Agents | Langchain

Skip to main content LangChainDocsUse casesIntegrationsAPIMoreCommunityTutorialsContributingAlso by LangChainChat our docsLangSmithLangChain HubLangServePython DocsSearchCTRLKGet startedIntroductionInstallationQuickstartLangChain Expression LanguageInterfaceHow toCookbookWhy use LCEL?LangChain Expression Language (LCEL)ModulesModel I/ORetrievalChainsMemoryAgentsAgent typesHow-toToolsToolkitsCallbacksModulesSecurityGuidesEcosystemModulesAgentsOn this pageAgentsSome applications require a flexible chain of calls to LLMs and other tools based on user input. The Agent interface provides the flexibility for such applications. An agent has access to a suite of tools, and determines which ones to use depending on the user input. Agents can use multiple tools, and use the output of one tool as the input to the next.There are two main types of agents:Action agents: at each timestep, decide on the next action using the outputs of all previous actionsPlan-and-execute agents: decide on the full sequence of actions up front, then execute them all without updating the planAction agents are suitable for small tasks, while

plan-and-execute agents are better for complex or long-running tasks that require maintaining long-term objectives and focus. Often the best approach is to combine the dynamism of an action agent with the planning abilities of a plan-and-execute agent by letting the plan-and-execute agent use action agents to execute plans.For a full list of agent types see agent types. Additional abstractions involved in agents are:Tools: the actions an agent can take. What tools you give an agent highly depend on what you want the agent to doToolkits: wrappers around collections of tools that can be used together a specific use case. For example, in order for an agent to
interact with a SQL database it will likely need one tool to execute queries and another to inspect tablesAction agentsAt a high-level an action agent:Receives user inputDecides which tool, if any, to use and the tool inputCalls the tool and records the output (also known as an "observation")Decides the next step using the history of tools, tool inputs, and observationsRepeats 3-4 until it determines it can respond directly to the userAction agents are wrapped in agent executors, chains which are responsible for calling the agent, getting back an action and action input, calling the tool that the action references with the generated input, getting the output of the tool, and then passing all that information back into the agent to get the next action it should take.Although an agent can be constructed in many ways, it typically involves these components:Prompt template: Responsible for taking the user input and previous steps and constructing a prompt
to send to the language modelLanguage model: Takes the prompt with user input and action history and decides what to do nextOutput parser: Takes the output of the language model and parses it into the next action or a final answerPlan-and-execute agentsAt a high-level a plan-and-execute agent:Receives user inputPlans the full sequence of steps to takeExecutes the steps in order, passing the outputs of past steps as inputs to future stepsThe most typical implementation is to have the planner be a language model, and the executor be an action agent. Read more here.Get startedLangChain offers several types of agents. Here's an example using one powered by OpenAI functions:import { initializeAgentExecutorWithOptions } from

"langchain/agents";import { ChatOpenAI } from "langchain/chat_models/openai";import { SerpAPI } from "langchain/tools";import { Calculator } from "langchain/tools/calculator";const tools = [new Calculator(), new SerpAPI()];const chat = new ChatOpenAI({ modelName: "gpt-4", temperature: 0 });const executor = await initializeAgentExecutorWithOptions(tools, chat, { agentType: "openai-functions", verbose: true,});const result = await executor.invoke({ input: "What is the weather in New York?",});console.log(result);/* The current weather in New York is 72°F with a wind speed of 1 mph coming from the SSW. The humidity is at 89% and the UV index is 0 out of 11. The cloud cover is 79% and there has been no rain.*/API Reference:initializeAgentExecutorWithOptions from langchain/agentsChatOpenAI from langchain/chat_models/openaiSerpAPI from langchain/toolsCalculator from langchain/tools/calculatorAnd here is the logged verbose output:[chain/start] [1:chain:AgentExecutor] Entering Chain run with input: { "input": "What is the weather in New York?", "chat_history": []}[llm/start] [1:chain:AgentExecutor > 2:llm:ChatOpenAI] Entering LLM run with input: { "messages": [[{ "lc": 1, "type": "constructor", "id": ["langchain", "schema", "SystemMessage"], "kwargs": { "content": "You are a helpful AI assistant.", "additional_kwargs": {} } }, { "lc": 1, "type": "constructor", "id": ["langchain", "schema", "HumanMessage"], "kwargs": { "content": "What is the weather in New York?", "additional_kwargs": {} } }]]}[llm/end] [1:chain:AgentExecutor > 2:llm:ChatOpenAI] [1.97s] Exiting LLM run with output: { "generations": [[{ "text": "", "message": { "lc": 1, "type": "constructor", "id": ["langchain", "schema", "AIMessage"], "kwargs": { "content": "", "additional_kwargs": { "function_call": { "name": "search", "arguments": "{\n \"input\": \"current weather in New York\"\n}" } } } } }]], "llmOutput": { "tokenUsage": { "completionTokens": 18, "promptTokens": 121, "totalTokens": 139 } }}[agent/action] [1:chain:AgentExecutor] Agent selected action: { "tool": "search", "toolInput": { "input": "current weather in New York"

}, "log": ""}[tool/start] [1:chain:AgentExecutor > 3:tool:SerpAPI] Entering Tool run with input: "current weather in New York"[tool/end] [1:chain:AgentExecutor > 3:tool:SerpAPI] [1.90s] Exiting Tool run with output: "1 am · Feels Like72° · WindSSW 1 mph · Humidity89% · UV Index0 of 11 · Cloud Cover79% · Rain Amount0 in ..."[llm/start] [1:chain:AgentExecutor > 4:llm:ChatOpenAI] Entering LLM run with input: { "messages": [[{ "lc": 1, "type": "constructor", "id": ["langchain", "schema", "SystemMessage"], "kwargs": { "content": "You are a helpful AI assistant.", "additional_kwargs": {} } }, { "lc": 1, "type": "constructor", "id": ["langchain", "schema", "HumanMessage"], "kwargs": { "content": "What is the weather in New York?", "additional_kwargs": {} } }, { "lc": 1, "type": "constructor", "id": ["langchain", "schema", "AIMessage"], "kwargs": { "content": "", "additional_kwargs": { "function_call": { "name": "search", "arguments": "{\"input\":\"current weather in New York\"}" } } } }, { "lc": 1, "type": "constructor", "id": ["langchain", "schema", "FunctionMessage"], "kwargs": { "content": "1 am · Feels Like72° · WindSSW 1 mph · Humidity89% · UV Index0 of 11 · Cloud Cover79% · Rain Amount0 in ...", "name": "search", "additional_kwargs": {} } }]]}[llm/end] [1:chain:AgentExecutor > 4:llm:ChatOpenAI] [3.33s] Exiting LLM run with output: { "generations": [[{ "text": "The current weather in New York is 72°F with a wind speed of 1 mph coming from the SSW. The humidity is at 89% and the UV index is 0 out of 11. The cloud cover is 79% and there has been no rain.", "message": { "lc": 1, "type": "constructor", "id": ["langchain", "schema", "AIMessage"], "kwargs": { "content": "The current weather in New York is 72°F with a wind speed of 1 mph coming from the SSW. The humidity is at 89% and the UV index is 0 out of 11. The cloud cover is 79% and there has been no rain.", "additional_kwargs": {} } } }]], "llmOutput": { "tokenUsage": { "completionTokens": 58, "promptTokens": 180, "totalTokens": 238 } }}[chain/end]

[1:chain:AgentExecutor] [7.73s] Exiting Chain run with output: { "output": "The current weather in New York is 72°F with a wind speed of 1 mph coming from the SSW. The humidity is at 89% and the UV index is 0 out of 11. The cloud cover is 79% and there has been no rain."}PreviousVector store-backed memoryNextAgent typesAction agentsPlan-and-execute agentsGet startedCommunityDiscordTwitterGitHubPythonJS/TSMoreHomepageBlog