Agent Chat

> Agent Chat is the command center for agentic workflows within Cursor. An agent is a large language model (LLM) with access to tool calls—in this case, the ability to make changes directly to your codebase.

Docs: Agent Chat

Overview

This demo covers all the key configurations and features of Agent Chat.

---

1. Agent Modes

> Cursor provides different modes optimized for different tasks. Choose the right mode for your workflow.

Mode	Use Case
Agent	Execute code changes across your codebase (default)
Plan	Create editable implementation plans for complex features
Debug	Find, test, and resolve difficult bugs
Ask	Learn about the codebase without making changes (read-only)

---

2. Model Selection

> You have full control over which model to use. Choose based on your task complexity and speed requirements.

- **Composer 1**: Cursor's own coding model, optimized for speed
- **Auto**: Let Cursor smartly select the best model for each task
- **Frontier models**: Claude, GPT-4, Gemini, and more

Quick Selection

Click the model dropdown in the chat input to quickly switch between models.

Model Settings

For more control, configure models in **Cursor Settings > Models**:

1. Open **Cursor** menu > **Cursor Settings**
2. Navigate to **Models**
3. Click **View All Models** to see all available options
4. Enable or disable models based on your needs

This is useful for: - Enabling new models as they become available - Disabling models you don't use to declutter the selector - Setting default models for different workflows

---

3. Adding Context

> Provide the agent with relevant context so it can make informed decisions.

Why This Matters

Context is everything when working with AI. Without the right context, even the best model will give generic or incorrect answers. The friction of adding context—switching between apps, copying documentation from ChatGPT or Claude's web interface, pasting error messages—creates constant context switching that slows you down more than you realize.

Cursor makes adding context fast and seamless. Instead of juggling browser tabs and copy-pasting between applications, you simply type `@` and reference what you need. This eliminates the hidden productivity tax of context switching, letting you stay in flow while giving the agent exactly what it needs to help you.

How to Add Context

Use the **Add Context** button (or type `@`) to include:

- **Files** – Specific files to reference or modify
- **Folders** – Entire directories for broader context
- **Docs** – Up-to-date documentation from popular libraries
- **Terminal** – Terminal output for debugging errors
- **Past Chats** – Reference previous conversations

---

4. Image Upload

> Upload images as reference for the agent—mockups, screenshots, or error messages.

Click the image upload button to attach images to your prompt.

---

5. Voice Input

> Speak your prompts instead of typing for faster iteration.

Click the microphone button to record your prompt.

---

6. Execution Mode

> Choose where your agent runs based on your needs.

Mode	Description
Local	Runs on your machine with full access to local files
Cloud	Runs on Cursor’s cloud infrastructure
Worktree	Run parallel agents in isolated worktrees (advanced)

---

7. Starting the Server

> Let the agent handle common tasks like starting your development server.

- Start the development server

The agent will inspect your project configuration and run the appropriate command.

Note: Configure terminal command permissions in **Cursor Settings > Agents**: - Ask every time (default) - Run in sandbox - Run everything

---

8. Making Code Changes

> Reference specific files when asking for changes to help the agent find the right code immediately.

- Change sidebar background to blue

After the agent makes changes: 1. Click **Review Changes** to see a diff 2. Click **Keep** to accept or **Undo** to revert

---

9. Fixing Bugs from Terminal Output

> Quickly fix bugs by piping terminal errors directly into the agent.

Step 1: Introduce a bug (for demo purposes) - Use /introduce-runtime-bug slash command

Step 2: View the error 1. Open the terminal panel (View > Terminal) 2. Select the terminal with the error output

Step 3: Send to agent 1. Select the error text in the terminal 2. Press `Cmd+L` to add to chat 3. Submit to have the agent fix it

---

10. Using Documentation

> Reference up-to-date library documentation so the agent uses current APIs and patterns.

Built-in Docs

Popular libraries are pre-indexed. Type `@docs` and search:

- Add validation with Zod

Adding Custom Docs

1. Click **Add Context** > **Docs**
2. Scroll to bottom, click **Add new doc**
3. Paste the documentation URL
4. Cursor will index it for future use

---

11. Git Diff Context

> Review changes against your main branch and generate commit messages.

Use `@Git (Diff with Main Branch)` to: - Understand what your feature branch changed - Generate commit messages based on actual changes - Get an overview of your current work status

- Explain changes against main

Context Window Management

> Monitor your agent's context usage to ensure optimal performance.

As conversations grow, the context window fills up. When it reaches capacity: 1. Cursor summarizes the conversation 2. Resets with the summary as context 3. Continues the task seamlessly

Best Practice: Use atomic tasks—one focused task per agent session. Start new agents for new tasks to keep context clean.

---

12. Demo: Implementing Dark Mode

> Walk through implementing a complete feature using Agent Chat.

This demo shows the full workflow: clicking a deep link, selecting a model, running the agent, and reviewing changes.

Steps

1. Click the deep link below to open the prompt
2. Click **Create Chat** in the dialog
3. Select **composer-1** for faster results
4. Submit and watch the agent implement the feature
5. Click **Review** to see all changes
6. Accept or undo changes as needed

- Implement Dark Mode

What to Highlight

- Deep links pre-fill prompts for consistent demos
- Model selection affects speed and quality
- Agent reads existing code to understand patterns
- Review mode shows all changes across files
- Changes can be accepted or reverted individually

Quick Reference

Action	Shortcut
Open Chat	`Cmd+L`
New Chat	Click “New Chat” button
Chat History	`Opt+Cmd+'`
Add Selection to Chat	Select text + `Cmd+L`
Voice Input	Click microphone button