T03 # Permission Modes

Technical Beginner

Tool access control, from safest to most permissive

Available Modes

Mode	Canonical name	Activation	Recommended use
Default	`default`	(none)	Daily development
Auto-accept edits	`acceptEdits`	`Shift+Tab`	Code reviews
Plan	`plan`	`Shift+Tab x2` or `/plan`	Analysis without modification
Don't Ask	`dontAsk`	`/permissions` or `permissions.allow` rules	Restrictive workflows, silent auto-deny if not pre-approved
Auto (AI classifier)	`auto`	`permissions.defaultMode: "auto"`	Long tasks, fewer interruptions
Full bypass	`bypassPermissions`	`--dangerously-skip-permissions`	Headless CI/CD, sandboxed
Fewer prompts	—	`/fewer-permission-prompts`	Generates an allowlist from transcripts (shipped as `/less-permission-prompts` in v2.1.111)

CLI activation: `claude --permission-mode <mode>` accepts `default`, `plan`, `acceptEdits`, `bypassPermissions`. **Persistent activation:** `permissions.defaultMode` key in `settings.json`.

Note: `--dangerously-skip-permissions` no longer prompts for writes to `.claude/skills/`, `.claude/agents/`, and `.claude/commands/` (v2.1.121); the rest of `.claude/` (settings, hooks) and `.git/` stay protected even under full bypass. The `auto` mode relies on a classifier model that evaluates each tool call before execution, less friction, not a security boundary.

Tool Whitelist

```
# Allow only specific tools
claude --allowedTools "Read,Grep,Glob"

# Block specific tools
claude --disallowedTools "Bash,Write"

# Useful combinations
claude --allowedTools "Read,Edit,Bash(git*)"
```

Configuration in settings.json

```
{
  "permissions": {
    "allow": [
      "Bash(git log*)",
      "Bash(npm test*)",
      "Read",
      "Edit"
    ],
    "deny": [
      "Bash(rm*)",
      "Bash(sudo*)"
    ]
  }
}
```

Permission Hierarchy

Permissions accumulate and are inherited in this order:

1. `~/.claude/settings.json`: global user
2. `.claude/settings.json`: project (shared)
3. `.claude/settings.local.json`: project (local, gitignored)
4. CLI flags: session only

Glob Patterns for Bash

```
# Allow git only
"Bash(git *)"

# Allow npm test and build
"Bash(npm test*)",
"Bash(npm run build*)"

# Allow file reading
"Bash(cat *)",
"Bash(ls *)"
```

Best Practices

CI/CD: Always use `--dangerously-skip-permissions` with a sandboxed environment (Docker, ephemeral container). Never on a shared production machine.

Sensitive projects: Restrict Bash tools with precise globs in `.claude/settings.json`. Commit this file so the whole team uses the same constraints.

Audit: Claude's actions are logged in `~/.claude/logs/`. Verifiable at any time.