



Your cheat sheet for GitHub Copilot CLI commands, syntax, and workflows.

Last updated: 2026-02-02

Three Interaction Modes

Mode	How to Use	Best For
Interactive	<code>copilot</code>	Exploration, multi-turn conversations, iteration
Plan	<code>/plan</code> or <code>Shift+Tab</code>	Complex tasks, reviewing approach before coding
Programmatic	<code>copilot -p "prompt"</code>	Automation, scripts, CI/CD pipelines

Essential Slash Commands

Core Commands

Command	Description
<code>/help</code>	Show available commands
<code>/clear</code>	Clear conversation history
<code>/model</code>	Show or switch AI model
<code>/exit</code>	End the session
<code>/plan</code>	Create implementation plan before coding
<code>/review</code>	Run code-review agent on staged/unstaged changes
<code>/delegate</code>	Hand off task to Copilot coding agent on GitHub
<code>/diff</code>	Review changes made in current directory (experimental)

Session Management

Command	Description
<code>/session</code>	Show session info and workspace summary
<code>/usage</code>	Display session usage metrics

<code>/context</code>	Show context window token usage
<code>/compact</code>	Summarize conversation to reduce context
<code>/share</code>	Export session as markdown or GitHub gist
<code>/rename</code>	Rename the current session
<code>/resume</code>	Switch to a different session

Permissions

Command	Description
<code>/allow-all</code>	Auto-approve all permission prompts (use with caution)
<code>/yolo</code>	Alias for <code>/allow-all</code>

Directory Access

Command	Description
<code>/add-dir <path></code>	Add a directory to allowed list
<code>/list-dirs</code>	Show all allowed directories
<code>/cwd</code> or <code>/cd</code>	View or change working directory

Authentication

Command	Description
<code>/login</code>	Log in to GitHub Copilot
<code>/logout</code>	Log out of GitHub Copilot

Configuration

Command	Description
<code>/theme</code>	View or set terminal theme
<code>/terminal-setup</code>	Enable multiline input support
<code>/user</code>	Manage GitHub accounts
<code>/feedback</code>	Submit feedback to GitHub
<code>/init</code>	Initialize Copilot instructions for repository

@ Syntax for Context

File References

```
@filename.js      # Single file
@src/api/users.js # File with path
@src/             # Entire directory
@src/**/*.ts     # Glob pattern
```

Multiple Files

```
> @file1.js @file2.js Compare these implementations
> @src/api/ @tests/ Generate tests for all API endpoints
```

Best Practices

- Start specific, expand if needed
- Use glob patterns for targeted searches
- Combine files for cross-file analysis

Built-in Agents

Agent	How to Invoke	Purpose
Plan	<code>/plan</code> or <code>Shift+Tab</code>	Step-by-step implementation plans
Code-review	<code>/review</code>	Focused review of staged/unstaged changes
Explore	<i>Automatic</i>	Codebase analysis (used internally)
Task	<i>Automatic</i>	Tests, builds, lints (success = brief summary, failure = full details)

Use `/agent` to browse and select from your custom agents.

 [Agents Documentation](#)

Custom Agents

Create `AGENTS.md` or `*.agent.md` files:

```
---
description: Frontend specialist with expertise in React and TypeScript
tools: ["read", "edit", "search"]
---

## Frontend Agent

You are a frontend specialist with expertise in React and TypeScript.

**Focus Areas:**
- Component architecture
```

- Accessibility (WCAG 2.1 AA)
- Performance optimization

💡 **Required:** The `description` field in YAML frontmatter is required. Other fields like `name`, `tools`, and `target` are optional. Tool aliases: `read`, `edit`, `search`, `execute`, `web`, `agent`.

Skills System

Using Skills

Skills are **automatically triggered** based on your prompt matching the skill's description:

```
> Review this code for security issues
# Your "security-audit" skill activates automatically

> Generate tests for the login function
# Your "generate-tests" skill activates automatically
```

Managing Installed Skills

Command	Purpose
<code>/skills list</code>	Show all installed skills
<code>/skills info <name></code>	Get skill details
<code>/skills add <name></code>	Enable a skill
<code>/skills remove <name></code>	Disable a skill
<code>/skills reload</code>	Reload after editing

Skills trigger automatically when your prompt matches their description - no manual activation needed.

Creating Skills

Create `~/.copilot/skills/skill-name/SKILL.md`:

```
---
name: my-skill
description: What this skill does and when to use it
---

# My Skill

Instructions for the skill...
```

Required properties: `name` (lowercase, hyphens), `description` . Optional: `license` .

 **Official docs:** [About Agent Skills](#)

MCP Servers

Common Servers

Server	Purpose
<code>github</code>	Issues, PRs, repositories (included by default)
<code>filesystem</code>	Enhanced file operations
<code>postgres</code>	Database inspection

Using MCP

```
> Get issue #42 details           # Uses GitHub MCP
> List open PRs                   # Uses GitHub MCP
> Create a PR for this branch    # Uses GitHub MCP
```

Plugins

Extend Copilot CLI with community plugins:

Command	Purpose
<code>/plugin list</code>	See installed plugins
<code>/plugin marketplace</code>	Browse available plugins
<code>/plugin install <name></code>	Install a plugin

Common Workflows

Security Review

```
copilot -p "Review @src/auth/ for security vulnerabilities"
```

Code Review

```
copilot
> /review                                # Review staged changes
> @src/api/users.js Review for security, performance, best practices
```

Test Generation

```
copilot -p "@src/utils/validation.js Generate Jest tests with edge cases"
```

Debugging

```
copilot
> @src/api/payments.js Users report $10.20 + $5.10 shows as $15.299999
> Debug why this happens
```

Git Commit Message

```
copilot -p "Generate commit message for: $(git diff --staged)"
```

PR Description

```
copilot -p "Generate PR description for: $(git log main..HEAD --oneline)"
```

Model Selection

Model	Best For
claude-sonnet-4.5	Default, balanced
claude-opus-4.5	Complex architecture decisions
gpt-5-mini	Quick tasks (non-premium)
gpt-4.1	Routine code generation (non-premium)

```
> /model claude-opus-4.5    # Switch model
> /model                    # See available models
```

Session Persistence

Save and Resume

```
# Save current session
> /rename feature-auth

# Later, resume it
copilot --resume feature-auth

# Or continue last session
copilot --continue
```

CI/CD Integration

Basic Usage

```
copilot -p "Security review of @$file" --silent >> review.md
```

Pre-commit Hook Example

```
#!/bin/bash
STAGED=$(git diff --cached --name-only --diff-filter=ACM | grep -E '\.(js|ts)$')
for file in $STAGED; do
  copilot -p "Quick security review of @$file - critical issues only"
done
```

Quick Tips

1. Use **-p** for one-off questions - Faster than interactive mode
2. Reference files with **@** - Gives Copilot full context
3. Use **/plan** for complex tasks - Review approach before coding
4. Switch models for different tasks - Opus for architecture, mini for routine
5. Save sessions - Resume work later with full context
6. Use **--silent** in scripts - Cleaner CI/CD output

Keyboard Shortcuts

Shortcut	Action
Shift+Tab	Toggle Plan Mode
Ctrl+C	Cancel current operation
Esc	Cancel current input or exit menus

<code>Ctrl+L</code>	Clear the screen
<code>!command</code>	Run shell command directly (e.g., <code>!git status</code>)

Resources

- [GitHub Copilot CLI for Beginners Course Repository](#)
- [GitHub Copilot CLI Docs](#)
- [MCP Server Registry](#)

Generated from GitHub Copilot CLI for Beginners course materials.