
    words = _tokenize(text)

    if not words:
        return 0.0

    # Average word length normalized to [0,1] range
    # Assumption: average English word is ~5 chars, so normalize around that
    avg_word_len = sum(len(word) for word in words) / len(words)
    word_len_score = min(avg_word_len / 10, 1.0)

    # Text length score - favor reasonable length responses
    ideal_length = 100 # words
    length_score = min(len(words) / ideal_length, 1.0)

    # Simple sentence structure check (periods, question marks, etc.)
    sentence_count = len(re.findall(r'[.!?]+', text)) + 1
    sentence_ratio = min(sentence_count / (len(words) / 15), 1.0)

    # Combine scores
    fluency_score = (word_len_score + length_score + sentence_ratio) / 3
```

```
return fluency_score
```

Lambda 請求格式

您的 Lambda 函數會以此格式接收資料：

```
[
  {
    "id": "sample-001",
    "messages": [
      {
        "role": "user",
        "content": [
          {
            "type": "text",
            "text": "Do you have a dedicated security team?"
          }
        ]
      },
      {
        "role": "nova_assistant",
        "content": [
          {
            "type": "text",
            "text": "As an AI developed by Company, I don't have a dedicated security team in the traditional sense. However, the development and deployment of AI systems like me involve extensive security measures, including data encryption, user privacy protection, and other safeguards to ensure safe and responsible use."
          }
        ]
      }
    ],
    "reference_answer": {
      "compliant": "No",
      "explanation": "As an AI developed by Company, I do not have a traditional security team. However, the deployment involves stringent safety measures, such as encryption and privacy safeguards."
    }
  }
]
```

Note

訊息結構包含巢狀content陣列，符合輸入資料格式。最後一個具有 角色的訊息nova_assistant包含模型產生的回應。

Lambda 回應格式

您的 Lambda 函數必須以下列格式傳回資料：

```
[
  {
    "id": "sample-001",
    "aggregate_reward_score": 0.75,
    "metrics_list": [
      {
        "name": "accuracy",
        "value": 0.85,
        "type": "Metric"
      },
      {
        "name": "fluency",
        "value": 0.90,
        "type": "Reward"
      }
    ]
  }
]
```

回應欄位：

- id：必須符合輸入範例 ID
- aggregate_reward_score：整體分數（通常為 0.0 到 1.0）
- metrics_list：具有下列項目的個別指標陣列：
 - name：指標識別符（例如 "accuracy"、"fluency"）
 - value：指標分數（通常為 0.0 到 1.0）
 - type：「指標」（用於報告）或「獎勵」（用於訓練）

IAM 許可

所需的許可

您的 SageMaker AI 執行角色必須具有叫用 Lambda 函數的許可。將此政策新增至 SageMaker AI 執行角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:region:account-id:function:function-name"
    }
  ]
}
```

Lambda 執行角色

您的 Lambda 函數的執行角色需要基本的 Lambda 執行許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

其他許可：如果您的 Lambda 函數存取其他 AWS 服務（例如，用於參考資料的 Amazon S3、用於記錄的 DynamoDB），請將這些許可新增至 Lambda 執行角色。

執行評估任務

1. 準備您的資料

- 根據資料格式要求格式化評估資料
- 將您的 JSONL 檔案上傳至 Amazon S3：s3://your-bucket/eval-data/eval_data.jsonl

2. 設定您的配方

使用您的組態更新範例配方：

- model_name_or_path 設定為您的模型位置
- lambda_arn 設定為您的獎勵函數 ARN
- output_s3_path 設定為您想要的輸出位置
- 視需要調整inference參數

將配方儲存為 rft_eval_recipe.yaml

3. 執行評估

使用提供的筆記本執行評估任務：[Nova 模型評估筆記本](#)

4. 監控進度

透過下列方式監控您的評估任務：

- SageMaker AI 主控台：檢查任務狀態和日誌
- CloudWatch Logs：檢視詳細的執行日誌
- Lambda 日誌：偵錯獎勵函數問題

了解評估結果

輸出格式

評估任務會以 JSONL 格式將結果輸出至您指定的 Amazon S3 位置。每行包含一個範例的評估結果：

```
{
  "id": "sample-001",
  "aggregate_reward_score": 0.75,
  "metrics_list": [
```

```
{
  "name": "accuracy",
  "value": 0.85,
  "type": "Metric"
},
{
  "name": "fluency",
  "value": 0.90,
  "type": "Reward"
}
]
```

Note

RFT 評估任務輸出與 Lambda 回應格式相同。評估服務會在不修改的情況下傳遞 Lambda 函數的回應，確保獎勵計算與最終結果之間的一致性。

解譯結果

彙總獎勵分數：

- 範圍：通常為 0.0（最差）到 1.0（最佳），但取決於您的實作
- 目的：總結整體效能的單一數字
- 用量：比較模型，追蹤訓練的改善

個別指標：

- 指標類型：分析的資訊指標
- 獎勵類型：RFT 訓練期間使用的指標
- 解譯：較高的值通常表示效能更好（除非您設計反向指標）

效能基準

什麼構成「良好」效能取決於您的使用案例：

分數範圍	解譯	Action
0.8 - 1.0	卓越	模型已準備好進行部署
0.6 - 0.8	好	次要改善可能有益
0.4 - 0.6	公平	需要大幅改善
0.0 - 0.4	不佳	檢閱訓練資料和獎勵函數

Important

這些是一般準則。根據業務需求、基準模型效能、領域特定限制，以及進一步訓練的成本利益分析，定義您自己的閾值。

使用 MLflow 監控 HyperPod 任務

您可以使用 MLflow 在 SageMaker HyperPod 上追蹤和監控訓練任務。請依照下列步驟設定 MLflow，並將其連接到您的訓練配方。

建立 MLflow 應用程式

範例 AWS CLI 命令

```
aws sagemaker-mlflow create-mlflow-app \
  --name <app-name> \
  --artifact-store-uri <s3-bucket-name> \
  --role-arn <role-arn> \
  --region <region-name>
```

範例輸出

```
{
  "Arn": "arn:aws:sagemaker:us-east-1:111122223333:mlflow-app/app-LGZE0Z2UY4NZ"
}
```

產生預先簽章的 URL

範例 AWS CLI 命令

```
aws sagemaker-mlflow create-presigned-mlflow-app-url \  
  --arn <app-arn> \  
  --region <region-name> \  
  --output text
```

範例輸出

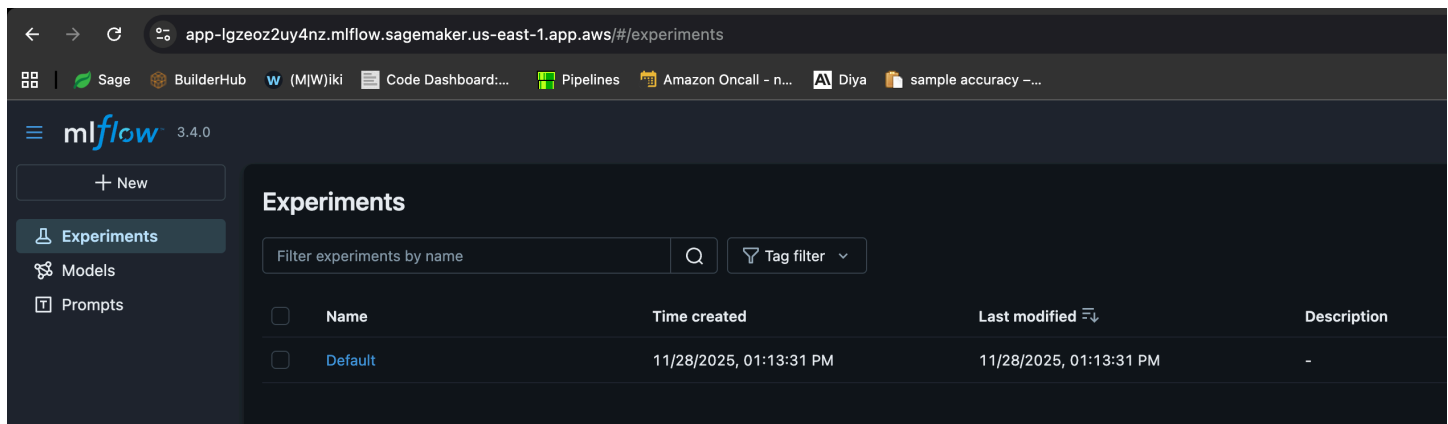
```
https://app-LGZE0Z2UY4NZ.mlflow.sagemaker.us-east-1.app.aws/auth?  
authToken=eyJhbGciOiJIUzI1NiJ9.eyJhdXRoVG9rZW5JZCI6IkxETVBPUyIsImZhc0NyZWRLbnRpYWxzIjoiQWdWNGhD
```

開啟預先簽章的 URL 並檢視應用程式

Click

```
https://app-LGZE0Z2UY4NZ.mlflow.sagemaker.us-east-1.app.aws/auth?  
authToken=eyJhbGciOiJIUzI1NiJ9.eyJhdXRoVG9rZW5JZCI6IkxETVBPUyIsImZhc0NyZWRLbnRpYWxzIjoiQWdWNGhD
```

檢視

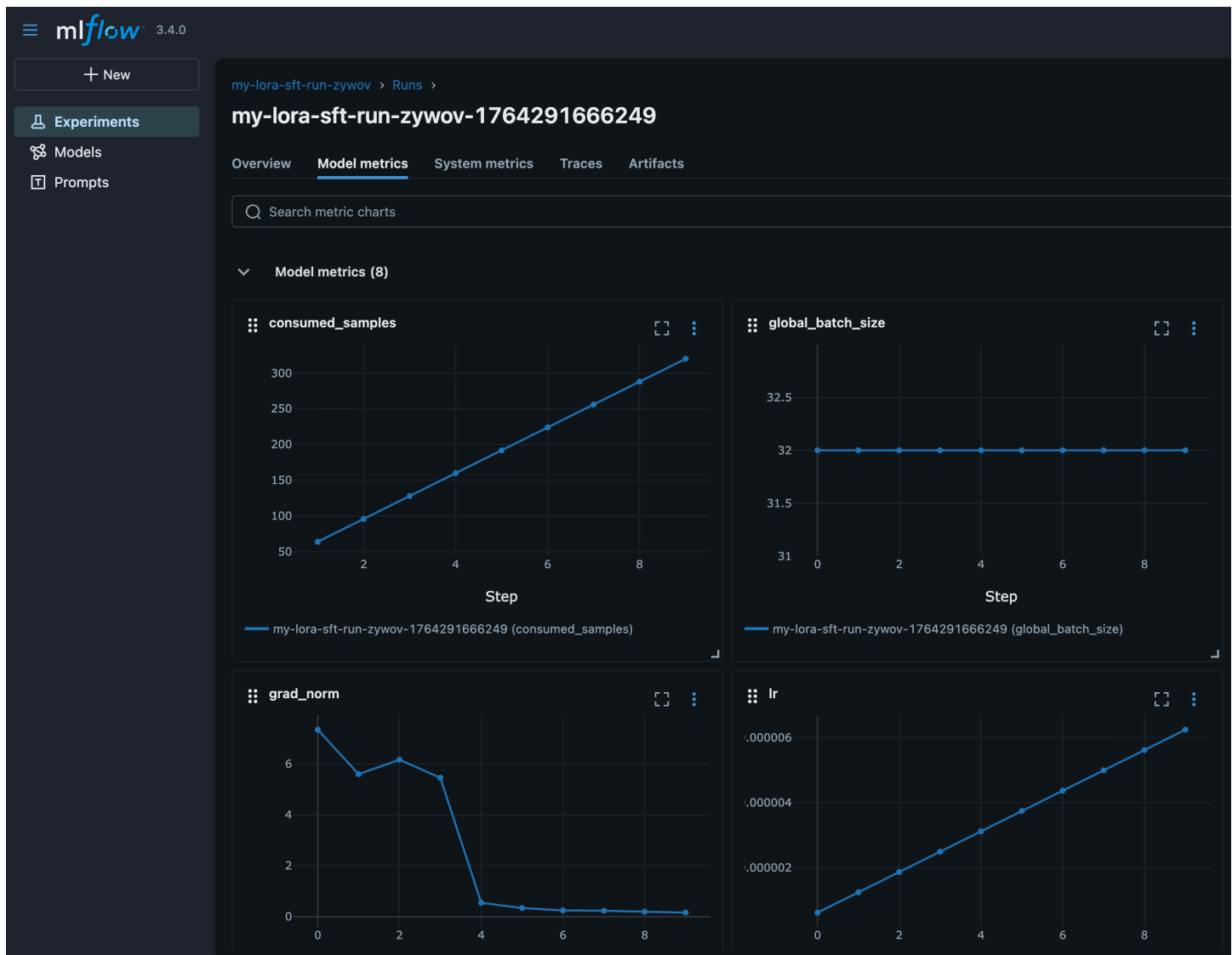


在 SageMaker HyperPod 配方的執行區塊下傳遞至配方

Recipe

```
run  
  mlflow_tracking_uri: arn:aws:sagemaker:us-east-1:111122223333:mlflow-app/app-  
  LGZE0Z2UY4NZ
```

檢視



反覆訓練

概觀

反覆訓練是透過不同訓練方法的多個訓練週期重複微調模型的程序：訓練、評估、分析錯誤、調整 data/objectives/hyperparameters，每個回合都從先前的檢查點開始。此方法可讓您以系統化的方式鎖定模型失敗模式、納入解決特定弱點的精選範例，以及因應隨時間變化的需求。

優於單次通過訓練的優點：

- 目標改善：解決透過評估發現的特定故障模式
- 自適應改進：回應分佈轉移或不斷變化的產品需求

- 風險緩解：逐步驗證改進，而不是承諾進行單一長訓練
- 資料效率：將資料收集工作集中在模型效能不佳的領域
- 課程訓練：多輪訓練搭配越來越高品質的資料

運作方式

檢查點位置和存取

在每個訓練任務完成後，資訊清單檔案會在訓練組態中 `output_path` 參數指定的輸出位置中產生。

存取您的檢查點

- 導覽至 S3 `output_path` 中指定的
- 下載並擷取 `output.tar.gz` 檔案
- 在 中開啟 `manifest.json` 檔案
- 尋找 `checkpoint_s3_bucket` 參數，其中包含訓練模型的 S3 URI

`manifest.json` 結構範例

```
{
  "checkpoint_s3_bucket": "s3://customer-escrow-<account-number>-smtj-<unique-
  identifier>/<job-name>/stepID",
  ...
}
```

了解託管儲存貯體

由於 Amazon Nova 權重是專有的，經過訓練的模型檢查點會存放在受管帳戶中的託管 S3 儲存貯體中，而不是複製到您的帳戶。AWS 這些託管儲存貯體：

- 安全地包含您的自訂模型權重
- 可由其他 AWS 服務（推論、評估和後續訓練任務）參考
- 只能透過 IAM 許可存取 AWS 您的帳戶
- 在您的帳戶中產生標準 S3 儲存費用（請參閱成本考量）

您可以在下一次訓練執行 `model_name_or_path` 中使用託管儲存貯體路徑做為，以繼續反覆訓練。

使用檢查點進行反覆訓練

設定您的下一個訓練任務，以使用先前的檢查點做為基本模型：

```
run:
  name: "my-iterative-training-job"
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-identifier>/<previous-job-name>"
  data_s3_path: s3://<bucket>/<data-file>.jsonl
  replicas: 4
```

何時使用反覆訓練

理想的使用案例

當您有下列情況時，請使用反覆訓練：

- 回饋迴圈 – 能夠收集真實世界的故障案例並系統化地解決它們
- 動態環境 – 不斷發展的文件、APIs或支援需要定期更新模型的主題
- 強大的評估 – 強大的基準和評估架構（請參閱以下範例），以自信地衡量改進
- ML 操作功能 – 管理多個訓練週期和版本控制的資源

強大的評估架構範例

- 具有通過/失敗閾值的自動化基準測試套件
- 具有評分者間可靠性指標的人工評估通訊協定
- 涵蓋邊緣案例和對手輸入的紅隊測試案例
- 測量生產影響的 A/B 測試基礎設施

常見模式

SFT → RFT 管道：經常使用的反覆模式涉及：

- SFT first – 透過示範範例教導模型如何解決問題
- RFT 秒 – 使用獎勵訊號在更廣泛的問題空間中最佳化效能

當模型一開始效能不佳時，此序列至關重要：如果沒有先透過 SFT 建立基本問題解決功能，則近乎零的準確性模型上的 RFT 將無法改善效能。

何時不使用反覆訓練

避免針對下列項目進行反覆訓練：

- 穩定、定義明確的任務 – 具有一致需求的靜態資料已實現接近上限的效能
- 簡單分類問題 – 具有足夠單次訓練的窄小任務
- 資源限制 – 缺乏管理多個訓練週期的專用 ML 操作功能
- 邊際收益 – 當額外負荷無法證明最低效能改善

工作流程範例：SFT → RFT

此範例示範推理模型的常見反覆訓練模式。

步驟 1：初始 SFT 訓練

使用資料集設定和啟動 SFT 訓練任務：

```
run:
  name: "initial-sft-training"
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: "nova-lite-2/prod"
  data_s3_path: s3://<bucket>/sft-training-data.jsonl
  validation_data_s3_path: s3://<bucket>/sft-validation-data.jsonl
```

理由：SFT 提供額外的示範，將模型輸出塑造成您想要的格式和語音，建立基礎功能。

訓練完成後

- 請注意在您的訓練任務中 `output_path` 設定的
- `output.tar.gz` 從該位置下載
- 擷取並尋找 `manifest.json`
- 複製 `checkpoint_s3_bucket` 值

步驟 2：SFT 檢查點上的 RFT 訓練

使用 SFT 檢查點建立新的 RFT 訓練任務：

```
run:
  name: "rft-on-sft-checkpoint"
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-identifier>/<initial-sft-training>"
  data_s3_path: s3://<bucket>/rft-training-data.jsonl
  reward_lambda_arn: <your-reward-function-arn>
```

理由：RFT 訓練建置在 SFT 基礎上，允許模型開發更複雜的推理模式，以您的獎勵函數最佳化。

步驟 3：評估和反覆運算

在 RFT 檢查點上執行評估，以評估效能：

```
run:
  name: "evaluate-rft-checkpoint"
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-identifier>/<rft-on-sft-checkpoint>"
  data_s3_path: s3://<bucket>/evaluation-data.jsonl
```

如果未滿足目標指標，請使用調整後的資料或超參數繼續反覆運算。

Important

訓練技術 (LoRA 與完整排名) 必須在所有反覆運算中保持一致：

- 如果您搭配 LoRA 使用 SFT，則必須搭配 LoRA 使用 RFT
- 如果您使用具有完整排名的 SFT，則必須使用具有完整排名的 RFT
- 您無法在 LoRA 和 Full Rank 中線之間切換

Important

如果 KMS 金鑰用於 Amazon 擁有的輸出 S3 儲存貯體中的加密，則該相同的 KMS 金鑰必須用於所有未來的反覆運算。

監控反覆運算的進度

您可以設定任務的 MLflow，[透過 MLflow](#) 追蹤指標。

建立 MLflow 應用程式

使用 Studio UI：如果您透過 Studio UI 建立訓練任務，預設 MLflow 應用程式會自動建立，並在進階選項下依預設選取。

使用 CLI：如果您使用 CLI，則必須建立 MLflow 應用程式並將其做為訓練任務 API 請求的輸入傳遞。

```
mlflow_app_name="<enter your MLflow app name>"
role_arn="<enter your role ARN>"
bucket_name="<enter your bucket name>"
region="<enter your region>"

mlflow_app_arn=$(aws sagemaker create-mlflow-app \
  --name $mlflow_app_name \
  --artifact-store-uri "s3://$bucket_name" \
  --role-arn $role_arn \
  --region $region)
```

存取 MLflow 應用程式

使用 CLI：建立預先簽章的 URL 以存取 MLflow 應用程式 UI：

```
aws sagemaker create-presigned-mlflow-app-url \
  --arn $mlflow_app_arn \
  --region $region \
  --output text
```

使用 Studio UI：Studio UI 會顯示存放在 MLflow 中的關鍵指標，並提供 MLflow 應用程式 UI 的連結。

要追蹤的關鍵指標

跨反覆運算監控這些指標，以評估改善並追蹤任務進度：

對於 SFT

- 訓練損失曲線
- 使用的範例數量和處理範例的時間

- 保留測試集的效能準確性
- 格式合規（例如，有效的 JSON 輸出速率）
- 網域特定評估資料的複雜度

對於 RFT

- 訓練的平均獎勵分數
- 獎勵分佈（高獎勵回應的百分比）
- 驗證獎勵趨勢（注意過度擬合）
- 任務特定的成功率（例如，程式碼執行通過率、數學問題準確性）

一般

- 反覆運算之間的基準效能差異
- 代表性範例的人工評估分數
- 生產指標（如果反覆部署）

判斷何時停止

在下列情況下停止反覆運算：

- 效能穩定 – 其他訓練不再有意義地改善目標指標
- 技術切換有幫助 – 如果一個技術穩定，請嘗試切換（例如，SFT → RFT → SFT）以突破效能上限
- 實現的目標指標 – 符合您的成功條件
- 偵測到迴歸 – 新的反覆運算會降低效能（請參閱下列轉返程序）

如需詳細評估程序，請參閱評估一節。

最佳實務

從小開始並逐漸擴展

從最少的資料集和單一訓練 epoch 開始，以在擴展之前驗證您的方法。這可建立信心，並有助於及早識別問題。

建立明確的成功指標

在開始之前定義量化和定性指標：

使用案例的成功指標範例

- 回答問題 – 完全符合準確性、F1 分數、人工偏好評分
- 程式碼產生 – 單位測試通過率、編譯成功、執行時間
- 原因任務 – 步驟準確性、最終答案正確性、獎勵分數
- 內容產生 – 一致性分數、事實準確性、風格遵循

實作自動化評估

設定自動化評估管道來追蹤每個回合後的效能，以便快速迭代和目標比較。

維持嚴格的版本控制

每次反覆運算的文件：

- 資料集版本和修改
- 模型檢查點位置
- 超參數變更
- 效能指標和差異
- 定性觀察

這會建立機構知識並啟用偵錯。

專注於資料品質而非數量

分析先前回合中的失敗案例，並新增有針對性且高品質的範例，而不只是增加資料集大小。

計劃反覆運算預算

將 3-5 次反覆運算規劃為典型範圍：

- 1-2 次反覆運算 - 通常足以進行簡單的改善或最終修飾

- 3-5 次反覆運算 – 適用於需要多個精簡週期的複雜任務
- 5 次以上的反覆運算 – 可能表示減少傳回或需要不同的方法

根據運算預算和效能改善率進行調整。

實作復原功能

如果反覆運算引入迴歸：

- 識別迴歸 – 比較跨檢查點的評估指標
- 返回先前的檢查點 – 使用較早檢查點的 S3 路徑做為您的 `model_name_or_path`
- 調整訓練方法 – 在重試之前修改資料、超參數或技術
- 記錄失敗 – 記錄導致迴歸避免重複的原因

復原範例

```
run:
  name: "rollback-to-iteration-2"
  model_type: amazon.nova-2-lite-v1:0:256k
  # Use iteration 2 checkpoint instead of failed iteration 3
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-identifier>/
<iteration-2-job-name>"
```

成本考量

檢查點儲存

- 位置 – 存放在託管儲存貯體中的檢查點會產生標準 S3 儲存費用，並向 AWS 您的帳戶收費
- 保留 – 除非明確刪除，否則檢查點會無限期保留
- 管理 – 實作生命週期政策來封存或刪除不再需要的舊檢查點

成本最佳化秘訣

- 驗證較新的反覆運算後刪除中繼檢查點
- 將檢查點封存至 S3 Glacier，以較低成本長期保留
- 根據您的合規和實驗需求設定保留政策

限制

模型系列一致性

反覆訓練時，您必須在所有反覆運算中使用相同的模型類型。

初始訓練

```
run:
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: "nova-lite-2/prod"
```

後續反覆運算必須使用相同的 `model_type`

```
run:
  model_type: amazon.nova-2-lite-v1:0:256k # Must match original
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-identifier>/<job-name>"
```

訓練技術一致性

訓練技術必須在反覆運算之間保持一致：

- LoRA 訓練的模型只能使用 LoRA 反覆訓練
- Full-Rank-trained模型只能使用完整 Rank 反覆訓練

LoRA 轉接器如何在反覆訓練中運作

- 每個 LoRA 訓練反覆運算都會產生新的轉接器權重
- 新的轉接器取代（而非使用 堆疊）先前的轉接器
- 基本模型會保持凍結狀態；只會更新轉接器

技術相容性矩陣

初始訓練	可以使用 反覆運算
SFT (Full-Rank)	SFT (Full-Rank)、RFT (Full-Rank)

初始訓練	可以使用 反覆運算
SFT (LoRA)	SFT (LoRA)、RFT (LoRA)
RFT (Full-Rank)	RFT (Full-Rank)
RFT (LoRA)	RFT (LoRA)

在開始任務之前驗證相容性

- 檢查您的先前訓練配方，以識別模型類型和訓練技術 (LoRA 與 Full-Rank)
- 確保您的新配方同時符合模型類型和技術
- 檢閱 manifest.json 以確認檢查點路徑正確

疑難排解

錯誤：「偵測到不相容的模型訓練技術」

原因：訓練技術 (LoRA 與 Full-Rank) 不符合檢查點的技術。

解決方案：確保您的配方使用與原始模型相同的訓練技術：

- 如果檢查點已使用 LoRA 訓練，請在新配方中使用 LoRA
- 如果檢查點已使用 Full-Rank 訓練，請在新配方中使用 Full-Rank

錯誤：「從 model_name_or_path 擷取之任務的基礎模型不符合 model_type」

原因：中指定的模型類型與檢查點中的實際模型model_type不相符。

解決方案：驗證：

- 配方model_type中的 符合原始模型類型
- 中的檢查點 S3 路徑model_name_or_path正確
- 您正在使用正確 manifest.json 檔案的路徑

正確組態的範例

```
run:
  model_type: amazon.nova-2-lite-v1:0:256k # Must match checkpoint's model
  model_name_or_path: "s3://customer-escrow-<account-number>-smtj-<unique-identifier>/<job-name>"
```

錯誤：「找不到模型組態」

原因：中的 S3 路徑 `model_name_or_path` 無效或無法存取。

解決方法：

- 驗證 S3 路徑是否已從 `manifest.json` 檔案正確複製
- 確保您的 IAM 角色具有存取託管儲存貯體的許可
- 確認先前訓練任務已成功完成
- 檢查路徑中是否有錯字

反覆運算後的效能迴歸

症狀：評估指標會在新的訓練反覆運算後下降。

解決方法：

- 轉返 – 使用先前的檢查點做為基礎模型
- 分析 – 檢閱失敗反覆運算的訓練日誌和資料品質
- 調整 – 修改超參數（降低學習率）、改善資料品質，或降低訓練 epoch
- 重試 – 透過調整啟動新的反覆運算

SageMaker 推論

自訂 Amazon Nova 模型現在可在 SageMaker 推論上使用。透過 SageMaker 上的 Amazon Nova，您可以從訓練過的自訂 Amazon Nova 模型開始取得預測或推論。SageMaker 提供廣泛的機器學習 (ML) 基礎架構和模型部署選項，有助於滿足您所有的機器學習推論需求。透過 SageMaker 推論，您可以擴展模型部署、更有效地在生產環境中管理模型，並減少營運負擔。

SageMaker 為您提供各種推論選項，例如取得低延遲推論的即時端點，以及批次請求的非同步端點。透過為您的使用案例利用適當的推論選項，您可以確保高效的模型部署和推論。如需 SageMaker 推論的詳細資訊，請參閱[部署模型以進行推論](#)。

Important

SageMaker 推論僅支援全階自訂模型和 LoRA 合併模型。對於未合併的 LoRA 模型和基礎模型，請使用 Amazon Bedrock。

功能

下列功能適用於 SageMaker 推論上的 Amazon Nova 模型：

模型功能

- 產生文字

部署和擴展

- 具有自訂執行個體選擇的即時端點
- Auto Scaling – 根據流量模式自動調整容量，以最佳化成本和 GPU 使用率。如需詳細資訊，請參閱[自動擴展 Amazon SageMaker 模型](#)。
- 串流 API 支援即時產生字符

監控和最佳化

- 用於監控和警示的 Amazon CloudWatch 整合
- 透過 VPC 組態的可用區域感知延遲最佳化

開發工具

- AWS CLI 支援 – 如需詳細資訊，請參閱[AWS SageMaker 的 CLI 命令參考](#)。
- 透過 SDK 支援進行筆記本整合

支援的模型和執行個體

建立 SageMaker 推論端點時，您可以設定兩個環境變數來設定部署：CONTEXT_LENGTH和MAX_CONCURRENCY。

- CONTEXT_LENGTH – 每個請求的總字符長度上限（輸入 + 輸出）

- MAX_CONCURRENCY – 端點將服務的並行請求數目上限

下表列出支援的 Amazon Nova 模型、執行個體類型和支援的組態。MAX_CONCURRENCY 值代表每個 CONTEXT_LENGTH 設定支援的並行上限：

模型	執行個體類型	支援的組態
Amazon Nova Micro	ml.g5.12xlarge	CONTEXT_LENGTH : 400 0 , MAX_CONCURRENCY : 32 CONTEXT_LENGTH : 800 0 , MAX_CONCURRENCY : 16
	ml.g5.24xlarge	CONTEXT_LENGTH : 800 0 , MAX_CONCURRENCY : 32
	ml.g6.12xlarge	CONTEXT_LENGTH : 400 0 , MAX_CONCURRENCY : 32 CONTEXT_LENGTH : 800 0 , MAX_CONCURRENCY : 16
	ml.g6.24xlarge	CONTEXT_LENGTH : 800 0 , MAX_CONCURRENCY : 32
	ml.g6.48xlarge	CONTEXT_LENGTH : 800 0 , MAX_CONCURRENCY : 32
	ml.p5.48xlarge	CONTEXT_LENGTH : 800 0 , MAX_CONCURRENCY : 32 CONTEXT_LENGTH : 16000 , MAX_CONCURRENCY : 2 CONTEXT_LENGTH : 24000 , MAX_CONCURRENCY : 1
Amazon Nova Lite	ml.g6.48xlarge	CONTEXT_LENGTH : 400 0 , MAX_CONCURRENCY : 32

模型	執行個體類型	支援的組態
	ml.p5.48xlarge	CONTEXT_LENGTH : 8000 , MAX_CONCURRENCY : 16
		CONTEXT_LENGTH : 8000 , MAX_CONCURRENCY : 32
		CONTEXT_LENGTH : 16000 , MAX_CONCURRENCY : 2
		CONTEXT_LENGTH : 24000 , MAX_CONCURRENCY : 1
Nova 2 Lite	ml.p5.48xlarge	CONTEXT_LENGTH : 8000 , MAX_CONCURRENCY : 32
		CONTEXT_LENGTH : 16000 , MAX_CONCURRENCY : 2
		CONTEXT_LENGTH : 24000 , MAX_CONCURRENCY : 1

Note

顯示的 MAX_CONCURRENCY 值是每個 CONTEXT_LENGTH 設定的上限。您可以使用相同並行的較低內容長度，但超過這些值會導致 SageMaker 端點建立失敗。

例如，在具有 ml.g5.12xlarge 的 Amazon Nova Micro 上：

- CONTEXT_LENGTH=2000、→ MAX_CONCURRENCY=32 有效
- CONTEXT_LENGTH=8000、MAX_CONCURRENCY=32→ 已拒絕（內容長度為 8000 的並行限制為 16）
- CONTEXT_LENGTH=8000、→ MAX_CONCURRENCY=4 有效
- CONTEXT_LENGTH=8000、→ MAX_CONCURRENCY=16 有效
- CONTEXT_LENGTH=10000 → 拒絕（此執行個體的最大內容長度為 8000）

支援 AWS 的區域

下表列出可在 SageMaker 推論上使用 Amazon Nova 模型 AWS 的區域：

區域名稱	區域代碼	可用性
美國東部 (維吉尼亞北部)	us-east-1	Available
美國西部 (奧勒岡)	us-west-2	Available

支援的容器映像

下表依區域列出 SageMaker 推論上 Amazon Nova 模型的容器映像 URIs。每個區域可使用兩個映像標籤：版本化標籤 (v1.0.0) 和最新標籤 (SM-Inference-latest)。對於生產部署，我們建議您使用版本控制的標籤。

區域	容器映像 URIs
us-east-1	708977205387.dkr.ecr.us-east-1.amazonaws.com/ nova-inference-repo:SM-Inference-latest
us-west-2	176779409107.dkr.ecr.us-west-2.amazonaws.com/ nova-inference-repo:SM-Inference-latest

最佳實務

如需在 SageMaker 上部署和管理模型的最佳實務，請參閱 [SageMaker 的最佳實務](#)。

支援

如需 SageMaker 推論上 Amazon Nova 模型的問題和支援，請透過主控台或您的 AWS 客戶經理聯絡 AWS Support。

主題

- [開始使用](#)
- [API 參考](#)

- [評估託管在 SageMaker 推論上的模型](#)
- [在 Amazon SageMaker 推論濫用偵測中部署 Amazon Nova Forge 模型](#)

開始使用

本指南說明如何在 SageMaker 即時端點上部署自訂的 Amazon Nova 模型、設定推論參數，以及叫用模型進行測試。

先決條件

以下是在 SageMaker 推論上部署 Amazon Nova 模型的先決條件：

- 建立 AWS 帳戶 - 如果您還沒有，請參閱[建立 AWS 帳戶](#)。
- 必要的 IAM 許可 - 確保您的 IAM 使用者或角色已連接下列受管政策：
 - AmazonSageMakerFullAccess
 - AmazonS3FullAccess
- 必要的 SDKs/CLI 版本 - 下列 SDK 版本已在 SageMaker 推論上使用 Amazon Nova 模型進行測試和驗證：
 - 適用於資源型 API 方法的 SageMaker Python SDK v3.0.0+ (sagemaker>=3.0.0)
 - Boto3 1.35.0+ 版 (boto3>=1.35.0) 用於直接 API 呼叫。本指南中的範例使用此方法。
- 增加服務配額 - 針對您計劃用於 Amazon SageMaker SageMaker 服務配額（例如 ml.p5.48xlarge for endpoint usage）。如需支援的執行個體類型清單，請參閱 [the section called “支援的模型和執行個體”](#)。若要請求提高配額，請參閱[請求提高配額](#)。如需 SageMaker 執行個體配額的相關資訊，請參閱 [SageMaker 端點和配額](#)。

步驟 1：設定 AWS 登入資料

使用下列其中一種方法設定您的 AWS 登入資料：

選項 1：AWS CLI（建議）

```
aws configure
```

出現提示時，輸入您的 AWS 存取金鑰、私密金鑰和預設區域。

選項 2：AWS credentials 檔案

建立或編輯 `~/.aws/credentials`：

```
[default]
aws_access_key_id = YOUR_ACCESS_KEY
aws_secret_access_key = YOUR_SECRET_KEY
```

選項 3：環境變數

```
export AWS_ACCESS_KEY_ID=your_access_key
export AWS_SECRET_ACCESS_KEY=your_secret_key
```

 Note

如需 AWS 登入資料的詳細資訊，請參閱[組態和登入資料檔案設定](#)。

初始化 AWS 用戶端

使用下列程式碼建立 Python 指令碼或筆記本，以初始化 AWS SDK 並驗證您的登入資料：

```
import boto3

# AWS Configuration - Update these for your environment
REGION = "us-east-1" # Supported regions: us-east-1, us-west-2
AWS_ACCOUNT_ID = "YOUR_ACCOUNT_ID" # Replace with your AWS account ID

# Initialize AWS clients using default credential chain
sagemaker = boto3.client('sagemaker', region_name=REGION)
sts = boto3.client('sts')

# Verify credentials
try:
    identity = sts.get_caller_identity()
    print(f"Successfully authenticated to AWS Account: {identity['Account']}")

    if identity['Account'] != AWS_ACCOUNT_ID:
        print(f"Warning: Connected to account {identity['Account']}, expected {AWS_ACCOUNT_ID}")

except Exception as e:
    print(f"Failed to authenticate: {e}")
```

```
print("Please verify your credentials are configured correctly.")
```

如果身分驗證成功，您應該會看到確認 AWS 帳戶 ID 的輸出。

步驟 2：建立 SageMaker 執行角色

SageMaker 執行角色是一種 IAM 角色，可授予 SageMaker 代表您存取 AWS 資源的許可，例如用於模型成品的 Amazon S3 儲存貯體和用於記錄的 CloudWatch。

建立執行角色

Note

建立 IAM 角色需要 `iam:CreateRole` 和 `iam:AttachRolePolicy` 許可。在繼續之前，請確定您的 IAM 使用者或角色具有這些許可。

下列程式碼會建立具有部署 Amazon Nova 自訂模型必要許可的 IAM 角色：

```
import json

# Create SageMaker Execution Role
role_name = f"SageMakerInference-ExecutionRole-{AWS_ACCOUNT_ID}"

trust_policy = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Service": "sagemaker.amazonaws.com"},
            "Action": "sts:AssumeRole"
        }
    ]
}

iam = boto3.client('iam', region_name=REGION)

# Create the role
role_response = iam.create_role(
    RoleName=role_name,
    AssumeRolePolicyDocument=json.dumps(trust_policy),
```

```

    Description='SageMaker execution role with S3 and SageMaker access'
)

# Attach required policies
iam.attach_role_policy(
    RoleName=role_name,
    PolicyArn='arn:aws:iam::aws:policy/AmazonSageMakerFullAccess'
)

iam.attach_role_policy(
    RoleName=role_name,
    PolicyArn='arn:aws:iam::aws:policy/AmazonS3FullAccess'
)

SAGEMAKER_EXECUTION_ROLE_ARN = role_response['Role']['Arn']
print(f"Created SageMaker execution role: {SAGEMAKER_EXECUTION_ROLE_ARN}")

```

使用現有的執行角色（選用）

如果您已有 SageMaker 執行角色，您可以改用它：

```

# Replace with your existing role ARN
SAGEMAKER_EXECUTION_ROLE_ARN = "arn:aws:iam::YOUR_ACCOUNT_ID:role/
YOUR_EXISTING_ROLE_NAME"

```

若要尋找帳戶中現有的 SageMaker 角色：

```

iam = boto3.client('iam', region_name=REGION)
response = iam.list_roles()
sagemaker_roles = [role for role in response['Roles'] if 'SageMaker' in
    role['RoleName']]
for role in sagemaker_roles:
    print(f"{role['RoleName']}: {role['Arn']}")

```

Important

執行角色必須與 `sagemaker.amazonaws.com` 和 具有信任關係，才能存取 Amazon S3 和 SageMaker 資源。

如需 SageMaker 執行角色的詳細資訊，請參閱 [SageMaker 角色](#)。

步驟 3：設定模型參數

設定 Amazon Nova 模型的部署參數。這些設定控制模型行為、資源配置和推論特性。如需每個支援的執行個體類型和支援的 `CONTEXT_LENGTH` 和 `MAX_CONCURRENCY` 值清單，請參閱 [the section called “支援的模型和執行個體”](#)。

必要參數

- `IMAGE`：Amazon Nova 推論容器的 Docker 容器映像 URI。這將由 提供 AWS。
- `CONTEXT_LENGTH`：模型內容長度。
- `MAX_CONCURRENCY`：每次反覆運算的序列數目上限；設定在 GPU 上的單一批次內可同時處理個別使用者請求（提示）的數量限制。範圍：大於 0 的整數。

選用的產生參數

- `DEFAULT_TEMPERATURE`：控制產生時的隨機性。範圍：0.0 到 2.0 (0.0 = 確定性，較高 = 更多隨機)。
- `DEFAULT_TOP_P`：Hucleus 取樣閾值用於權杖選取。範圍：1e-10 到 1.0。
- `DEFAULT_TOP_K`：將權杖選擇限制為最熱門的 K 權杖。範圍：整數 -1 或更高 (-1 = 無限制)。
- `DEFAULT_MAX_NEW_TOKENS`：回應中要產生的字符數量上限（即最大輸出字符）。範圍：整數 1 或更高。
- `DEFAULT_LOGPROBS`：每個字符傳回的日誌機率。範圍：整數 1 到 20。

設定您的部署

```
# AWS Configuration
REGION = "us-east-1" # Must match region from Step 1

# ECR Account mapping by region
ECR_ACCOUNT_MAP = {
    "us-east-1": "708977205387",
    "us-west-2": "176779409107"
}

# Container Image
IMAGE = f"{ECR_ACCOUNT_MAP[REGION]}.dkr.ecr.{REGION}.amazonaws.com/nova-inference-repo:SM-Inference-latest"
print(f"IMAGE = {IMAGE}")
```

```

# Model Parameters
CONTEXT_LENGTH = "16000"          # Maximum total context length
MAX_CONCURRENCY = "2"             # Maximum concurrent sequences

# Optional: Default generation parameters (uncomment to use)
DEFAULT_TEMPERATURE = "0.0"       # Deterministic output
DEFAULT_TOP_P = "1.0"             # Consider all tokens
# DEFAULT_TOP_K = "50"            # Uncomment to limit to top 50 tokens
# DEFAULT_MAX_NEW_TOKENS = "2048" # Uncomment to set max output tokens
# DEFAULT_LOGPROBS = "1"          # Uncomment to enable log probabilities

# Build environment variables for the container
environment = {
    'CONTEXT_LENGTH': CONTEXT_LENGTH,
    'MAX_CONCURRENCY': MAX_CONCURRENCY,
}

# Add optional parameters if defined
if 'DEFAULT_TEMPERATURE' in globals():
    environment['DEFAULT_TEMPERATURE'] = DEFAULT_TEMPERATURE
if 'DEFAULT_TOP_P' in globals():
    environment['DEFAULT_TOP_P'] = DEFAULT_TOP_P
if 'DEFAULT_TOP_K' in globals():
    environment['DEFAULT_TOP_K'] = DEFAULT_TOP_K
if 'DEFAULT_MAX_NEW_TOKENS' in globals():
    environment['DEFAULT_MAX_NEW_TOKENS'] = DEFAULT_MAX_NEW_TOKENS
if 'DEFAULT_LOGPROBS' in globals():
    environment['DEFAULT_LOGPROBS'] = DEFAULT_LOGPROBS

print("Environment configuration:")
for key, value in environment.items():
    print(f" {key}: {value}")

```

設定部署特定的參數

現在為您的 Amazon Nova 模型部署設定特定參數，包括模型成品位置和執行個體類型選擇。

設定部署識別符

```

# Deployment identifier - use a descriptive name for your use case
JOB_NAME = "my-nova-deployment"

```


指定模型成品位置

提供存放訓練過之 Amazon Nova 模型成品的 Amazon S3 URI。這應該是模型訓練或微調任務的輸出位置。

```
# S3 location of your trained Nova model artifacts
# Replace with your model's S3 URI - must end with /
MODEL_S3_LOCATION = "s3://your-bucket-name/path/to/model/artifacts/"
```

選取模型變體和執行個體類型

```
# Configure model variant and instance type
TESTCASE = {
    "model": "lite2",                # Options: micro, lite, lite2
    "instance": "ml.p5.48xlarge"     # Refer to "Supported models and instances" section
}

# Generate resource names
INSTANCE_TYPE = TESTCASE["instance"]
MODEL_NAME = JOB_NAME + "-" + TESTCASE["model"] + "-" + INSTANCE_TYPE.replace(".", "-")
ENDPOINT_CONFIG_NAME = MODEL_NAME + "-Config"
ENDPOINT_NAME = MODEL_NAME + "-Endpoint"

print(f"Model Name: {MODEL_NAME}")
print(f"Endpoint Config: {ENDPOINT_CONFIG_NAME}")
print(f"Endpoint Name: {ENDPOINT_NAME}")
```

命名慣例

程式碼會自動為 AWS 資源產生一致的名稱：

- 模型名稱：{JOB_NAME}-{model}-{instance-type}
- 端點組態：{MODEL_NAME}-Config
- 端點名稱：{MODEL_NAME}-Endpoint

步驟 4：建立 SageMaker 模型和端點組態

在此步驟中，您將建立兩個基本資源：參考 Amazon Nova 模型成品的 SageMaker 模型物件，以及定義模型部署方式的端點組態。

SageMaker 模型：封裝推論容器映像、模型成品位置和環境組態的模型物件。這是可重複使用的資源，可部署到多個端點。

端點組態：定義部署的基礎設施設定，包括執行個體類型、執行個體計數和模型變體。這可讓您與模型本身分開管理部署設定。

建立 SageMaker 模型

下列程式碼會建立參考 Amazon Nova 模型成品的 SageMaker 模型：

```
try:
    model_response = sagemaker.create_model(
        ModelName=MODEL_NAME,
        PrimaryContainer={
            'Image': IMAGE,
            'ModelDataSource': {
                'S3DataSource': {
                    'S3Uri': MODEL_S3_LOCATION,
                    'S3DataType': 'S3Prefix',
                    'CompressionType': 'None'
                }
            },
            'Environment': environment
        },
        ExecutionRoleArn=SAGEMAKER_EXECUTION_ROLE_ARN,
        EnableNetworkIsolation=True
    )
    print("Model created successfully!")
    print(f"Model ARN: {model_response['ModelArn']}")

except sagemaker.exceptions.ClientError as e:
    print(f"Error creating model: {e}")
```

重要參數：

- **ModelName：**模型的唯一識別符
- **Image：**Amazon Nova 推論的 Docker 容器映像 URI
- **ModelDataSource：**模型成品的 Amazon S3 位置
- **Environment：**在步驟 3 中設定的環境變數
- **ExecutionRoleArn：**步驟 2 中的 IAM 角色
- **EnableNetworkIsolation：**將設定為 True 以增強安全性（防止容器進行傳出網路呼叫）

建立端點組態

接著，建立定義部署基礎設施的端點組態：

```
# Create Endpoint Configuration
try:
    production_variant = {
        'VariantName': 'primary',
        'ModelName': MODEL_NAME,
        'InitialInstanceCount': 1,
        'InstanceType': INSTANCE_TYPE,
    }

    config_response = sagemaker.create_endpoint_config(
        EndpointConfigName=ENDPOINT_CONFIG_NAME,
        ProductionVariants=[production_variant]
    )
    print("Endpoint configuration created successfully!")
    print(f"Config ARN: {config_response['EndpointConfigArn']}")

except sagemaker.exceptions.ClientError as e:
    print(f"Error creating endpoint configuration: {e}")
```

重要參數：

- VariantName：此模型變體的識別符（單一模型部署使用「主要」）
- ModelName：參考上面建立的模型
- InitialInstanceCount：要部署的執行個體數目（從 1 開始，視需要稍後擴展）
- InstanceType：在步驟 3 中選取 ML 執行個體類型

驗證資源建立

您可以驗證您的資源是否已成功建立：

```
# Describe the model
model_info = sagemaker.describe_model(ModelName=MODEL_NAME)
print(f"Model Status: {model_info['ModelName']} created")

# Describe the endpoint configuration
config_info =
    sagemaker.describe_endpoint_config(EndpointConfigName=ENDPOINT_CONFIG_NAME)
```

```
print(f"Endpoint Config Status: {config_info['EndpointConfigName']} created")
```

步驟 5：部署端點

下一步是透過建立 SageMaker 即時端點來部署 Amazon Nova 模型。此端點會託管您的模型，並提供安全的 HTTPS 端點來提出推論請求。

端點建立通常需要 15-30 分鐘，因為會 AWS 佈建基礎設施、下載模型成品，以及初始化推論容器。

建立端點

```
import time

try:
    endpoint_response = sagemaker.create_endpoint(
        EndpointName=ENDPOINT_NAME,
        EndpointConfigName=ENDPOINT_CONFIG_NAME
    )
    print("Endpoint creation initiated successfully!")
    print(f"Endpoint ARN: {endpoint_response['EndpointArn']}")
except Exception as e:
    print(f"Error creating endpoint: {e}")
```

監控端點建立

下列程式碼會輪詢端點狀態，直到部署完成：

```
# Monitor endpoint creation progress
print("Waiting for endpoint creation to complete...")
print("This typically takes 15-30 minutes...\n")

while True:
    try:
        response = sagemaker.describe_endpoint(EndpointName=ENDPOINT_NAME)
        status = response['EndpointStatus']

        if status == 'Creating':
            print(f"# Status: {status} - Provisioning infrastructure and loading model...")
        elif status == 'InService':
            print(f"# Status: {status}")
            print("\nEndpoint creation completed successfully!")
```

```

        print(f"Endpoint Name: {ENDPOINT_NAME}")
        print(f"Endpoint ARN: {response['EndpointArn']}")
        break
    elif status == 'Failed':
        print(f"# Status: {status}")
        print(f"Failure Reason: {response.get('FailureReason', 'Unknown')}")
        print("\nFull response:")
        print(response)
        break
    else:
        print(f"Status: {status}")

except Exception as e:
    print(f"Error checking endpoint status: {e}")
    break

time.sleep(30) # Check every 30 seconds

```

確認端點已就緒

一旦端點成為 InService，您就可以驗證其組態：

```

# Get detailed endpoint information
endpoint_info = sagemaker.describe_endpoint(EndpointName=ENDPOINT_NAME)

print("\n=== Endpoint Details ===")
print(f"Endpoint Name: {endpoint_info['EndpointName']}")
print(f"Endpoint ARN: {endpoint_info['EndpointArn']}")
print(f"Status: {endpoint_info['EndpointStatus']}")
print(f"Creation Time: {endpoint_info['CreationTime']}")
print(f>Last Modified: {endpoint_info['LastModifiedTime']}")

# Get endpoint config for instance type details
endpoint_config_name = endpoint_info['EndpointConfigName']
endpoint_config =
    sagemaker.describe_endpoint_config(EndpointConfigName=endpoint_config_name)

# Display production variant details
for variant in endpoint_info['ProductionVariants']:
    print(f"\nProduction Variant: {variant['VariantName']}")
    print(f"  Current Instance Count: {variant['CurrentInstanceCount']}")
    print(f"  Desired Instance Count: {variant['DesiredInstanceCount']}")
    # Get instance type from endpoint config
    for config_variant in endpoint_config['ProductionVariants']:

```

```
if config_variant['VariantName'] == variant['VariantName']:
    print(f" Instance Type: {config_variant['InstanceType']}")
    break
```

對端點建立失敗進行故障診斷

常見的失敗原因：

- 容量不足：請求的執行個體類型在您的區域無法使用
 - 解決方案：嘗試不同的執行個體類型或請求提高配額
- IAM 許可：執行角色缺少必要的許可
 - 解決方案：確認角色可存取 Amazon S3 模型成品和必要的 SageMaker 許可
- 找不到模型成品：Amazon S3 URI 不正確或無法存取
 - 解決方案：驗證 Amazon S3 URI 並檢查儲存貯體許可，確保您位於正確的區域
- 資源限制：端點或執行個體超過帳戶限制
 - 解決方案：透過 Service Quotas 或 Support 請求提高服務配額 AWS

Note

如果您需要刪除失敗的端點並重新開始：

```
sagemaker.delete_endpoint(EndpointName=ENDPOINT_NAME)
```

步驟 6：叫用端點

一旦您的端點為 InService，您就可以傳送推論請求，以從 Amazon Nova 模型產生預測。SageMaker 支援同步端點（即時串流/非串流模式）和非同步端點（以 Amazon S3 為基礎進行批次處理）。

設定執行時間用戶端

使用適當的逾時設定建立 SageMaker 執行期用戶端：

```
import json
import boto3
import botocore
from botocore.exceptions import ClientError
```

```
# Configure client with appropriate timeouts
config = botocore.config.Config(
    read_timeout=120,      # Maximum time to wait for response
    connect_timeout=10,    # Maximum time to establish connection
    retries={'max_attempts': 3} # Number of retry attempts
)

# Create SageMaker Runtime client
runtime_client = boto3.client('sagemaker-runtime', config=config, region_name=REGION)
```

建立通用推論函數

下列 函數會同時處理串流和非串流請求：

```
def invoke_nova_endpoint(request_body):
    """
    Invoke Nova endpoint with automatic streaming detection.

    Args:
        request_body (dict): Request payload containing prompt and parameters

    Returns:
        dict: Response from the model (for non-streaming requests)
        None: For streaming requests (prints output directly)
    """
    body = json.dumps(request_body)
    is_streaming = request_body.get("stream", False)

    try:
        print(f"Invoking endpoint ({'streaming' if is_streaming else 'non-streaming'})...")

        if is_streaming:
            response = runtime_client.invoke_endpoint_with_response_stream(
                EndpointName=ENDPOINT_NAME,
                ContentType='application/json',
                Body=body
            )

            event_stream = response['Body']
            for event in event_stream:
                if 'PayloadPart' in event:
                    chunk = event['PayloadPart']
                    if 'Bytes' in chunk:
```

```

        data = chunk['Bytes'].decode()
        print("Chunk:", data)

    else:
        # Non-streaming inference
        response = runtime_client.invoke_endpoint(
            EndpointName=ENDPOINT_NAME,
            ContentType='application/json',
            Accept='application/json',
            Body=body
        )

        response_body = response['Body'].read().decode('utf-8')
        result = json.loads(response_body)
        print("# Response received successfully")
        return result

except ClientError as e:
    error_code = e.response['Error']['Code']
    error_message = e.response['Error']['Message']
    print(f"# AWS Error: {error_code} - {error_message}")
except Exception as e:
    print(f"# Unexpected error: {str(e)}")

```

範例 1：非串流聊天完成

使用聊天格式進行對話互動：

```

# Non-streaming chat request
chat_request = {
    "messages": [
        {"role": "user", "content": "Hello! How are you?"}
    ],
    "max_tokens": 100,
    "max_completion_tokens": 100, # Alternative to max_tokens
    "stream": False,
    "temperature": 0.7,
    "top_p": 0.9,
    "top_k": 50,
    "logprobs": True,
    "top_logprobs": 3,
    "reasoning_effort": "low", # Options: "low", "high"
    "allowed_token_ids": None, # List of allowed token IDs
    "truncate_prompt_tokens": None, # Truncate prompt to this many tokens
    "stream_options": None
}

```



```
}  
  
response = invoke_nova_endpoint(chat_request)
```

回應範例：

```
{  
  "id": "chatcmpl-123456",  
  "object": "chat.completion",  
  "created": 1234567890,  
  "model": "default",  
  "choices": [  
    {  
      "index": 0,  
      "message": {  
        "role": "assistant",  
        "content": "Hello! I'm doing well, thank you for asking. I'm here and  
ready to help you with any questions or tasks you might have. How can I assist you  
today?"  
      },  
      "logprobs": {  
        "content": [  
          {  
            "token": "Hello",  
            "logprob": -0.123,  
            "top_logprobs": [  
              {"token": "Hello", "logprob": -0.123},  
              {"token": "Hi", "logprob": -2.456},  
              {"token": "Hey", "logprob": -3.789}  
            ]  
          },  
          # Additional tokens...  
        ]  
      },  
      "finish_reason": "stop"  
    }  
  ],  
  "usage": {  
    "prompt_tokens": 12,  
    "completion_tokens": 28,  
    "total_tokens": 40  
  }  
}
```

範例 2：簡單文字完成

使用完成格式產生簡單的文字：

```
# Simple completion request
completion_request = {
    "prompt": "The capital of France is",
    "max_tokens": 50,
    "stream": False,
    "temperature": 0.0,
    "top_p": 1.0,
    "top_k": -1, # -1 means no limit
    "logprobs": 3, # Number of log probabilities to return
    "allowed_token_ids": None, # List of allowed token IDs
    "truncate_prompt_tokens": None, # Truncate prompt to this many tokens
    "stream_options": None
}

response = invoke_nova_endpoint(completion_request)
```

回應範例：

```
{
  "id": "cmpl-789012",
  "object": "text_completion",
  "created": 1234567890,
  "model": "default",
  "choices": [
    {
      "text": " Paris.",
      "index": 0,
      "logprobs": {
        "tokens": [" Paris", "."],
        "token_logprobs": [-0.001, -0.002],
        "top_logprobs": [
          {" Paris": -0.001, " London": -5.234, " Rome": -6.789},
          {".": -0.002, ",": -4.567, " ": -7.890}
        ]
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
```

```

        "prompt_tokens": 6,
        "completion_tokens": 2,
        "total_tokens": 8
    }
}

```

範例 3：串流聊天完成

```

# Streaming chat request
streaming_request = {
    "messages": [
        {"role": "user", "content": "Tell me a short story about a robot"}
    ],
    "max_tokens": 200,
    "stream": True,
    "temperature": 0.7,
    "top_p": 0.95,
    "top_k": 40,
    "logprobs": True,
    "top_logprobs": 2,
    "reasoning_effort": "high", # For more detailed reasoning
    "stream_options": {"include_usage": True}
}

invoke_nova_endpoint(streaming_request)

```

串流輸出範例：

```

Chunk: data: {"id":"chatcmpl-029ca032-fa01-4868-80b7-
c4cb1af90fb9","object":"chat.completion.chunk","created":1772060532,"model":"default","choices":
[{"index":0,"delta":
{"role":"assistant","content":"","reasoning_content":null},"logprobs":null,"finish_reason":null}]}
Chunk: data: {"id":"chatcmpl-029ca032-fa01-4868-80b7-
c4cb1af90fb9","object":"chat.completion.chunk","created":1772060532,"model":"default","choices":
[{"index":0,"delta":{"content":" Once","reasoning_content":null},"logprobs":
{"content":[{"token":"\u2581Once","logprob":-0.6078429222106934,"bytes":
[226,150,129,79,110,99,101],"top_logprobs":
[{"token":"\u2581Once","logprob":-0.6078429222106934,"bytes":
[226,150,129,79,110,99,101]}]},{"token":"\u2581In","logprob":-0.7864127159118652,"bytes":
[226,150,129,73,110]}]}]}]},"finish_reason":null,"token_ids":null}}]}
Chunk: data: {"id":"chatcmpl-029ca032-fa01-4868-80b7-
c4cb1af90fb9","object":"chat.completion.chunk","created":1772060532,"model":"default","choices":
[{"index":0,"delta":{"content":" upon","reasoning_content":null},"logprobs":

```

```

{"content":[{"token":"\u2581upon","logprob":-0.0012345,"bytes":
[226,150,129,117,112,111,110],"top_logprobs":
[{"token":"\u2581upon","logprob":-0.0012345,"bytes":[226,150,129,117,112,111,110]}],
{"token":"\u2581a","logprob":-6.789,"bytes":
[226,150,129,97]}]}]},"finish_reason":null,"token_ids":null}}]
Chunk: data: {"id":"chatcmpl-029ca032-fa01-4868-80b7-
c4cb1af90fb9","object":"chat.completion.chunk","created":1772060532,"model":"default","choices":
[{"index":0,"delta":{"content":" a","reasoning_content":null},"logprobs":
{"content":[{"token":"\u2581a","logprob":-0.0001234,"bytes":
[226,150,129,97],"top_logprobs":[{"token":"\u2581a","logprob":-0.0001234,"bytes":
[226,150,129,97]},{token":"\u2581time","logprob":-9.123,"bytes":
[226,150,129,116,105,109,101]}]}]}]},"finish_reason":null,"token_ids":null}}]
Chunk: data: {"id":"chatcmpl-029ca032-fa01-4868-80b7-
c4cb1af90fb9","object":"chat.completion.chunk","created":1772060532,"model":"default","choices":
[{"index":0,"delta":{"content":" time","reasoning_content":null},"logprobs":
{"content":[{"token":"\u2581time","logprob":-0.0023456,"bytes":
[226,150,129,116,105,109,101],"top_logprobs":
[{"token":"\u2581time","logprob":-0.0023456,"bytes":[226,150,129,116,105,109,101]},
{"token":"","logprob":-6.012,"bytes":
[44]}]}]}]},"finish_reason":null,"token_ids":null}}]

# Additional chunks...

Chunk: data: {"id":"chatcmpl-029ca032-fa01-4868-80b7-
c4cb1af90fb9","object":"chat.completion.chunk","created":1772060532,"model":"default","choices":
[{"index":0,"delta":{"},"logprobs":null,"finish_reason":"stop"}],"usage":
{"prompt_tokens":15,"completion_tokens":87,"total_tokens":102}}
Chunk: data: [DONE]

```

範例 4：多模式聊天完成

針對影像和文字輸入使用多模式格式：

```

# Multimodal chat request (if supported by your model)
multimodal_request = {
    "messages": [
        {
            "role": "user",
            "content": [
                {"type": "text", "text": "What's in this image?"},
                {"type": "image_url", "image_url": {"url": "data:image/
jpeg;base64,..."}}
            ]
        }
    ]
}

```

```
],
  "max_tokens": 150,
  "temperature": 0.3,
  "top_p": 0.8,
  "stream": False
}

response = invoke_nova_endpoint(multimodal_request)
```

回應範例：

```
{
  "id": "chatcmpl-345678",
  "object": "chat.completion",
  "created": 1234567890,
  "model": "default",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "The image shows..."
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 1250,
    "completion_tokens": 45,
    "total_tokens": 1295
  }
}
```

步驟 7：清除資源（選用）

為了避免產生不必要的費用，請刪除您在本教學課程中建立 AWS 的資源。SageMaker 端點會在執行時產生費用，即使您沒有主動提出推論請求。

Important

刪除資源是永久的，無法復原。在繼續之前，請確定您不再需要這些資源。

刪除端點

```
import boto3

# Initialize SageMaker client
sagemaker = boto3.client('sagemaker', region_name=REGION)

try:
    print("Deleting endpoint...")
    sagemaker.delete_endpoint(EndpointName=ENDPOINT_NAME)
    print(f"# Endpoint '{ENDPOINT_NAME}' deletion initiated")
    print("Charges will stop once deletion completes (typically 2-5 minutes)")
except Exception as e:
    print(f"# Error deleting endpoint: {e}")
```

Note

端點刪除是非同步的。您可以監控刪除狀態：

```
import time

print("Monitoring endpoint deletion...")
while True:
    try:
        response = sagemaker.describe_endpoint(EndpointName=ENDPOINT_NAME)
        status = response['EndpointStatus']
        print(f"Status: {status}")
        time.sleep(10)
    except sagemaker.exceptions.ClientError as e:
        if e.response['Error']['Code'] == 'ValidationException':
            print("# Endpoint successfully deleted")
            break
        else:
            print(f"Error: {e}")
            break
```

刪除端點組態

刪除端點後，移除端點組態：

```
try:
```

```
print("Deleting endpoint configuration...")
sagemaker.delete_endpoint_config(EndpointConfigName=ENDPOINT_CONFIG_NAME)
print(f"# Endpoint configuration '{ENDPOINT_CONFIG_NAME}' deleted")
except Exception as e:
    print(f"# Error deleting endpoint configuration: {e}")
```

刪除模型

移除 SageMaker 模型物件：

```
try:
    print("Deleting model...")
    sagemaker.delete_model(ModelName=MODEL_NAME)
    print(f"# Model '{MODEL_NAME}' deleted")
except Exception as e:
    print(f"# Error deleting model: {e}")
```

API 參考

SageMaker 上的 Amazon Nova 模型使用標準 SageMaker 執行期 API 進行推論。如需完整的 API 文件，請參閱[測試部署的模型](#)。

端點調用

SageMaker 上的 Amazon Nova 模型支援兩種叫用方法：

- 同步調用：將 [InvokeEndpoint](#) API 用於即時、非串流推論請求。
- 串流調用：使用 [InvokeEndpointWithResponseStream](#) API 進行即時串流推論請求。

要求格式

Amazon Nova 模型支援兩種請求格式：

聊天完成格式

使用此格式進行對話互動：

```
{
  "messages": [
    {"role": "user", "content": "string"}
```

```
],
"max_tokens": integer,
"max_completion_tokens": integer,
"stream": boolean,
"temperature": float,
"top_p": float,
"top_k": integer,
"logprobs": boolean,
"top_logprobs": integer,
"reasoning_effort": "low" | "high",
"allowed_token_ids": [integer],
"truncate_prompt_tokens": integer,
"stream_options": {
  "include_usage": boolean
}
}
```

文字完成格式

使用此格式產生簡單的文字：

```
{
  "prompt": "string",
  "max_tokens": integer,
  "stream": boolean,
  "temperature": float,
  "top_p": float,
  "top_k": integer,
  "logprobs": integer,
  "allowed_token_ids": [integer],
  "truncate_prompt_tokens": integer,
  "stream_options": {
    "include_usage": boolean
  }
}
```

多模式聊天完成格式

針對影像和文字輸入使用此格式：

```
{
  "messages": [
    {
```



```

    "role": "user",
    "content": [
      {"type": "text", "text": "What's in this image?"},
      {"type": "image_url", "image_url": {"url": "data:image/jpeg;base64,..."}}
    ]
  },
  "max_tokens": integer,
  "temperature": float,
  "top_p": float,
  "stream": boolean
}

```

請求參數

- **messages** (陣列) : 適用於聊天完成格式。具有 **role** 和 **content** 欄位的訊息物件陣列。內容可以是純文字的字串，也可以是多模式輸入的陣列。
- **prompt** (字串) : 適用於文字完成格式。要從中產生的輸入文字。
- **max_tokens** (整數) : 回應中要產生的字符數目上限。範圍：1 或更高。
- **max_completion_tokens** (整數) : 完成聊天時 **max_tokens** 的替代方案。要產生的完成權杖數目上限。
- **temperature** (浮點數) : 控制產生時的隨機性。範圍：0.0 到 2.0 (0.0 = 確定性，2.0 = 最大隨機性)。
- **top_p** (浮點數) : Nucleus 取樣閾值。範圍：1e-10 到 1.0。
- **top_k** (整數) : 將權杖選擇限制為 K 最有可能的權杖。範圍：-1 或更高 (-1 = 無限制)。
- **stream** (布林值) : 是否串流回應。將 設為 **true** 以進行串流，將 設為 **false** 以進行非串流。
- **logprobs** (布林值/整數) : 對於聊天完成，請使用布林值。對於文字完成，請針對要傳回的日誌機率使用整數。範圍：1 到 20。
- **top_logprobs** (整數) : 最有可能傳回日誌機率的字符數量 (僅限聊天完成)。
- **reasoning_effort** (字串) : 推理工作的程度。選項：「低」、「高」 (僅限 Nova 2 Lite 自訂模型的聊天完成)。
- **allowed_token_ids** (陣列) : 允許產生的字符 IDs 清單。將輸出限制為指定的字符。
- **truncate_prompt_tokens** (整數) : 如果超過限制，請將提示截斷為許多字符。
- **stream_options** (物件) : 串流回應的選項。包含 **include_usage** 布林值，以在串流回應中包含字符用量。

回應格式

回應格式取決於呼叫方法和請求類型：

聊天完成回應（非串流）

對於同步聊天完成請求：

```
{
  "id": "chatcmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "chat.completion",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Hello! I'm doing well, thank you for asking. How can I help you
today?",
        "refusal": null,
        "reasoning": null,
        "reasoning_content": null
      },
      "logprobs": {
        "content": [
          {
            "token": "Hello",
            "logprob": -0.31725305,
            "bytes": [72, 101, 108, 108, 111],
            "top_logprobs": [
              {
                "token": "Hello",
                "logprob": -0.31725305,
                "bytes": [72, 101, 108, 108, 111]
              },
              {
                "token": "Hi",
                "logprob": -1.3190403,
                "bytes": [72, 105]
              }
            ]
          }
        ]
      }
    }
  ]
}
```

```

    },
    "finish_reason": "stop",
    "stop_reason": null,
    "token_ids": [9906, 0, 358, 2157, 1049, 11, 1309, 345, 369, 6464, 13]
  }
],
"usage": {
  "prompt_tokens": 9,
  "completion_tokens": 12,
  "total_tokens": 21,
  "prompt_tokens_details": {
    "cached_tokens": 0
  }
},
"prompt_token_ids": [9906, 0, 358]
}

```

文字完成回應（非串流）

對於同步文字完成請求：

```

{
  "id": "cml-123e4567-e89b-12d3-a456-426614174000",
  "object": "text_completion",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "text": "Paris, the capital and most populous city of France.",
      "logprobs": {
        "tokens": ["Paris", ",", " the", " capital"],
        "token_logprobs": [-0.31725305, -0.07918124, -0.12345678, -0.23456789],
        "top_logprobs": [
          {
            "Paris": -0.31725305,
            "London": -1.3190403,
            "Rome": -2.1234567
          },
          {
            ",": -0.07918124,
            " is": -1.2345678
          }
        ]
      }
    }
  ]
}

```

```

    },
    "finish_reason": "stop",
    "stop_reason": null,
    "prompt_token_ids": [464, 6864, 315, 4881, 374],
    "token_ids": [3915, 11, 279, 6864, 323, 1455, 95551, 3363, 315, 4881, 13]
  }
],
"usage": {
  "prompt_tokens": 5,
  "completion_tokens": 11,
  "total_tokens": 16,
  "prompt_tokens_details": {
    "cached_tokens": 0
  }
}
}
}

```

聊天完成串流回應

對於串流聊天完成請求，回應會以伺服器傳送事件 (SSE) 的形式傳送：

```

data: {
  "id": "chatcmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "chat.completion.chunk",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "delta": {
        "role": "assistant",
        "content": "Hello",
        "refusal": null,
        "reasoning": null,
        "reasoning_content": null
      },
      "logprobs": {
        "content": [
          {
            "token": "Hello",
            "logprob": -0.31725305,
            "bytes": [72, 101, 108, 108, 111],
            "top_logprobs": [

```

```

        "token": "Hello",
        "logprob": -0.31725305,
        "bytes": [72, 101, 108, 108, 111]
      }
    ]
  },
  "finish_reason": null,
  "stop_reason": null
}
],
"usage": null,
"prompt_token_ids": null
}

data: {
  "id": "chatcmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "chat.completion.chunk",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "delta": {
        "content": "! I'm"
      },
      "logprobs": null,
      "finish_reason": null,
      "stop_reason": null
    }
  ],
  "usage": null
}

data: {
  "id": "chatcmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "chat.completion.chunk",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "delta": {},

```

```
    "finish_reason": "stop",
    "stop_reason": null
  }
],
"usage": {
  "prompt_tokens": 9,
  "completion_tokens": 12,
  "total_tokens": 21,
  "prompt_tokens_details": {
    "cached_tokens": 0
  }
}
}

data: [DONE]
```

文字完成串流回應

對於串流文字完成請求：

```
data: {
  "id": "cmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "text_completion",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "text": "Paris",
      "logprobs": {
        "tokens": ["Paris"],
        "token_logprobs": [-0.31725305],
        "top_logprobs": [
          {
            "Paris": -0.31725305,
            "London": -1.3190403
          }
        ]
      },
      "finish_reason": null,
      "stop_reason": null
    }
  ],
  "usage": null
}
```

```
}

data: {
  "id": "cmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "text_completion",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "text": ", the capital",
      "logprobs": null,
      "finish_reason": null,
      "stop_reason": null
    }
  ],
  "usage": null
}

data: {
  "id": "cmpl-123e4567-e89b-12d3-a456-426614174000",
  "object": "text_completion",
  "created": 1677652288,
  "model": "nova-micro-custom",
  "choices": [
    {
      "index": 0,
      "text": "",
      "finish_reason": "stop",
      "stop_reason": null
    }
  ],
  "usage": {
    "prompt_tokens": 5,
    "completion_tokens": 11,
    "total_tokens": 16
  }
}

data: [DONE]
```

回應欄位說明

- `id` : 完成的唯一識別符
- `object` : 傳回的物件類型 ("chat.completion"、"text_completion"、"chat.completion.chunk")
- `created` : 建立完成時的 Unix 時間戳記
- `model` : 用於完成的模型
- `choices` : 完成選項陣列
- `usage` : 字符用量資訊，包括提示、完成和字符總數
- `logprobs` : 字符的日誌機率資訊（請求時）
- `finish_reason` : 模型停止產生的原因 ("stop"、"length"、"content_filter")
- `delta` : 串流回應中的增量內容
- `reasoning` : 使用 `reasoning_effort` 時合理的內容
- `token_ids` : 產生文字的字符 IDs 陣列

如需完整的 API 文件，請參閱 [InvokeEndpoint API 參考](#) 和 [InvokeEndpointWithResponseStream API 參考](#)。

評估託管在 SageMaker 推論上的模型

本指南說明如何使用 [Inspect AI](#) 開放原始碼評估架構，評估部署在 SageMaker 推論端點上的自訂 Amazon Nova 模型。

Note

如需實作演練，請參閱 [SageMaker Inspect AI 快速入門筆記本](#)。

概觀

您可以使用 AI 研究社群的標準化基準，評估部署在 SageMaker 端點上的自訂 Amazon Nova 模型。此方法可讓您：

- 大規模評估自訂的 Amazon Nova 模型（微調、分割或以其他方式調整）
- 跨多個端點執行個體以平行推論執行評估
- 使用 MMLU、TruthfulQA 和 HumanEval 等基準比較模型效能
- 與您現有的 SageMaker 基礎設施整合

支援的模型

SageMaker 推論提供者適用於：

- Amazon Nova 模型 (Nova Micro、Nova Lite、Nova Lite 2)
- 透過 vLLM 或 OpenAI 相容推論伺服器部署的模型
- 支援 OpenAI 聊天完成 API 格式的任何端點

先決條件

開始前，請確保您具備以下條件：

- AWS 帳戶 具有建立和叫用 SageMaker 端點許可的
- AWS 透過 AWS CLI、環境變數或 IAM 角色設定的登入資料
- Python 3.9 或更新版本

必要的 IAM 許可

您的 IAM 使用者或角色需要下列許可：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sagemaker:InvokeEndpoint",
        "sagemaker:DescribeEndpoint"
      ],
      "Resource": "arn:aws:sagemaker:*:*:endpoint/*"
    }
  ]
}
```

步驟 1：部署 SageMaker 端點

在執行評估之前，您需要執行模型的 SageMaker 推論端點。

如需使用 Amazon Nova 模型建立 SageMaker 推論端點的說明，請參閱 [the section called “開始使用”](#)。

一旦您的端點處於 InService 狀態，請記下要用於評估命令的端點名稱。

步驟 2：安裝評估相依性

建立 Python 虛擬環境並安裝必要的套件。

```
# Create virtual environment
python3.12 -m venv venv
source venv/bin/activate

# Install uv for faster package installation
pip install uv

# Install Inspect AI and evaluation benchmarks
uv pip install inspect-ai inspect-vals

# Install AWS dependencies
uv pip install aioboto3 boto3 botocore openai
```

步驟 3：設定 AWS 登入資料

選擇下列其中一種身分驗證方法：

選項 1：AWS CLI（建議）

```
aws configure
```

出現提示時，輸入您的 AWS 存取金鑰 ID、私密存取金鑰和預設區域。

選項 2：環境變數

```
export AWS_ACCESS_KEY_ID=your_access_key
export AWS_SECRET_ACCESS_KEY=your_secret_key
export AWS_DEFAULT_REGION=us-west-2
```

選項 3：IAM 角色

如果在 Amazon EC2 或 SageMaker 筆記本上執行，會自動使用執行個體的 IAM 角色。

驗證登入資料

```
import boto3
```

```
sts = boto3.client('sts')
identity = sts.get_caller_identity()
print(f"Account: {identity['Account']}")
print(f"User/Role: {identity['Arn']}")
```

步驟 4：安裝 SageMaker 提供者

SageMaker 提供者可讓 Inspect AI 與您的 SageMaker 端點通訊。在[快速入門筆記本](#)中簡化了提供者安裝程序。

步驟 5：下載評估基準

複製檢查 Evals 儲存庫以存取標準基準：

```
git clone https://github.com/UKGovernmentBEIS/inspect_evals.git
```

此儲存庫包含基準，例如：

- MMLU 和 MMLU-Pro（知識和推理）
- TruthfulQA（真實性）
- HumanEval（程式碼產生）
- GSM8K（數學推理）

步驟 6：執行評估

使用您的 SageMaker 端點執行評估：

```
cd inspect_evals/src/inspect_evals/

inspect eval mmlu_pro/mmlu_pro.py \
  --model sagemaker/my-nova-endpoint \
  -M region_name=us-west-2 \
  --max-connections 256 \
  --max-retries 100 \
  --display plain
```

金鑰參數

參數	預設	Description
<code>--max-connections</code>	10	對端點的平行請求數量。使用執行個體計數擴展（例如，10 個執行個體 × 25 = 250）。
<code>--max-retries</code>	3	重試失敗請求的嘗試。使用 50-100 進行大型評估。
<code>-M region_name</code>	us-east-1	AWS 部署端點的區域。
<code>-M read_timeout</code>	600	請求以秒為單位的逾時。
<code>-M connect_timeout</code>	60	連線逾時，以秒為單位。

調校建議

對於多執行個體端點：

```
# 10-instance endpoint example
--max-connections 250    # ~25 connections per instance
--max-retries 100       # Handle transient errors
```

設定 `--max-connections` 過高可能會壓倒端點並導致限流。設定太低，未充分利用容量。

步驟 7：檢視結果

啟動檢查 AI 檢視器以分析評估結果：

```
inspect view
```

檢視器會顯示：

- 整體分數和指標
- 每個範例結果與模型回應
- 錯誤分析和失敗模式

管理端點

更新端點

若要使用新模型或組態更新現有端點：

```
import boto3

sagemaker = boto3.client('sagemaker', region_name=REGION)

# Create new model and endpoint configuration
# Then update the endpoint
sagemaker.update_endpoint(
    EndpointName=EXISTING_ENDPOINT_NAME,
    EndpointConfigName=NEW_ENDPOINT_CONFIG_NAME
)
```

刪除端點

```
sagemaker.delete_endpoint(EndpointName=ENDPOINT_NAME)
```

加入自訂基準

您可以使用下列工作流程，將新基準新增至檢查 AI：

1. 研究基準的資料集格式和評估指標
2. 在 中檢閱類似的實作 `inspect_evals/`
3. 建立任務檔案，將資料集記錄轉換為檢查 AI 範例
4. 實作適當的求解器和評分器
5. 使用小型測試執行進行驗證

範例任務結構：

```
from inspect_ai import Task, task
from inspect_ai.dataset import hf_dataset
from inspect_ai.scorer import choice
from inspect_ai.solver import multiple_choice

@task
def my_benchmark():
    return Task(
        dataset=hf_dataset("dataset_name", split="test"),
        solver=multiple_choice(),
```

```
scorer=choice()  
)
```

疑難排解

常見問題

端點限流或逾時

- 減少 `--max-connections`
- 增加 `--max-retries`
- 檢查端點 CloudWatch 指標是否有容量問題

身分驗證錯誤

- 確認已正確設定 AWS 登入資料
- 檢查 IAM 許可包括 `sagemaker:InvokeEndpoint`

模型錯誤

- 確認端點處於 `InService` 狀態
- 檢查模型是否支援 OpenAI 聊天完成 API 格式

相關資源

- [檢查 AI 文件](#)
- [檢查 Evals 儲存庫](#)
- [SageMaker 開發人員指南](#)
- [部署模型以進行推論](#)
- [設定 AWS CLI](#)

在 Amazon SageMaker 推論濫用偵測中部署 Amazon Nova Forge 模型

AWS 致力於負責任地使用 AI。為了協助防止潛在的濫用，當您在 Amazon SageMaker 推論中部署 Amazon Nova Forge 模型時，SageMaker 推論會實作自動濫用偵測機制，以識別可能違反 AWS [可接受的使用政策和服務條款](#)，包括[負責任的 AI 政策](#)。

我們的濫用偵測機制是完全自動化的，因此沒有人工審查或存取使用者輸入或模型輸出。

自動濫用偵測包括：

- 分類內容 – 我們使用分類器來偵測使用者輸入和模型輸出中的有害內容（例如煽動暴力的內容）。分類器是一種演算法，用於處理模型輸入和輸出，並指派傷害類型和可信度。我們可能會在 Amazon Nova Forge Model 用量上執行這些分類器。分類程序會自動執行，不涉及對使用者輸入或模型輸出進行人工審查。
- 識別模式 – 我們使用分類器指標來識別潛在的違規和重複行為。我們可能會編譯和共用匿名分類器指標。Amazon SageMaker Inference 不會儲存使用者輸入或模型輸出。
- 偵測和封鎖兒童性濫用資料 (CSAM) – 您對您（和最終使用者）上傳至 Amazon SageMaker Inference 的內容負責，並且必須確保此內容沒有非法影像。為了協助停止 CSAM 的傳播，在 Amazon SageMaker 推論中部署 Amazon Nova Forge 模型時，SageMaker 推論可能會使用自動濫用偵測機制（例如雜湊比對技術或分類器）來偵測明顯的 CSAM。如果 Amazon SageMaker Inference 在您的映像輸入中偵測到明顯的 CSAM，Amazon SageMaker Inference 會封鎖請求，而且您會收到自動錯誤訊息。Amazon SageMaker Inference 也可以向國家缺少和入侵兒童中心 (NCMEC) 或相關當局提交報告。我們非常重視 CSAM，並將繼續更新我們的偵測、封鎖和報告機制。相關法律可能會要求您採取其他動作，而且您必須對這些動作負責。

一旦我們的自動濫用偵測機制發現潛在違規，我們可能會請求您使用 Amazon SageMaker 推論以及遵守服務條款的相關資訊。如果您沒有回應、不願意或無法遵守這些條款或政策，AWS 可能會暫停您對 Amazon SageMaker 推論的存取。如果我們的自動化測試偵測到模型回應與我們的條款和政策不一致，則可能會向您收取失敗的推論任務的費用。

如果您有其他問題，請聯絡 AWS Support。如需詳細資訊，請參閱 [Amazon SageMaker FAQs](#)。

Amazon Bedrock 參考。

訓練並測試 Amazon Nova 模型後，您可以將模型部署到 Amazon Bedrock 以進行生產規模推論。部署程序涉及使用 CreateCustomModel API 建立 Amazon Bedrock 模型、從受管 Amazon S3 儲存貯體將您的模型成品匯出到模型模型，然後一旦模型處於 ACTIVE 狀態，即可使用隨需或佈建輸送量推論來設定端點。

在 SageMaker 中建立自訂模型後，您可以使用 CreateCustomModel API 從 SageMaker 託管將其部署到 Amazon Bedrock，以執行推論。然後，您可以使用 CreateCustomModelDeployment 來建立 OD 推論端點，或設定參數效率微調 (PEFT) 模型的佈建輸送量推論。您可以設定完整排名自訂模型的佈建輸送量推論。

您也可以使用 Amazon Nova Forge SDK 部署自訂的 Amazon Nova 模型。Amazon Nova Forge SDK 提供簡化的體驗，讓您從訓練任務或 S3 模型檢查點擷取相關資訊，並將其發佈至 Amazon Bedrock。如需詳細資訊，請參閱 [Amazon Nova Forge SDK](#)。

如需為自訂模型設定 Amazon Bedrock 推論的詳細步驟，請參閱[將自訂 Amazon Nova 模型部署至 Amazon Bedrock](#)。

下一節提供自訂模型隨需推論的詳細資訊。

部署自訂模型以進行隨需推論

透過模型自訂任務 (微調、蒸餾或持續預先訓練) 成功建立自訂模型後，您可以為模型設定隨需推論。

若要設定自訂模型的隨需推論，請使用自訂模型部署來部署模型。部署自訂模型之後，使用部署的 Amazon Resource Name (ARN) 作為 `InvokeModel` 或 `Converse API` 操作中的 `modelId` 參數。您可以使用部署的模型搭配 Amazon Bedrock 功能，例如實驗場、代理程式和知識庫，進行隨需推論。

主題

- [支援的模型](#)
- [部署自訂模型](#)
- [使用部署進行隨需推論](#)
- [刪除自訂模型部署](#)

支援的模型

您可以為下列模型設定隨需推論：

- Amazon Nova Canvas
- Amazon Nova Lite
- Amazon Nova Micro
- Amazon Nova Pro

部署自訂模型

您可以使用 Amazon Bedrock 主控台 AWS Command Line Interface 或 AWS SDKs 部署自訂模型。如需使用部署進行推論的相關資訊，請參閱[使用部署進行隨需推論](#)。

主題

- [部署自訂模型 \(主控台\)](#)
- [部署自訂模型 \(AWS Command Line Interface\)](#)
- [部署自訂模型AWS SDKs\)](#)

部署自訂模型 (主控台)

您可以如下所示從自訂模型頁面部署自訂模型。您也可以從具有相同欄位的隨需自訂模型頁面部署模型。若要尋找此頁面，請在導覽窗格的推論和評估中選擇隨需自訂模型。

部署自訂模型

1. AWS 管理主控台 使用[具有 Amazon Bedrock 許可的 IAM 角色](#)登入，然後開啟位於 <https://console.aws.amazon.com/nova/> 的 Amazon Bedrock 主控台。
2. 從左側導覽窗格中，選擇基礎模型下的自訂模型。
3. 在模型索引標籤中，選擇您要部署之模型的選項按鈕。
4. 選擇設定推論，然後選擇隨需部署。
5. 在部署詳細資訊中，提供下列資訊：
 - 部署名稱 (必要)：輸入部署的唯一名稱。
 - 描述 (選用)：輸入部署描述。
 - 標籤 (選用)：新增成本分配和資源管理的標籤。
6. 選擇建立。當狀態顯示 Completed 時，自訂模型已準備好進行隨需推論。如需使用自訂模型的詳細資訊，請參閱[使用部署進行隨需推論](#)。

部署自訂模型 (AWS Command Line Interface)

若要使用 部署用於隨需推論的自訂模型 AWS Command Line Interface，請使用 `create-custom-model-deployment` 命令搭配自訂模型的 Amazon Resource Name (ARN)。此命令使用 [CreateCustomModelDeployment](#) API 操作。它會傳回部署的 ARN，您可以在提出推論請求時將其用作 `modelId`。如需使用部署進行推論的相關資訊，請參閱[使用部署進行隨需推論](#)。

```
aws bedrock create-custom-model-deployment \  
--model-deployment-name "Unique name" \  
--model-arn "Custom Model ARN" \  

```

```
--description "Deployment description" \
--tags '[
    {
        "key": "Environment",
        "value": "Production"
    },
    {
        "key": "Team",
        "value": "ML-Engineering"
    },
    {
        "key": "Project",
        "value": "CustomerSupport"
    }
]' \
--client-request-token "unique-deployment-token" \
--region region
```

部署自訂模型AWS SDKs)

若要部署用於隨需推論的自訂模型，請搭配自訂模型的 Amazon Resource Name (ARN) 使用 [CreateCustomModelDeployment](#) API 操作。回應會傳回部署的 ARN，您可以在提出推論請求時將其用作 modelId。如需使用部署進行推論的相關資訊，請參閱[使用部署進行隨需推論](#)。

下列程式碼說明如何使用適用於 Python 的 SDK (Boto3) 部署自訂模型。

```
def create_custom_model_deployment(bedrock_client):
    """Create a custom model deployment
    Args:
        bedrock_client: A boto3 Bedrock client for making API calls

    Returns:
        str: The ARN of the created custom model deployment

    Raises:
        Exception: If there is an error creating the deployment
    """

    try:
        response = bedrock_client.create_custom_model_deployment(
            modelDeploymentName="Unique deployment name",
            modelArn="Custom Model ARN",
            description="Deployment description",
```

```
tags=[
    {'key': 'Environment', 'value': 'Production'},
    {'key': 'Team', 'value': 'ML-Engineering'},
    {'key': 'Project', 'value': 'CustomerSupport'}
],
clientRequestToken=f"deployment-{uuid.uuid4()}"
)

deployment_arn = response['customModelDeploymentArn']
print(f"Deployment created: {deployment_arn}")
return deployment_arn

except Exception as e:
    print(f"Error creating deployment: {str(e)}")
    raise
```

使用部署進行隨需推論

部署用於隨需推論的自訂模型後，您可以使用它，透過提出推論請求來產生回應。對於 `InvokeModel` 或 `Converse` 作業，您可以使用部署的 Amazon Resource Name (ARN) 作為 `modelId`。

如需提出推論請求的資訊，請參閱下列主題：

- [提交提示詞並使用模型推論產生回應](#)
- [執行模型推論的必要條件](#)
- [使用 API 提交提示詞並產生回應](#)

刪除自訂模型部署

使用模型完成隨需推論後，您可以刪除部署。刪除部署後，您便無法將其用於隨需推論，但不會刪除基礎自訂模型。

您可以使用 Amazon Bedrock 主控台、AWS Command Line Interface 或 AWS SDKs 刪除自訂模型部署。

Important

刪除自訂模型部署是不可復原的。在繼續刪除之前，請確定您不再需要部署。如果您需要再次使用自訂模型進行隨需推論，則必須建立新的部署。

主題

- [刪除自訂模型部署 \(主控台\)](#)
- [刪除自訂模型部署 \(AWS Command Line Interface\)](#)
- [刪除自訂模型部署AWS SDKs\)](#)

刪除自訂模型部署 (主控台)

刪除自訂模型部署

1. 在導覽窗格的推論和評估下，選擇隨需自訂模型。
2. 選擇您要刪除的自訂模型部署。
3. 選擇刪除。
4. 在確認對話方塊中，輸入部署名稱以確認刪除。
5. 選擇刪除以確認刪除。

當刪除正在進行時，部署狀態會變更為 `Deleting`。完成後，部署將從清單中移除。

刪除自訂模型部署 (AWS Command Line Interface)

若要使用 刪除自訂模型部署 AWS Command Line Interface，請使用 `delete-custom-model-deployment` 命令搭配您的部署識別符。

```
aws bedrock delete-custom-model-deployment \
--custom-model-deployment-identifier "deployment-arn-or-name" \
--region region
```

刪除自訂模型部署AWS SDKs)

若要以程式設計方式刪除自訂模型部署，請使用 [DeleteCustomModelDeployment](#) API 操作搭配部署的 Amazon Resource Name (ARN) 或名稱。下列程式碼說明如何使用適用於 Python (Boto3) 的 SDK 來刪除自訂模型部署。

```
def delete_custom_model_deployment(bedrock_client):
    """Delete a custom model deployment

    Args:
```

```
bedrock_client: A boto3 Bedrock client for making API calls

Returns:
    dict: The response from the delete operation

Raises:
    Exception: If there is an error deleting the deployment
"""

try:
    response = bedrock_client.delete_custom_model_deployment(
        customModelDeploymentIdentifier="Deployment identifier"
    )

    print(f"Deployment deletion initiated")
    return response

except Exception as e:
    print(f"Error deleting deployment: {str(e)}")
    raise
```

基於自訂模型的隨需推論

隨需 (OD) 推論可讓您在自訂 Amazon Nova 模型上執行推論，而無需維護佈建的輸送量端點。這可協助您最佳化成本並有效率地進行擴展。使用隨需推論時，會根據使用量向您收費，以詞元數計量，包括進出詞元。

相容性要求

適用下列相容性要求：

- Amazon Nova Pro、Lite 和 Micro 自訂理解模型支援 OD 推論。Nova 自訂內容產生模型不支援隨需推論。
- 2025 年 7 月 16 日之後訓練的 Amazon Nova 自訂理解模型支援隨需推論。2025 年 7 月 16 日之前訓練的自訂模型與隨需推論不相容。
- Amazon Bedrock 自訂：使用 Amazon Bedrock 自訂自訂的模型，以及使用 Amazon Bedrock 從教師模型蒸餾的學生模型，均支援隨需推論。
- SageMaker AI 自訂：對於 SageMaker AI 中自訂的模型，只有在 Amazon Bedrock 上託管的參數高效微調 (PEFT) 模型才支援隨需推論。這包括直接偏好最佳化及 PEFT。經微調的全秩模型不支援隨需推論。

模型訓練與推論

在 2025 年 7 月 16 日之後使用 PEFT 在 Amazon Bedrock 或 SageMaker AI 上訓練新的自訂 Amazon Nova Pro、Lite 或 Micro 模型時，模型會自動與佈建和隨需推論選項相容。您可以在部署模型時選取偏好的推論方法。

若要搭配 2025 年 7 月 16 日之後訓練的模型使用隨需推論，請完成下列步驟：

1. 使用 [Amazon Bedrock 自訂 API](#) 或 [SageMaker AI 自訂 API](#) 建立新的微調任務。
2. 使用 [CreateCustomModel API](#) 將新訓練的模型部署到 Amazon Bedrock。
3. 使用 CustomModelDeployment API 部署以進行隨需推論。

速率限制

下列每分鐘請求數 (RPM) 和每分鐘詞元數 (TPM) 限制適用於隨需推論請求：

Base Model for Custom Model	RPM per Custom Model Deployment	TPM per Custom Model Deployment
Nova 2 Lite	2,000	4,000,000

若要進一步了解 Amazon Nova 適用的配額，請參閱 [Amazon Nova 的配額](#)。

延遲

您可以預期基本模型調用與轉接器之間的端至端延遲差異 (即到第一個詞元的時間 (TTFT)) 為 20-55%。確切的延遲值因模型大小而異，且符合業界標準。

自訂 Amazon Nova 模型的限制

Amazon Nova 自訂不支援 SageMaker 上的下列功能。

- 對執行個體進行 SSH 以尋找指標

由於已實施安全控制，因此您無法在訓練 algo-1 執行個體中對主節點進行 SSH，以尋找記憶體統計資料或 NVIDIA 統計資料，並驗證訓練步驟。

- 支援 SageMaker 訓練器 SDK

SageMaker 訓練器目前無法開始訓練任務。今日您只能透過估算器 API 開始訓練任務。

- SageMaker 訓練任務無法存取暖集區

由於已實施安全控制，因此 SageMaker 暖集區無法用來將執行個體保留在暖集區中，直到存留時間為止。

- 自訂模型合併

目前不支援合併多個模型。這表示無法使用基本模型建立多個 LoRA 轉接器並執行多合併操作。

- 支援的可觀測性工具

[TensorBoard](#) 是唯一支援檢視 SageMaker 訓練任務指標的可觀測性工具。目前不支援 [MLFlow](#) 或 [WandB](#)。如需在 SageMaker 中使用 TensorBoard 的詳細資訊，請參閱 [SageMaker 中的 TensorBoard](#)。

Amazon Nova Forge

Amazon Nova Forge 是first-of-its-kind的服務，為組織提供使用 Nova 建置自己的前沿模型最簡單且最具成本效益的方式。

Amazon Nova Forge 引進「開放式訓練」模型的概念，可讓組織存取各種早期模型檢查點，並在模型訓練的每個階段將專屬資料與 Amazon 策劃的資料集混合。這可讓模型最大限度地從專屬資料中學習，同時將忘記基礎技能的風險降至最低，例如推理。

Nova Forge 提供下列主要功能：

- 跨模型開發的所有階段存取檢查點，並在新 Nova 模型廣泛推出之前加以利用
- 使用 Amazon Nova 策劃的訓練資料混合您的專屬資料
- 在您的環境中使用獎勵函數執行強化學習
- 使用最佳化的按鈕配方，透過視覺化工作流程或命令列界面使用 Nova 建置
- 使用 Responsible AI Toolkit，讓模型在訓練過程中符合 Amazon Nova 的負責任 AI 準則，並實作執行時間控制，以在推論期間調整模型回應。

先決條件

主題

- [訂閱 Nova Forge](#)
- [其他先決條件](#)

訂閱 Nova Forge

若要請求存取 Amazon Nova Forge 服務，請將下列標籤新增至主控台 IAM 角色：值為 true 的金鑰 forge-subscription。將此標籤新增至角色後，請前往 SageMaker AI 主控台 > 模型訓練和自訂，然後按一下 Nova Forge。在此頁面上，您將找到有關服務、定價資訊和功能的詳細資訊。您可以從此頁面請求訂閱，然後管理您的訂閱。

1. 角色應具有呼叫 api 的許可 ListAttachedRolePolicy，且回應應包含 AdministratorAccess 或 AmazonSageMakerFullAccess 政策。
2. 登入角色應具有呼叫 api 的許可 ListRoleTags，且回應標籤應包含 tag.key=forge-subscription。

其他先決條件

也請確定下列先決條件已完成：

1. [一般先決條件](#)
2. 使用者的其他步驟：將受限制執行個體群組 (RIG) 新增至 SageMaker HyperPod 叢集（若要完成以下步驟，請參閱[此處](#)）

Nova Forge 存取和設定

若要加入 Nova Forge，請遵循下列 2 步驟程序：

- 步驟 1：訂閱 Nova Forge
- 步驟 2：設定 HyperPod 基礎設施

取得 Nova Forge 文件

若要取得 Nova Forge 文件，請遵循下列步驟：

```
mkdir NovaForgeHyperpodCLI
cd NovaForgeHyperpodCLI
aws s3 cp s3://nova-forge-c7363-206080352451-us-east-1/v1/ ./ --recursive
pip install -e .
```

步驟 1：訂閱 Nova Forge

快速摘要：

1. 確認您具有 Amazon Web Services 帳戶的管理員存取權。
2. 導覽至 SageMaker AI 主控台並請求存取 Nova Forge。
3. 等待 Nova 團隊在您的訂閱請求核准後傳送電子郵件確認。
4. 使用標籤 `forge-subscription` 標記您的執行角色。存取 Nova Forge 功能和檢查點需要此標籤。將下列標籤新增至您的執行角色：
 - 索引鍵：`forge-subscription`
 - 值：`true`

詳細指南

若要訂閱 Nova Forge 並有效使用自訂服務，Amazon Web Services 客戶必須擁有其 Amazon Web Services 帳戶的管理員存取權，或讓其管理員授予其管理員存取權。本文件概述下列各項所需的步驟：

- 安全管理員存取
- 設定政策以訂閱 Nova Forge
- 存取自訂配方
- 設定自訂
- 監控工作流程
- 評估自訂模型檢查點

選項 A

流程 1：帳戶使用者必須聯絡帳戶管理員，以請求下列項目：

- 透過 IAM 將 `forge-subscription` 標籤新增至帳戶（如需步驟，請參閱附錄 A）。
- 透過 IAM 新增 `ListRoleTags` 和 `ListAttachedRolePolicies` 許可（如需步驟，請參閱附錄 B）。

❗ Required setup for requesting subscription
To request Nova Forge subscription, your IAM role must have the "forge-subscription" tag attached and the "ListRoleTags" and "ListAttachedRolePolicies" policies added.

Learn more ↗

Nova Forge

Request subscription Documentation ↗

Choose from multiple Nova model checkpoints to introduce your data at the optimal stage, mix Amazon-curated training data from each stage with your data, integrate your proprietary tools as reward functions, customize the responsible AI controls, and build your own custom frontier model.

Subscription

Request subscription to access Nova Forge resources. Once approved, your payer account will be eligible for a 1-month free trial. Infrastructure costs will apply during the free trial. See Nova Forge pricing details below.

Status

⊖ Not subscribed

選項 B

流程：帳戶使用者必須聯絡帳戶管理員，請求管理員存取帳戶。

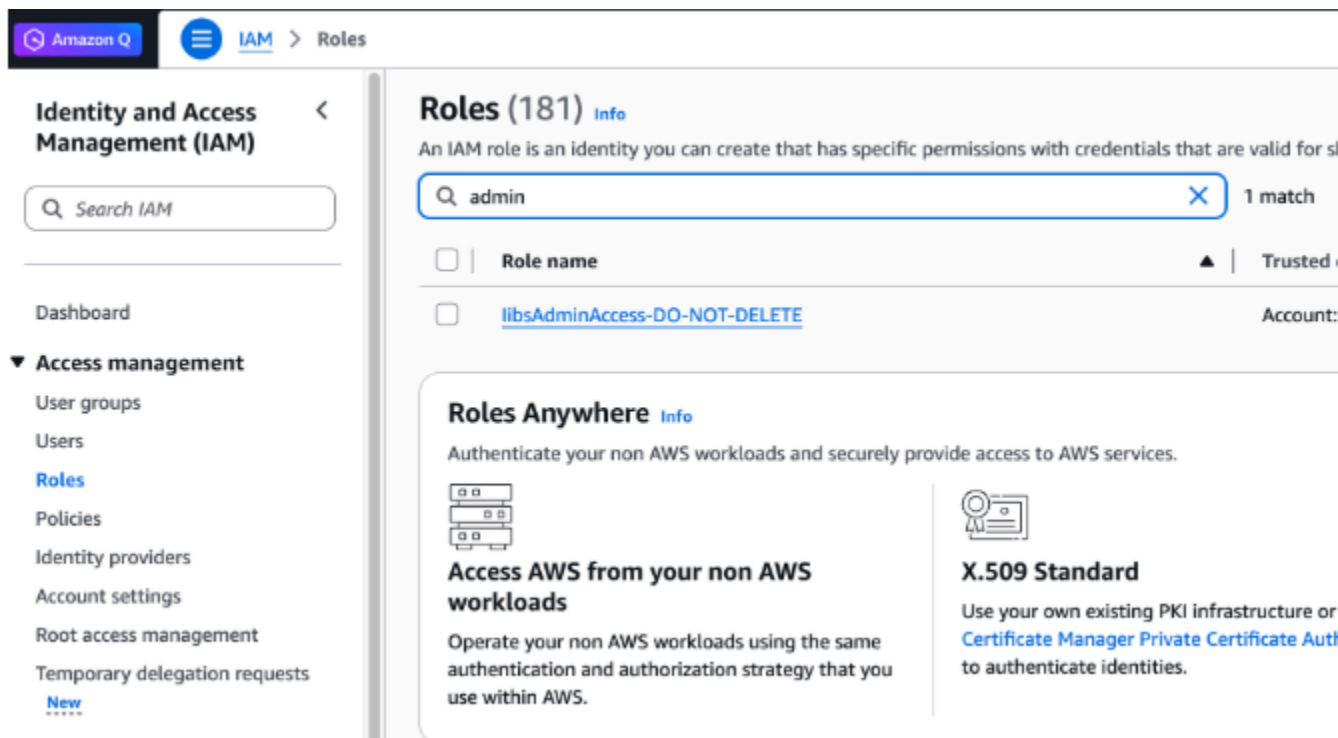
- 授予管理員存取權後，請遵循流程 2 中的步驟。

流程 2。具有管理員存取權的 Amazon Web Services 帳戶

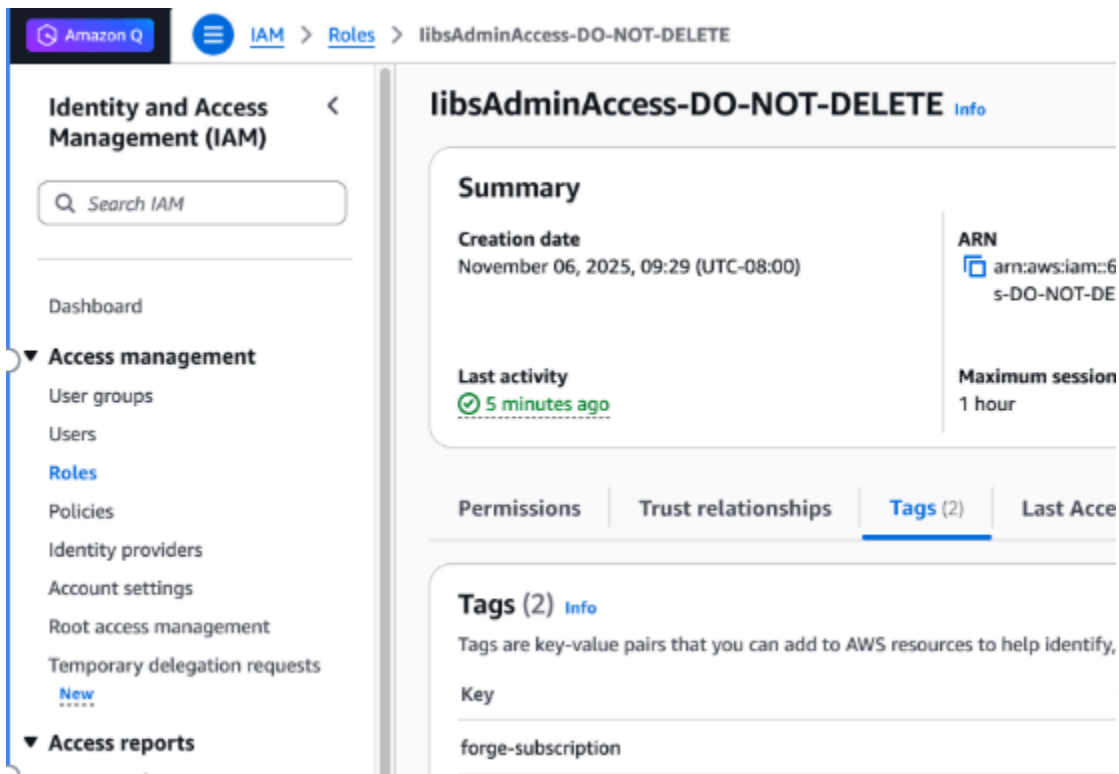
- 透過 IAM 將 forge-subscription 標籤新增至帳戶。請參閱附錄 A 中的步驟

附錄 A：將 forge-subscription 政策新增至 Amazon Web Services 帳戶

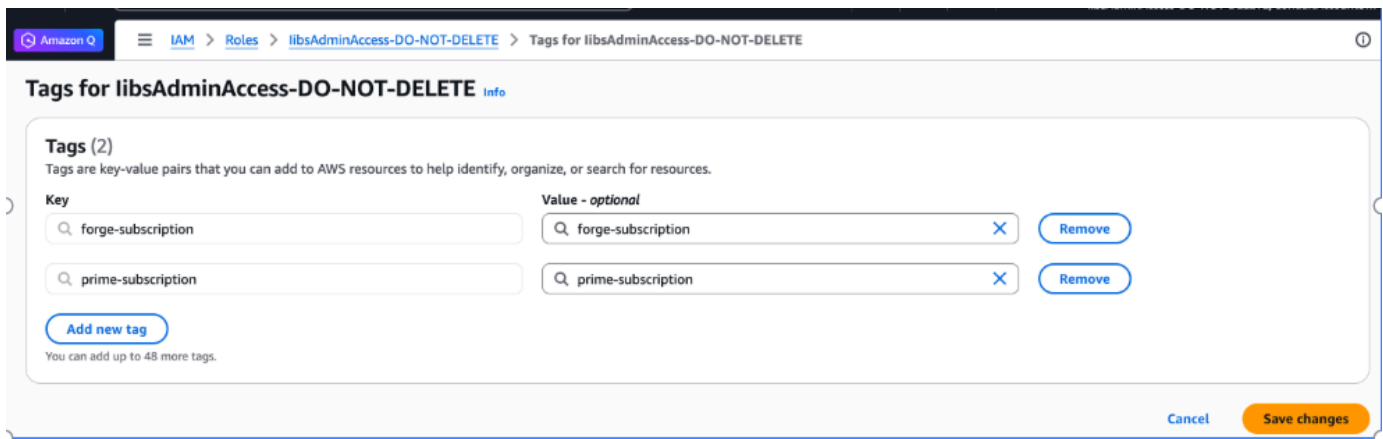
- 前往 Amazon Web Services IAM 儀表板。按一下左側的角色。搜尋管理員並按一下管理員角色



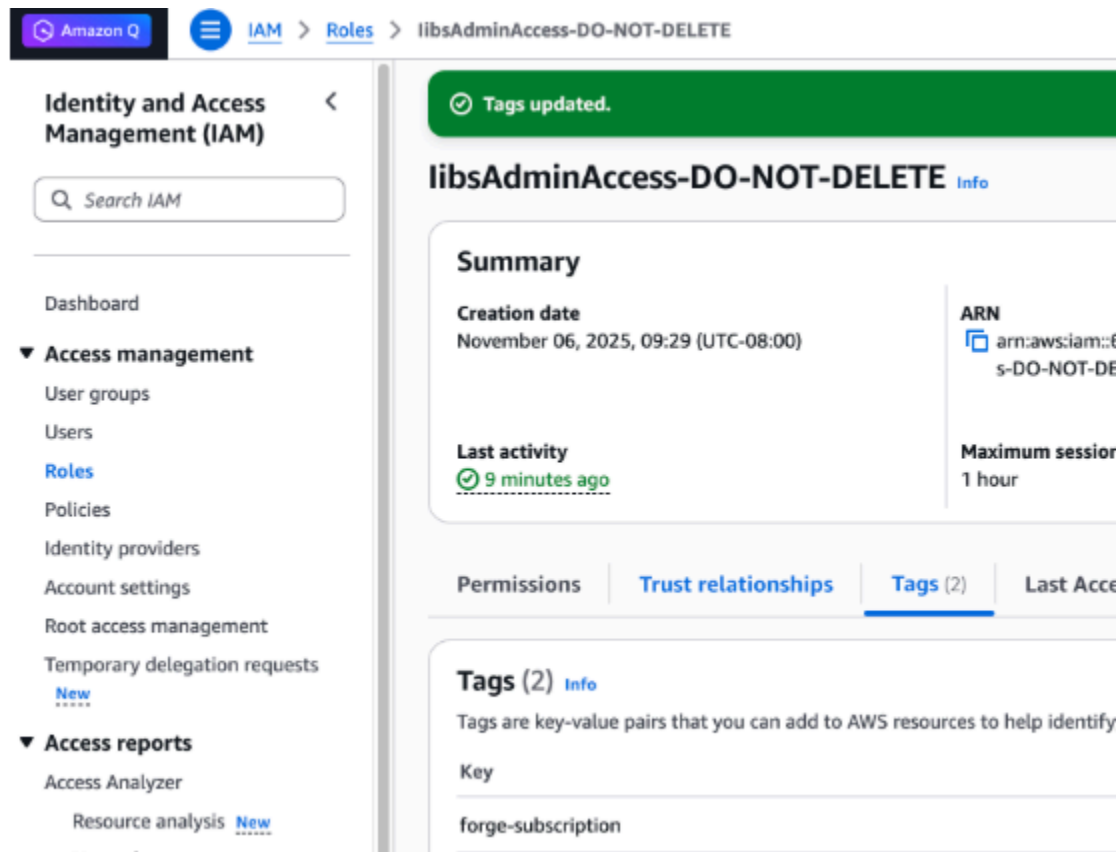
- 選取 <AssumedRoleToUse> (例如 libsAdminAccess)。按一下標籤索引標籤。



3. 按一下管理索引標籤。新增標籤。在金鑰下輸入「forge-subscription」，然後按一下儲存變更

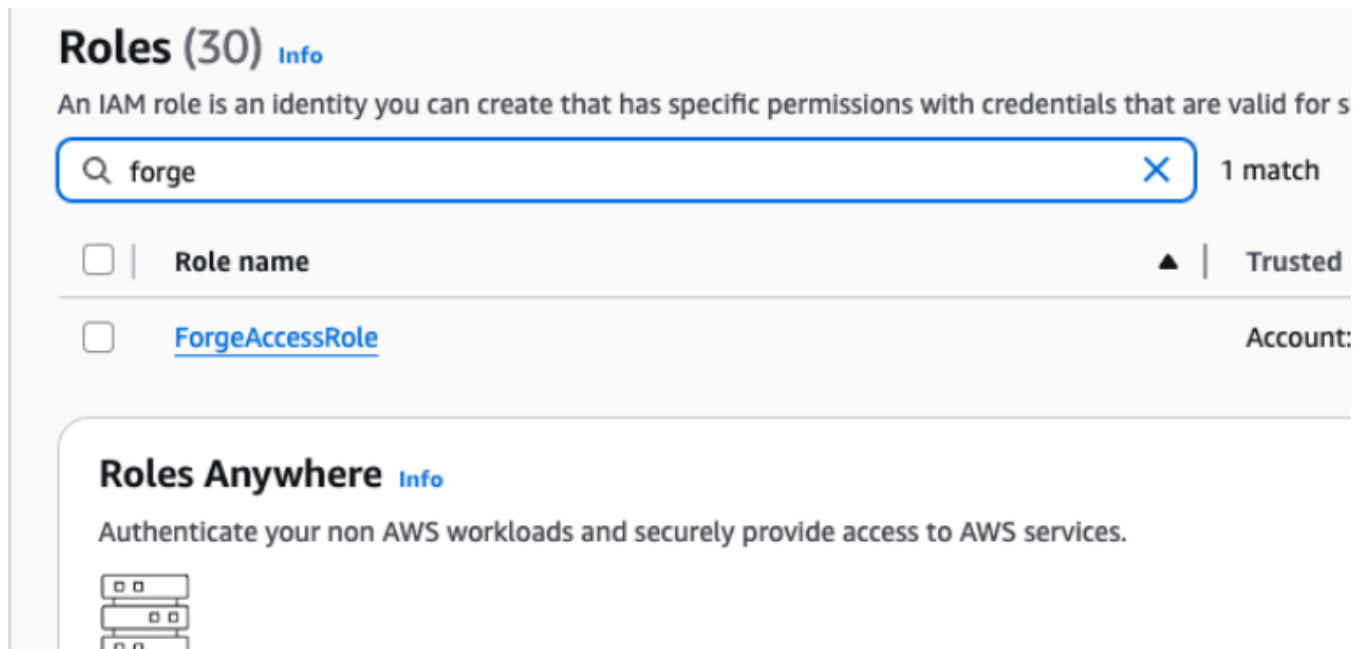


4. 確保您在標籤區段中將 forge-subscription 視為索引鍵

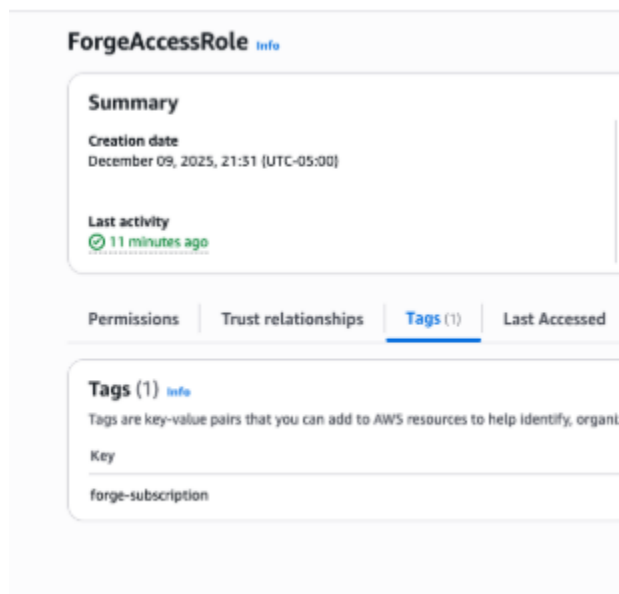


附錄 B：依管理員將 ListRoleTags 和 ListAttachedPolicies 政策新增至非管理員角色的 Amazon Web Services 帳戶

1. 前往 Amazon Web Services IAM 儀表板。按一下左側的角色。搜尋 <AssumedRoleToUse> (例如 ForgeAccessRole)，然後按一下 <AssumedRoleToUse> (例如 ForgeAccessRole) 角色



2. 按一下 <AssumedRoleToUse> (例如 ForgeAccessRole) 角色，然後選取標籤。新增類型為 "forge-subscription" 的新標籤



3. 在許可下，新增許可：新增許可 → 建立內嵌政策 → 新增下列政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```

        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": [
            "iam:ListRoleTags",
            "iam:ListAttachedRolePolicies"
        ],
        "Resource": "*"
    }
]
}

```

ForgeAccessRole [Info](#)

Summary

Creation date December 09, 2025, 21:31 (UTC-05:00)	ARN arn:aws:iam::i
Last activity 15 minutes ago	Maximum session 1 hour

[Permissions](#) |
 [Trust relationships](#) |
 [Tags \(1\)](#) |
 [Last Accessed](#) |
 [Revoke sessi](#)

Permissions policies (2) [Info](#)

You can attach up to 10 managed policies.

Search

- ☐ Policy name ↗
- ☐ AmazonSageMakerFullAccess
- ☐ forge-access-role

forge-access-role

```

1- {
2-   "Version": "2012-10-17",
3-   "Statement": [
4-     {
5-       "Sid": "VisualEditor0",
6-       "Effect": "Allow",
7-       "Action": [
8-         "iam:ListRoleTags",
9-         "iam:ListAttachedRolePolicies"
10-      ],
11-       "Resource": "*"
12-     }
13-   ]
14- }
    
```

步驟 2. 設定 HyperPod 基礎設施

遵循使用啟用 Forge 功能設定環境的[研討會指示](#)來設定必要的基礎設施。

內容管制設定

如果您需要存取 Nova Forge，Amazon Nova Lite 1.0 和 Pro 1.0 模型可使用可自訂的內容管制設定 (CCMS)。CCMS 允許調整內容管制控制，以符合特定業務需求，同時維護基本的負責任 AI 防護措施。若要判斷商業模型是否適合 CCMS，請聯絡 AWS 客戶經理。

如需搭配自訂模型設定和使用 CCMS 的詳細資訊，請參閱[負責任的 AI Toolkit 和內容管制一節](#)。

持續訓練前和中訓練

Note

訂閱後提供詳細文件

Nova Forge CPT 提供超越標準 CPT 的進階功能，包括存取中繼檢查點，以及與 Nova 訓練前 corpus 混合的資料。這些功能可更有效率地調整網域，並更好地保留模型的一般功能。

什麼是中繼檢查點？為什麼需要中繼檢查點？

中繼檢查點是在模型達到其最終生產就緒狀態之前，儲存在訓練前不同階段的 Amazon Nova 模型快照。在模型開發期間，Amazon Nova 會經歷多個訓練階段：具有持續學習率的初始預先訓練、學習率逐步下降、內容延伸訓練，以及最後的指示遵循一致性和安全性訓練。對於 CPT，中繼檢查點通常比最終產品檢查點更喜歡，因為它們更可整形且接受網域適應。Prod 檢查點經過廣泛的指示遵循一致性和安全性訓練，可最佳化模型以供一般對話使用，但可在 CPT 期間抵禦學習新的領域特定模式。相反地，部分和完全預先訓練的純文字檢查點會保留模型的預先訓練特性。它們尚未嚴重轉向特定行為，使它們更有效率地開始適應網域。執行大規模 CPT (>10B 權杖) 時，從中繼檢查點開始，通常會導致收斂速度更快、訓練穩定性更好，以及更有效地取得領域知識。不過，對於小規模 CPT (<10B 權杖) 或需要保留指示遵循功能時，Prod 檢查點可能更適合，因為它允許網域適應，同時維持模型的對話能力。

CPT 需要多個中繼檢查點，因為它們提供不同層級的模型可塑性，影響模型吸收新領域知識的效率。最終產品檢查點經過廣泛的指示遵循一致性和安全性訓練，可將其最佳化以供一般對話使用，但可抵禦學習新的特定領域模式。換句話說，它已透過訓練後強化。相反地，較舊的檢查點會保留模型的預先訓練特性，並且尚未嚴重轉向特定行為，使它們更具可塑性且接受網域適應性。

為了獲得最佳訓練效率，提供了多個中繼檢查點。

有哪些檢查點可用？

新版本 2.0

有三個 Amazon Nova Lite 2.0 檢查點。

- PRE-TRAINED - 【nova-lite-2/pretraining-text-RD】：這是 Amazon Nova 預先訓練的持續學習率和漸進階段之後的檢查點，其中模型會在數兆個字符上進行訓練。
- MID-TRAINED - 【nova-lite-2/pretraining-text-CE】：此檢查點允許以比預先訓練更保守的學習速率引入中繼量的非結構化資料，吸收特定領域的知識，同時避免災難性忘記。
- POST-TRAINED - 【nova-lite-2/prod】：這是模型完全對齊的最終檢查點，已經過所有相關和訓練後步驟。

下表詳細說明訓練前和中訓練的不同條件。

資料類型	執行	使用檢查點
大規模的非結構化原始網域資料（文件、日誌、文章、程式碼等）	持續預先訓練	預先訓練
大規模的非結構化原始網域資料（文件、日誌、文章、程式碼等）	中階訓練	預先訓練
少量的非結構化原始資料。結構化推理追蹤/CoT 資料	中階訓練	中階訓練
結構化示範（高品質輸入輸出對、策劃的任務指示、多迴轉對話）	完整微調	中階訓練
結構化示範（高品質輸入輸出對、策劃的任務指示、多迴轉對話）	參數高效率微調	訓練後

要使用哪個檢查點？

部分預先訓練的純文字和完全預先訓練的純文字檢查點通常會收斂得更快，且需要較少的訓練步驟才能進行網域調整。不過，他們沒有指令調校，需要經過訓練後步驟，才能執行有用的任務並遵循指示。GA 檢查點可能需要更多步驟來調整，但為小規模實驗提供更安全的起點，即使經過 CPT 訓練，也會在訓練功能後維持其中一部分。

一般而言，對於大型訓練資料集 (>10B 權杖)，從僅部分預先訓練的文字或僅完全預先訓練的文字檢查點開始，以獲得更有效率且穩定的訓練，因為模型的知識庫將會大幅修改。使用小型資料集 (<10B 字元)，使用 GA 檢查點來保留指示遵循功能，同時適應網域。

如何使用 Nova 2.0 的資料混合？

使用新的網域資料執行 CPT 時，將新資料與先前在模型訓練前階段使用的一些資料混合使用會非常有益。將舊資料與新網域資料混合可解決兩個問題：

- 忘記控制：保留模型的現有技能和知識，以防止災難性忘記。如果沒有資料混合，僅在窄域資料上進行訓練會導致模型覆寫一般功能。例如，僅根據法律文件訓練的模型可能會失去其編寫程式碼或執行數學的能力。混合一般網域資料集可在取得新網域時保留這些一般技能。
- 最佳化穩定性：透過錨定模型的內部表示來維持訓練穩定性。在 CPT 期間，修改模型學習到的功能，資料混合提供來自各種來源的漸層，可順暢地引導此適應。如果沒有它，在窄分佈上進行訓練可能會導致梯度不穩定，其中模型的表示式移動太大，導致訓練差異、損失尖峰或現有功能摺疊。這是穩定性-持久性權衡：模型應足夠可整形，以學習新的領域知識，但足夠穩定，不會破壞其已知的內容。

新 CPT 資料混合功能

存取 Amazon Nova 預先訓練資料和檢查點是 Amazon Nova CPT 自訂的核心產品之一。Amazon Nova CPT 自訂可讓您輕鬆地將網域資料與 Amazon Nova 的預先訓練 corpus 混合。此外，可以變更特定 Amazon Nova 資料類別（例如程式碼、數學、推理等）的取樣率，以及其控制成補充網域資料的比例。這允許強化與使用案例一致的功能，同時將模型調整為特定網域。

尋找最佳混合比率

Amazon Nova 資料與網域資料的最佳比率取決於資料集的網域、複雜性、大小、品質，以及維護一般功能的重要性。此比率必須透過實驗探索。決定要混合多少 Amazon Nova 資料的實驗架構如下。

選取代表性的網域資料子集（例如 5B 權杖），並在所有實驗執行中保持此常數。

執行小規模 CPT 實驗只會改變混合於以下項目的 Amazon Nova 資料量：

- 不混合：100% 網域 → 僅限 5B 網域（總計 5B）
- 輕度混合：90% 網域 → 5B 網域 + ~0.56B Amazon Nova（總計 ~5.56B）
- 中等混合：70% 網域 → 5B 網域 + ~2.14B Amazon Nova（總計 ~7.14B）
- 重度混合：50% 網域 → 5B 網域 + 5B Amazon Nova（總計 10B）

在網域和一般網域基準中評估 上的每個檢查點。也請在任何訓練之前評估起始檢查點 (Amazon Nova 檢查點)。

- 客戶網域效能在執行期間是否大致保持不變？通常應該，因為每次執行都會看到相同數量的網域字符。如果網域效能隨著更多混合而改善，Amazon Nova 資料會提供有用的正規化。
- 隨著混合增加，一般基準分數是否會改善？
 - 預期的行為是，隨著新增更多 Amazon Nova 資料，一般功能應該會單調改善。
 - 測量多個一般基準：MMLU（一般知識）、HumanEval（編碼）、GSM8K（數學）或感興趣的特定基準。
- 選取維持網域效能的混合比率，同時為使用案例提供可接受的一般功能。將更多資料混合納入額外的訓練成本。

找出最佳混合比率後，請使用具有所選混合比率的完整網域資料集來執行完整規模的 CPT。

解析資料混合類別

在下面，我們剖析資料混合中的每個可用類別，以便您最佳地決定哪些資料類別最適用於您的整體資料混合。

如何啟用資料混合

使用跨資料集類別的適當百分比分佈，將 `data_mixing` 區段新增至您的配方。nova_data 百分比的總和必須為 100。

Nova 2.0 組態與資料混合

```
# Note:
# This recipe can run on p5.48xlarge

# Run config
display_name: "Nova Lite Pretrain on P5 GPU"
versions: ["2.0"]
```

```

instance_types: ["ml.p5.48xlarge"]

run:
  name: "my-cpt-run"      # A descriptive name for your training job
  model_type: "amazon.nova-2-lite-v1:0:256k" # Model variant specification, do not
change
  model_name_or_path: "nova-lite-2/prod" # Base model path, do not change
  replicas: 8             # Number of compute instances for training, allowed values are 4,
8, 16, 32
  data_s3_path: ""       # Customer data paths
  validation_data_s3_path: "" # Customer validation data paths
  output_s3_path: ""     # Output artifact path, SageMaker HyperPod job-specific
configuration - not compatible with standard SageMaker Training jobs

## Training specific configs
training_config:
  task_type: cpt
  max_length: 8192          # Maximum context window size (tokens)
  global_batch_size: 64     # Global batch size, allowed values are 32, 64, 128,
256.

  trainer:
    max_steps: 10           # The number of training steps to run total
    val_check_interval: 10  # The number of steps between running validation
    limit_val_batches: 2    # Batches of the validation set to use each trigger

  model:
    hidden_dropout: 0.0     # Dropout for hidden states, must be between 0.0 and
1.0
    attention_dropout: 0.0  # Dropout for attention weights, must be between 0.0
and 1.0

  optim:
    optimizer: adam
    lr: 1e-5                # Learning rate
    name: distributed_fused_adam # Optimizer algorithm, do not change
    adam_w_mode: true       # Enable AdamW mode
    eps: 1e-06              # Epsilon for numerical stability
    weight_decay: 0.0       # L2 regularization strength, must be between 0.0 and
1.0
    adam_beta1: 0.9         # Beta1 for Adam optimizer
    adam_beta2: 0.95        # Beta2 for Adam optimizer
    sched:
      warmup_steps: 10      # Learning rate warmup steps

```

```

    constant_steps: 0          # Steps at constant learning rate
    min_lr: 1e-6              # Minimum learning rate, must be lower than lr

data_mixing:
  dataset_catalog: cpt_text_lite
  sources:
    nova_data:  # percent inputs for Nova data must sum to 100%; use 0% if you want to
    exclude a data grouping
      agents: 20
      business-and-finance: 4
      scientific: 10
      code: 5
      factual-and-news: 5
      longform-text: 6
      health-and-medicine: 1
      humanities-and-education: 1
      legal: 1
      math: 9
      additional-languages: 15
      social-and-personal-interest: 11
      entertainment: 0.5
      reasoning: 10
      other: 0.5
      tables: 1
    customer_data: # percent input of customer data. 100 = use only customer data, 0 =
    use only the nova_data mix above
      percent: 25

```

這些類別的意義

注意：Nova 2.0 reasoning-code 包含無法在 Nova 1.0 中使用的其他推理特定類別（例如 reasoning-math、reasoning-instruction-following）。

類別和資訊標籤摘要：

類別名稱	資訊詳細資訊
agents	在 AI 系統中專注於自動決策、任務完成和目標導向行為的訓練資料
baseline	專注於一般理解、基本溝通和核心語言功能的基本語言資料
chat	顯示自然對話流程、內容維護和適當社交互動的對話交流

類別名稱	資訊詳細資訊
code	來自各種程式設計語言和平台的程式設計原始程式碼、文件 and 技術討論。
factuality	著重於準確性、來源驗證和事實評估的參考資料和已驗證資訊
identity	專注於一致角色特徵、值和互動風格的人格架構和行為模式
long-context	延伸文字和複雜的敘述，著重於在冗長的交換之間維持一致性和相關性
math	數學內容，包括教科書、問題、解決方案和數學討論。
rai	強調道德 AI 原則、安全考量和負責任技術部署的案例和案例
instruction-following	根據不同層級的使用者提示和指示詞，精確執行任務的範例
stem	涵蓋科學、技術、工程和數學的技術內容，包括解決問題和理論概念
planning	顯示策略思維、step-by-step 任務明細和高效資源分配的序列
reasoning-chat	專注於邏輯討論和結構化對話流程的分析對話案例
reasoning-code	專注於系統化解決方案開發的程式設計挑戰和演算法問題
reasoning-factuality	著重於關鍵評估和驗證程序的資訊評估案例
reasoning-instruction-following	專注於系統化解譯和系統化執行的複雜任務分析
reasoning-math	專注於邏輯進展和解決方案策略的數學問題解決案例
reasoning-planning	專注於實現目標的系統性方法的策略決策案例
reasoning-rag	著重於情境理解和相關應用程式的資訊擷取和合成案例
reasoning-rai	專注於系統性評估 AI 安全和公平性的道德決策案例

類別名稱	資訊詳細資訊
reasoning-stem	專注於系統分析和解決方案開發的科學問題解決案例
rag	有效結合擷取的外部知識與產生的回應，以提供準確、情境式資訊的範例
translation	多語言內容對顯示準確的翻譯，同時保留內容、色調和文化細微差別

參數指南

- `dataset_catalog`：目前唯一值是 `cpt_text_lite`，直到我們啟用多模式訓練為止。
- `nova_data`：混合時 Nova 資料的個別類別百分比。它們應該加到 1.0。
- `customer_data`：混入 Nova 資料的客戶資料百分比。

訓練中使用的字符總數可從 $\text{max\_length} * \text{global\_batch\_size} * \text{計算 max\_steps}$

限制

目前的 CPT 僅支援文字資料，不支援任何客戶多模態資料集。

受監督的微調

簡介

受監督的微調會將資料集與輸入輸出對用於感興趣的任務。換句話說，您會提供提示（問題、指示等）的範例，以及正確或所需的回應，並繼續在這些回應上訓練模型。模型的權重經過調整，可將受監督損失降至最低，通常是其預測和目標回應權杖之間的交叉熵。

何時使用 SFT？

當您有明確定義且具有明確所需輸出的任務時，SFT 是最佳的。如果您可以明確說出「給予 X 輸入，正確/想要的輸出是 Y」，而且您可以收集這類 X-Y 映射的範例，則監督式微調是很好的選擇。SFT Excel 的一些案例包括：

- 結構化或複雜的分類任務：例如，將內部文件或合約分類為許多自訂類別。使用 SFT，模型可以學習這些特定類別遠優於單獨提示。
- 具有已知答案的問題回答或轉換任務：例如，微調模型以回答公司知識庫的問題，或在格式之間轉換資料，其中每個輸入都有正確的回應。

- **格式和樣式一致性：**如果您需要模型一律以特定格式或色調回應，您可以微調正確格式/色調的範例。例如，針對顯示特定品牌語音或樣式的提示回應對進行訓練，可以在其輸出中教導該樣式的模型。指示追蹤行為通常透過 SFT 最初針對精心策劃的良好助理行為範例進行教學。

當您可以指定正確的行為時，SFT 是教導 LLM 新技能或行為的最直接方式。它利用模型現有的語言理解，並專注於您的任務。當差距是知識而非行為時，請勿使用 SFT；它不會讓模型學習新的事實、術語或最近的事件。在這些情況下，偏好在大型網域內 corpora 或擷取擴增產生的持續預先訓練，以在推論時帶來外部知識。當您可以測量品質但無法標記單一右答案時，使用可驗證的獎勵或 LLM-as-judge 進行微調可能較 SFT 為佳。

根據 Nova 模型的任務複雜性和效能，無需調校，每個任務規劃數千到數萬個示範，資料品質、一致性和多樣性比原始磁碟區更重要。

何時使用參數效率和何時使用完整排名 SFT？

Nova 自訂配方可讓您執行參數效率，特別是 LoRA 或完整排名 SFT。如果您想要直接、具成本效益的模型更新，或資料很少，則偏好參數效率方法，因此您可以訓練小型轉接器，同時讓大部分骨幹保持不變（完整排名 SFT 更新所有模型參數）。

SFT 的資料混合

資料混合可讓您將自訂訓練資料集與 Nova 的專屬訓練資料結合。此功能適用於 Nova 1.0 和 Nova 2.0 模型。

Nova 專屬資料類型：Nova 同時支援文字和多模式 SFT 資料類型。它被組織成多個資料類別，每個類別都包含與對應類別相關的任務混合。

Nova 專屬資料類別：文字資料集包含數個類別，包括：自動決策、任務完成、目標導向資料集（代理程式）、推理和非理性的精確任務執行資料集 (reasoning-instruction-following、指示遵循)、展現策略思維的序列和 step-by-step 任務明細（規劃）、負責任的 AI (rai)、長內容、事實、數學、莖等。同樣地，多模態資料集包含影片、螢幕擷取畫面、圖表等。

資料混合功能可讓您將自己的微調訓練範例與用於微調 Nova 的 Nova 資料集範例混合。這可以防止過度擬合您的自訂訓練和 Nova 功能的「災難性忘記」，或在從新的預先訓練檢查點訓練時協助您建置功能。

若要混合 Nova 資料，您只需在 training_config 區段下，將 data_mixing 區塊新增至配方 YAML 檔案。文字和多模態資料混合區塊具有不同的內容。請參閱對應的配方。

支援模型

- Nova 2.0 Lite

支援的模態

- 文字
- 多模態

YAML 組態範例

Nova 2.0 組態範例

```
run:
  name: my-lora-sft-run
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: nova-lite-2/prod
  data_s3_path: s3://my-bucket-name/train.jsonl
  replicas: 4
  output_s3_path: s3://my-bucket-name/outputs/
  mlflow_tracking_uri: ""
  mlflow_experiment_name: "my-lora-sft-experiment"
  mlflow_run_name: "my-lora-sft-run"

training_config:
  max_steps: 100
  save_steps: 10
  save_top_k: 5
  max_length: 32768
  global_batch_size: 32
  reasoning_enabled: true
  lr_scheduler:
    warmup_steps: 15
    min_lr: 1e-6
  optim_config:
    lr: 1e-5
    weight_decay: 0.0
    adam_beta1: 0.9
    adam_beta2: 0.95
  peft:
```

```
peft_scheme: "lora"
lora_tuning:
  alpha: 64
  lora_plus_lr_ratio: 64.0
```

Nova 2.0 文字資料混合

```
data_mixing:
  dataset_catalog: sft_1p5_text_chat
  sources:
    customer_data:
      percent: 50
    nova_data:
      agents: 1
      baseline: 10
      chat: 0.5
      code: 10
      factuality: 0.1
      identity: 1
      long-context: 1
      math: 2
      rai: 1
      instruction-following: 13
      stem: 0.5
      planning: 10
      reasoning-chat: 0.5
      reasoning-code: 0.5
      reasoning-factuality: 0.5
      reasoning-instruction-following: 45
      reasoning-math: 0.5
      reasoning-planning: 0.5
      reasoning-rag: 0.4
      reasoning-rai: 0.5
      reasoning-stem: 0.4
      rag: 1
      translation: 0.1
```

Nova 2.0 多模態資料混合

```
data_mixing:
```

```
dataset_catalog: sft_1p5_mm_chat
sources:
  customer_data:
    percent: 50
  nova_data:
    charts: 1
    chat: 38
    code: 20
    docs: 3
    general: 2
    grounding: 1
    rag: 4
    screenshot: 4
    text: 8
    translation: 4
    video: 15
```

模型檢查點

Nova 2.0 檢查點

- 預先訓練 **【nova-lite-2/pretraining-text-RD】**：在持續學習率和漸進式階段之後的檢查點，其中模型是以數兆個字符進行訓練。【第 2 階段的結果】
- MID-TRAINED **【nova-lite-2/pretraining-text-CE】**：允許具有中繼非結構化資料量的客戶，以比預先訓練更保守的學習率介紹其資料，吸收特定領域的知識，同時避免災難性忘記。【階段 3 的結果】
- 最終 **【nova-lite-2/prod】**：已通過所有訓練前和訓練後步驟的完全對齊最終檢查點。【階段 4 的結果】

訓練階段：

- 階段 1：PT Ckpt，初始預先訓練，持續學習率
- 階段 2：PT Ckpt、學習率下降
- 階段 3：PT Ckpt、內容延伸訓練
- 階段 4：指示遵循一致性和安全性訓練

訓練方法

訓練方法選擇指南

資料類型	資料磁碟區	執行	使用檢查點
大規模的非結構化原始網域資料 (文件、日誌、文章、程式碼等)	1T+ 權杖	持續預先訓練	持續學習結束率 (CLR)
大規模的非結構化原始網域資料	100B+ 權杖	中階訓練	CLR 結束
非結構化原始資料的較小磁碟區；結構化推理追蹤/CoT 資料	1B+ 權杖	中階訓練	Nova 基礎模型
結構化示範 (高品質輸入輸出對、精心策劃的任務指示、多迴轉對話)	1K+ 範例	受管微調 (SFT)	Nova 基礎模型

開始之前的必要項目

- 我們假設您已使用具有作用中容量的限制執行個體群組 (RIG) 設定 SMHP 叢集。如果沒有，請參閱此處，讓您的 SMHP 叢集和 RIG 設定完成 [【文件連結、研討會連結】](#)
- 您需要 p5.48xlarge EC2 執行個體才能執行此配方。有效執行此配方所需的執行個體數量下限如下：
 - Nova Lite 2.0 - 4 p5.48xlarge
- 使用[此處](#)提供的指示安裝 Forge Specific SageMaker HyperPod CLI
- 確認您可以使用 連線至叢集 hyperpod get-clusters
 - 請注意，此命令會列出您帳戶中的所有 SMHP 叢集
- 確認您的訓練和選擇性驗證資料可在 SMHP 叢集的執行角色可存取的 S3 儲存貯體中使用。如需資料準備，請參閱下一節。
- 完成設定。如果您尚未完成設定，請遵循下列[指南](#)。
- 驗證：完成設定後，請確認您可以在命令下成功執行

```
aws sagemaker describe-cluster --cluster-name <cluster-name> --region <region>

hyperpod connect-cluster --cluster-name cluster-name
```

實現成功 SFT 的系統性方法

- 資料準備：遵循已建立的準則，將資料集建立、清除或重新格式化為所需的結構。確保輸入、輸出和輔助資訊（例如推理追蹤或中繼資料）正確對齊並格式化。
- 訓練組態：定義模型的訓練方式。使用 Amazon SageMaker HyperPod 時，此組態會寫入 YAML 配置文件檔案中，其中包含：
 - 資料來源路徑（訓練和驗證資料集）
 - 金鑰超參數（訓練步驟的數量、學習率、批次大小）
 - 選用元件（分散式訓練參數等）
 - 資料混合設定（定義客戶和 Nova 資料類別的比例）
- 最佳化 SFT 超參數：我們建議的 SFT 配方參數值是很好的起點和強大的選擇。如果您想要針對您的使用案例進一步最佳化它們，請使用不同的參數組合執行多個 SFT 執行，並挑選最佳的執行。您可以依照您選擇的超參數最佳化方法選取參數組合。一個簡單的方法是改變一個參數的值（預設值*0.5、預設值、預設值*2），同時保留其他參數的其他預設值，針對您要最佳化的每個參數重複此操作，並視需要反覆執行。LoRA 最相關的參數是學習率、alpha（擴展參數）、訓練和暖機步驟的 epoch 數目；對於完整排名，主要是學習率、epoch 數目和暖機步驟。

實驗定序和資料混合

- 如果您只有一組任務的 SFT 資料 (train/dev/test)，並且只關心這些任務的測試效能
 - 在不混合【最終】Nova 檢查點的情況下執行 SFT。使用預設 SFT 超參數，並選擇性地針對您的使用案例進行最佳化。監控驗證指標和/或評估較大資料集的中繼檢查點。
- 如果您只有一組任務的 SFT 資料 (train/dev/test)，並關心感興趣的領域中這些任務和一般基準測試的效能
 - 首先在訓練前檢查點 (PRE-TRAINED 或 MID-TRAINED 檢查點，而非 FINAL) 上使用 Nova 資料混合 SFT。使用中繼檢查點可讓模型更好地整合您的自訂資料與 Nova 的專屬資料，同時維持強大的一般功能。

- 執行較短的 SFT 訓練執行時，混合了不同數量的 Nova 資料（例如 10%、25%、50%、75%）和 Nova 資料類別選擇，以補充您的使用案例（例如，如果您關心以下功能的一般指示，請在類別後挑選指示）。監控驗證指標並評估混合是否有助於一般基準的效能。選取可在任務和一般效能上獲得最佳效能組合的訓練組合和檢查點。根據使用案例，可以使用強化微調 (RFT) 進一步改善任務和一般效能。

準備 SFT 的資料集

Nova 2.0：使用 Converse API 格式 <https://docs.aws.amazon.com/bedrock/latest/userguide/conversation-inference-call.html>。Nova 2.0 資料格式可包含其他推理欄位：https://docs.aws.amazon.com/bedrock/latest/APIReference/API_runtime_ReasoningContentBlock.html

推理內容會先擷取模型的中繼思考步驟，再產生最終答案。assistant 然後，使用 reasoningContent 欄位來包含推理追蹤。使用純文字推理內容、避免標記標籤，例如 <thinking>和 </thinking>除非任務特別要求，並確保推理內容清晰且與問題解決程序相關。

評估方法

先決條件

- 來自訓練任務manifest.json檔案的檢查點 S3 URI（適用於訓練模型）
- 以正確格式上傳至 S3 的評估資料集
- 評估結果的輸出 S3 路徑

現成基準：使用現成基準驗證一般任務的效能。如需詳細資訊，請參閱：<https://docs.aws.amazon.com/sagemaker/latest/dg/nova-hp-evaluate.html>

使用您自己的資料

您也可以提供自訂資料，方法是將它們格式化為以下格式，然後視需要使用下列容器來取得推論結果以及校正的日誌機率。

使用下列結構為每個任務建立 jsonl：

```
{  
  "metadata": "{key:4, category:'apple'}",
```

```

"system": "arithmetic-patterns, please answer the following with no other words: ",
"query": "What is the next number in this series? 1, 2, 4, 8, 16, ?",
"response": "32"
}

```

在評估任務的推論階段產生的輸出將具有下列結構：

```

{
  "prompt": "[{'role': 'system', 'content': 'arithmetic-patterns, please answer the following with no other words: '}, {'role': 'user', 'content': 'What is the next number in this series? 1, 2, 4, 8, 16, ?'}]",
  "inference": "['32']",
  "gold": "32",
  "metadata": "{key:4, category:'apple'}"
}

```

欄位描述：

- prompt：傳送至模型的格式化輸入
- inference：模型產生的回應
- gold：來自輸入資料集的預期正確答案、來自輸入的回應欄位
- metadata：從輸入傳遞的選用中繼資料

準備評估組態

啟動評估任務的命令。使用 "--override-parameters" 修改配方中的任何項目。

```

hyperpod start-job -n kubeflow \
  --recipe evaluation/nova/nova_micro_p5_48x1_bring_your_own_dataset_eval \
  --override-parameters '{
    "instance_type": "p5.48xlarge",
    "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-latest",
    "recipes.run.name": "<your-eval-job-name>",
    "recipes.run.model_name_or_path": "<checkpoint-s3-uri>",
    "recipes.run.output_s3_path": "s3://<your-bucket>/eval-results/"
  }'

```

```
"recipes.run.data_s3_path": "s3://<your-bucket>/eval-data.jsonl"
}'
```

最佳實務

- 優先考慮磁碟區的資料品質：高品質、多樣化和代表性的訓練資料比大量低品質資料更有價值。
- 包含 reasoning-instruction-following 類別：使用資料混合時，請包含「reasoning-instruction-following」類別，以維持跨任務的強大一般效能。
- 使用預設學習率：從預設學習率 (LoRA 為 1e-5、完整排名 SFT 為 5e-6) 開始，並僅在需要時根據驗證指標進行調整。
- Balance Nova 資料混合：混合最大 50% 的新資料，以獲得最佳延遲效能平衡。較高的百分比可以改善一般功能，但可以增加訓練時間。
- 監控驗證指標：定期在訓練期間評估中繼檢查點，以提早偵測過度擬合或效能降低。
- 對代表性資料集進行測試：確保您的評估資料集準確代表您的生產使用案例，以進行有意義的效能評估。

準備訓練任務組態

Hyper 參數

資料混合以外的完整超參數集：

```
## Run config
run:
  name: my-lora-sft-run
  model_type: amazon.nova-2-lite-v1:0:256k
  model_name_or_path: nova-lite-2/prod
  data_s3_path: s3://my-bucket-name/train.jsonl # SageMaker HyperPod (SMHP) only and
not compatible with SageMaker Training jobs. Note replace my-bucket-name with your
real bucket name for SMHP job
  replicas: 4 # Number of compute instances for training, allowed
values are 4, 8, 16, 32
  output_s3_path: s3://my-bucket-name/outputs/ # Output artifact path
(Hyperpod job-specific; not compatible with standard SageMaker Training jobs). Note
replace my-bucket-name with your real bucket name for SMHP job
```



```

## MLFlow configs
mlflow_tracking_uri: "" # Required for MLFlow
mlflow_experiment_name: "my-lora-sft-experiment" # Optional for MLFlow. Note: leave
this field non-empty
mlflow_run_name: "my-lora-sft-run" # Optional for MLFlow. Note: leave this field non-
empty

training_config:
    max_steps: 100 # Maximum training steps. Minimal is 4.
    save_steps: 10 # This parameter suggests after how many training steps the
checkpoints will be saved. Should be less than or equal to max_steps(please override
this value with a numerical value equal or less than max_steps value; min: 4)
    save_top_k: 5 # Keep top K best checkpoints. Note supported only
for SageMaker HyperPod jobs. Minimal is 1.
    max_length: 32768 # Sequence length (options: 8192, 16384, 32768
[default], 65536)
    global_batch_size: 32 # Global batch size (options: 32, 64, 128)
    reasoning_enabled: true # If data has reasoningContent, set to true;
otherwise False

    lr_scheduler:
        warmup_steps: 15 # Learning rate warmup steps. Recommend 15% of
max_steps
        min_lr: 1e-6 # Minimum learning rate, must be between 0.0 and 1.0

    optim_config: # Optimizer settings
        lr: 1e-5 # Learning rate, must be between 0.0 and 1.0
        weight_decay: 0.0 # L2 regularization strength, must be between 0.0
and 1.0
        adam_beta1: 0.9 # Exponential decay rate for first-moment estimates,
must be between 0.0 and 1.0
        adam_beta2: 0.95 # Exponential decay rate for second-moment
estimates, must be between 0.0 and 1.0

    peft: # Parameter-efficient fine-tuning (LoRA)
        peft_scheme: "lora" # Enable LoRA for PEFT
        lora_tuning:
            alpha: 64 # Scaling factor for LoRA weights ( options: 32, 64,
96, 128, 160, 192),
            lora_plus_lr_ratio: 64.0 # LoRA+ learning rate scaling factor (0.0-100.0)

```

LoRA 最相關的參數是學習率、alpha (擴展參數)、訓練和暖機步驟的 epoch 數目；對於完整排名，主要是學習率、epoch 數目和暖機步驟。配方會預先填入建議的預設值。

設定資料混合區塊

使用跨資料集類別的適當百分比分佈，將 data_mixing 區段新增至您的配方。

以下說明每個可用的 Nova 資料類別。

Nova 2.0 組態與資料混合

```
data_mixing:
  dataset_catalog: sft_1p5_text_chat      # Nova text dataset catalog
  sources:
    customer_data:
      percent: 50                        # Percent of overall mix to draw from customer data
    nova_data:                          # The remainder will be drawn from Nova data. The
categories below must add to 100
  agents: 1                             # autonomous decision-making, task completion, goal-
oriented behavior in AI systems
  baseline: 10                          # [New in Nova 1.5]
  chat: 0.5                             # Conversational exchanges demonstrating natural
dialogue flow
  code: 10                              # Programming examples and solutions spanning
multiple languages
  factuality: 0.1                       # [New in Nova 1.5]
  identity: 1                           # [New in Nova 1.5]
  long-context: 1                       # [New in Nova 1.5]
  math: 2                               # [New in Nova 1.5]
  rai: 1                                # ethical AI principles, safety considerations, and
responsible technology deployment
  instruction-following: 13             # precise task execution based on varying levels of
user prompts and directives
  stem: 0.5                             # Technical content covering science, technology,
engineering, and mathematics
  planning: 10                          # Sequences demonstrating strategic thinking and
step-by-step task breakdown
  reasoning-chat: 0.5
  reasoning-code: 0.5
  reasoning-factuality: 0.5
  reasoning-instruction-following: 45
  reasoning-math: 0.5
  reasoning-planning: 0.5
  reasoning-rag: 0.4
  reasoning-rai: 0.5
```

```
reasoning-stem: 0.4
rag: 1 # combining retrieved external knowledge with
generated responses
translation: 0.1
```

這些類別代表什麼意思？

Nova 2.0 文字資料類別

類別名稱	資訊詳細資訊
代理程式	在 AI 系統中專注於自動決策、任務完成和目標導向行為的訓練資料
基準	專注於一般理解、基本溝通和核心語言功能的基本語言資料
chat	顯示自然對話流程、內容維護和適當社交互動的對話交流
code	來自各種程式設計語言和平台的程式設計原始程式碼、文件和技术討論。
事實	著重於準確性、來源驗證和事實評估的參考資料和已驗證資訊
身分	專注於一致角色特徵、價值和互動風格的人格架構和行為模式
長內容	延伸文字和複雜的敘述，著重於在冗長的交換之間維持一致性和相關性
數學運算	數學內容，包括教科書、問題、解決方案和數學討論。
rai	強調道德 AI 原則、安全考量和負責任技術部署的案例和案例
instructions-flowing	根據不同層級的使用者提示和指示詞，精確執行任務的範例

類別名稱	資訊詳細資訊
主體	涵蓋科學、技術、工程和數學的技術內容，包括解決問題和理論概念
規劃	顯示策略思維、step-by-step任務明細和高效資源配置的序列
推理聊天	專注於邏輯討論和結構化對話流程的分析對話案例
推理程式碼	專注於系統化解決方案開發的程式設計挑戰和演算法問題
推理事實	著重於關鍵評估和驗證程序的資訊評估案例
reasoning-instruction-following	專注於系統化解譯和系統化執行的複雜任務分析
推理數學	專注於邏輯進展和解決方案策略的數學問題解決案例
推理規劃	專注於實現目標的系統性方法的策略決策案例
推理布	著重於情境理解和相關應用程式的資訊擷取和合成案例
reasoning-rai	專注於系統性評估 AI 安全和公平性的道德決策案例
reasoning-stem	專注於系統分析和解決方案開發的科學問題解決案例
布	有效地將擷取的外部知識與產生的回應結合，以提供準確、情境式資訊的範例
translation	多語言內容對顯示準確的翻譯，同時保留內容、色調和文化細微差別

多模式資料混合 (Nova 2.0)

```
data_mixing:
  dataset_catalog: sft_1p5_mm_chat          # Nova text dataset catalog
  sources:
    customer_data:
      percent: 50                          # Percent of overall mix to draw from customer data
    nova_data:                             # The remainder will be drawn from Nova data. The
categories below must add to 100
  charts: 1
  chat: 38
  code: 20
  docs: 3
  general: 2
  grounding: 1
  rag: 4
  screenshot: 4
  text: 8
  translation: 4
  video: 15
```

注意：Nova 2.0 包含 Nova 1.0 中不提供的影片資料類別支援。

這些類別代表什麼意思？

Nova 2.0 多模式資料類別

類別名稱	資訊詳細資訊
圖表	圖形、圓餅圖、長條圖、折線圖和其他統計視覺化的視覺化呈現和說明，以協助模型有效地了解和傳達量化資訊
chat	對話資料搭配專注於情境對話理解和以影像為基礎的互動的視覺元素
code	專注於視覺化程式碼解譯、IDE 螢幕擷取畫面和技術圖表的程式設計界面和開發環境

類別名稱	資訊詳細資訊
文件	文件中心資料結合文字、影像、配置和格式，以訓練模型了解和處理各種文件類型和結構，以協助處理 PDF 內容辨識等概念
一般	專注於廣泛理解影像、圖形和隨附的描述性文字的多樣化視覺文字內容
依據	視覺參考資料和標籤影像，著重於將語言概念連接到真實世界視覺呈現
布	多模式擷取範例，示範如何有效地結合和參考視覺和文字外部知識，以產生準確的情境式回應
螢幕擷取畫面	應用程式界面擷取和數位顯示影像，著重於了解軟體界面和數位互動
text	從純文字 SFT Nova 資料集類別建立的內容文字資料平衡集區，以提供一般功能
translation	專注於影像和文化視覺元素中文字的多語言解釋的跨語言視覺內容
影片	以動作為基礎的視覺內容，著重於時間理解和循序視覺敘述理解

如何啟動任務

如果您只需要取得啟動第一次 SFT 執行的基本詳細資訊，也可以參考 README。

容器資訊：

容器資訊和啟動命令

模型	技術	Subcategory	映像 URI	Hyperpod 啟動器命令
新版本 2.0	微調	SFT 文字	708977205387.dkr.ecr.us-	Hyperpod start-job \ -n kubeflow

模型	技術	Subcategory	映像 URI	Hyperpod 啟動器命令
			east-1.amazons.com/nova-fine-tune-repo:SM-HP-SFT-V2-latest	\ --recipe fine-tuning/nova/nova_2_0/nova_lite/SFT/nova_lite_2_0_p5_gpu_sft \ --override-parameters '{ "instance_type" : "ml.p5.48xlarge" , "container" : "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-latest" }'

模型	技術	Subcategory	映像 URI	Hyperpod 啟動器命令
新版本 2.0	微調	SFT 文字 + 資料混合	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-DATAMIXING-latest	Hyperpod start-job \ -n kubeflow \ --recipe fine-tuning/nova/forge/nova_2_0/nova_lite/SFT/nova_lite_2_0_p5_gpu_sft_t_ext_with_datamix \ --override-parameters '{ "instance_type" : "ml.p5.48xlarge" , "container" : "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-DATAMIXING-latest" }'

模型	技術	Subcategory	映像 URI	Hyperpod 啟動器命令
新版本 2.0	微調	SFT MM	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-latest	Hyperpod start-job \ -n kubeflow \ --recipe fine-tuning/nova/nova_2_0/nova_lite/SFT/nova_lite_2_0_p5_gpu_sft \ --override-parameters '{ "instance_type" : "ml.p5.48xlarge" , "container" : "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-latest" }'

模型	技術	Subcategory	映像 URI	Hyperpod 啟動器命令
新版本 2.0	微調	SFT MM + 資料混合	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-DATAMIXING-latest	Hyperpod start-job \ -n kubeflow \ --recipe fine-tuning/nova/forge/nova_2_0/nova_lite/SFT/nova_lite_2_0_p5_gpu_sft_mm_with_datamix \ --override-parameters '{ "instance_type" : "ml.p5.48xlarge" , "container" : "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-SFT-V2-DATAMIXING-latest" }'

完成所有設定後，從 sagemaker-hyperpod-cli 儲存庫的根目錄開始，導覽至預設 nova sft 配方資料夾

- cd /src/hyperpod_cli/sagemaker_hyperpod_recipes/recipes_collection/recipes/training/nova/
- 在這裡，您可以根據基本模型的選擇，選擇是否要執行 nova 1 還是 nova 2 配方。

對於 Nova 2.0 軸：

- 如果您想要使用一般 sft 任務，您應該可以在此資料夾下看到一個配方
 - `cd /src/hyperpod_cli/sagemaker_hyperpod_recipes/recipes_collection/recipes/fine-tuning/nova_2_0/nova_lite/SFT`，然後您應該能夠在此資料夾下看到一個名為 `nova_lite_2_0_p5x8_gpu_sft.yaml` 的配方
- 如果您想要使用資料混合 sft 任務，您可以導覽至 sft Forge 配方資料夾
 - `cd /src/hyperpod_cli/sagemaker_hyperpod_recipes/recipes_collection/recipes/fine-tuning/nova/forge/nova_2_0/nova_lite/SFT`，您應該能夠在此資料夾下看到一個名為 `nova_lite_2_0_p5x8_gpu_sft_with_datamix.yaml` 的配方：
- 編輯任務所需配方中的區段，例如名稱、`data_s3_path`、`validation_s3_path`、`put_s3_path` 和 `max_steps`。由於我們正在執行 sft，因此在此不適用 `epochs` 的概念。

資料混合組態看起來會相同，但底部會有類似這樣的額外資料混合區段

```
data_mixing:
  dataset_catalog: sft_text_lite
  sources:
    nova_data: # percent inputs for Nova data must sum to 100%; use 0% if you want to
exclude a data grouping
    agents: 20
    business-and-finance: 20
    scientific: 20
    code: 20
    factual-and-news: 20
    longform-text: 0
    health-and-medicine: 0
    humanities-and-education: 0
    legal: 0
    math: 0
    additional-languages: 0
    social-and-personal-interest: 0
    entertainment: 0
    reasoning: 0
    other: 0
    tables: 0
    customer_data: # percent input of customer data. 100 = use only customer data, 0 =
use only the nova_data mix above
    percent: 25
```

這裡有兩種最上層的資料類別：

- `nova_data`：這是實際的資料混合，並細分為更多類別。總和必須達到 100%
 - 您可以在以下找到這些類別的完整明細，包括字符計數
- `customer_data`：這是您在 yaml 頂端 `data_s3_path` 金鑰中參考的訓練資料。此處提供的百分比決定 `nova_data` 的結果百分比。例如，在上述百分比選擇中，在訓練期間，我們將使用 25% 的 `customer_data` 和 75% 的 `nova_data`，其中 15% 將是客服人員、15% 將是 `business-and-finance`、15% 將是科學、15% 將是程式碼，而 15% 將是 `factual-and-news`

秘訣：再次執行 `pip install -e`。您已準備好提交任務！

我們將覆寫這裡的幾個參數，以使用資料混合：

```
hyperpod start-job \
  -n kubeflow \
  --recipe fine-tuning/nova/forg/nova_2_0/nova_lite/SFT/
nova_lite_2_0_p5x8_gpu_sft_with_datamix \
  --override-parameters '{
    "instance_type": "ml.p5.48xlarge",
    "recipes.run.name": "nova-sft-datamixing",
    "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-
SFT-V2-Datamix",
    "recipes.run.data_s3_path": "s3://sft-data/sft_train_data.jsonl",
    "recipes.run.validation_data_s3_path": "s3://sft-data/sft_val_data.jsonl",
    "recipes.run.output_s3_path": "s3://sft-data/output/
  }'
```

您的輸出應包含任務名稱，如下所示：

```
# MY Desktop # % hyperpod start-job \
  -n kubeflow \
  --recipe training/nova/forg/nova_2_0/nova_lite/sft/
nova_lite_2_0_p5x8_gpu_pretrain_with_datamix \
  --override-parameters '{
    "instance_type": "ml.p5.48xlarge",
    "recipes.run.name": "nova-sft-datamixing",
    "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-fine-tune-repo:SM-HP-
SFT-V2-Datamix",
    "recipes.run.data_s3_path": "s3://sft-data/sft_train_data.jsonl",
    "recipes.run.validation_data_s3_path": "3://sft-data/sft_val_data.jsonl",
    "recipes.run.output_s3_path": "3://sft-data/output/
```

```
}'
```

輸出會像這樣：

```
Final command: python3 /local/home/my/Downloads/sagemaker-hyperpod-cli/
src/hyperpod_cli/sagemaker_hyperpod_recipes/main.py recipes=training/nova/
nova_micro_p5x8_gpu_pretrain cluster_type=k8s cluster=k8s base_results_dir=/local/home/
niphaded/Downloads/sagemaker-hyperpod-cli/results cluster.pullPolicy="IfNotPresent"
cluster.restartPolicy="OnFailure" cluster.namespace="kubeflow"
instance_type="p5d.48xlarge" container="900867814919.dkr.ecr.us-east-1.amazonaws.com/
nova-fine-tune-repo:sft-datamix-rig-final"
Prepared output directory at /local/home/my/Downloads/sagemaker-hyperpod-cli/results/
my-sft-run-wzdyn/k8s_templates
Found credentials in shared credentials file: ~/.aws/credentials
Helm script created at /local/home/my/Downloads/sagemaker-hyperpod-cli/results/my-sft-
run-wzdyn/niphaded-sft-run-wzdyn_launch.sh
Running Helm script: /local/home/my/Downloads/sagemaker-hyperpod-cli/results/my-sft-
run-wzdyn/niphaded-sft-run-wzdyn_launch.sh

NAME: my-sft-run-wzdyn
LAST DEPLOYED: Tue Aug 26 16:21:06 2025
NAMESPACE: kubeflow
STATUS: deployed
REVISION: 1
TEST SUITE: None
Launcher successfully generated: /local/home/my/Downloads/sagemaker-hyperpod-cli/src/
hyperpod_cli/sagemaker_hyperpod_recipes/launcher/nova/k8s_templates/SFT

{
  "Console URL": "https://us-east-1.console.aws.amazon.com/sagemaker/home?region=us-
east-1#/cluster-management/hyperpod-eks-ga-0703"
}
```

您可以使用 `Hyperpod list-pods -n kubeflow --job-name my-sft-run-wzdyn` 檢視任務的狀態

```
hyperpod list-pods -n kubeflow --job-name my-sft-run-wzdyn
{
  "pods": [
    {
      "PodName": "my-sft-run-wzdyn-master-0",
      "Namespace": "kubeflow",
```

```

    "Status": "Pending",
    "CreationTime": "2025-08-26 16:21:06+00:00"
  },
  {
    "PodName": "my-sft-run-wzdyn-worker-0",
    "Namespace": "kubeflow",
    "Status": "Pending",
    "CreationTime": "2025-08-26 16:21:06+00:00"
  }
]
}

```

或直接使用 `kubectl` 命令來尋找它們。

例如

```
kubectl get pods -o wide -w -n kubeflow | (head -n1 ; grep my-sft-run)
```

NAME	READY	STATUS
RESTARTS AGE IP NODE		NOMINATED NODE
READINESS GATES		
my-sft-run-5suc8-master-0	0/1	Completed 0
3h23m 172.31.32.132 hyperpod-i-00b3d8a1bf25714e4	<none>	<none>
my-sft-run-5suc8-worker-0	0/1	Completed 0
3h23m 172.31.44.196 hyperpod-i-0aa7ccfc2bd26b2a0	<none>	<none>
my-sft-run-5suc8-worker-1	0/1	Completed 0
3h23m 172.31.46.84 hyperpod-i-026df6406a7b7e55c	<none>	<none>
my-sft-run-5suc8-worker-2	0/1	Completed 0
3h23m 172.31.28.68 hyperpod-i-0802e850f903f28f1	<none>	<none>

專業提示：請務必一律使用 `-o wide` 旗標，因為任務執行所在的 EKS 節點可協助您更快地在 UI AWS 中尋找日誌

如何監控任務

您可以透過以下三種方式之一來檢視日誌：

a) 使用 CloudWatch

您的日誌可在包含 CloudWatch 下 Hyperpod 叢集的 Amazon Web Services 帳戶中使用。若要在瀏覽器中檢視它們，請導覽至您帳戶中的 CloudWatch 首頁，並搜尋您的叢集名稱。例如，如果您的叢集稱為 `my-hyperpod-rig`，則日誌群組會有 字首：

- 日誌群組：`/aws/sagemaker/Clusters/my-hyperpod-rig/{UUID}`
- 進入日誌群組後，您可以使用 `- hyperpod-i-00b3d8a1bf25714e4` 等節點執行個體 ID 來尋找您的特定日誌。
 - 此處的 `i-00b3d8a1bf25714e4` 代表訓練任務執行所在的 Hyperpod 易用機器名稱。請記得在先前的命令 `kubectl` 中，如何取得 `Pod -o wide -w -n kubeflow | (head -n1 ; grep my-cpt-run)` 輸出，我們擷取稱為 `NODE` 的資料欄。
 - 在此情況下，「主要」節點執行是在 `Hyperpod-i-00b3d8a1bf25714e4` 上執行，因此我們將使用該字串來選取要檢視的日誌群組。選取描述 `SagemakerHyperPodTrainingJob/rig-group/【NODE】` 的。

您的日誌看起來應該像這樣：

b) 使用 CloudWatch Insights

如果您的任務名稱方便使用，但不希望進行上述所有步驟，您可以直接查詢 `/aws/sagemaker/Clusters/my-hyperpod-rig/{UUID}` 下的所有日誌，以尋找個別日誌。

CPT

```
fields @timestamp, @message, @logStream, @log
| filter @message like /(?(i)Starting CPT Job/
| sort @timestamp desc
| limit 100
```

若要完成任務，請將啟動 SFT 任務取代為完成 SFT 任務

然後，您可以按一下結果，並挑選「Epoch 0」，因為這會是您的主節點。

C) 使用 aws cli

您可以選擇使用結尾您的日誌。執行此操作之前，請使用 `aws --version` 檢查您的 `aws cli` 版本。也建議您使用此公用程式指令碼，以協助追蹤終端機中的即時日誌

適用於 V1：

```
aws logs get-log-events \
  --log-group-name /aws/sagemaker/YourLogGroupName \
  --log-stream-name YourLogStream \
```

```
--start-from-head | jq -r '.events[].message'
```

適用於 V2：

```
aws logs tail /aws/sagemaker/YourLogGroupName \  
  --log-stream-name YourLogStream \  
  --since 10m \  
  --follow
```

D) 設定 ML 流程：

您可以透過 MLFlow 追蹤指標。

建立 MLflow 應用程式

使用 Studio UI：如果您透過 Studio UI 建立訓練任務，預設 MLflow 應用程式會自動建立，並依預設在進階選項下選取。

使用 CLI：如果您使用 CLI，則必須建立 MLflow 應用程式並將其做為訓練任務 API 請求的輸入傳遞。

```
mlflow_app_name="<enter your MLflow app name>"  
role_arn="<enter your role ARN>"  
bucket_name="<enter your bucket name>"  
region="<enter your region>"  
  
mlflow_app_arn=$(aws sagemaker create-mlflow-app \  
  --name $mlflow_app_name \  
  --artifact-store-uri "s3://$bucket_name" \  
  --role-arn $role_arn \  
  --region $region)
```

存取 MLflow 應用程式

使用 CLI：建立預先簽章的 URL 以存取 MLflow 應用程式 UI：

```
aws sagemaker create-presigned-mlflow-app-url \  
  --arn $mlflow_app_arn \  
  --region $region \  
  --output text
```


設定 ML 流程後，您可以在配方中傳遞 URI，或在啟動任務時使用覆寫。您可以在 README 中找到如何執行此操作的範例。

如何在 SFT 之後評估您的模型？

先決條件

- 來自訓練任務 manifest.json 檔案的檢查點 S3 URI（適用於訓練模型）
- 以正確格式上傳至 S3 的評估資料集
- 評估結果的輸出 S3 路徑

現成基準：使用現成基準來驗證一般任務的效能。如需詳細資訊，請參閱此處。

攜帶您自己的資料：

您也可以提供自訂資料，方法是將它們格式化為下列格式，然後使用下列容器取得推論結果，並視需要取得校正的日誌機率。

具有下列結構的每個任務的 Crate jsonl：

```
{
  "metadata": "{key:4, category:'apple'}",
  "system": "arithmetic-patterns, please answer the following with no other words: ",
  "query": "What is the next number in this series? 1, 2, 4, 8, 16, ?",
  "response": "32"
}
```

在評估任務的推論階段產生的輸出將具有下列結構：

```
{
  "prompt": "[{'role': 'system', 'content': 'arithmetic-patterns, please answer the following with no other words: '}, {'role': 'user', 'content': 'What is the next number in this series? 1, 2, 4, 8, 16, ?'}]",
  "inference": "['32']",
  "gold": "32",
  "metadata": "{key:4, category:'apple'}"
}
```

欄位描述：

- 提示：傳送至模型的格式化輸入
- 推論：模型產生的回應
- 金色：來自輸入資料集的預期正確答案、來自輸入的回應欄位
- 中繼資料：從輸入傳遞的選用中繼資料

準備評估組態

啟動評估任務的命令。使用「`--override-parameters`」來修改配方中的任何項目。

```
hyperpod start-job -n kubeflow \
  --recipe evaluation/nova/nova_micro_p5_48xl_bring_your_own_dataset_eval \
  --override-parameters '{
    "instance_type": "p5.48xlarge",
    "container": "708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-latest",
    "recipes.run.name": "<your-eval-job-name>",
    "recipes.run.model_name_or_path": "<checkpoint-s3-uri>",
    "recipes.run.output_s3_path": "s3://<your-bucket>/eval-results/",
    "recipes.run.data_s3_path": "s3://<your-bucket>/eval-data.jsonl"
  }'
```

啟動您的評估任務

任務啟動具有對應映像之不同配方的命令。

評估任務啟動命令

模型	技術	Subcategory	映像 URI	命令
新版本 2.0	評估	評估	708977205387.dkr.ecr.us-east-1.amazonaws.com/nova-evaluation-repo:SM-HP-Eval-latest	Hyperpod start-job -n kubeflow \ --recipe evaluation/nova/nova_micro_p5_48xl_bring_your_own_dataset_eval \ --override-

模型	技術	Subcategory	映像 URI	命令
				<pre>parameters { "instance_type" : "ml.p5.48 xlarge" , "container" : "70897720 5387.dkr.ecr.us- east-1.amaz onaws.com/ nova-evaluation- repo:SM-HP- Eval-latest" }</pre>

經驗教訓和秘訣

- SFT 資料集的品質至關重要。您應該盡一切努力篩選掉低品質的資料。如果您在複雜性和準確性方面有一小部分非常高品質的資料，您可以考慮將其放在訓練結束時，以協助模型更好地收斂。
- 我們利用文字和多模態 (MM) 資料集來混合資料。我們的文字資料集實驗顯示，新增 Nova 的專屬「reasoning-instruction-following」類別可大幅改善一般基準的效能。如果您關心在對資料集執行 SFT 之後迴歸的一般基準，建議您在資料混合策略中包含此類別。
- 對於 MM 資料集，我們的實驗指出，將超過 20% 的影片類別整合到混合中有助於維持一般基準效能。
- 此外，具有資料混合的 SFT 對學習率非常敏感，因此我們的調查結果建議使用預設學習率進行微調，即 1e-5 for LoRA 和 5e-6 for FR。
- 最後，如果您混合 Nova 專屬資料集，則會在延遲和效能之間產生取捨，因此我們的問題清單建議混合最大 50% 的 作為良好的平衡。

強化學習

Note

訂閱後提供詳細文件

Nova Forge 提供進階強化學習功能，可讓您選擇在自己的環境中使用遠端獎勵函數。客戶可以選擇整合自己的端點來執行即時真實世界意見回饋的驗證，甚至使用自己的協調器來協調環境中的代理多迴轉評估。

使用您自己的協調器進行代理多迴轉評估

對於需要多轉對話或獎勵函數超過 15 分鐘逾時的 Forge 使用者，Nova Forge 提供自攜協調 (BYOO) 功能。這可讓您協調環境中的代理程式多迴轉評估（例如，使用化學工具來評分分子設計，或獎勵有效任務完成和懲罰衝突的機器人模擬）。

主題

- [架構概觀](#)
- [設定和執行](#)

架構概觀

BYOO 架構透過客戶管理的基礎設施，提供推展和產生程序的完整控制權。

訓練 VPC：

- 推展：透過將推展產生委派給客戶基礎設施來協調訓練
- 培訓人員：根據收到的推展執行模型權重更新

客戶 VPC（例如 EC2 上的 ECS）：

- Proxy Lambda：接收推展請求並與客戶基礎設施協調
- 推展回應 SQS：將已完成推展傳回至訓練基礎設施的佇列
- 產生請求 SQS：模型產生請求的佇列
- 產生回應 SQS：模型產生回應的佇列

- 客戶容器：實作自訂協同運作邏輯（可以使用提供的入門套件）
- DynamoDB：儲存和擷取整個協同運作程序的狀態

工作流程：

1. 推展會將推展產生委派給 Proxy Lambda
2. Proxy Lambda 推送推展 API 請求以產生請求 SQS
3. 客戶容器處理請求、管理多迴轉互動，以及呼叫獎勵函數
4. 容器會視需要從 DynamoDB 存放和擷取狀態
5. 容器會將推展回應推送至推展回應 SQS
6. 推展會將完成的推展傳送給訓練人員以進行權重更新

設定和執行

如需詳細的設定指示、配方組態、請求和回應格式，以及環境範例，請參閱提供給 Nova Forge 訂閱者的機密文件。若要取得 Nova Forge 文件，請遵循下列步驟：

```
aws s3 cp s3://nova-forge-c7363-206080352451-us-east-1/v1/ ./ --recursive
```

下載資產後，您可以在 docs 資料夾下找到所有文件。

負責任的 AI Toolkit 和內容管制

負責任的 AI 工具組

Nova Forge 提供負責任的 AI Toolkit，其中包含訓練和評估資料，以在訓練過程中使模型符合 Amazon Nova 負責任的 AI 準則，以及在推論期間控制執行時間，以調節模型回應。

訓練資料 – 強調負責任 AI 原則、安全考量和負責任技術部署的案例和案例可用於資料混合，以便在繼續訓練期間以負責任的方式調整模型。

評估 – 測試模型偵測和拒絕不適當、有害或不正確內容的能力的評估可作為基準任務，以確定基本模型效能和自訂模型效能之間的差異。

執行時間控制 – 根據預設，Amazon Nova 的執行時間會在推論期間控制中等模型回應。若要修改這些執行時間控制項，請聯絡 AWS 帳戶管理員，請求 Amazon Nova 的可自訂內容管制設定。

安全是 AWS 與其使用者之間의 共同責任。變更基礎模型或使用持續的預先訓練來改善特定使用案例的效能，可能會影響新模型的安全性、公平性和其他屬性。強大的適應方法可將基本模型內建的安全、公平性和其他保護變更降至最低，同時將模型未自訂任務對模型效能的影響降至最低。需要對代表使用案例的資料集上的應用程式End-to-end測試，以判斷測試結果是否符合對安全性、公平性和其他屬性的特定期望，以及整體有效性。如需詳細資訊，請參閱 Amazon Web Services 負責任地使用 AI 指南、Amazon Web Services 負責任的 AI 政策、Amazon Web Services 可接受的使用政策和 Amazon Web Services 服務條款。

負責任的使用

使用 AI 模型建置安全、安全和信任措施是 AWS 和客戶之間共同責任。我們的目標是使我們的模型與 [AWS 可接受使用政策](#) 保持一致，在提供令人滿意的客戶體驗的同時，減少不想要的結果。我們負責任的 AI 原則 (RAI) 以 [負責任 AI 的核心維度](#) 為基礎，如下列清單所述。對於其中每個層面，我們都制定了指導方針，以控管整個模型開發生命週期的決策。此生命週期涵蓋了從初始資料收集和預訓練，到部署後執行時期風險緩解實作的每個階段。

- 公平性 - 考慮對不同利害關係人群組的影響
- 可解釋性 - 了解並評估系統輸出
- 隱私權與安全性 - 適當取得、使用和保護資料和模型
- 安全 - 防止有害的輸出和濫用
- 可控制性 - 具有監控和引導 AI 系統行為的機制
- 準確性和穩健性 - 實現正確的系統輸出，即使是非預期或對抗性輸入
- 控管 - 將最佳實務納入 AI 供應鏈，包括供應商和部署者
- 透明度 - 讓利害關係人能夠針對其與 AI 系統的互動做出明智的選擇

指導方針

我們用來引導模型開發的指導方針包括但不限於管制讚美、促進或推廣下列項目的內容：

- 參與危險活動、自我傷害或使用危險物質。
- 使用、濫用或交易受控物質、煙草或酒精。
- 肢體暴力或血腥。
- 兒童虐待或兒童性虐待內容。
- 動物虐待或動物販運。
- 將個人或群組視為需對刻意欺騙、破壞具有一般公眾公信力的機構，或危及人類健康或生計負責的錯誤資訊。
- 惡意軟體、惡意內容或任何有助於網路犯罪的內容。
- 對個人或群組的不尊重、歧視或刻板印象。
- 侮辱、褻瀆、淫穢手勢、性露骨語言、色情、仇恨符號或仇恨群組。
- 科學、教育或參考上下文以外的完整裸體。
- 對群組基於人口統計特性的偏見。

建議

使用適當性：由於 AI 模型的輸出具有機率性，Amazon Nova 可能會產生不正確或不適當的內容。客戶應針對其使用案例評估輸出的準確性和適當性，特別是當這些輸出將直接面向最終使用者時。此外，如果在產生後續決策的客戶工作流程中使用 Amazon Nova，客戶必須評估其使用案例的潛在風險，並實作適當的人工監督、測試和其他使用案例特定保護措施，以減輕此類風險。

提示詞最佳化：如果遇到 Amazon Nova 的審核，請考慮檢查所用提示是否符合上述指導方針。最佳化提示詞以降低產生意外結果的可能性，這是使用 Amazon Nova 模型產生預期輸出的建議策略。請注意使用者控制輸入的位置，包括可能影響模型效能的像素內容。如需更多詳細資訊，請參閱本使用者指南中的提示詞指導方針一節。

隱私權：Amazon Nova 可在 中使用 Amazon Bedrock。Amazon Bedrock 是一項受管服務，不會儲存或檢閱客戶提示，也不會在客戶之間或 Amazon Bedrock 合作夥伴之間共用客戶提示、提示和完成。AWS 不會使用透過 Amazon Bedrock 服務產生的輸入或輸出來訓練 Amazon Bedrock 模型，包括 Amazon Nova。如需詳細資訊，請參閱 AWS 服務條款第 [50.3](#) 節和 AWS [資料隱私權常見問答集](#)。如需服務特定的隱私權資訊，請參閱 [Amazon Bedrock 常見問答集](#) 文件的「隱私權與安全」一節。Amazon Nova 會採取步驟來避免完成可解釋為請求私有資訊的提示詞。如果使用者擔心他們的私有資訊已包含在 Amazon Nova 完成內容中，使用者應[在此處](#)聯絡我們。

安全性：包括 Amazon Nova 在內的所有 Amazon Bedrock 模型都隨附企業安全性，讓客戶能夠建置支援通用資料安全性和合規標準的生成式 AI 應用程式，包括 GDPR 和 HIPAA。客戶可以使用 AWS PrivateLink 在自訂 Amazon Nova 和內部部署網路之間建立私有連線，而無需將客戶流量暴露到網際網路。客戶資料一律在傳輸中和靜態時加密，客戶可以使用自己的金鑰來加密資料，例如使用 [AWS Key Management Service](#)。客戶可以使用 AWS Identity and Access Management 安全地控制對資源的 Amazon Bedrock 存取，包括自訂的 Amazon Nova。此外，Amazon Bedrock 還提供全面的監控和記錄功能，以支援客戶控管和稽核要求。例如，Amazon CloudWatch 可協助追蹤稽核所需的用量指標，並 AWS CloudTrail 協助監控 API 活動和疑難排解問題，因為 Amazon Nova 已與其他 AWS 系統整合。客戶也可以選擇將中繼資料、提示和完成項目存放在自己的加密 Amazon Simple Storage Service (Amazon S3) 儲存貯體中。

智慧財產權：AWS 為一般可用 Amazon Nova 模型的輸出提供無上限的智慧財產權 (IP) 保證涵蓋範圍（請參閱 [服務條款](#) 第 50.10 節）。這表示客戶會受到保護，免於因這些 Amazon Nova 模型產生的輸出而遭第三方主張 IP 侵權或盜用的索賠（包括著作權索賠）。此外，我們針對服務使用的標準 IP 彌償可保護客戶免於因本服務（包括 Amazon Nova 模型）和用於訓練它們的資料而遭第三方宣告控訴 IP 侵權（包括著作權主張）。

Amazon Nova 的配額

您的 AWS 帳戶 具有 Amazon Bedrock 的預設配額，先前稱為限制。若要檢視 Amazon Bedrock 的服務配額，請執行下列其中一項操作：

- 請遵循[檢視服務配額](#)中的步驟，然後選取 Amazon Bedrock 做為服務。
- 請參閱 中的 [Amazon Bedrock 端點和配額](#) AWS 一般參考。

Note

如果配額標示為無法透過 Service Quotas 調整，您可以透過[提高限制表單](#)提交請求，以考慮提高配額。

為了維持服務的效能並確保適當使用 Amazon Bedrock，指派給帳戶的預設配額可能會根據區域因素、付款歷史記錄、欺詐使用和/或配額增加請求的核准而更新。

請求提高 Amazon Nova 配額

您可以按照以下步驟請求提高帳戶配額：

- 如果配額在 Amazon Bedrock 端點的可調整欄中標記為是，並在 中的配額中標記為是 AWS 一般參考，您可以依照 Service Quotas 使用者指南中的 Service [Service Quotas](#) 使用者指南中的[請求配額增加](#)中的步驟進行調整。 <https://docs.aws.amazon.com/general/latest/gr/bedrock.html>
- Amazon Bedrock 端點中的隨需模型調用配額和 中的配額 AWS 一般參考 無法透過 Service Quotas 調整。 <https://docs.aws.amazon.com/general/latest/gr/bedrock.html>請聯絡您的 AWS 帳戶 經理以考慮增加。

Note

由於需求量龐大，將優先考慮所產生流量足以消耗其現有配額配置的客戶。如果您不符合此條件，可能會拒絕您的請求。

- 即使配額在 AWS 一般參考的 [Amazon Bedrock 端點和配額](#)的可調整欄中標記為否，您也可以透過[提高限制表單](#)提交請求，考慮提高配額，

監控 Amazon Nova

監控是維護 Amazon Nova 及其他 AWS 解決方案可靠性、可用性和效能的重要部分。AWS 提供下列監控工具來監看 Amazon Nova、在發生錯誤時回報，並適時採取自動動作：

由於 Amazon Nova 可透過 Amazon Bedrock 使用，請遵循[監控 Amazon Bedrock 的健康情況和效能](#)的所有監控指引。

Amazon Nova 的安全性

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模式](#)將其描述為雲端的安全性，和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 中執行 AWS 服務的基礎設施 AWS 雲端。AWS 也為您提供可安全使用的服務。在[AWS 合規計畫](#)中，第三方稽核人員會定期測試和驗證我們安全的有效性。若要了解適用於 Amazon Nova 的合規計畫，請參閱[AWS 合規計畫的服務範圍](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。您也必須負責其他因素，包括資料的敏感度、公司的要求，以及適用的法律和法規。

由於 Amazon Nova 可透過 Amazon Bedrock 使用，請遵循 [Amazon Bedrock 中的安全性](#)的所有安全指導。

使用 建立 Amazon Nova 資源 AWS CloudFormation

使用 透過 Amazon Bedrock 部署 Amazon Nova 模型時 CloudFormation，請遵循[使用 建立 Amazon Bedrock 資源 CloudFormation](#)的所有 CloudFormation 指導。

如需詳細資訊，包括 YOUR-RESOURCE-PLURAL 的 JSON 和 YAML 範本範例，請參閱AWS CloudFormation 《使用者指南》[中的 YOUR-SERVICE-RESOURCE-TOPNODE-IN-CLOUDFORMATION](#)，修改 [local-phrases.ent](#) 檔案中的此連結。

進一步了解 CloudFormation

若要進一步了解 CloudFormation，請參閱下列資源：

- [AWS CloudFormation](#)
- [AWS CloudFormation 使用者指南](#)
- [CloudFormation API 參考](#)
- [AWS CloudFormation 命令列界面使用者指南](#)

Amazon Nova 開發人員指南的文件歷史記錄

下表說明 Amazon Nova 2.0 的文件版本。

變更	描述	日期
Amazon Nova 2.0 的初始版本	具有新模型的 Amazon Nova 2.0 開發人員指南初始版本：Nova 2 Lite、Amazon Nova Sonic 和 。本指南介紹增強型多模態功能、推理功能和改善所有模型層的效能。	2025 年 12 月 2 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。