Building AI Agents in LangChain: Hub, ReAct, and AgentExecutor

1. Introduction

LangChain enables developers to build **AI Agents** that can autonomously reason, act, and interact with tools or APIs. Modern agents integrate frameworks like **ReAct** for reasoning and action chaining, and **LangChain Hub** for sharing and reusing agent configurations.

Key Capabilities:

- Multi-step reasoning and decision-making.
- Tool calling and structured output handling.
- Reusable components via LangChain Hub.

2. Core Components

2.1 Prompts

Prompts guide LLM reasoning. Types include:

- **Simple Prompt:** Direct instruction.
- **Template Prompt:** Supports dynamic variables.
- **System + Human Prompt:** System defines behavior; human provides query.

Example:

Prompt Example

```
"You are an AI agent. Use available tools to answer queries,
reason step-by-step, and provide the final answer in JSON."
```

2.2 Tools

Tools are external functions that the agent can call.

- **Name:** Identifier used by the agent.
- **Description:** Explains functionality.
- **Function:** Python function implementation.
- **Input Schema:** Optional JSON schema for arguments.

Example Tool: Currency Conversion

Tool Example

```
def get_conversion_factor(base_currency, target_currency):
    return exchange_rate
```

2.3 Agents

Agents orchestrate LLM reasoning and tool usage.

- **Zero-Shot Agent:** Relies on tool descriptions without examples.
- **Conversational Agent:** Maintains chat/memory context.
- **ReAct Agent:** Uses the **Reason + Act** paradigm.

2.4 ReAct Architecture

The **ReAct (Reason + Act)** framework allows agents to interleave reasoning and action dynamically. It consists of:

- **Observation:** Agent perceives the current state or input.
- **Reasoning:** LLM generates a thought process, deciding what action to take.
- **Action:** Executes a tool, API call, or internal function.
- **Reflection:** Updates context/memory with the results.

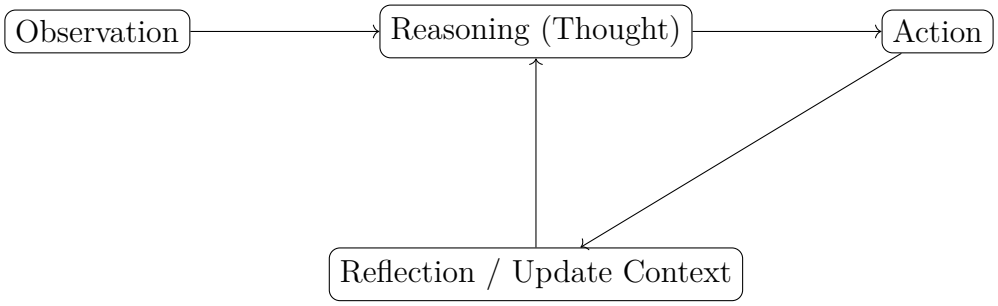
ReAct Loop Example:

ReAct Step-by-Step

```
User query: "Convert 100 USD to EUR"

Observation: Agent sees the query
Thought: "I need to use the currency conversion tool"
Action: convert_currency("USD", "EUR", 100)
Observation: Tool returns 92 EUR
Thought: "I have the answer"
Action: Return final JSON {"USD": 100, "EUR": 92}
```

Diagram (Conceptual):



2.5 AgentExecutor

AgentExecutor runs the agent loop:

1. Receives input from the user.
2. Chooses tools or reasoning steps via LLM.
3. Executes tools.
4. Updates memory or context.
5. Returns final output.

Key Parameters:

- `tools`: List of tool objects.
- `llm`: Language model instance.
- `agent`: Agent type (e.g., "zero-shot-react-description", "conversational-react").
- `handle_parsing_errors`: Bool to retry malformed outputs.

- `verbose`: Bool for logging each step.
- `max_iterations`: Limit reasoning/action steps.

2.6 LangChain Hub

The Hub allows:

- Sharing tools, prompts, and agent configurations.
- Reusing prebuilt agents for common tasks.
- Downloading and deploying agent templates quickly.

Example:

Hub Example

```
agent = load_agent_from_hub("langchain/agent-currency-converter")
```

3. Building a Multi-Tool ReAct Agent

3.1 Define Tools

Tools Definition

```
from langchain_community.tools import DuckDuckGoSearchRun

# Search tool
search_tool = DuckDuckGoSearchRun()

# Weather tool
@tool
def get_weather_data(city: str) -> str:
    """
    Fetch current weather data for a city
    """
    url = f'https://api.weatherstack.com/current?access_key=
    {weather_api_key}&query={city}'
    response = requests.get(url)
    return response.json()
```

3.2 Create ReAct Agent

ReAct Agent Creation

```
from langchain.agents import create_react_agent, AgentExecutor
from langchain import hub

# Pull standard ReAct prompt from LangChain Hub
prompt = hub.pull("hwchase17/react")

# Create agent
agent = create_react_agent(
    llm=llm,
    tools=[search_tool, get_weather_data],
    prompt=prompt
)

# Wrap agent with AgentExecutor
agent_executor = AgentExecutor(
    agent=agent,
    tools=[search_tool, get_weather_data],
    verbose=True
)
```

3.3 Run Agent Examples

Example: Multi-Step Reasoning with Weather (Verbose)

```
response = agent_executor.invoke({
    "input": "Find the capital of Bangladesh and Then tell me weather
    of this city"
})
print(response['output'])

# Verbose AgentExecutor Output:
# > Entering new AgentExecutor chain...
# Action: duckduckgo_search
```

Action Input: capital of BangladeshFrom Wikipedia, the free encyclopedia. Capital and largest city of Bangladesh . This article is about the capital city. Bangladesh , [a] officially the People's Republic of Bangladesh , [b] is a country in South Asia. It is the eighth-most populous country in the world and among the most densely populated with a population of over 171 million within an area of 148,460 square kilometres (57,320 sq mi). Exact time now, time zone, time difference, sunrise/sunset time and key facts for Dhaka, Bangladesh .Dhaka is the capital of Bangladesh . Latitude: 23.71. Longitude: 90.41. Bangladesh is a country in Asia, known for the Sundarbans mangroves and Bengal Delta. It has a population of 175.7 million, making it the 8th largest country in the world. Provides an overview of Bangladesh , including key dates and facts about this South Asian nation.People's republic of bangladesh : facts. Capital : Dhaka. Area: 148,460 sq km.The capital of Bangladesh is Dhaka. Now I need to find the weather of Dhaka.

Action: get_weather_data
Action Input: Dhaka{'request': {'type': 'City', 'query': 'Dhaka, Bangladesh', 'language': 'en', 'unit': 'm'}, 'location': {'name': 'Dhaka', 'country': 'Bangladesh', 'region': '', 'lat': '23.723', 'lon': '90.409', 'timezone_id': 'Asia/Dhaka', 'localtime': '2025-10-09 20:03', 'localtime_epoch': 1760040180, 'utc_offset': '6.0'}, 'current': {'observation_time': '02:03 PM', 'temperature': 29, 'weather_code': 116, 'weather_icons': ['https://cdn.worldweatheronline.com/images/wsymbols01_png_64/wsymbol_0004_black_low_cloud.png'], 'weather_descriptions': ['Partly Cloudy '], 'astro': {'sunrise': '05:53 AM', 'sunset': '05:38 PM', 'moonrise': '07:09 PM', 'moonset': '07:59 AM', 'moon_phase': 'Waning Gibbous', 'moon_illumination': 95}, 'air_quality': {'co': '306.85', 'no2': '7.35', 'o3': '158', 'so2': '21.05', 'pm2_5': '21.15', 'pm10': '21.45', 'us-epa-index': '2', 'gb-defra-index': '2'}, 'wind_speed': 4, 'wind_degree': 7, 'wind_dir': 'N', 'pressure': 1009, 'precip': 0, 'humidity': 70, 'cloudcover': 31, 'feelslike': 32, 'uv_index': 0, 'visibility': 10, 'is_day': 'no'}}

I now know the final answer

```
# Final Answer: The capital of Bangladesh is Dhaka. The current weather
in Dhaka is Partly Cloudy with a temperature of 29°C, but it feels like
32°C. The wind speed is 4 km/h from the North, and the humidity is 70%.

# > Finished chain.

# {'input': 'Find the capital of Bangladesh and Then tell me weather of
this city',

#  'output': 'The capital of Bangladesh is Dhaka. The current weather in
Dhaka is Partly Cloudy with a temperature of 29°C, but it feels like
32°C. The wind speed is 4 km/h from the North, and the humidity is 70%.'}
```

Notes:

- The agent uses **ReAct reasoning**: Observe → Reason → Act → Reflect.
- Multi-step tasks (like finding a capital and then weather) are handled automatically.
- DuckDuckGo and Weather tools can be combined seamlessly in one chain.
- Verbose mode shows the intermediate reasoning steps.

4. Best Practices

- Provide descriptive tool names and usage instructions.
- Keep prompts explicit and clear.
- Use `handle_parsing_errors=True` to handle malformed LLM outputs.
- Limit `max_iterations` to prevent infinite loops.
- Leverage Hub templates to reduce setup time.

5. Summary

LangChain provides a structured framework to build **autonomous AI agents**:

- **Prompts** guide reasoning.

- **Tools** perform external actions.
- **Agents** decide reasoning and action sequence.
- **ReAct** enables reasoning + acting dynamically.
- **AgentExecutor** manages the loop of perception, reasoning, action, and response.
- **Hub** allows sharing and reusing agent templates and tools.

Together, these components enable developers to create **multi-step, intelligent, and tool-using agents** capable of complex real-world tasks.