ChatGPT

{ "meta": { "artifact": "cursor_best_practices_last30days", "time_window": { "from": "2025-10-01", "to": "2025-10-30" }, "generated_at": "2025-10-30T14:19:00Z", "tool": "Cursor 2.0 Research Agent", "version": "1.0" }, "items": [{ "id": "bp-plan-mode", "category": "planning", "practice": "Use Plan Mode to break down complex tasks before writing code", "why_it_matters": "Plan Mode helps the agent gather relevant context and clarify requirements upfront, significantly improving the quality of generated code [1] [2]. It reduces misinterpretation by letting the AI analyze your codebase and ask questions, solving about 80% of communication issues during coding [2].", "how_to_apply": ["Enter Plan Mode (Shift+Tab) and describe the high-level feature or change you need", "Answer any clarifying questions the agent asks to refine the plan [3] [2]", "Review and edit the generated to-do list (plan) for completeness, then instruct the agent to build from the plan when satisfied [4]"], "usage_example": { "context": "Implementing a new feature that spans multiple files (e.g., adding user profile settings)", "steps": ["Activate Plan Mode and prompt the agent with an overview of the 'user settings' feature", "Provide answers when the agent asks for details (e.g., desired settings, UI requirements)", "Inspect the AI-generated Markdown plan (listing files and changes), tweak any steps if needed, then run the plan"], "expected_outcome": "Cursor generates a detailed plan and then writes the necessary code across all relevant files according to that plan, ensuring nothing important is missed", "snippet": ""], "checks": ["Confirm the plan covers all sub-tasks and affected files before execution", "Verify that clarifying questions from the agent have been answered to avoid ambiguities"], "caveats": ["Plan Mode may consume extra tokens for the planning step, but it saves time and tokens overall by reducing corrections [5]", "Using a stronger model for planning can yield better results; the Auto mode typically selects a powerful model (e.g. GPT-5 or Claude 4.5) for Plan Mode automatically [6]"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "Introducing Plan Mode – Cursor Blog", "url": "https://cursor.com/blog/plan-mode", "published_date": "2025-10-07", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Boosting Productivity with Plan Mode (Forum Discussion)", "url": "https://forum.cursor.com/t/boosting-productivity-with-cursor-s-new-plan-mode/136015", "published_date": "2025-10-03", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-parallel-plan", "category": "planning", "practice": "Leverage background and parallel planning to explore multiple solutions", "why_it_matters": "Cursor 2.0 allows planning and building in parallel, so you can use different AI models or simultaneous agents to tackle a complex design from multiple angles. This can reveal better approaches and saves time by running tasks concurrently [7].", "how_to_apply": ["After drafting a plan, choose to build it in the background so you can continue working while the agent executes the plan asynchronously [7]", "For tricky problems, run Plan Mode with two agents (or models) in parallel – each will create its own plan. Compare the resulting plans and pick the best or merge their insights [7] [8]"], "usage_example": { "context": "Designing a complex algorithm or feature with uncertain implementation strategies", "steps": ["Use Plan Mode with Model A (e.g., a highly reasoning model) to generate Plan X, and simultaneously use Plan Mode with Model B to generate Plan Y", "Allow both plans to build in the background, producing two separate branches of code", "Review Plan X vs Plan Y outcomes, then merge the superior approach or components from each into the main project"], "expected_outcome": "Two alternative implementations of the feature are produced for evaluation, without serial wait time. The team can select the most effective solution or combine the best parts of each", "snippet": ""], "checks": ["Ensure each parallel plan runs in an isolated worktree (Cursor handles this to avoid conflicts) and that all planned changes have been applied before merging [9]", "Review each plan’s code output thoroughly to decide which solution to adopt"], "caveats": ["Running multiple planning agents will consume more token budget; use parallel plans primarily for very complex or high-stakes tasks", "Background plan execution relies on git worktrees – be mindful that any external changes to the repo during this process could complicate merging"], "evidence_level": "medium", "volatile": false, "sources": [{ "title": "New Coding Model and Agent Interface – Cursor Changelog 2.0", "url": "https://cursor.com/changelog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 2.0.x – what are parallel agents doing? (Forum)", "url":

"https://forum.cursor.com/t/cursor-2-0-x-what-are-parallel-agents-doing/138911", "published_date": "2025-10-25", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-multi-model", "category": "background_agents", "practice": "Run multiple agents in parallel on the same task (using different models) and pick the best result", "why_it_matters": "Parallel agents offer diverse solutions. Having several AI models attempt a challenging problem simultaneously and then selecting the best outcome significantly improves quality [10]. Each model may approach the task differently, so this \"ensemble\" approach often yields a more robust final result [11].", "how_to_apply": ["In Cursor's Agents interface, start multiple agent instances for your task (up to 8 in Cursor 2.0) [12] ", "Assign a different AI model to each agent (e.g., one using Cursor's Composer, another using GPT-4/5, etc.) [11] ", "Give all agents the same prompt or objective and allow them to work in parallel. Once they complete, compare their outputs side-by-side and select the best approach [13] "], "usage_example": { "context": "Optimizing a complex algorithm implementation", "steps": ["Launch two agents for the task, one with a fast model (Composer) and one with a highly intelligent model (GPT-5 Codex)", "Both agents attempt to implement the optimized algorithm concurrently in isolated worktrees", "Evaluate which agent's solution is more efficient or correct by reviewing their code and test results, then merge the superior solution into your main branch"], "expected_outcome": "The best solution among the parallel runs is identified and used, often combining correctness and efficiency that a single attempt might not achieve", "snippet": ""], "checks": ["Verify that each agent's changes are isolated (Cursor uses separate git worktrees for parallel agents to avoid conflicts) [14] ", "Run tests or benchmarks on each solution to objectively determine which implementation performs better"], "caveats": ["Parallel runs will use more resources and API calls – ensure you have the necessary compute and plan budget for multiple agents", "Merging code from different solutions can be non-trivial if they diverge significantly; you will usually choose one agent's output and discard the others"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "Introducing Cursor 2.0 and Composer (Cursor Blog)", "url": "https://cursor.com/blog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 2.0.x – parallel agents discussion (Forum)", "url": "https://forum.cursor.com/t/cursor-2-0-x-what-are-parallel-agents-doing/138911", "published_date": "2025-10-25", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 2.0 Multi-Agent Approach (Grow Fast Blog)", "url": "https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025", "published_date": "2025-10-30", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-parallel-features", "category": "performance", "practice": "Assign independent features to separate agents to build them in parallel", "why_it_matters": "Parallelizing feature development with multiple agents can drastically shorten development time. Instead of building features one after another, up to eight agents can work concurrently in isolated workspaces [14], collapsing what would be sequential hours of work into the span of a single run [15].", "how_to_apply": ["Identify tasks or features that don't depend on each other (e.g., backend API, frontend component, and a test suite) and spin up a Cursor agent for each one", "Give each agent a clear, separate goal. For example, Agent A implements feature X while Agent B simultaneously works on feature Y. Cursor will create a separate git worktree for each to prevent conflicts [14] ", "After all agents finish, review each agent's changes and merge the features into the main branch, resolving any minor overlaps"], "usage_example": { "context": "Implementing multiple new features in the same sprint (such as user auth, a UI dashboard, and performance optimizations)", "steps": ["Launch three agents in parallel, each assigned to one feature (authentication flow, dashboard UI, performance tuning)", "Each agent works on its feature in its own branch simultaneously [16] ", "Once they complete (roughly at the same time), test each feature and then integrate all changes together into the application"], "expected_outcome": "All three features are completed roughly in the time it would normally take to finish one, significantly accelerating the development cycle", "snippet": ""], "checks": ["Ensure each parallel agent's scope is well-defined to avoid overlapping edits in the same files", "After parallel development, run the full test suite on the integrated codebase to catch any interactions between the newly added features"], "caveats": ["Running many agents at once can strain system resources; monitor performance if you use the maximum parallel agents", "Merging parallel work requires careful integration – even with isolated worktrees, there may be merge conflicts or subtle interactions to resolve in the final code"],

"evidence_level": "medium", "volatile": false, "sources": [{ "title": "New Coding Model and Agent Interface – Cursor Changelog 2.0", "url": "https://cursor.com/changelog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 2.0: Composer, Multi-Agent Dev (GrowFast Blog)", "url": "https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025", "published_date": "2025-10-30", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-team-rules", "category": "rules_engine", "practice": "Define team-wide rules and custom commands to enforce consistent standards", "why_it_matters": "By using Cursor's Team Rules and Commands, organizations can inject coding standards, best practices, and project-specific context into every developer's AI assistant. This ensures uniform code quality and reduces drift across projects [17]. All team members get the same guidance (e.g., testing requirements, style guides) without manual setup on each repo [18].", "how_to_apply": ["In the Cursor web dashboard, create global rules or commands for your team (e.g., a rule requiring docstrings on all functions) [19] ", "Team admins should include any architectural guidelines or forbidden patterns as rules. These rules are then automatically applied in every Cursor session for the team, so the agent follows them by default [18] ", "Leverage this for things like: ensuring new code has unit tests, using specific logging patterns, or including security checks — all described in the shared team rule files"], "usage_example": { "context": "A team wants to enforce a coding style and testing policy", "steps": ["The lead adds a Team Rule (via dashboard) that functions must include type hints and a unit test when created", "Each developer's Cursor agent now always suggests type annotations and a corresponding test file when generating new functions, because the team rules are automatically in its context", "The Cursor agent also respects team-specific commands (e.g., a custom 'init microservice' command) that set up boilerplate according to company standards"], "expected_outcome": "All developers get consistent AI suggestions that adhere to the team's conventions, leading to a cohesive codebase and less rework for style or testing omissions", "snippet": ""], "checks": ["Verify that the intended rules are active by checking Cursor's settings or doing a trial run (e.g., see if the agent produces a test as per the rule)", "Keep the team rules updated as standards evolve (changes in these central rules propagate to everyone immediately)"], "caveats": ["Only team administrators can edit team-wide rules; developers should communicate needed changes", "Overly rigid rules might limit the AI's flexibility – ensure rules are focused on essential standards", "Team rules apply to all projects in the team, which may not be ideal if some repositories have different requirements (consider scoping rules if needed)"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "New Coding Model and Agent Interface – Cursor Changelog 2.0", "url": "https://cursor.com/changelog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 1.7 vs 1.6 – Upgrade Guide (Skywork blog)", "url": "https://skywork.ai/blog/cursor-1-7-vs-1-6-2025-comparison-upgrade-guide/", "published_date": "2025-10-01", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 2.0 Guide – Team Commands (Grow Fast Blog)", "url": "https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025", "published_date": "2025-10-30", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-hooks-safety", "category": "security", "practice": "Implement custom Hooks to enforce security policies and audit agent actions", "why_it_matters": "Cursor's Hooks feature lets you intercept and customize the agent's execution loop, providing a safety net for sensitive operations. With Hooks, you can automatically enforce rules like removing secrets from prompts or blocking dangerous shell commands, ensuring the AI doesn't violate security or compliance guidelines [20].", "how_to_apply": ["Write Hook scripts (in Cursor's supported format) to define checks at runtime. For example, create a hook that scans every agent-generated diff for API keys or passwords and redacts them before the agent proceeds [20] ", "Use another hook to intercept shell commands the agent tries to run; allow or block them based on a whitelist/blacklist (e.g., prevent `sudo` or destructive file operations) [20] ", "Enable these hooks in Cursor (ensure hooks are placed in the project's hooks directory or added via the team dashboard in 2.0 for enterprise) so they automatically execute during agent runs"], "usage_example": { "context": "Developing in a repository that contains secret keys and performing operations that could be risky", "steps": ["Create a `beforeCommand` hook that checks if the agent's command matches a forbidden pattern (like deleting databases) and stops it with a

warning", "Create a `beforePrompt` hook that removes any lines containing known secret patterns (like AWS keys) from the agent's context before it sees them", "Run the agent normally. When it encounters a scenario triggering the hook (e.g., trying to print an API key), the hook code executes, sanitizing the context or aborting the action"], "expected_outcome": "The AI assistant adheres to your security rules automatically: no credentials are exposed in its outputs [20], and potentially harmful commands are blocked unless explicitly approved. The development process stays safe by design", "snippet": ""], "checks": ["Test each hook with a intentional dummy case (e.g., have the agent echo a fake secret) to confirm the hook intercepts it as expected", "Use the Cursor audit log (for enterprise accounts) or console feedback to verify hooks are firing and see what actions were taken [21]"], "caveats": ["Hooks are a powerful feature but can be complex to write; ensure thorough testing of your hooks to avoid false positives or breaking the agent's normal behavior", "As of Cursor 2.0, Hooks may still be marked beta [22], so expect some limitations or need for updates in future releases", "Distributing and managing hooks across a team is easier with enterprise features (dashboard distribution) [23] – otherwise, hooks must be shared and installed manually on each project"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "Browser Controls, Plan Mode, and Hooks – Cursor Changelog 1.7", "url": "https://cursor.com/changelog/1-7", "published_date": "2025-09-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Changelog 2.0 – Hooks (excerpt)", "url": "https://cursor.com/changelog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor 1.7 vs 1.6 – Hooks and governance (Skywork)", "url": "https://skywork.ai/blog/cursor-1-7-vs-1-6-2025-comparison-upgrade-guide/", "published_date": "2025-10-01", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-ai-tdd", "category": "testing_tdd", "practice": "Have the AI follow a test-driven development approach for new code", "why_it_matters": "Test-driven development (TDD) pairs excellently with AI coding. Writing tests first gives the agent a concrete target and prevents it from \"hallucinating\" requirements. The failing tests act as a specification that the AI must satisfy, resulting in code that is verified to meet the requirements once the tests pass [24].", "how_to_apply": ["When prompting for a new feature or function, include instructions for the agent to write tests before the implementation. For example: \"*Write tests for this functionality, then implement it and keep fixing the code until all tests pass.*\" [24]", "Enable Cursor's automated test execution (e.g., YOLO mode or using the integrated terminal to run tests) so the agent can run the tests it wrote. The agent should iteratively debug and update the code based on test failures until the test suite passes [25]"], "usage_example": { "context": "Adding a new data processing function to an existing project", "steps": ["Prompt the agent: \"*First write a unit test for a function that processes input X into output Y (cover edge cases), then write the function and adjust it until the test passes.*\"", "The agent generates a test file defining the expected behavior, then writes the function code", "On running the test, some cases fail; the agent automatically revises the code to fix those issues [25]. It repeats running and fixing until the test suite passes completely"], "expected_outcome": "The final function and accompanying tests are delivered together. The code is confirmed to satisfy the specified behavior because all the agent-written tests are green, embodying true TDD where code meets pre-defined criteria", "snippet": ""], "checks": ["Review the AI-generated tests to ensure they correctly and comprehensively capture the requirements (you can add more test cases if needed and let the agent handle them)", "Run the full test suite (including the new tests) yourself to double-check everything passes in a clean environment"], "caveats": ["The agent might occasionally get stuck or take many iterations if the tests are complex. Be prepared to intervene (e.g., simplify a test or provide a hint) if progress stalls [26]", "Make sure the tests themselves are correct – the AI will optimize to pass the tests given, so flawed tests could lead to misleading success", "Automated test runs consume time; keep the feedback loop efficient by focusing on a small set of tests at a time"], "evidence_level": "medium", "volatile": false, "sources": [{ "title": "Reddit: MVP Development with Cursor (user workflow)", "url": "https://www.reddit.com/r/ChatGPTCoding/comments/1kl5adu/mvp_development_with_cursor_my_full_setup_tdd/", "published_date": "2025-04-15", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Steve Sewell (Builder.io) on Cursor TDD (Blog)", "url": "https://www.builder.io/blog/cursor-tips", "published_date": "2025-03-11", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-code-review-diff", "category": "safe_edits", "practice": "Use

Cursor’s diff view to review agent changes and prefer incremental edits", "why_it_matters": "Reviewing the AI’s code modifications before accepting them is crucial. Cursor 2.0’s improved code review interface shows all agent changes across files in one place [27], allowing you to catch unintended edits. By keeping the AI’s changes small and focused, you maintain control and reduce the risk of introducing errors.", "how_to_apply": ["After the agent makes changes, open the unified diff or \"Review Changes\" view to inspect every file modification in context [28]. Check that each change aligns with your expectations", "Work in small batches: ask the agent to perform one logical edit at a time (e.g., refactor one function or fix one bug) rather than a massive project-wide change. This way the diff is manageable and you can easily identify any wrong changes", "Only accept the diff once you’ve verified it. If you spot an incorrect change, either manually fix it or prompt the agent to correct that specific part before proceeding"], "usage_example": { "context": "Refactoring code with the help of the AI", "steps": ["Request the agent to update a module's function naming for clarity, limited to that module", "When the agent finishes, use Cursor’s diff viewer to see all files it changed (function definitions, call sites, etc.) [27]", "Confirm that only the intended names are changed and no other logic was altered. If an unrelated change appears, revert or edit it out before accepting the rest"], "expected_outcome": "Each AI-assisted change is vetted by the developer. Mistakes or extraneous edits are caught in the diff review, resulting in safer, minimal modifications that preserve existing functionality", "snippet": ""], "checks": ["Ensure tests still pass after applying the diff, or run a quick manual test of the affected feature", "If the diff spans multiple files, pay extra attention to whether the modifications stay consistent (e.g., a function signature change is reflected in all call sites)"], "caveats": ["For very large diffs, consider discarding and breaking the task into smaller pieces – it's harder to verify huge changes thoroughly", "Sometimes an agent’s change might be correct but hard to understand; don’t hesitate to ask the AI to explain the diff or add comments as needed, or simplify the change", "Always have version control backups – if a diff introduces issues, you can revert and try a different approach"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "New Coding Model and Agent Interface – Cursor Changelog 2.0", "url": "https://cursor.com/changelog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Grow Fast Blog – Multi-Agent Coding (code review UI)", "url": "https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025", "published_date": "2025-10-30", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-docs-context", "category": "docs_web", "practice": "Provide the AI with real-time documentation or web knowledge for unfamiliar APIs", "why_it_matters": "The AI’s training data may be outdated on new frameworks or libraries. By injecting up-to-date documentation into Cursor (via the in-app browser or tools like Context7), you prevent the agent from guessing and instead guide it with correct, version-specific information [29]. This leads to accurate code that uses APIs properly, avoiding deprecated or incorrect patterns.", "how_to_apply": ["When using a library or API that the agent might not know well, fetch its official docs into Cursor. You can use the `@Web` context menu or the Browser Agent to search and open the documentation page inside the editor", "Alternatively, use a Model Context Protocol (MCP) server like *Context7*. For example, append **\"use context7\"** in your prompt when asking about a library function. This command tells Cursor to pull the latest relevant docs and examples for that library straight into the conversation [29]", "Once the documentation snippet is present, ask the agent to proceed with implementation. The agent will now reference the actual docs (visible in the prompt context) and produce code that matches the documented usage"], "usage_example": { "context": "Integrating a third-party API (v3.0) that the AI hasn't seen before", "steps": ["Type a query like: *"How do I authenticate with [API Name] v3.0? use context7*" in Cursor", "Cursor fetches the official authentication guide for that API and inserts it into the prompt [30]", "Now ask the agent to implement the authentication code. The agent uses the documentation context to include the correct endpoints and parameters as per v3.0, rather than guessing"], "expected_outcome": "The AI writes integration code that is aligned with the latest official documentation (e.g., correct function names, arguments, and usage patterns), resulting in code that works on the first try without trial-and-error", "snippet": ""], "checks": ["Verify that the fetched documentation truly corresponds to the library version you're using (some tools might grab the latest version; ensure that's what you need)", "If the doc is lengthy, identify

the specific section you need and consider summarizing or focusing the agent on that (to keep the context relevant and within limits)"], "caveats": ["Using external documentation tools may require setup (Context7 needs installation and a prompt trigger, web search needs internet access). Ensure these are configured in Cursor", "Be cautious of documentation from unofficial sources. Stick to official docs or reputable guides to avoid introducing wrong information", "Including very large documentation dumps can use a lot of context window. Provide only the necessary pieces (or use summarization) for efficiency"], "evidence_level": "medium", "volatile": false, "sources": [{ "title": "Reddit – Context7 for Docs (ChatGPTCoding thread)", "url": "https://www.reddit.com/r/ChatGPTCoding/comments/1k6uyjp/i_just_found_out_about_context7_mcp_server_and/", "published_date": "2025-04-01", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor Forum – User Best Practices (Context7 rule)", "url": "https://forum.cursor.com/t/add-the-best-practices-section-to-the-documentation/129131", "published_date": "2025-08-13", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-sandbox-mode", "category": "security", "practice": "Run the AI in sandbox mode for any system-altering commands", "why_it_matters": "Cursor’s sandboxed terminal feature ensures that when the AI executes shell commands, they run in an isolated environment with restricted access (only the project folder, no internet) [31]. This prevents accidental damage (like deleting wrong files or leaking data) if the agent issues a risky command. Keeping sandbox mode on by default protects your system and repository.", "how_to_apply": ["Leave Cursor’s sandbox setting enabled (on macOS 2.0 it’s default [31]; on other platforms or older versions, use allowlist mode so unknown commands go to sandbox [32])", "Let the agent run commands in the sandbox. If a command fails due to sandbox restrictions and you deem it safe, you can explicitly allow it. Cursor will usually prompt if it suspects the sandbox blocked a necessary action [33] ", "For enterprise teams, consider enforcing sandbox mode team-wide via admin controls [34] to ensure no agent in the organization executes unconfined commands"], "usage_example": { "context": "The AI agent needs to execute a build script and run database migrations as part of its plan", "steps": ["Cursor executes these commands inside a secure sandbox, so they can only affect the workspace files and cannot access the internet or outside system files [31] ", "The build runs and the migration touches only the local project database file. If the agent accidentally tries a disallowed action (like sending data out or modifying /etc configs), the sandbox blocks it", "Upon completion, you see the results and can merge the changes knowing no unintended system changes occurred"], "expected_outcome": "The agent can perform typical development tasks safely. Any destructive or network-related commands are contained; if they were truly needed, you would be alerted and could run them manually with oversight. Your system and secrets remain secure throughout the AI’s operations", "snippet": ""], "checks": ["Monitor the agent's terminal output; if something fails due to sandbox (e.g., a network call), decide if it's necessary and safe to run outside the sandbox", "Keep the sandbox on especially for operations like file system changes, installation of packages, or running tests that might have side effects"], "caveats": ["Sandboxed execution may limit certain legitimate actions (for instance, downloading dependencies). You might need to manually intervene for those or temporarily allow network access for that step", "Ensure the sandbox covers what you expect. On some platforms, sandbox support may vary; always use the latest Cursor version for the best sandbox features", "If a command consistently needs to run outside (and is safe), you can add it to the allowlist so it won’t be sandboxed. But be very cautious with expanding the allowlist"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "New Coding Model and Agent Interface – Cursor Changelog 2.0", "url": "https://cursor.com/changelog/2-0", "published_date": "2025-10-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Browser Controls, Plan Mode, Hooks – Changelog 1.7", "url": "https://cursor.com/changelog/1-7", "published_date": "2025-09-29", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Grow Fast Blog – Sandbox and admin controls", "url": "https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025", "published_date": "2025-10-30", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-model-selection", "category": "performance", "practice": "Choose models strategically – use fast models for iteration, and high-accuracy models for complex steps", "why_it_matters": "Different AI models have different strengths. Cursor’s new Composer model, for example, is ~4× faster than GPT-4-class models [35]. This speed keeps you in flow

during development, albeit with a slight trade-off in raw “intelligence.” By using the fast model for routine coding and switching to a more powerful model for particularly tricky or critical tasks, you get the best of both – quick feedback and top-notch quality [35].", "how_to_apply": ["During iterative coding (e.g., debugging, minor feature builds), use a faster model like Composer. Its rapid responses let you test ideas and refine code without long waits [35]", "When facing a complex algorithm, architectural decision, or if the fast model’s output isn’t satisfactory, switch to a more advanced model (GPT-4/5, Claude, etc.) for that query. These models might be slower but can provide deeper insight or more correct solutions for complex problems", "You can let Cursor’s Auto mode handle this (it often picks a strong model for complex prompts [6]), or manually select the model in the agent settings depending on the task at hand"], "usage_example": { "context": "Implementing a feature with multiple steps and edge cases", "steps": ["Use Composer for the initial coding of the feature. It generates code quickly for the main scenarios, and you rapidly iterate as it fixes minor bugs in seconds", "For a complicated edge-case calculation that Composer struggled with, switch the agent to GPT-4 and re-prompt. The more advanced model produces a correct, though slower, solution for that part", "Switch back to Composer for remaining straightforward tasks. Finally, use GPT-4 once more to review the entire code for any subtle issues"], "expected_outcome": "Development moves swiftly for the majority of the work, thanks to the fast model, and the critical portions are handled with the high-accuracy model. The final code is both quickly produced and thoroughly vetted, leveraging each model’s strengths", "snippet": ""], "checks": ["Keep an eye on Cursor’s model selection (displayed above the chat). If a task feels too slow, you might not need a heavyweight model – consider switching to a faster one", "Conversely, if a model is repeatedly making mistakes on a complicated task, upgrading to a smarter model can save time in the long run"], "caveats": ["Faster models may occasionally miss nuances or require a few more iterations – don’t use them blindly for complex logic without tests", "Using multiple models in one session can sometimes cause small inconsistencies in coding style or approach; be prepared to do a final cleanup for consistency", "Higher-tier models often cost more API usage – use them when they matter most"], "evidence_level": "medium", "volatile": false, "sources": [{ "title": "Grow Fast Blog – Composer vs Frontier Models", "url": "https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025", "published_date": "2025-10-30", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor Plan Mode Discussion (model choice)", "url": "https://forum.cursor.com/t/boosting-productivity-with-cursor-s-new-plan-mode/136015", "published_date": "2025-10-09", "accessed_at": "2025-10-30T14:19:00Z" }] }, { "id": "bp-context-ignore", "category": "context_scoping", "practice": "Exclude large, irrelevant, or sensitive files from the AI’s context indexing", "why_it_matters": "By default, Cursor indexes your entire project, which can include thousands of lines that aren’t relevant (vendor libraries, build artifacts) or contain secrets. Ignoring these files improves the AI’s responsiveness and accuracy – it won’t be distracted or accidentally leak sensitive info from them [36] [37]. This is both a performance boost and a security measure.", "how_to_apply": ["Create a `.cursorignore` file in your project root (similar to a .gitignore). Add patterns for files and directories that the AI should skip indexing [38] [37] – e.g., `node_modules/`, `dist/`, `**/*.env`, `*.pem`, large data files, etc.", "Cursor automatically respects your `.gitignore` as well [37], but use `.cursorignore` for anything additional (like internal docs or secrets) that you want excluded from the AI’s context. Once set, these files won’t be considered by the agent unless you explicitly open them"], "usage_example": { "context": "A repository with many generated files and some secret config files", "steps": ["Add common noisy patterns to `.cursorignore`: e.g., `logs/`, `*.log`, `tmp/`, `**/__pycache__`, `*.env` [39] [40]", "Restart or re-index Cursor. The indexing process will skip those files. Now the agent’s suggestions are based only on the core codebase, and it never sees the credentials in the `.env` file during normal operations", "Development is smoother – Cursor responds faster without sifting through irrelevant files, and sensitive data stays out of its prompts"], "expected_outcome": "Cursor focuses on the code that matters. Unnecessary files aren’t embedded or referenced by the AI, which means fewer spurious suggestions (like info from libraries or old logs) and reduced risk of exposing secrets [36]", "snippet": ""], "checks": ["If the agent ever mentions content from an ignored file, double-check that the ignore pattern is correct and that you didn’t manually provide that file to the

agent (remember, opening a file in the editor can still give it context access even if ignored for indexing [41] [42])", "Use the Cursor settings panel for indexing to verify which files are included or skipped. This can help confirm your ignore rules are working"], "caveats": ["`.cursorignore` prevents the AI from automatically indexing those files, but it does not 100% block them. If you open an ignored file and ask the AI about it, it can still read it [41]. So keep truly sensitive files closed or outside of Cursor entirely when possible", "Keep your ignore list updated as the project grows – new large data files or build outputs should be added to keep the index clean"], "evidence_level": "high", "volatile": false, "sources": [{ "title": "NSTech Blog – Excluding Files (Cursor best practices)", "url": "https://ai-learningpath.nstech.com.br/pages/cursor/cursor_excluding_files.html", "published_date": "2025-08-01", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Educative Course – Codebase Indexing (Context management)", "url": "https://www.educative.io/courses/advanced-cursor-ai/mastering-context-codebase-indexing-and--references", "published_date": "2025-10-15", "accessed_at": "2025-10-30T14:19:00Z" }, { "title": "Cursor Forum – .cursorignore caveat (security)", "url": "https://forum.cursor.com/t/big-security-risk-cursorignore-doesnt-seem-to-work-envs-files-being-sent-as-context/14027", "published_date": "2024-08-31", "accessed_at": "2025-10-30T14:19:00Z" }] }] }

---

[1] [3] [4] Introducing Plan Mode · Cursor
https://cursor.com/blog/plan-mode

[2] [5] [6] Boosting Productivity with Cursor's New "Plan" Mode - Discussions - Cursor - Community Forum
https://forum.cursor.com/t/boosting-productivity-with-cursor-s-new-plan-mode/136015

[7] [14] [18] [19] [21] [23] [31] New Coding Model and Agent Interface · Cursor
https://cursor.com/changelog/2-0

[8] [9] [11] Cursor 2.0.x - what are parallel agents doing? - Discussions - Cursor - Community Forum
https://forum.cursor.com/t/cursor-2-0-x-what-are-parallel-agents-doing/138911

[10] Introducing Cursor 2.0 and Composer · Cursor
https://cursor.com/blog/2-0

[12] Cursor 2.0 Has Arrived — And Agentic AI Coding Just Got Wild | by Joe Njenga | AI Software Engineer | Oct, 2025 | Medium
https://medium.com/ai-software-engineer/cursor-2-0-has-arrived-and-agentic-ai-coding-just-got-wild-65bbbd3be4ec

[13] [15] [16] [28] [34] [35] Cursor 2.0: How Composer and Multi-Agent Coding Are Redefining Software Development (UK Guide 2025) | Grow Fast Blog
https://www.grow-fast.co.uk/blog/cursor-2-composer-multi-agent-development-uk-guide-2025

[17] Cursor 1.7 vs 1.6 (2025): Upgrade Guide & Key Differences
https://skywork.ai/blog/cursor-1-7-vs-1-6-2025-comparison-upgrade-guide/

[20] [22] [27] Changelog · Cursor
https://cursor.com/changelog

[24] MVP Development with Cursor - my full setup (TDD, rules, planning, agents, more) : r/ChatGPTCoding
https://www.reddit.com/r/ChatGPTCoding/comments/1kl5adu/mvp_development_with_cursor_my_full_setup_tdd/

[25] [26] How I use Cursor (+ my best tips)
https://www.builder.io/blog/cursor-tips

[29] [30] I just found out about Context7 MCP Server and it's awesome! : r/ChatGPTCoding
https://www.reddit.com/r/ChatGPTCoding/comments/1k6uyjp/i_just_found_out_about_context7_mcp_server_and/

[32] [33] Browser Controls, Plan Mode, and Hooks · Cursor
https://cursor.com/changelog/1-7

[36] [38] [39] [40] Cursor: Excluding Files/Directories
https://ai-learningpath.nstech.com.br/pages/cursor/cursor_excluding_files.html

[37] Mastering Context: Codebase Indexing and @-References
https://www.educative.io/courses/advanced-cursor-ai/mastering-context-codebase-indexing-and--references

[41] [42] BIG SECURITY RISK! .cursorignore doesn't seem to work, .envs files being sent as context - Bug Reports - Cursor - Community Forum
https://forum.cursor.com/t/big-security-risk-cursorignore-doesnt-seem-to-work-envs-files-being-sent-as-context/14027