LangChain

LangChain

How to build an agent

Learn how to build an agent -- from choosing realistic task examples, to building the MVP to testing quality and safety, to deploying in production.

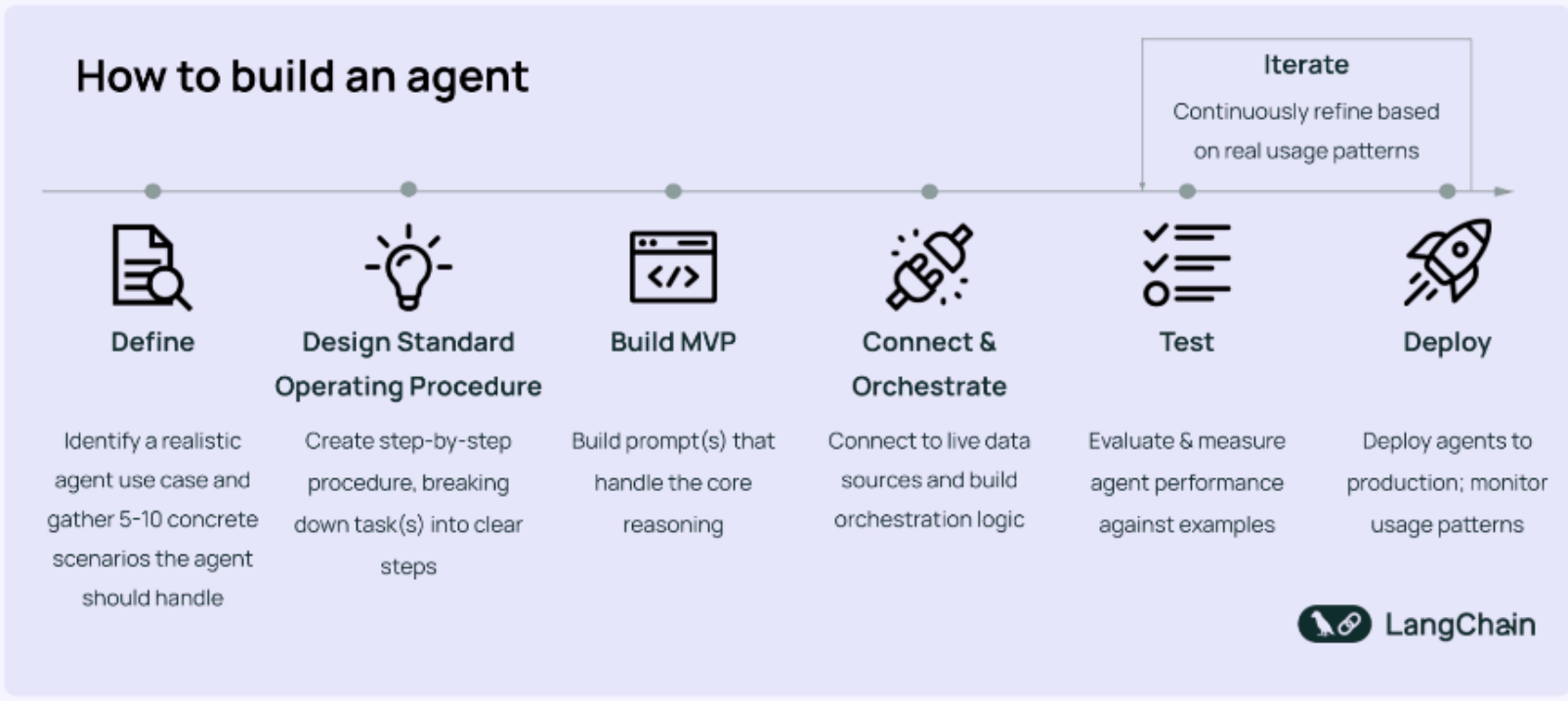
6 MIN READ JUL 9, 2025

SLIDE TO EXPLORE

Follow  **OM NALINDE** to learn more about AI Agents

Table of Contents

Introduction



While seemingly every company is talking about building agents this year, far fewer have done it. It's easy to let your imagination run wild with how agents can transform your business, but many teams are unsure where to begin, how to make progress, and where to set expectations.

In this guide, we'll walk through a framework for going from idea to impact—illustrated with a real-world example of building an email agent.

Introduction

LangChain

Stage	Define	Design Standard Operating Procedure	Build MVP for Core Prompts	Connect & Orchestrate	Test & Iterate	Deploy, Scale & Refine
Description	*Identify a realistic agent use case and gather 5-10 concrete scenarios the agent should handle*	*Create a step-by-step procedure, breaking down the task(s) into clear steps a human would follow*	*Design agent architecture; Build prompt(s) that handle the core reasoning*	*Connect to live data sources and build orchestration logic*	*Evaluate agent against examples; Identify failures & iterate to improve reliability & performance*	*Deploy agent & monitor how users interact; continuously refine based on real usage patterns.*
Duration	~1-2 week(s)	~1-2 week(s)	~1-2 week(s)	~2-3 week(s)	~1-2 month(s)	Ongoing
Output	Clear agent task scope & concrete use cases	Clear Standard Operating Procedure of step-by-step workflow, incorporating identified scenarios	Agent architecture and core prompt(s) that works with static data	End-to-end agent with live data	Agent with systematically tested performance	Improved agent with expanded capabilities
Stakeholders	Product Owner, Subject Matter Expert	Product Manager, Subject Matter Expert	Product Manager, AI & Prompt Engineer	Engineers	Product Manager, QA Engineer	PM, Engineers, Support
Example: Building an Email Agent	Example tasks: • Categorize email by priority • Schedules meetings • Answer product questions	Step-by-step procedure: • With incoming emails, label with email category & response priority • Checks calendar availability & schedule meeting • Draft response & queue for human review	Email agent architecture design with core prompts, incl. email routing & response generation; Test with manually provided context and sample data	Example integrations: • Gmail & Outlook for email access • Google & Outlook Calendar • CRM database for sender context	Evaluate email responses over success criteria: response quality, accuracy, and professional tone	Monitor traffic, use cases, and feedback. Iterate to improve performance and add additional features

Step 1: Define your agent's job with examples

Choose something realistic and something that requires an agent.

Pick something you could teach a smart intern. If your best intern could never complete the task given enough time and resources, the task may be unrealistic or too ambitious. Prove you can get the basics down before activating expert mode.

Start by coming up with 5-10 concrete examples of the task. This serves two purposes:

- First, it validates that your idea is well-scoped - not too trivial or vague
- Second, gives you a benchmark for measuring performance later.

Example: Building an Email Agent

At this step, we'd define what tasks our agent needs to handle, which likely includes:

- Prioritize urgent emails from key stakeholders
- Schedule meetings based on calendar availability
- Ignore spam or emails that don't require responses
- Answer product questions based on company documentation

Step 1: Define your agent’s job with examples

Red flags to avoid:

- If you can’t come up with concrete examples, your scope is probably too broad.
- Using an agent when traditional software would work better (e.g., when the logic is simple, fixed, and already implemented elsewhere)**.** Agents are slow, expensive, and can be finicky at times. If traditional software gets the job done - just use that!
- Expecting magic that doesn't exist (e.g., connecting to APIs or datasets that don’t exist or can’t be built yet)

Step 2: Design operating procedure

Write up a detailed standard operating procedure (SOP), with step-by-step instructions for how a human would perform the task or process.

This step helps confirm that you've chosen a problem with a clear, reasonable scope. It also surfaces the key steps, decisions, and tools your agent will likely need to handle—laying the groundwork for what to build.

Example: Building an Email Agent

For our email agent, a step-by-step procedure could look like below:

- Analyze email content and sender context to categorize response priority
- Checks calendar availability; schedules video conference meeting
- Draft a response based on the email, sender, and scheduling context
- Send the email after a quick human review and approval

Writing this out helps ensure the task is scoped appropriately, and surfaces the tools and logic our agent will need to handle.

Step 3: Build MVP with prompt

Choosing a place to start is important. If your agent is complex, trying to do it all in one go is too ambitious. Start by designing the agent's architecture outlined by the SOP: how it will flow, what decisions it needs to make, and where LLM reasoning is essential.

Then, build an MVP by focusing on the **most critical LLM reasoning task(s)** (e.g., classification, decision-making) and **creating a prompt that handles them well.** Most agents fail because the LLM can't reason well enough for the task. Getting a single prompt working with hand-fed data will help you build up confidence before proceeding to build the full agent. **Prompt engineering tools like LangSmith** can help streamline this process, from managing prompt versions, to testing across scenarios or datasets, and tracking performance over time as you iterate.

Keep it simple by:

- Starting with manual inputs for any data or context the prompt needs (hold off on automation for now)
- Testing against your outlined examples from Step 1 to validate performance across common use cases
- Focusing on getting the LLM reasoning right

Step 3: Build MVP with prompt

Example: Building an Email Agent

At this stage, we're identifying and solving *one* high-leverage reasoning task to start with.

For our email agent, that might mean focusing just on **classifying emails by urgency and intent** (e.g., meeting request, support questions), as this is a foundational step that the rest of the agent depends on.

Start by writing a core prompt that does just this, with hand-fed inputs like:

- Email content: *"Can we meet next week about LangChain's product roadmap?"*
- Sender: "*Jeff Bezos*", *Title: "CEO of Amazon"*
- Output: *Intent = "Meeting Request", Urgency = "High"*

Once the model consistently gets this right across your test cases, you'll have confidence that the core logic is sound—**and a strong foundation to build on.**

Step 4: Connect & Orchestrate

Now that we have a working prompt, it's time to **connect the prompt to real data and user inputs.**

Start by identifying what context or data the prompt needs—such as email content, calendar availability, and documentation of products—and plan how to access it programmatically (e.g., via APIs, databases, or file systems).

Then, write orchestration logic to connect the right data into your prompt. In simple cases, this might just mean passing inputs directly. For more complex workflows, you may need agentic logic to decide which data sources to query, when to call them, and how to combine their outputs before prompting the LLM.

Example: Building an Email Agent

For our email agent, this step could involve integrating with the **Gmail API** (to read incoming emails), **Google Calendar API** (to check availability), and a **CRM or contact database** (to enrich sender context).

We'd then build orchestration logic like the following :

1. A new email triggers the agent
2. The agent fetches sender info from the CRM or via web search
3. It passes the full context into the prompt to determine urgency and whether a response is needed
4. If a meeting is appropriate, it checks calendar availability and proposes times
5. The agent drafts a response
6. After human review, it sends the email

Step 5: Test & Iterate

Begin by **manually testing** your MVP using the examples you defined in Step 1. The goal is to verify that your agent is producing reasonable, accurate outputs for your core use cases. If your system involves multiple LLM calls or steps, it's helpful to **set up tracing** using tools like **LangSmith** to visualize the flow and debug how decisions are made at each stage.

Once manual testing is solid, **scale to automated testing** to ensure consistency and catch edge cases. Teams will often beef up examples to a few dozen to get a better sense of the agent's strengths and weaknesses. This also helps you quantify performance before adding more complexity:

- Run all examples (original + new) programmatically through your agent
- Define automated success metrics — this forces clarity around your agent's expected behavior
- Use human review selectively to catch issues that metrics might miss

Example: Building an Email Agent

For the email agent, we'd want to define and test success across several key areas:

- **Tone and Safety:** Responses should be professional, respectful, and free of hallucinated or inappropriate content
- **Intent & Priority Detection:** Emails should be correctly categorized and prioritized based on sender and content
- **Tool Usage Efficiency:** The agent should trigger only the necessary tools (e.g., avoid checking the calendar if no scheduling is required)
- **Draft Quality:** Suggested replies should be clear, relevant, and accurate based on the input context

Step 6: Deploy, Scale, and Refine

Once your MVP is performing reliably, begin expanding its scope—adding new capabilities, broader use cases, or even multi-agent workflows. For every new feature or integration, **repeat the testing process** from Step 5 to ensure you're not breaking existing functionality.

When ready, deploy to production in users' hands. **LangGraph Platform** allows you to quickly ship, scale, and manage your agents with one-click deployment.

Monitor how people actually use your agent. Tools like LangSmith let you trace your agent's actions in real time, making it easier to spot spikes in cost, accuracy issues, or latency. Real-world usage often differs from your initial assumptions, and these insights can reveal gaps, surface unexpected needs, and guide prioritization during your next iteration.

The key is treating launch as the beginning of iteration, not the end of development.

Example: Building an Email Agent

After deploying our email agent, we might discover unaddressed use cases through monitoring traffic and common use cases.

These emerging patterns signal opportunities to expand scope. From there, we can iteratively add new integrations and update our prompts and orchestration logic—always validating each addition with tests and user feedback before scaling further.

Conclusion

This process is designed to help you build agents that are grounded in clear use cases, tested against real examples, and shaped by real-world feedback. It’s not just about getting an agent to run, but about building something useful, reliable, and aligned with how people actually work.

Whether you're automating email triage or orchestrating complex workflows, these six steps offer a practical path from idea to impact. But the work doesn’t stop at deployment—**the best agents are built through iteration.**

So start small, stay user-focused, and keep refining.

Interested in more content like this?

Follow me :

OM NALINDE

linkedin.com/in/that-aum