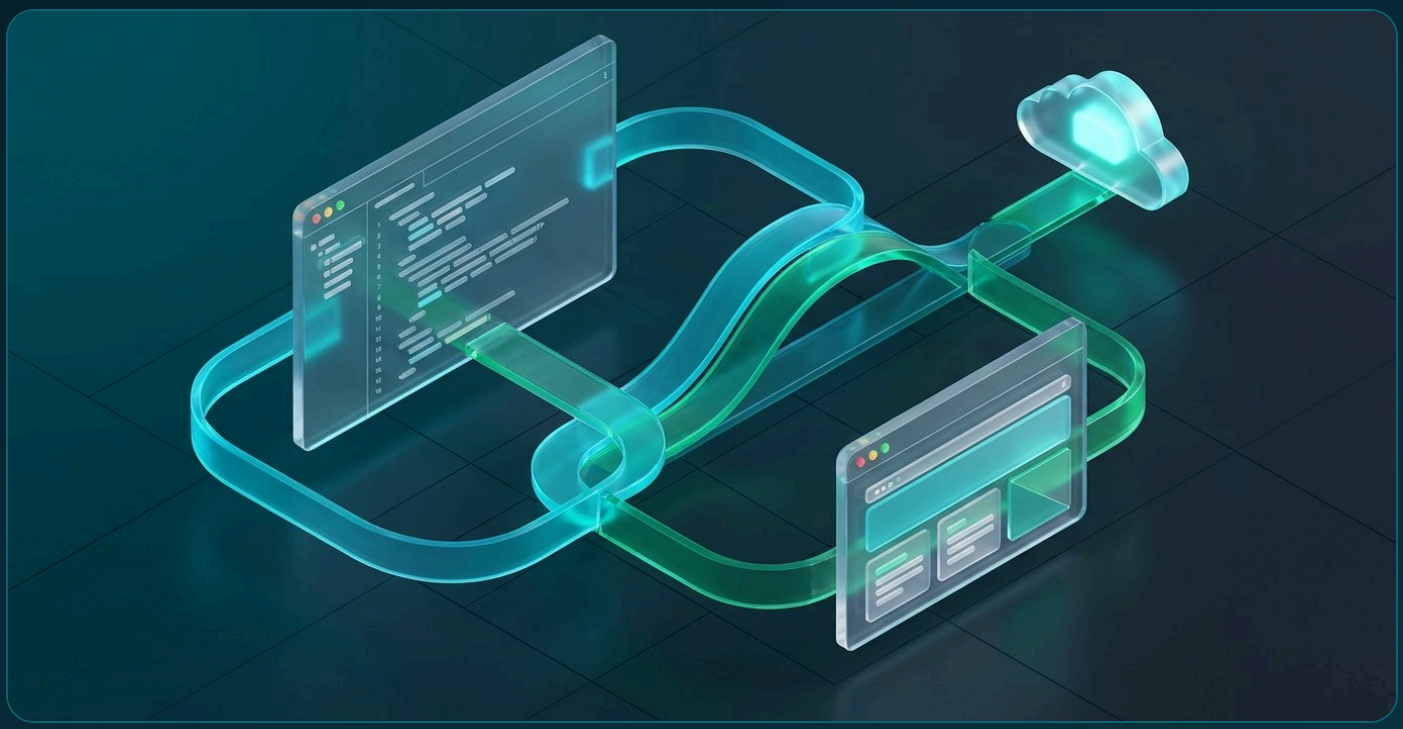
AI IDES & AGENTIC WORKFLOWS • FIELD GUIDE

Windsurf & Devin Desktop: Cascade & Autonomous Tasks

Master flow-state agentic coding in Windsurf and Devin Desktop—combining Cascade & Devin Local multi-file edits, Memories & Rules, live Previews, MCP integrations, and cloud-to-desktop Devin handoffs.

Cascade & Devin Local | Agent Command Center | Rules, Memories & AGENTS.md | Live Previews & Click-to-Edit | MCP Servers (100-Tool Cap) | Local-to-Cloud Devin Handoffs



Stay in Agentic Flow

Unify real-time cursor, terminal, and file-graph awareness across `Cascade` and the Rust-powered `Devin Local` harness.

Codify Team Standards

Combine scoped `.devin/rules/` & `.windsurf/rules/`, slash-invoked Workflows, `AGENTS.md`, and auto-saved Memories.

Scale Local to Cloud

Dispatch backlog tickets to parallel Devin Cloud sessions, then pull draft PRs into Devin Desktop for interactive Live Preview polish.

From Windsurf Editor to Devin Desktop: The Flow Mental Model

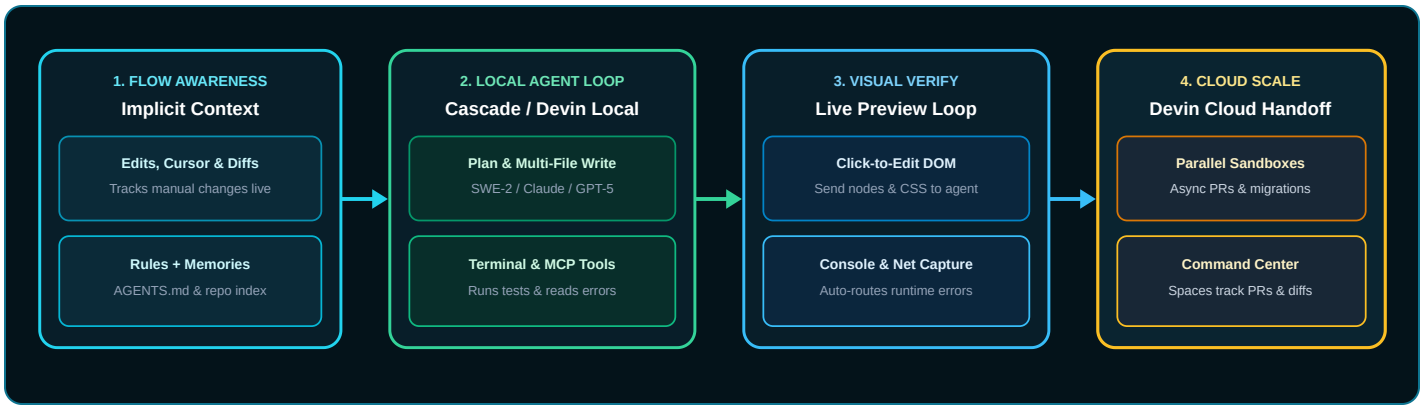
Following Cognition AI's July 2025 acquisition of Windsurf and the **June 2, 2026 rebrand to Devin Desktop**, Windsurf's signature real-time **Flows** architecture merged with Devin's autonomous cloud engineering engine. Today, one workspace orchestrates local keystroke-speed edits and parallel cloud PR agents side by side.

Quick Navigation

1. Flow Architecture & Rebrand Map p. 2
2. Mode & Pricing Decision Matrix p. 3
3. Annotated 7-Control IDE HUD Tour p. 4
4. Cascade, Planning & Multi-File Loops p. 5
5. Rules, Memories & Workflows Anatomy p. 6
6. Agent Command Center & Spaces p. 4, 8
7. Live Previews, Click-to-Edit & MCP p. 7
8. Local-to-Cloud Devin Handoffs p. 8
9. 5 Copy-Paste Rules & Workflow Recipes p. 9
10. Security, ZDR & Team Governance p. 10
11. Cross-Tool Ticket-to-Merge Pipeline p. 11
12. Troubleshooting Clinic & Pre-Flight p. 12

The 4-Stage End-to-End Agentic Flow Loop

Unlike chat sidebars that force you to manually paste files and stack traces, Windsurf / Devin Desktop maintains a live **Flow state** linking your editor buffer, terminal exit codes, browser DOM, and remote cloud sessions:



1 What Changed in Late 2026

- **Rebrand (June 2, 2026):** Windsurf Editor became **Devin Desktop**; `windsurf.com` redirects to `devin.ai/desktop`.
- **Devin Local (July 1, 2026):** Rewritten in Rust (~30% more token-efficient), now the default local agent alongside legacy `Cascade`.
- **Agent Client Protocol (ACP):** Run Claude Agent, Codex, or OpenCode inside the same Command Center Spaces.

2 Backward Compatibility Guarantee

- **Rules & Workflows:** `.devin/rules/` takes precedence, while `.windsurf/rules/`, `.windsurfrules`, and `AGENTS.md` still load seamlessly.
- **Ignore Files:** `.devinignore`, `.windsurfignore`, and `.codeiumignore` are all honored during indexing.
- **Unified Quotas:** One subscription covers Devin Desktop, Devin CLI, and Devin Cloud sessions.

Core Principle: Match Latency to Autonomy

Keep tight visual, architectural, and exploratory work inside your local editor loop (Supercomplete → Cascade / Devin Local → Live Preview), and offload well-specified, multi-hour, or parallelizable tickets to Devin Cloud.

Choosing the Right Agent Mode & Plan Tier

Burning frontier reasoning quota on boilerplate completions slows you down, while relying on Tab autocomplete for a 14-file schema migration invites regressions. Use this matrix to pick the optimal execution surface every time.

Mode / Surface	Execution Scope	Best-Fit Engineering Tasks	Context & Tools	Quota & Latency Profile
Supercomplete / Tab `Tab` • `Cmd+I`	Single cursor, next-edit jump, or inline selection diff	Writing functions, type signatures, test cases, repetitive edits	Open tabs, recent keystrokes, local AST, clipboard	**Unlimited** on all plans (incl. Free); sub-200ms latency
Cascade Chat / Ask Read-only Q&A	Zero file mutations; explains code, traces bugs, drafts specs	Onboarding to unfamiliar services, root-cause tracing, PR Q&A	Indexed repo, `@file`, `@symbol`, terminal, MCP reads	Low quota draw; ideal with fast native `SWE-2`
Cascade Write / Devin Local Default Agent	Multi-file edits, shell commands, test loops, Live Preview	Feature builds, refactors (2–25 files), click-to-edit UI, CI fixes	Full Flow state, terminal stderr, Rules, Memories, MCP	Draws daily/weekly quota; Devin Local (Rust) saves ~30%
Planning Mode Plan → Approve	Generates persistent Markdown blueprint before writing code	Cross-cutting architecture, DB migrations, auth rewrites	Deep repo search + human checkpoint before mutations	Higher upfront reasoning; prevents costly rework loops
Devin Autonomous Cloud Async Sandbox	Isolated cloud VM, git branches, CI runs, draft PRs in parallel	Backlog triage, dependency upgrades, test sprints, Linear tickets	Repo clone, Devin Secrets, Playbooks, GitHub/Linear	Runs unattended (10+ concurrent); billed from shared quota

Late-2026 Plans, Quotas & Model Economics (As of Nov 2026)

In March–April 2026, Cognition retired legacy monthly prompt credits and $2.25 ACU packs on self-serve tiers in favor of **daily & weekly refreshing quotas** shared across Devin Desktop, Devin CLI, and Devin Cloud, with optional overages billed at raw API token rates.

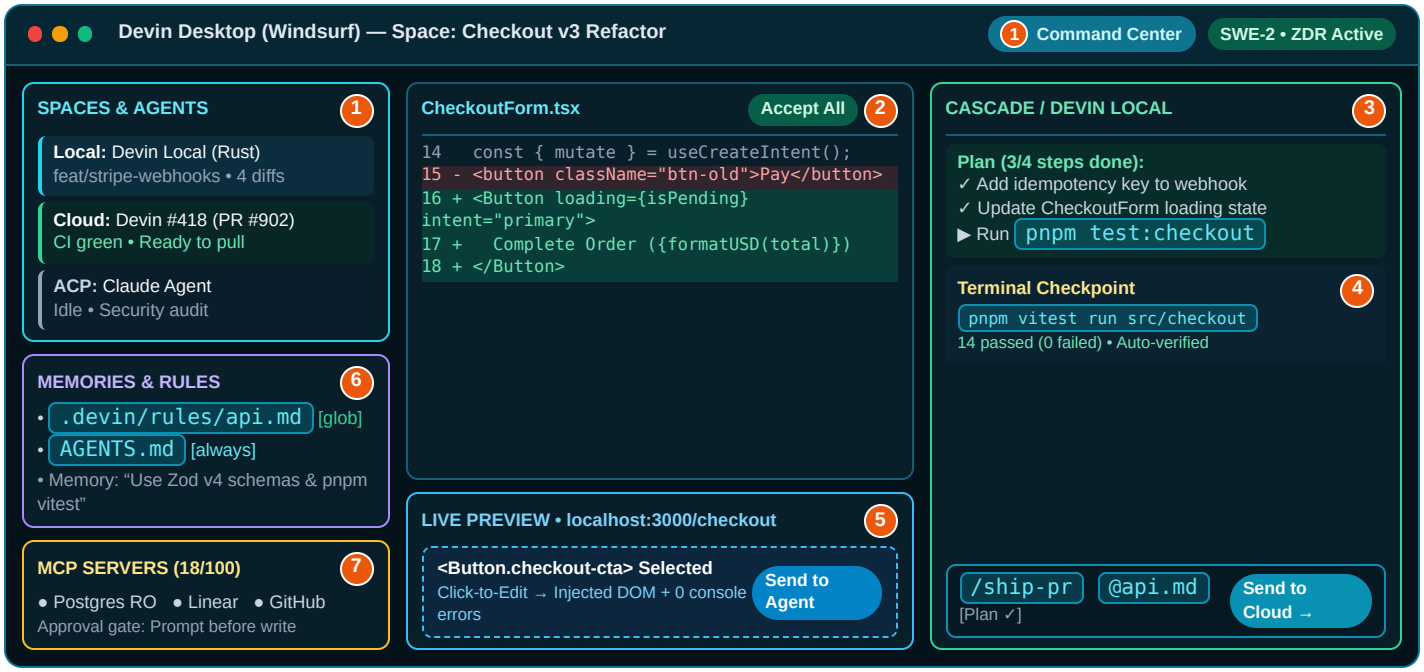
Plan Tier	Published Price	Included Agent & Editor Quotas	Model Access	Governance & ZDR
Free	**$0 / mo** (1 user)	Unlimited Tab & inline edits; light daily agent quota	Limited base models (SWE-1 Lite / entry)	No ZDR opt-out; may train models
Pro	**$20 / mo** (1 user)	Standard daily/weekly quota (Desktop, CLI, Cloud); API overages	Frontier (Claude, GPT-5, Gemini) + `SWE-2`	Opt-in ZDR from profile settings
Max	**$200 / mo** (1 user)	Heavy daily/weekly quotas & higher concurrent cloud sessions	Full frontier + high-reasoning + `SWE-2`	Opt-in ZDR; priority routing
Teams	**$80/mo** + $40/seat	Up to 100–200 users; shared credit pool & team Spaces	Full frontier + `SWE-2`; admin allowlists	**ZDR default ON**; central billing
Enterprise	**Custom** (Unlim.)	Unlimited concurrent agents; custom ACU or token contracts	Custom routing, VPC / AWS GovCloud	SOC 2 Type II, SSO/SCIM, org locks

Model Routing Shortcut (Cut Quota Burn 50%)
Default everyday refactors and tests to Cognition's native **SWE-2**. Switch to frontier reasoning models only for Planning Mode or gnarly concurrency bugs.

When to Toggle Planning Mode First
If a task touches **>5 files**, modifies a DB schema, or changes public API contracts, toggle **Planning Mode** first to approve the checklist before editing disk.

7-Control Command HUD: Cascade, Live Preview & Spaces

Devin Desktop elevates the classic Windsurf editor into a unified **Agent Command Center**. Master these seven high-leverage controls to move seamlessly between multi-agent Spaces, local code diffs, DOM previews, and persistent Memories.



Control-by-Control Breakdown

1 **Agent Command Center & Spaces:** Kanban board grouping local sessions, Devin Cloud PRs, git worktrees, and third-party ACP agents (Claude Agent, Codex) by feature.

2 **Inline Diff & Hunk Review Bar:** Review multi-file hunks individually or accept all; manual edits you type here immediately update Flow Awareness without re-prompting.

3 **Cascade / Devin Local Panel + Mode Switch:** Toggle Write vs. Chat vs. Planning Mode, invoke `/workflows`, attach `@files` or rules, and escalate any thread to Devin Cloud.

4 **Terminal Execution Checkpoint:** Agent runs tests, linters, and builds inside a sandboxed shell, automatically ingesting stack traces until green.

5 **Live Browser Preview & Click-to-Edit:** Embedded webview (or Chromium proxy) lets you click any DOM node or console error and inject its exact selector and styles into context.

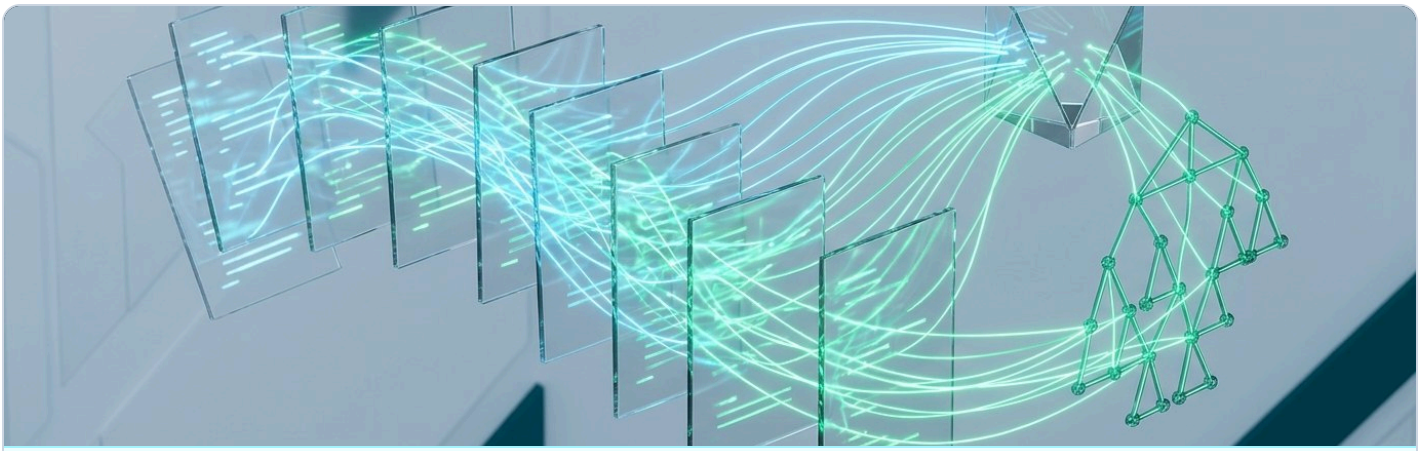
6 **Memories & Rules Inspector:** Inspect active `.devin/rules/`, `.windsurf/rules/`, `AGENTS.md`, and auto-generated Memories; prune stale facts in one click.

7 **MCP Server & Tool Budget Meter:** Displays connected MCP servers against the **100 active tools cap** and enforces per-tool approval prompts for write actions.

Pro Tip — Parallel Worktrees: Open two Spaces on separate git worktrees so Devin Local builds a frontend flow while a second local agent refactors backend tests with zero branch collisions.

Mastering Flow Awareness, Planning & Terminal Loops

The secret behind Windsurf's Cascade—and its Rust-rewritten successor **Devin Local**—is *bidirectional Flow state*: the agent watches your manual edits, open buffers, and terminal output in real time, then closes the loop by running verification commands automatically.



Multi-File Orchestration: Cascade / Devin Local indexes your repository AST, tracks real-time human edits, plans coordinated diffs across layers, and loops against live compiler & test-runner output until checks pass.

The 3-Step High-Precision Multi-File Refactor Protocol

1 Anchor & Plan

Enable **Planning Mode** and tag only the entry-point symbols (`@CheckoutRouter`, `@OrderSchema`). Let the agent map all downstream callers and produce a step-by-step checklist before mutating disk.

2 Scoped Diffs

Approve the plan and let Devin Local apply atomic hunks across types, services, and UI components. If you hand-edit a naming convention mid-flight, Flow Awareness automatically adapts subsequent files.

3 Terminal Verify

End every prompt with an explicit verification gate (e.g., *"Run `pnpm typecheck && pnpm vitest run` and fix any regressions"*). The agent reads stderr directly and self-corrects without copy-pasting.

Cascade (Legacy) vs. Devin Local (Rust)

- **~30% Higher Token Efficiency:** Devin Local's Rust harness streams targeted symbol slices instead of re-sending full file buffers on each turn.
- **Shared CLI & Desktop Harness:** Start a local session in Devin Desktop and resume the exact same session in `devin` CLI over SSH or tmux.
- **Revert Checkpoints:** Every agent step creates an instant rollback snapshot so you can undo a misguided 12-file attempt cleanly.

Context Hygiene Rules for Big Repos

- **Start Fresh Per Feature:** Long 40-turn threads dilute attention and inflate API overages; start a new session once a PR merges.
- **Pin Verification Commands:** Put exact test/lint commands in `AGENTS.md` so the agent never guesses `npm` vs. `pnpm`.
- **Use `@mention` Sparingly:** Mentioning 15 whole files wastes quota; let semantic search + AST indexing pull only relevant functions.

Anti-Pattern to Avoid: The "Vague Fix" Loop

Never prompt *"Fix the broken checkout."* Instead, trigger the failing test or reproduce the error in the integrated terminal first—Flow Awareness captures the exact stack trace, exit code, and offending line so Cascade fixes root causes in 1 turn instead of 5.

The 5-Part Customization Blueprint: Rules, Memories & `AGENTS.md`

Repeating your architecture standards every morning wastes quota and invites drift. Devin Desktop / Windsurf layers five distinct customization mechanisms—each with clear precedence, activation triggers, and hard character caps.

COLOR-CODED BLUEPRINT — `.DEVIN/RULES/API-LAYER.MD` (OR `.WINDSURF/RULES/API-LAYER.MD`) + `AGENTS.MD`

```
--- trigger: glob | globs: ["src/api/**/*.ts"] | description: API error & validation rules ---
Architecture (AGENTS.md / Rules): Use Zod v4 for all request schemas; return Result<T, AppError> instead of throwing across service boundaries.
Auto-Memory (Local Inspector): Repo uses pnpm 9, Node 22, and Postgres 16 via Docker Compose port 5439.
Workflow (.devin/workflows/new-endpoint.md): Invoked via /new-endpoint → scaffold route, Zod schema, unit test, run pnpm vitest.
Guardrail (Hard Cap & Scope): Keep each workspace rule ≤ 12,000 chars (global ≤ 6,000 chars); never auto-run destructive DB migrations.
```

1. Frontmatter Trigger
2. Durable Team Rules
3. Persistent Memories
4. Slash Workflows
5. Caps & Guardrails

Directory Precedence & Character Limits (Late 2026)

Mechanism & Path	Scope & Git	Activation Trigger	Hard Limit	Best Used For
Directory Rules `.devin/rules/*.md` `.windsurf/rules/*.md`	Workspace (committed to git)	`always_on`, `glob`, `model_decision`, or `manual` (`@rule`)	**12,000 chars** per rule file	Domain conventions (API, React, SQL, testing) scoped by folder
Cross-Agent Spec `AGENTS.md` (root / subdirs)	Repo-wide (universal standard)	Auto-loaded when working inside root or matching subdirectory	Keep concise (<4,000 chars)	Build/test commands & architecture shared with Codex / Claude
Global Rules `global_rules.md`	User-local (all repos)	Always applied to every local session across workspaces	**6,000 chars** total	Personal tone, TypeScript preferences, commit style
Workflows `.devin/workflows/*.md` `.windsurf/workflows/*.md`	Workspace (committed to git)	Explicit slash invocation (e.g., `/ship-pr`, `/audit-sql`)	Multi-step Markdown SOP	Repeatable multi-step checklists, release prep, migrations
Memories Local Inspector DB	Local per workspace	Auto-saved by Cascade / Devin Local or pinned by user	Auto-pruned; audit weekly	Ephemeral quirks, local ports, env hints learned during debugging

Choosing the Right Rule Trigger

- `always_on`: Use sparingly (max 1–2 core files) so you don't bloat every prompt's system context.
- `glob`: Gold standard—attach `["src/db/**/*.sql"]` only when touching SQL files.
- `model_decision`: Write a razor-sharp 1-line `description` so the router activates it reliably.
- `manual`: Reserve for heavy migration or security-audit playbooks invoked via `@rule-name`.

Migrating from Legacy `.windsurfrules`

- Legacy root `.windsurfrules` still loads, but splitting it into scoped `.devin/rules/*.md` files cuts irrelevant token overhead.
- **Precedence:** When both `.devin/rules/` and `.windsurf/rules/` exist, `.devin/rules/` wins.
- **Cursor Import:** Devin Desktop natively imports `.cursor/rules/*.mdc` into `.devin/rules/`.

Visual DOM Iteration & External Tooling via MCP

Stop describing CSS bugs in paragraphs or switching between six dashboards. **Live Previews** wire your running web app directly into Cascade / Devin Local, while **Model Context Protocol (MCP)** servers connect databases, issue trackers, and observability logs.

1 Live Browser Previews & Click-to-Edit

1. **Launch Preview:** Click the Web icon in the agent toolbar or prompt *"Start dev server and open Preview."* Opens an IDE webview tab or instrumented Chrome/Arc window.
2. **Click-to-Edit DOM:** Select any button, card, or modal to inject its exact DOM node, computed CSS, and component path directly into your prompt.
3. **Auto-Capture Errors:** Console exceptions, hydration mismatches, and failed `fetch()` calls stream straight to Cascade / Devin Local with one click.

One-Click Deploys: Push prototypes to shareable staging URLs directly from the toolbar.

2 MCP Server Architecture & Tool Budget

- **Global Config:** `~/.config/devin/mcp_config.json` (Windows: `%APPDATA%\devin\mcp_config.json`).
- **Project Config:** `.devin/mcp_config.json` in repo root for team-shared tools (Devin Local reads both).
- **100 Active Tools Cap:** Hard cap of **100 active MCP tools**. Disable unused tools per server in the MCP panel to preserve context budget and routing accuracy.
- **Approval Gate by Default:** Devin Local prompts for human approval before executing any MCP tool call unless allowlisted.

Production-Ready `mcp_config.json` Example (Read-Only DB + Linear + GitHub)

```
{
  "mcpServers": {
    "linear": {
      "command": "npx", "args": ["-y", "@linear/mcp-server"],
      "env": { "LINEAR_API_KEY": "${env:LINEAR_API_KEY}" }
    },
    "postgres-readonly": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-postgres", "${env:READONLY_DB_URL}"]
    },
    "github": {
      "serverUrl": "https://api.githubcopilot.com/mcp/",
      "headers": { "Authorization": "Bearer ${env:GITHUB_PAT}" }
    }
  }
}
```

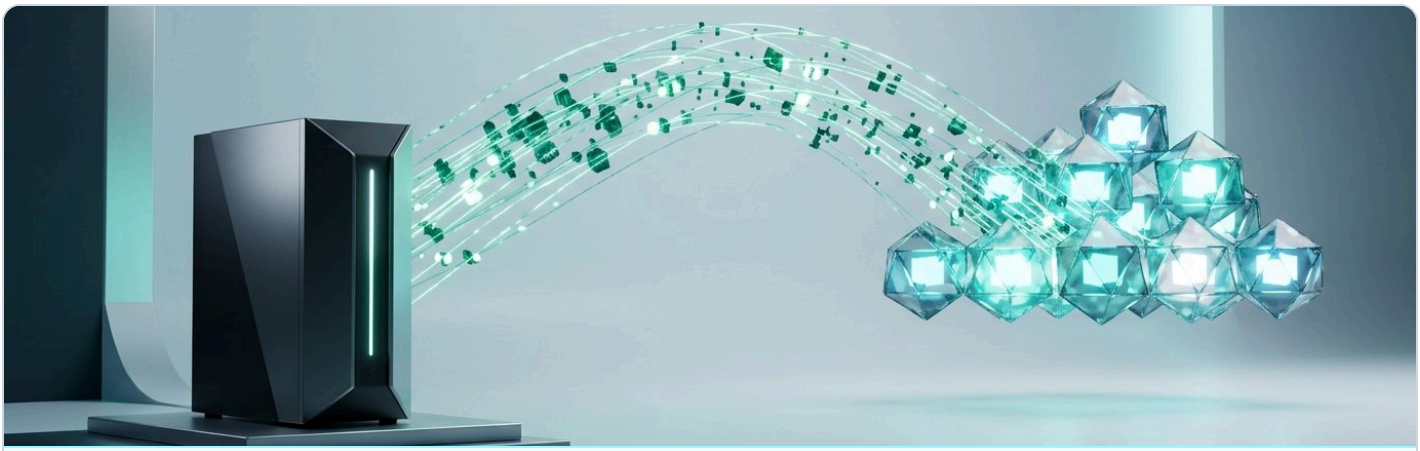
High-Leverage MCP Servers for Full-Stack Teams

MCP Server	Transport	What Cascade / Devin Local Can Do	Recommended Guardrail
Linear / Jira	STDIO / HTTP	Read acceptance criteria, link branches, post PR comments	Allowlist reads; require approval on writes
Postgres / Supabase	STDIO	Inspect live schemas, explain slow queries, verify seed data	Use a **read-only** role; no prod write creds
Playwright / Browser	STDIO	Run headless E2E flows, capture screenshots, assert a11y	Restrict to `localhost` and staging
Sentry / Datadog	Remote HTTP	Pull live stack traces, breadcrumbs, and release diffs	Scope token to read-only project issues

Security Note: Never Hardcode Secrets in `.devin/mcp_config.json`
Always reference environment variables (`${env:VAR_NAME}`) or your OS keychain, and keep personal tokens in global `~/.config/devin/mcp_config.json` rather than committing them to git.

Local Cascade vs. Autonomous Devin Cloud Sessions

The biggest productivity unlock in Devin Desktop is knowing **where** a task should run. Keep high-touch architectural and visual work local in Cascade / Devin Local, and dispatch bounded, verifiable work to parallel **Devin Cloud** sandboxes inside the same Command Center Space.



Seamless Continuity: Escalate any local Planning Mode spec to Devin Cloud with one click, or pull a Devin Cloud draft PR directly into your local Devin Desktop worktree for final Live Preview inspection.

Local vs. Cloud Delegation Matrix

Task Dimension	Keep Local (Cascade / Devin Local)	Delegate to Devin Cloud (Autonomous)
Spec Clarity	Exploratory, ambiguous UX, or evolving product architecture requiring rapid back-and-forth	Clear acceptance criteria, reproducible bug report, or well-defined migration playbook
Verification Signal	Needs human visual eye in Live Preview, local hardware/GPU, or private VPN services	Machine-verifiable via unit/E2E tests, TypeScript compiler, linters, or CI checks
Parallelism	1–2 active local worktrees competing for your laptop CPU/RAM and attention	3–10+ concurrent cloud sandboxes clearing backlog bugs, upgrades, or test debt overnight
Ideal Examples	Designing a new dashboard layout, debugging a tricky race condition, live Click-to-Edit polish	Upgrading React 18→19, adding 40 API integration tests, fixing 6 Sentry errors into draft PRs

1 Desktop → Cloud Escalation Flow

1. Draft and approve a spec in local **Planning Mode** while context is fresh.
2. Click **Send to Cloud** in the Agent Command Center to spawn an isolated Devin Cloud VM on a new git branch.
3. Continue coding on your primary feature locally while Devin Cloud implements, runs CI, and opens a draft PR.

2 Cloud → Desktop Pull-Down Flow

1. Watch the Devin Cloud card move to *Ready for Review* inside your Command Center Space.
2. Click **Open in Local Worktree** to check out the branch instantly without stashing your current local work.
3. Launch Live Preview, apply final Click-to-Edit tweaks in Devin Local, and push to merge.

Rule of Thumb: The “20-Minute Spec Test”

If you can write a crisp 5-bullet acceptance spec + an exact test command in under 2 minutes, and the work takes >20 minutes to execute, **delegate it to Devin Cloud**.

5 Production-Ready Recipes to Drop Into Your Repo Today

Copy these battle-tested configurations directly into `.devin/` (or `.windsurf/`), `AGENTS.md`, and your Cascade / Devin Local prompt box to eliminate repetitive guidance and standardize every PR.

#	Recipe Name	File / Invocation	Outcome & When to Trigger
1	**Scoped API Rule**	`.devin/rules/api.md`	Auto-enforces Zod validation & typed `Result` errors on all API routes
2	`/ship-pr` **Workflow**	`.devin/workflows/ship-pr.md`	Runs typecheck, tests, diff audit, and drafts a structured PR description
3	**Universal** `AGENTS.md`	`AGENTS.md` (repo root)	Single source of truth for Devin Desktop, Devin Cloud, CLI & ACP agents
4	**Click-to-Edit UI Fix**	Live Preview + Cascade	Fixes responsive/a11y regressions in 1 turn using selected DOM context
5	**Cloud Migration Brief**	Command Center → Cloud	Dispatches a bounded, test-verified refactor to an async Devin Cloud VM

Recipe 1 — `.devin/rules/api.md` (Glob Rule)

```
---
trigger: glob
globs: ["src/api/**/*.ts", "src/routes/**/*.ts"]
description: Typed API handler & Zod conventions
---
- Validate req.body & params with Zod v4 schemas.
- Return `Result<T, ApiError>`; never throw raw strings.
- Never log PII, auth headers, or raw payment tokens.
- Add a Vitest integration test for every new route.
```

Recipe 2 — `.devin/workflows/ship-pr.md` (`/ship-pr`)

```
# /ship-pr — Pre-Merge Verification Workflow
1. Run `git status` and review all staged/unstaged diffs.
2. Execute `pnpm typecheck && pnpm lint && pnpm test`.
3. If any check fails, fix root cause and re-run until
green.
4. Scan diff for leftover `console.log` or hardcoded
secrets.
5. Output a concise PR summary: What, Why, Verification,
Risk.
```

Recipe 4 — Live Preview Click-to-Edit Prompt

```
[Selected DOM: <div.pricing-grid> + <button.cta-pro>]
Fix mobile overflow at 375px viewport: stack cards to 1
col,
ensure 44px min tap target on CTA, preserve WCAG AA
contrast.
Verify in Live Preview with 0 console or hydration
errors.
```

Recipe 3 — Root `AGENTS.md` (Cross-Agent Spec)

```
# Repository Engineering Guide (AGENTS.md)
## Commands
- Install: `pnpm install --frozen-lockfile`
- Dev: `pnpm dev` (port 3000) | Test: `pnpm vitest run`
- Typecheck & Lint: `pnpm typecheck && pnpm biome check
.`

## Architecture & Guardrails
- Stack: Next.js 16 App Router, TypeScript strict,
Drizzle ORM.
- Never edit `src/db/migrations/` manually; use `pnpm
db:gen`.
- All new features require unit tests + updated Zod
schemas.
```

Recipe 5 — Devin Cloud Autonomous Handoff Prompt

```
Goal: Migrate all `moment` calls in `src/billing/` to
`date-fns` v4.
Constraints:
- Preserve exact UTC ISO-8601 output and timezone
formatting.
- Remove `moment` from package.json only if zero imports
remain.
Verification:
- Run `pnpm typecheck && pnpm vitest run src/billing`.
- Open a Draft PR with before/after bundle-size delta.
```

> **Workflow Chaining Tip**
> You can call one workflow inside another (e.g., end `/new-endpoint` by instructing the agent to run `/ship-pr`) to guarantee every scaffolded feature passes CI before you even review the diff.

Enterprise Hardening: ZDR, Command Allowlists & Secrets Hygiene

An agentic IDE reads untrusted files, executes shell commands, and invokes external MCP servers with your developer credentials. Pair Cognition's built-in Zero Data Retention (ZDR) with defense-in-depth workspace guardrails.

1 Zero Data Retention (ZDR) & Training Controls

- **Teams & Enterprise:** ZDR is **enabled by default**; customer code is never saved persistently to disk or used to train foundation models without written consent.
- **Pro & Max:** Individuals must explicitly toggle ZDR / training opt-out in account profile settings.
- **Free Plan Warning:** Free tier has no ZDR opt-out—never open proprietary or regulated repositories on Free accounts.
- **Compliance Posture:** SOC 2 Type II certified; federal deployments run on AWS GovCloud with mandatory ZDR.

2 Ignore Files & Real Secrets Isolation

- **Multi-File Enforcement:** Indexing skips `.gitignore`, `node_modules`, and paths listed in `.devinignore`, `.windsurfignore`, or `.codeiumignore`.
- **Don't Rely on Ignore Alone:** Ignore rules govern indexing/viewing—terminal output or prompt injection can still expose local `.env` files.
- **Best Practice:** Keep production secrets out of the workspace; inject at runtime via 1Password/Vault or **Devin Settings → Secrets** for cloud sessions.

5-Layer Defense-in-Depth Governance Checklist

Attack Surface	Real-World Risk (2025–2026 Lessons)	Mandatory Team Control
Auto-Execution / Turbo Mode	Prompt injection in cloned repos or PR comments can trigger destructive shell commands	Disable blanket Turbo mode; allowlist safe read/test commands (`pnpm test`, `git diff`) and require approval for `curl`, `rm`, or deploys
MCP Server Permissions	Unvetted STDIO servers or local config tampering can exfiltrate tokens outside IDE logs	Keep Devin Local's default MCP approval prompt ON; enforce an org-level MCP server allowlist; bind DB MCPs to read-only roles
Open VSX Extensions	Supply-chain malware (e.g., 2026 GlassWorm Open VSX campaign) runs with full dev privileges	Enforce an enterprise extension allowlist; pin verified publishers and audit installed extensions quarterly
Generated Code & Deps	Hallucinated packages (slopsquatting) or subtle auth/SQLi flaws introduced in multi-file diffs	Enforce GitHub branch protections + automated SAST/SCA (Semgrep, Snyk, CodeQL) on every Devin & local PR
Quota & Overage Metering	Runaway frontier reasoning loops or bloated context windows trigger unexpected API-rate overages	Default everyday tasks to `SWE-2`; set monthly Team spend caps; monitor usage telemetry in Devin Admin

Starter `.devinignore` (Mirror to `.windsurfignore`)

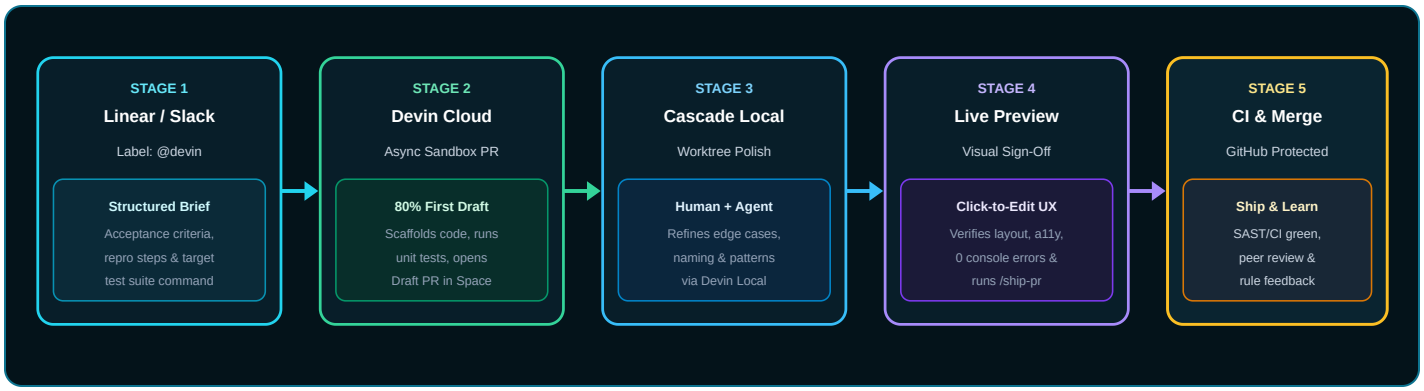
```
# Secrets & credentials
.env*
*.pem
*.key
secrets/
# Heavy build artifacts & fixtures
dist/
coverage/
test-fixtures/large-dumps/
```

Admin Tip: Policy-Enforced Allowlists

Local allowlists inside a developer's user config can be weakened accidentally. On Teams & Enterprise plans, deploy system-level Rules, MCP allowlists, and Auto-Execution ceilings from the **Devin Admin Console** so policies apply uniformly across every laptop and cloud sandbox.

The Ticket-to-Merge Pipeline: Linear → Devin Cloud → Cascade → Ship

High-velocity engineering teams stop treating cloud agents and desktop IDEs as separate worlds. Wire Linear/Slack, Devin Cloud, Devin Desktop (Cascade / Devin Local), Live Preview, and GitHub CI into one continuous shipping conveyor belt.



Stage-by-Stage Operator Playbook

1 Triggering Clean Cloud Runs (Stages 1–2)

- **Tag `@devin` in Linear or Slack** only when the ticket has a concrete repro or acceptance checklist.
- **Share `AGENTS.md` & Rules:** Because Devin Cloud reads the exact same `.devin/rules/` and `AGENTS.md` as Devin Desktop, its draft PR already matches your stack conventions.
- **Parallelize Routine Work:** Queue 3–5 independent bugs or API endpoints at the start of your morning standup; let Devin Cloud draft them while you focus on deep architecture.

2 Local Polish & Compounding Rules (Stages 3–5)

- **Pull Into a Worktree:** Open the cloud PR in Devin Desktop without disrupting your main branch; review hunks and run Live Preview for UI flows.
- **Invoke `/ship-pr`:** Standardize typecheck, lint, test, and PR summary generation before requesting human peer review.
- **Compound Every Review Comment:** Whenever a human reviewer catches a recurring pattern, spend 30 seconds adding 1 bullet to `.devin/rules/` so neither Cascade nor Devin Cloud repeats the mistake.

Weekly Team Rhythm: The 10-Minute “Rule & Memory Gardening” Ritual

Every Friday, have one engineer prune outdated local Memories, promote high-value debugging discoveries into git-committed `.devin/rules/*.md` files, and archive unused MCP servers. Teams that garden their rules weekly cut agent rework turns by **40%+** and keep daily quota burn predictable.

Troubleshooting Clinic (Top 7 Fixes) & Pre-Flight Checklist

When an agentic loop stalls, ignores a rule, or burns quota too fast, diagnose the exact root cause in seconds with this field-tested clinic—then run the 6-point pre-flight check on every new workspace.

#	Symptom	Root Cause	Instant Fix
1	**Rule file silently ignored**	File exceeded 12,000-char workspace cap, invalid YAML frontmatter, or glob mismatch	Trim under 12k chars, verify `trigger: glob` syntax, and confirm active badge in the Rules Inspector
2	**MCP tools missing or failing**	Exceeded the 100 active tools cap, invalid JSON, or missing env var in `mcp_config.json`	Disable unused tools in MCP panel to stay <100; validate JSON and export env vars before launching IDE
3	**Agent loops on same test error**	Stale conversation context or missing runtime detail (e.g., wrong Node/DB port)	Revert checkpoint, start a fresh session, run failing test in terminal once so Flow captures exact stderr
4	**Rapid daily quota exhaustion**	Defaulting routine edits to frontier high-reasoning models or 40+ turn monolithic chats	Switch everyday tasks to `SWE-2`, use Devin Local (Rust) for ~30% token savings, start fresh per PR
5	**Live Preview Click-to-Edit won't attach**	Dev server running on blocked port, non-Chromium browser, or HTTPS mixed-content block	Launch Preview from Cascade toolbar (IDE webview or Chrome/Arc proxy) and verify `localhost` port
6	**Wrong conventions across files**	Stale auto-generated Memory contradicting newer `.devin/rules/` or `AGENTS.md`	Open Memories Inspector, delete conflicting local memories, and codify the canonical rule in git
7	**Devin Cloud PR diverges from local**	Cloud sandbox lacked env setup, build script, or clear acceptance criteria	Add exact install/test commands to root `AGENTS.md` and store required keys in Devin Settings Secrets

6-Point Workspace Pre-Flight Checklist

- ☑ **1. Root `AGENTS.md` Committed:** Lists exact package manager, dev port, typecheck, lint, and test commands for local & cloud agents.
- ☑ **2. Scoped `.devin/rules/` Active:** Core rules split by `glob` trigger (each ≤12k chars); legacy `.windsurfrules` retired or synced.
- ☑ **3. ZDR & Ignore Files Verified:** Zero Data Retention confirmed in profile/org admin; `.devinignore` & `.windsurfignore` exclude secrets & dumps.
- ☑ **4. Safe Execution & MCP Caps Set:** Turbo mode capped; MCP active tools <100; DB MCPs scoped read-only with write-approval prompts ON.
- ☑ **5. `/ship-pr` Workflow Ready:** Reusable workflow in `.devin/workflows/` enforces green CI checks and diff hygiene before every PR.
- ☑ **6. Model Routing Tuned:** Everyday local loops default to `SWE-2` on Devin Local; Planning Mode & Devin Cloud reserved for heavy lifts.

Final Takeaway: Flow Locally, Scale Autonomously

Windsurf's evolution into Devin Desktop gives you the best of both worlds: sub-second local flow with Cascade / Devin Local & Live Previews, backed by parallel Devin Cloud engineers sharing the exact same rules, workflows, and verification gates.