

Glow.ai

10.000 Credits

kelly826088

Region TW-01 is offline. Your data has been backed up in region TW-03.

Create New

My Instances

Auto Deploy

Datadrive

Auto Scaling

Snapshots

Mesh Clustering

Storage Space

Billing

Profile

Create New

Accelerator Type

Training GPU

Inference GPU

Rendering GPU

CPU

Region

Global

	GPU Model	GPUs	GPU RAM	Region	Avail Unit	vCPUs	RAM	Storage	Price/h
<input type="radio"/>	NVIDIA L40S PCIe	1	48 GB	TW-04	7	12	80 GB	200 GB	0.650 Credit
<input type="radio"/>	NVIDIA A100-SXM4-80GB	1	80 GB	TW-04	8	16	80 GB	100 GB	1.200 Credits
<input type="radio"/>	NVIDIA H100 SXM5	1	80 GB	TW-02	7	16	224 GB	500 GB	2.960 Credits
<input type="radio"/>	NVIDIA H100 SXM5	1	80 GB	TW-02	8	16	224 GB	1000 GB	3.800 Credits
<input type="radio"/>	NVIDIA H100 SXM5	8							
<input checked="" type="radio"/>	NVIDIA GeForce RTX 4090	1	24 GB	TW-03	8	10	50 GB	100 GB	0.490 Credit

Official Image

Snapshot

Selected

WebUI, PyTorch 2.6.0, Python3.11, JupyterLab/VNC

2.7.0.dev20250312, Python3.11, JupyterLab/VNC

JupyterLab/VNC

Stable Diffusion WebUI Forge, JupyterLab/VNC

Foocus WebUI
Ubuntu20.04, CUDA12.4, PyTorch 2.6.0, Python3.11, JupyterLab/VNC, Foocus WebUI

WebUI Autogen Studio
Ubuntu20.04 CUDA12.2, OLLAMA, WebUI, N8N, Autogen Studio

DeepSeek-R1 70B_Q4
Ubuntu20.04 CUDA12.2, Python3.11, DeepSeek-R1 70B_Q4

Liger Kernel
Ubuntu20.04 CUDA12.2, Python3.11, NVCC, JupyterLab/VNC supported

Ubuntu24.04 CUDA11.8
Ubuntu 24.04 CUDA11.8

GEMMA2-WebUI
WebUI GEMMA2 9B

GEMMA2
ComfyUI GEMMA2

MimicTalk
Ubuntu20.04 CUDA12.2, MimicTalk

☐ NVIDIA GeForce RTX 4090 | 1 | 24 GB | TW-03 | 8 | 10 | 64 GB | 100 GB | 0.550 Credit |

<

1

>

ID	Name	Cost	Action
ins-dr7dvxrxj	NVIDIA GeForce RTX 4090 Region TW-03 · Running	0.060 Credit	Release Take Snapshot
<div>Access Monitor Billing Config Hardware Network Group</div>			
<div>SSH Port 22 HTTP Port 8080</div>			
SSH Command ssh -p 25414 root@tw-05.access.glows.ai		Service Port 25414	SSH Key Inactive
User root		Password *****	

以秒計費

GLOWS.AI

Create New

My Instances

Data Drive

Auto Scaling

Snapshots

Mesh Clustering

Space Management

ACCOUNT

Billing

Profile

Modify Quota

Space Storage

Storage

15GB

Expire

2025-03-30

Snapshot

Storage

10

+

GB

Data Drive

TW-01

2

+

GB

TW-02

1

+

GB

Cancel

Update

jackie691224@hotmail.com

Modify

Snapshot, Drive1, Drive2

開始使用 Open WebUI

① Open WebUI 不會建立任何外部連線，而且您的資料會安全地儲存在您本機伺服器上。

名稱

輸入您的全名

Email

輸入您的電子郵件

密碼

輸入您的密碼

登入

使用者 評估 語式 設定

一般 連線 模型 評估 文件 網頁搜尋 介面 資訊 圖片 管理 資料庫

Back

Ol

Reset Image

deepseek-r1:70b

deepseek-r1:70b

描述

新增標籤

可見性Public所有使用者可存取

模型參數

系統提示詞Write your model system prompt content here e.g.) You are Mario from Super Mario Bros, acting as an assistant.

新增參數

提示詞建議

知識若要在這處附加知識庫，請先將它們新增到「知識」工作區。

選擇知識庫

工具若要在這選擇工具包，請先將它們新增到「工具」工作區。

篩選器若要在這選擇篩選器，請先將它們新增到「語式」工作區。

動作若要在這選擇動作，請先將它們新增到「語式」工作區。

功能Vision Usage Citations

JSON 預覽

70b

deepseek-r1:70b

今天我能為您做些什麼？

+ Code Interpreter

建議

Tell me a fun fact about the Roman Empire

Give me ideas for what to do with my kids' art

Show me a code snippet of a website's sticky header

please introduce verilog testbench

Ol

deepseek-r1:70b

Today at 22:38

A Verilog testbench is a simulation environment used to verify the functionality of a digital design written in Verilog. It provides input stimuli to the design under test (DUT) and checks if the output responses are as expected. A well-written testbench helps ensure that the design behaves correctly for various input scenarios.

Key Components of a Verilog Testbench

1. Design Under Test (DUT): The Verilog module being tested.

2. Stimuli: Input signals applied to the DUT to exercise its functionality.

3. Expected Outputs: The correct outputs the DUT should produce for given inputs.

4. Monitoring: Mechanisms to observe and verify the DUT's behavior.

Structure of a Verilog Testbench

verilog

儲存 複製

1 module testbench;

2 // Declare signals for input, output, and internal connections

3 reg [7:0] input_signal; // Input signal

4 wire [7:0] output_signal; // Output signal

5

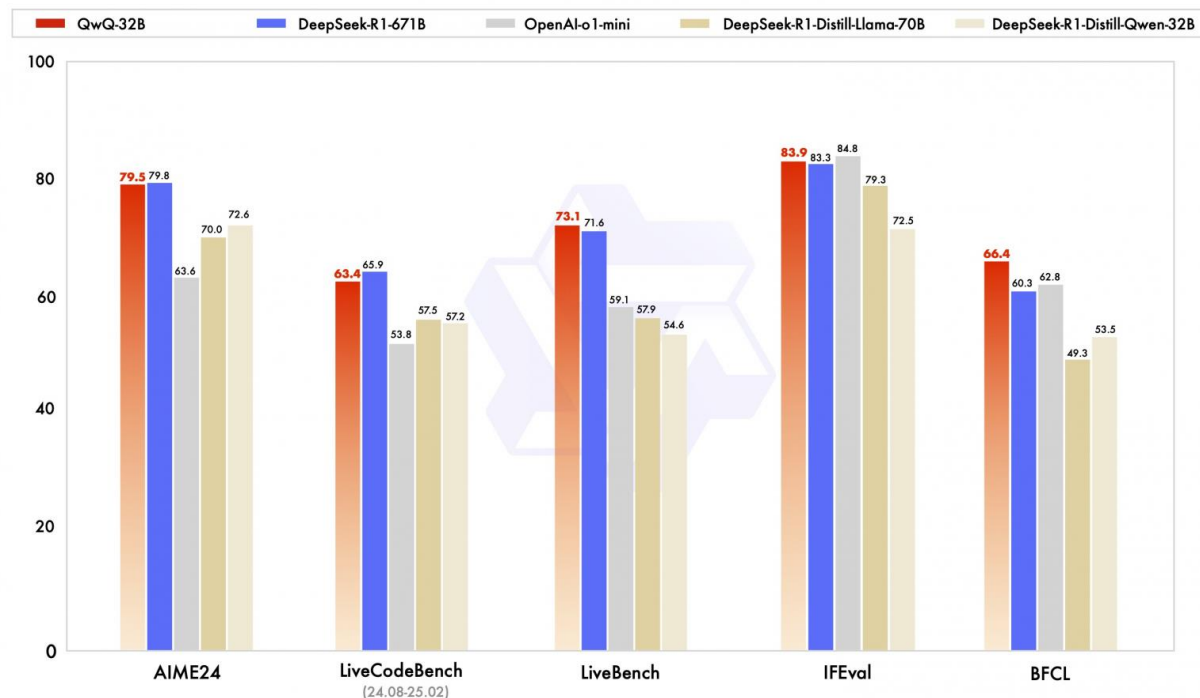
6 // Instantiate the DUT

7 dut_my_dut(input_signal, output_signal);

8

QWQ model

- 4090 -2
- 速度較deepseek R1快, 需要使用資源較少



請和我介紹你自己



qwq:32b 11:07 PM

<think>

好的，用户让我介绍自己。首先，我需要确定用户的需求是什么。可能他们想了解我的功能或者能力，或者是想确认我是否适合帮助他们解决问题。我应该先给出一个简洁的自我介绍，说明我是阿里巴巴集团旗下的AI助手，然后详细说明我的主要功能，比如回答问题、创作文字、编程等。还要提到我的多语言支持，这样用户知道我可以处理不同语言的需求。接下来，我需要询问用户有什么具体的问题或任务需要帮助，这样可以引导他们进一步说明需求，避免信息过载。同时，保持语气友好和开放，让用户感到欢迎和被重视。可能用户没有明确说明他们需要什么类型的信息，所以我的回答需要既全面又简洁，涵盖关键点，但不过于技术化，让用户容易理解。另外，考虑到用户可能有不同的使用场景，比如学习、工作或娱乐，我需要确保我的介绍适用于各种情况。最后，确保回复符合公司的指导原则，不包含任何敏感信息，并保持专业但亲切的语气。

</think>

你好！我是Qwen，阿里巴巴集团旗下的超大规模语言模型。我的设计目标是成为你的得力助手，无论是学习、工作还是日常生活中的问题，我都可以提供帮助。

我的主要能力包括：

1. **回答问题**：无论是学术问题、常识问题，还是专业领域的问题，我都可以尝试为你解答。
2. **创作文字**：我可以帮你写故事、写公文、写邮件、写剧本，甚至创作诗歌。
3. **逻辑推理**：我可以帮助你分析问题、进行逻辑推理，甚至解决一些复杂的数学问题。
4. **编程**：我熟悉多种编程语言，可以帮你写代码、调试程序或解释技术概念。
5. **多语言支持**：除了中文，我还支持英文、日语、韩语、法语、西班牙语等数十种语言。

我的设计特点：

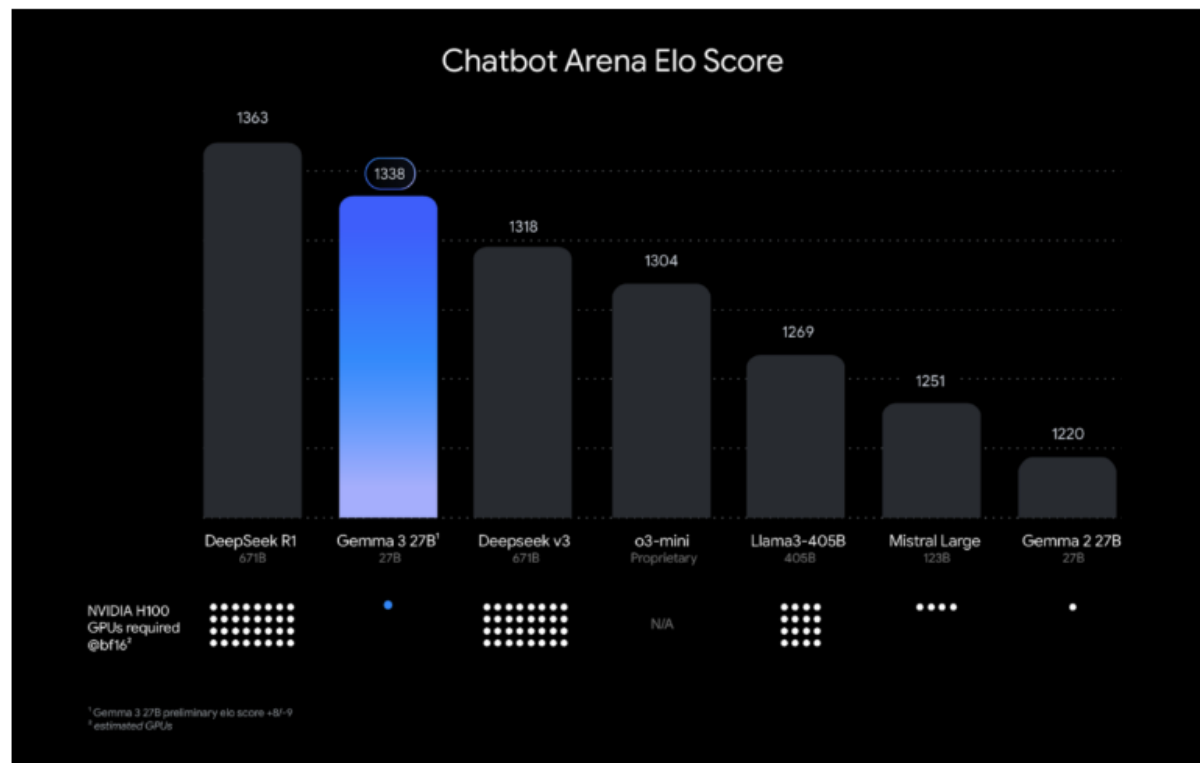
- **超大规模参数量**：这让我能够理解和生成更复杂、更自然的文本。
- **广泛的知识库**：我经过大量互联网文本的训练，具备跨领域的知识。
- **对话理解能力**：我能够理解与用户交互的对话历史，提供更连贯的对话体验。

如果你有任何问题或需要帮助，请随时告诉我！无论是学习上的困惑、工作中的挑战，还是生活中的小问题，我都会尽力为你提供支持。😊



Gemma3

- 27B, 4090 單卡可以運行
- Multi model (vision)



OI

gemma3:27b

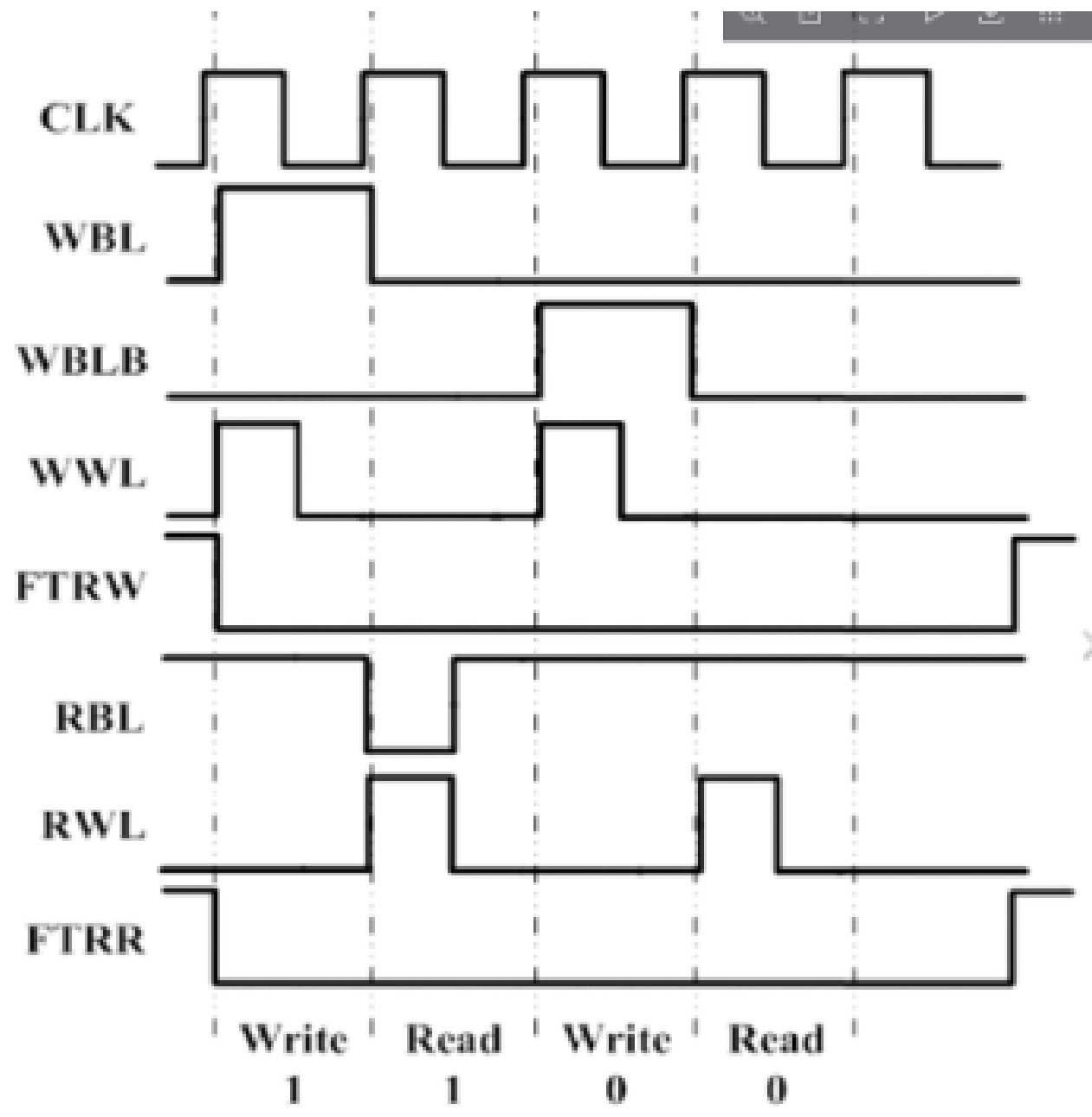
好的，以下是圖片中的文字：

中華民國技術士證

- 身份證統一編號：V123456789
- 出生日期：1980/12/24
- 技術士證號：000-123456
- 職類(項) 目稱：AI大神
- 生效日：民國113/08/07

旁註：勞動部 證





Llama4 16X17B vs. Gemma3 27B vs. Qwen2.5vl 72B

```
systemverilog
1 module memory_controller (
2     input clk,           // Clock
3     input wbl,           // Write Bit Line
4     input wblb,          // Write Bit Line Bar
5     input ww,            // Write Word Line
6     input ftrw,          // Fast Read/Write Control
7     output rbl,          // Read Bit Line
8     output rwl,          // Read Word Line
9     output ftrr          // Fast Read/Write Control
10 );
11
12 reg [7:0] state;        // 状态寄存器
13 reg [7:0] counter;      // 计数器
14
15 always @(posedge clk) begin
16     // 状态转换逻辑
17     case (state)
18         8'b00000000: begin // 初始化状态
19             if (wbl && wblb) begin
20                 state <= 8'b00000001; // 写入状态
21             end else if (!wbl && !wblb) begin
22                 state <= 8'b00000010; // 读取状态
23             end
24         end
25         8'b00000001: begin // 写入状态
26             if (ww) begin
27                 state <= 8'b00000100; // 写完成状态
28             end
29         end
30         8'b00000010: begin // 读取状态
31             if (ftrw) begin
32                 state <= 8'b00000100; // 读完成状态
33             end
34         end
35         default: state <= 8'b00000000; // 重置状态
36     endcase
37
38     // 计数器逻辑
39     if (counter < 10) begin
40         counter <= counter + 1'b1;
41     end else begin
42         counter <= 0;
43     end
44 end
45
46 // 输出逻辑
47 assign rbl = (state == 8'b00000100) ? 1'b1 : 1'b0;
48 assign rwl = (state == 8'b00000010) ? 1'b1 : 1'b0;
49 assign ftrr = (state == 8'b00000100) ? 1'b1 : 1'b0;
50
51 endmodule
```

```
systemverilog
1 module memory_controller (
2     input logic clk,
3     input logic rst,
4     input logic write_en,
5     input logic read_en,
6
7     output logic wbl,
8     output logic wblb,
9     output logic ww,
10    output logic ftrw,
11    output logic rbl,
12    output logic rwl,
13    output logic ftrr
14 );
15
16 // 状态定义
17 typedef enum logic [1:0] {
18     IDLE,
19     WRITE_PRE,
20     WRITE_EN,
21     WRITE_POST,
22     READ_PRE,
23     READ_EN,
24     READ_POST
25 } state_t;
26
27 state_t current_state, next_state;
28
29 // 寄存器
30 logic write_data;
31
```

```
// 状态逻辑
always_comb begin
    next_state = current_state;
    case (current_state)
        IDLE: begin
            if (write_en) begin
                next_state = WRITE_PRE;
            end else if (read_en) begin
                next_state = READ_PRE;
            end
        end
        WRITE_PRE: begin
            next_state = WRITE_EN;
        end
        WRITE_EN: begin
            next_state = WRITE_POST;
        end
        WRITE_POST: begin
            next_state = IDLE;
        end
        READ_PRE: begin
            next_state = READ_EN;
        end
        READ_EN: begin
            next_state = READ_POST;
        end
        READ_POST: begin
            next_state = IDLE;
        end
        default: next_state = IDLE;
    endcase
end

// 时序逻辑
always_ff @(posedge clk or posedge rst) begin
    if (rst) begin
        current_state <= IDLE;
    end else begin
        current_state <= next_state;
    end
end
```

```
// 输出逻辑
always_comb begin
    wbl = 0;
    wblb = 0;
    ww = 0;
    ftrw = 0;
    rbl = 0;
    rwl = 0;
    ftrr = 0;

    case (current_state)
        WRITE_PRE: begin
            ww = 1;
        end
        WRITE_EN: begin
            ftrw = 1;
        end
        READ_PRE: begin
            rwl = 1;
        end
        READ_EN: begin
            ftrr = 1;
        end
        default: ;
    endcase
end

endmodule
```

```
1 module memory_interface (
2     input logic clk,
3     input logic wbl,
4     input logic wblb,
5     input logic ww,
6     input logic wtrw,
7     input logic rbl,
8     input logic rwl,
9     input logic ftrr,
10    output logic [7:0] data_out,
11    input logic [7:0] data_in,
12    output logic read,
13    output logic write
14 );
15
16 logic [7:0] memory [0:255]; // 256-byte memory array
17 logic [7:0] address; // 8-bit address
18
19 always_ff @(posedge clk) begin
20     if (wbl && wblb && ww) begin // Write operation
21         write <= 1;
22         read <= 0;
23         memory[address] <= data_in;
24     end else if (rbl && rwl && ftrr) begin // Read operation
25         write <= 0;
26         read <= 1;
27         data_out <= memory[address];
28     end else begin
29         write <= 0;
30         read <= 0;
31     end
32 end
33
34 // Address counter (for simplicity, assume it increments on each clock cycle)
35 always_ff @(posedge clk) begin
36     address <= address + 1;
37 end
38
```

三份程式碼與時序圖的對應程度比較

最吻合（最正確）：第二位同學

- 狀態完整
- 訊號名稱與時序圖一致
- 實作方式與產業常見設計類似
- 易於擴充與維護

次佳：第三位同學

- 資料流與寫讀判斷直覺，但細節與題目時序圖對應不足
- 無狀態分層，適合作為testbench但不適合純controller設計
- 實際硬體實現可能無法滿足完整的控制邏輯需求

最不吻合：第一位同學

- 狀態設計過於簡化
- 控制訊號拉高時間不準確
- 讀寫切換與狀態遷移過於粗糙

Task / User request



AGENT



TOOLS



MEMORY



PLANNING



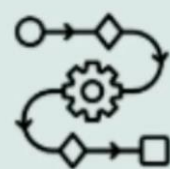
Action

AI Agent



協助進行核心財務決策，例如：預測潛在逾期付款風險、分析現金流異常，並主動提供決策建議以確保財務健康。

AI App
(AI Application)



輔助團隊完成標準化工作，例如：自動分類交易紀錄、生成財務報表摘要或推薦發送付款提醒郵件。

RPA
(Robotic Process Automation)



執行重複性、規則明確的任務，例如：自動輸入財務數據、計算基本指標，提升效率並減少人工操作。▶

客戶支持

AI代理自動處理客戶查詢和提供服務。



業務運營

AI代理優化業務流程和服務管理。



IT安全

AI代理監控網絡安全和問題排查。



人力資源

AI代理自動篩選簡歷和預測候選人。



醫療保健

AI代理協助醫療服務和患者管理。



個性化推薦

AI代理根據客戶偏好提供產品推薦。



市場研究

AI代理分析數據以了解市場趨勢。



金融管理

AI代理進行投資分析和風險評估。



智能製造

AI代理優化生產流程和質量控制。



物流優化

AI代理提升物流運營效率和成本管理。



n8n

- 開源免費的低代碼的自動化工作流平台。





AI agent chat

+ Add tag

Inactive ☐

Share

Saved

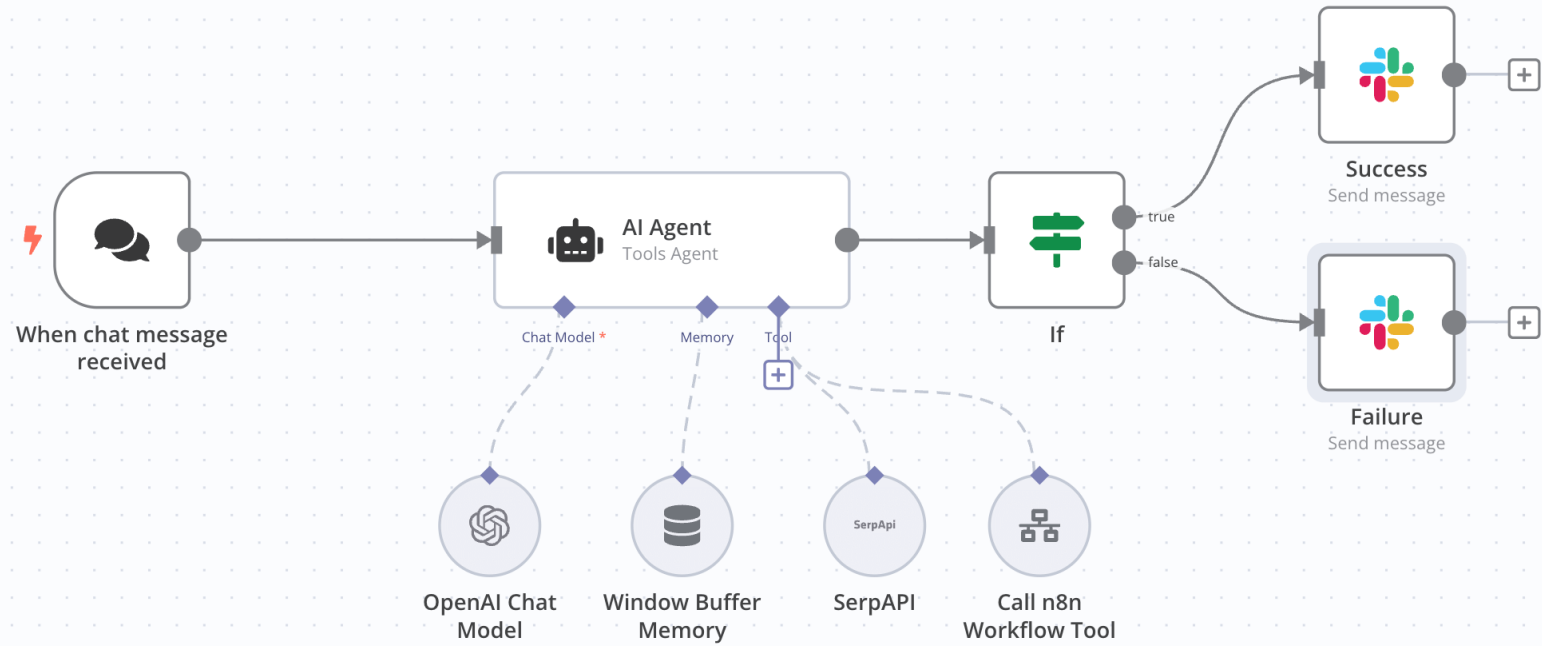


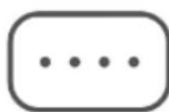
Star

51,309

Editor

Executions





應使用哪種節點類型？



觸發型節點

用於啟動特定動作或事件。



命令執行節點

用於執行特定命令或指令。



資料傳輸節點

用於在系統中移動或傳輸資料。



邏輯節點

用於執行邏輯操作或決策。