Welcome to Flowise

Build AI Agents Visually Without Coding

By : Ashu Mishra | Technical Product Manager

What is Flowise?

✓ Open-source platform for building LLM workflows visually

✓ Low-code/no-code approach - drag-and-drop interface

✓ Built on LangChain & LangGraph foundations

✓ Free forever (core platform) + optional managed cloud

✓ 12,000+ stars on GitHub - active community

✓ Production-ready - used by enterprises worldwide

The AI Development Challenge

❌ Building AI agents historically required deep coding expertise

❌ LLM APIs are complex to orchestrate effectively

❌ Non-technical teams locked out of AI innovation

❌ Rapid prototyping needed to stay competitive

✓ Flowise bridges: business ideas ↔ technical implementation

Flowise: AI Building with LEGO Blocks

- Flowise = LEGO for AI workflows
- Each colorful block = a 'node' with specific functionality
- Connect blocks to build complex intelligent systems
- No internal 'soldering' - Flowise handles the wiring
- ∞ Infinite combinations possible from modular pieces
- Modular design = reusable, maintainable components

Core Concept #1: Nodes

📦 Nodes = functional building blocks in your workflow

→ INPUT: receives data from previous node

⚙️ PROCESSING: does something with that data

→ OUTPUT: sends result to the next node

🔷 Node Types: LLM nodes, Agent nodes, Tool nodes, Memory nodes, and more

✨ Each node is self-contained and reusable

Core Concept #2: Workflows & Flows

🔀 Workflow = connected sequence of nodes

🔗 Each connection represents DATA FLOW (edge)

📊 Workflows can be: Linear (simple), Branching (decisions), Looping (iteration)

3️⃣ Three main types:

- Chatflow: Simple Q&A pattern
- AgentFlow: Multi-agent reasoning & coordination
- Sequential Agents: Conversational with persistent memory

Core Concept #3: Agents

🧠 Agents = LLM-powered decision makers

💭 Think, not just predict: agents REASON about problems

🛠️ Can choose and execute tools dynamically

🔄 Iterate: observe results → update reasoning → act again

Key Difference: LLM just generates text; Agent reasons AND acts

📚 Use case: Complex, multi-step problem-solving

Understanding Tools

- 🔧 Tools = external capabilities agents can call
- 📌 Built-in Tools: Web search, PDF scraper, Calculator, HTTP client
- 🎯 Custom Tools: Any API, database query, notification system
- 🎲 Deterministic: Agent picks the RIGHT tool based on reasoning
- 🔁 Feedback Loop: Tool results feed back → Agent refines
- 💡 Example: Agent uses search tool to find real-time data

Core Concept #4: RAG (Retrieval-Augmented Generation)

📚 RAG = Augment LLM knowledge with your actual documents

❌ Problem: LLMs have stale training data (knowledge cutoff)

✅ Solution: Let agents reference YOUR specific documents

🔄 RAG Pipeline: Load → Chunk → Embed → Store → Retrieve → Answer

🎯 Use Case: Build support bot using company docs

🚀 Result: Domain-specific, always current, accurate responses

Flowise Architecture: Three Layers

🖥️ Backend (Node.js): Processes flows, manages data, handles auth

🎨 Frontend (React): Visual editor for building workflows

🧩 Component Library: Reusable nodes powered by LangChain & LangGraph

🗄️ Database Options: SQLite (dev) → PostgreSQL (production)

🔐 Stack: TypeScript, OpenAPI, REST APIs

💡 Why it matters: Understanding the stack explains features

Node Type #1: Start Node

🚀 Every workflow starts here

📥 Input Types: Chat Input, File Upload, API Trigger, Schedule

💾 Flow State: Defines variables accessible throughout the workflow

🔄 Memory Options:

- Ephemeral: Fresh start each turn (no memory)
- Persist: Conversation history retained across turns

🛡️ Input Moderation: Filter inappropriate/harmful content

Node Type #2: LLM Node

🤖 The 'thinking' engine of your workflow

⚙️ Configuration: Model (GPT-4, Claude, Llama), Temperature, Max Tokens

🔐 Credentials: API keys securely managed

💬 Message Roles:

- System: Instructions & personality
- User: The actual query
- Assistant: Conversation history
- Developer: Meta-instructions for the agent

Node Type #3: Agent Node

🧠 Intelligent decision maker

🤔 LLM reasons: Which tool do I need? Do I need a tool at all?

🔄 Iterates: Observe result → Decide next action → Repeat

🛠️ Agent Types: Tool Agent, Conversational Agent, AutoGPT, CSV Agent

📊 Best For: Multi-step tasks, autonomous problem-solving

💡 Example: Agent decides if task needs search → calculation → response

Node Type #4: Direct Reply & Output Nodes

📤 Direct Reply: Returns static or templated response

🏁 End Node: Marks where the workflow concludes

📋 Formatting: Text, JSON, Structured data

📮 Return As: User Message or Assistant Message

🎯 Template Example: 'Hello {{user_name}}, your answer is {{output}}'

🌐 Destination: Return to user via chat, API, or integration

Node Type #5: Memory & State Management

💾 Agent Memory: Persistent conversation history

📌 Stores: Previous messages, custom variables, context

🗂️ Database: SQLite (development), PostgreSQL (production)

⏱️ Window Memory: Last N messages (optimize context window)

🎯 Custom State: User-defined variables (user_id, preferences)

💡 Example: Multi-turn conversation where agent learns from history

Workflow Architecture #1: Chatflow (Simple)

📝 Simplest pattern: Start → LLM → Output

🎯 Best For: FAQ bots, document Q&A, knowledge base

✨ Characteristics: No reasoning, no tools, straightforward

⚡ Speed: Fast execution, predictable results

📚 Use Cases:

- 'Chat with your PDF' applications
- Internal knowledge base Q&A
- Customer FAQ automation

Workflow Architecture #2: AgentFlow (Multi-Agent)

👥 Coordinate multiple specialized agents

🤝 Each agent has specific role: researcher, analyzer, presenter

🔗 Agents communicate and hand off work

📊 Supports: Sub-flow execution, hierarchical orchestration

🎯 Use Cases:

- Complex problem-solving
- Research pipelines (research → analysis → report)
- Document processing with expertise handoff

Workflow Architecture #3: AgentFlow V2 (Modern)

✨ Latest architecture - explicit workflow orchestration

🏗️ Native nodes (no external framework reliance)

🔀 Advanced Features: Branching logic, loops, human-in-the-loop

📦 Node Set: Start, Agent, LLM, Tool, Condition, End, Sub-flows

🎯 Advantages: More control, better debugging, clearer data flow

🚀 Perfect For: Complex, flexible, production-grade systems

Workflow Architecture #4: Sequential Agents

💬 Conversation-focused agent systems

🔄 Loop Pattern: Agent → Tool → Memory → Agent (repeat)

📝 Maintains conversation state across turns

✅ Built on LangGraph: Provides robust orchestration

🎯 Use Cases:

- Natural, flowing chatbots
- Assistant-like interactions
- Iterative problem-solving with memory

Advanced Pattern #1: RAG Pipeline

📂 Step 1 - Document Loading: PDF, DOCX, web pages, databases

✂️ Step 2 - Text Splitting: Chunking (size + overlap strategy)

🧮 Step 3 - Embedding: Convert text to vectors (numerical representation)

🗄️ Step 4 - Vector Storage: Pinecone, Weaviate, Milvus, Postgres pgvector

🔍 Step 5 - Retrieval: Semantic + keyword search combined

💡 Step 6 - Generation: LLM answers using retrieved context

Advanced Pattern #2: Agentic RAG

🤖 Agent controls the RAG process (not blind retrieval)

🎯 Agent Reasoning:

- 'Do I need to search? What keywords?'
- 'Is this document actually relevant?'
- 'Should I try a different search?'

✅ Query Refinement: Agent rephrases question for better results

🏆 Advantages: Higher accuracy, handles vague queries, adaptive

Advanced Pattern #3: Tool Integration & Function Calling

🔧 Tool Calling: Agent invokes external systems directly

📌 Built-in Tools: Web search, PDF scraper, calculator, HTTP

🎯 Custom Tools: Slack, Notion, Google Sheets, databases, APIs

⚙️ Function Calling (OpenAI): Agent outputs function calls with parameters

🔗 Parameter Mapping: Agent automatically fills tool inputs

💡 Example: Agent sends email, updates database, fetches real-time data

Advanced Pattern #4: Human-in-the-Loop (HITL)

⏸️ Pause workflow for human approval/review

🎯 Use Cases:

- High-stakes decisions (financial, legal)
- Data validation & quality checks
- Creative review (email drafts, reports)

✅ Benefits: Safety, quality control, learning

🔄 Resume: Agents remember context when restarting

👥 Trust: Keeps humans in the loop for critical decisions

Getting Started: Installation & Setup

- ☁️ Flowise Cloud: Managed hosting (easiest option)
- 🐳 Docker: Self-hosted in containers (flexible)
- 📦 npm: Local development (Node.js v18+)
- 💾 Database: SQLite (dev), PostgreSQL (production)
- 📋 System Requirements: 4GB RAM, Python (optional)
- 🚀 Command: npm install -g flowise && flowise

The Flowise Interface Tour

- 🎨 Visual Editor: Drag-drop canvas for building workflows
- 📦 Left Sidebar: Node library (searchable)
- ⚙️ Right Panel: Node configuration & properties
- 🎯 Toolbar: Save, deploy, test, settings
- 🔑 Credentials: Centralized API key management
- 🏪 Marketplace: Pre-built templates & flows to learn from

Managing Credentials & API Keys

🔐 Credentials Section: Centralized, encrypted storage

🔗 Supported: OpenAI, Anthropic, Google, Azure, Ollama (local)

🛡️ Encryption: Keys stored encrypted in database

🔄 Sharing: Credentials accessible to all flows or private

✅ On-Demand: Add credentials without code

💡 Best Practice: Rotate keys, use env variables in production

Deployment Best Practices

- Security: SSL/TLS, API authentication, input validation
- Monitoring: Track latency, errors, API usage, costs
- Scalability: Load balancing, worker nodes, async tasks
- Database: PostgreSQL (not SQLite) for production
- Versioning: Export flows as JSON, maintain changelog
- Testing: Staging environment before production

Real-World Use Case #1: Knowledge Assistants

📚 Build: RAG-based Q&A on internal documents

📄 Documents: Employee handbook, policies, FAQs, product docs

💬 Interaction: Natural language questions → synthesized answers

💡 Benefits:

- 24/7 self-service support
- Faster employee onboarding
- Reduced support ticket load
- Consistent information delivery

Real-World Use Case #2: Customer Support Agents

🛠️ Build: AI agent + KB + tool integration

🔧 Tools: Check order status, fetch customer history, escalate to human

🔄 Flow: Understand problem → Search KB → Fetch data → Offer solution

📈 Benefits:

- Instant responses 24/7
- Consistent service quality
- Reduce support ticket volume
- Escalate intelligently to humans

Real-World Use Case #3: Research & Analysis Agents

🔬 Build: Multi-agent system with specialized roles

👥 Agents:

- Researcher: Web search, data collection
- Analyzer: Processing, insights extraction
- Reporter: Synthesis, final report

📊 Use Cases:

- Market research automation
- Competitive analysis
- Literature review acceleration

Common Challenges & Solutions

🤥 Hallucination → Use RAG, add fact-checking, lower temperature

❌ Tool Reliability → Error handling, fallbacks, validation

💰 Cost Management → Use cheaper models, caching, limit API calls

⏱️ Latency Issues → Async workflows, caching, optimize searches

📊 Quality → Test extensively, monitor metrics, iterate on prompts

💡 Key: Start simple, measure, iterate iteratively

Flowise Ecosystem & Community

🌟 GitHub: 12,000+ stars, active contributions

🔄 Development: Monthly releases, feature requests welcomed

💬 Community: Discord, tutorials, templates, Q&A

☁️ Enterprise: Flowise Cloud (managed), custom deployments

🔌 Integrations: Growing node library, partnerships

🚀 Maturity: Production-ready, used by enterprises globally

Best Practices for Success

- Start Simple: Master Chatflows before AgentFlows
- Test Early: Validate assumptions with prototypes
- Use Templates: Learn from marketplace examples
- Version Control: Export flows as JSON, commit to git
- Document Everything: Explain decisions and intentions
- Monitor Metrics: Track latency, cost, accuracy
- Iterate Often: Add tools/memory incrementally

Your Learning Path: From Novice to Expert

- 📅 Week 1: Concepts (this course) + simple Chatflow
- 📅 Week 2: Build RAG chatbot + deployment
- 📅 Week 3: Add tools, explore AgentFlow
- 📅 Week 4: Multi-agent systems, human-in-the-loop
- 📅 Week 5+: Custom nodes, advanced patterns, production
- 💡 Pace: Take it step-by-step, each week builds on last

Essential Resources & Links

📖 Official Docs: docs.flowiseai.com

🔗 GitHub: github.com/FlowiseAI/Flowise

💬 Discord: Community support & discussions

🎨 Marketplace: Templates and pre-built flows

📺 YouTube: Tutorials & demos

📊 DataCamp: 'Flowise: A Guide With Demo Project' article

🌐 Blog: Latest features and announcements

Your AI Journey Starts Now

Build, Ship, Iterate. Transform Ideas Into Reality.