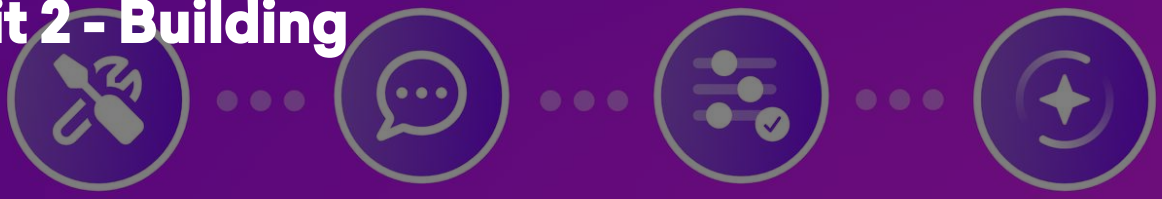
Unit 2 - Building



- 2.1 Unit Introduction
- 2.2 Building Your Agent
- 2.3 Configure the AI agent
- 2.4 Add the knowledge
- 2.5 Add the tools
- 2.6 Write the input
- 2.7 Wrap up

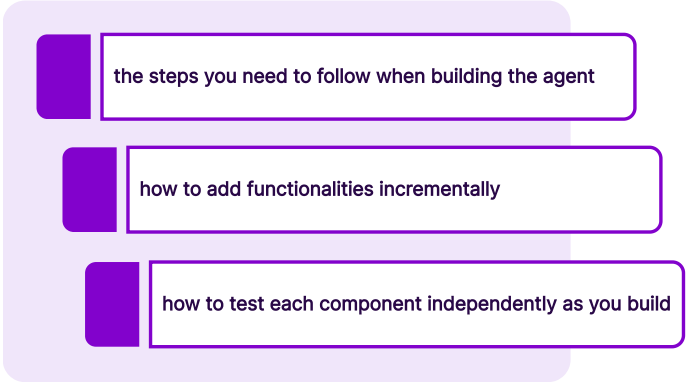
Unit 2
Building

2.1 Unit Introduction

You are on the second unit of the course Design and build a production-ready AI agent.

Now that you've planned your agent's objective, tools, and required data, **it's time to build it** step by step and test if it works.

In this unit you will learn:



Let's begin!

Continue to 2.2: Introduction

2.2 Building your agent

You don't build your AI agent in one go.

After you've planned it, you build it step by step, **testing each component** and improving it based on the results. This approach helps you make sure each component works. It also allows you to **catch problems early** and **understand exactly how each part works**, which helps you build a more effective and solid agent.

When building your agent, you will follow this process:

1. Configure your agent with initial **settings** and **instructions**
2. Add **knowledge** files that your agent can reference

3 Add **tools** that allow your agent to take actions

You will see each of these steps one by one.

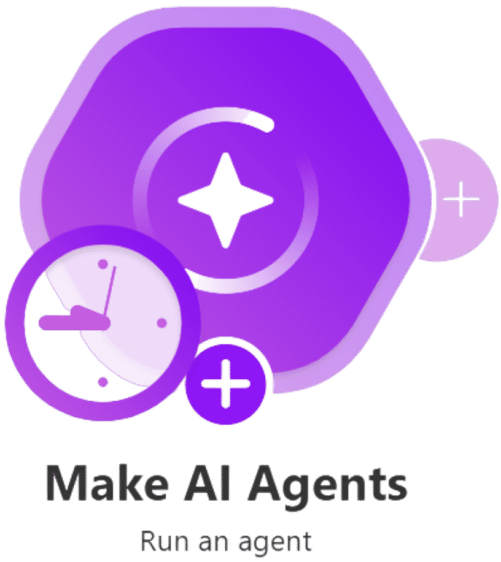
For each step, you will learn **what to do**, **what to avoid**, and **how to test** that it works correctly.

Let's dive in.

Continue to 2.3: Configure the AI agent

2.3 Configure the AI agent

To start building your agent you need to add the Run an agent module to a Make scenario and configure it.



This is where you **define how it will work and behave**. You choose which **AI model** to use and write **instructions** that tell your agent what to do.

> **If in your planning you identified that your agent is part of a larger workflow that includes other modules, add those modules to your scenario as well.**
>
> **These could include modules that provide input data before the agent runs, send the agent's outputs to specific channels, or process results after the agent completes.**

Continue to 2.3.1: Select the model

2.3.1 Select the model

The first element you need to define when building your agent is which AI model it should use.

In the course Tools and AI agent settings optimization, you learned how to select the right model based on capability and cost.

REVIEW UNIT 2

As a reminder, you need to **start with the highest-capability model** you can afford because it is more likely to use knowledge files correctly, follow multi-step instructions reliably, and handle complex logic without errors.

You will learn how to optimize it for cost and speed in the next unit.

Continue to 2.3.2: Tell your agent what to do

2.3.2 Tell your agent what to do

Next, you need to tell your agent what to do, what not to do, and what steps to follow in specific situations.

In Make, you specify this in the **Instructions** field of your agent configuration.

Your instructions should include **several key elements** that define how your agent operates:

Click each card to see the definitions.

Role	**What the agent is and what it handles**
Style	**How it communicates and the tone it uses**

Rules

Specific workflows and processes the agent should follow

Limits

What it should not do

Safety requirements

When the agent needs user approval

You've seen these concepts in the courses:

Information management and security

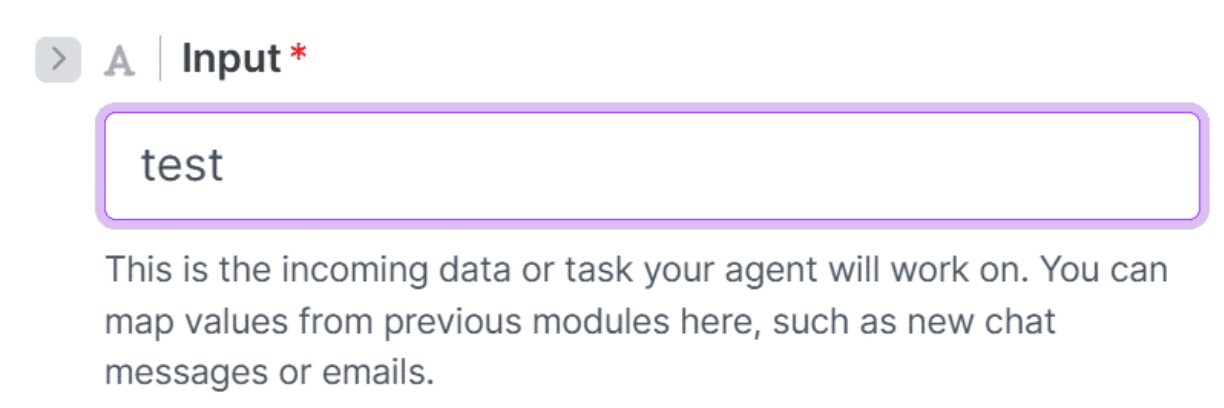
REVIEW UNIT 3

Context engineering

REVIEW UNIT 1

Use the information you have identified in the planning stage to add a **first version** of the instructions. These instructions will evolve as you test different components and discover what needs to be adjusted or

explained differently. For now, this initial version serves as your starting point.



To tell your agent which task it needs to carry out, you also need to provide the request in the **Input** field.

At this stage of development, you can add a **placeholder value** (like test) because you will test your agent through the chat interface to verify all components work. You don't need the actual input yet and you will define what it should look like later.

Later, after you finish testing and your agent works correctly, you will write the final version of the input based on what you learned during testing.

Continue to 2.4: Add the knowledge

2.4 Add the knowledge

After you configure your agent, the next step is to add the knowledge you identified in the planning stage.

In the course Information management and security, you learned why you need knowledge and how to prepare and add knowledge files to your agent in Make.

REVIEW UNIT 2

As a reminder, you **upload or select an existing knowledge file** in the Knowledge section of your agent configuration. The file should contain the reference information your agent needs to carry out its tasks.

After you add a knowledge file, you need to test that your agent can retrieve and use the information correctly.

Use the chat to ask your agent **a question that requires information from the knowledge file**. The agent should **query** the knowledge base, **retrieve** the relevant data, and **use** it in its response.

Check the Reasoning tab to monitor the agent's behavior.

✅

If the agent responds correctly, and you can see it queried the knowledge, your knowledge file is working fine.

❌

If something goes wrong, the agent either retrieved the wrong file or no file at all. In that case, adjust one of these elements: the **file description**, the **instructions**, or the **request**.

Keep adjusting until the agent works as you want.

Pro Tip

In the **Rules** of the agent's instructions **tell it to use a knowledge file for a specific task**.

For example, always use your knowledge base to find the most relevant office for each contact based on their country.

You have a customer support agent that helps assign the correct office to handle customer requests based on their location.

You store this information in a **CSV file** that your agent can access through the Knowledge. After adding it, you test immediately by asking ***Which office handles***

customers in Atlanta? The agent queries the new knowledge file and returns the correct answer.

If you have multiple knowledge files, add and test them one by one to ensure each works correctly before adding the next.

Continue to 2.5: Add the tools

2.5 Add the tools

The next step is to add tools and make sure they work properly.

In the planning stage, you have identified which tools your agent needs to **perform specific actions**. Now it's time to build them.

Remember that as you've seen in the course AI agents in Make you can build either module tools, scenario tools, or use MCP. Choose the one that works best with your needs.

REVIEW AI AGENTS IN MAKE

Add tools to your agent one at a time, not all at once.

This incremental approach helps you **identify issues with each tool** and make sure that they work properly, before adding others.

When you create a tool, remember the best practices for tools naming and description covered in the Context engineering course.

REVIEW CONTEXT ENGINEERING

After you add each tool, you test it immediately before adding the next tool.

This helps you **catch problems early** and understand exactly which tool causes issues if something goes wrong.

You test tools through the agent **chat interface** by sending a request that triggers the specific tool you just added. You then **verify that the agent calls the correct tool** and that you obtain the expected result.

When doing this test, you check the following aspects:

- **Request:** verify that the agent understands what action it needs to perform based on your request, even when some information is missing or incomplete.
- **Configuration:** confirm that the agent properly queries knowledge when needed and prepares all required data before calling the tool.
- **Execution:** check that the agent calls the correct tool with all necessary parameters filled in accurately.
- **Result:** validate that the action completes successfully and produces the expected outcome in the target system.

After you've added a tool to create a contact, you use the chat to send a request like: create contact John Doe in London with email *jdoe@example.com*.

After it runs, check that the agent derived the missing country from the city, queried the knowledge base to find the correct office, and created the contact in Google Sheets with all fields filled correctly.

If a test fails, you don't add more tools. Instead, you fix the current tool by adjusting one of these elements:

- **Tool name or description** to make it clearer when the agent should use it
- **Agent instructions** to specify the correct workflow or steps for using the tool
- **Tool field configuration,** such as enabling **Let AI Agent decide** for fields the agent should populate automatically
- **Tool type,** switching from a module tool to a scenario tool if you need more control over the logic or data

---

You continue adjusting and testing until the tool works as you want. Only then you can move to the next tool.

When building tools, keep in mind the security considerations and best practices from the Information management and security course, which cover how to handle sensitive information.

REVIEW UNIT 3

Continue this process of adding and testing tools one by one until you have added all the tools your agent needs.

Once you've added multiple tools, test whether your agent **calls them in the correct sequence** by sending requests that require the agent to use those tools together.

Pro Tip

If you discover the agent doesn't call them in the correct order, **add specific rules** to your instructions.

For example, if you added both a create contact tool and a search contact tool, and you want the agent to **create contacts only when they don't already exist**, test this workflow.

If the agent creates a duplicate contact instead of searching first, add an explicit rule to your instructions:

> *Before creating a contact, always search for an existing match by email. If found, refuse to create and inform the user the contact already exists*.

Make sure everything is working as it should to complete the building of your agent.

Continue to 2.6: Write the input

2.6 Write the input

During development, you used a placeholder value in the Input field (like test) because you tested your agent through the chat interface.

> Now that you understand how your agent works, you can write the **Input** in the agent configuration.

The input should provide the information your agent needs (whether from a chat message, previous modules, or other sources) along with clear instructions on what to do.

Write your input **based on what you learned during testing** about how to format requests so your agent can understand and execute them correctly.

- [ ] **Run a quick test** to verify your agent still works correctly with this input.
- [ ] **Send a real request through your scenario** (not just the chat interface) to confirm the agent receives the input properly, interprets it correctly, and executes the expected actions.

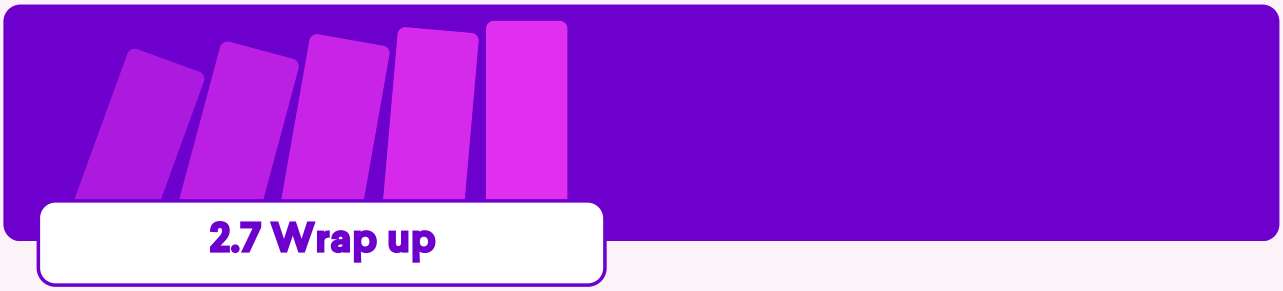
This test doesn't need to be extensive. You just need to **verify that switching from your placeholder to the final input didn't break anything**. You will learn how to do more extensive testing in the next unit.

> **If the agent behaves differently or encounters errors, adjust either the input format or your agent's instructions to handle the input correctly.**

Great, now you have a working agent!

In the next unit, you will learn how to do comprehensive testing to ensure it works correctly. You will also learn how to optimize cost, performance, and speed.

Continue to the wrap up for this unit



2.7 Wrap up

1

Building an AI agent requires a **structured, step-by-step approach** where you configure the agent first, then **add knowledge and tools incrementally**. You start by selecting a **high-capability model** and writing **clear instructions** that define the agent's role, behavior, and rules.

2

Add **one component at a time and test it** before moving to the next. After configuring your agent, you add knowledge files and **test that the agent can retrieve information correctly** by asking specific questions. Then you add tools one by one, **testing each tool through the chat** to verify the

agent calls it correctly and produces the expected result before adding the next tool.

3

Testing each component independently helps you **catch problems early** and **understand exactly what causes issues**. When you test a tool, you verify that the agent understands the request, prepares the data correctly, calls the right tool, and produces the expected outcome. **If something fails**, you adjust the tool configuration, agent instructions, or tool type before moving forward, ensuring each piece works before building on top of it.

Good job!

You now know how to build an AI agent step by step and how to verify everything works correctly before moving to production.

In the next unit, you'll learn how to ensure your agent works correctly and how to optimize its performance.

make | academy

Mark this task complete to continue to the next unit.