AI INFRASTRUCTURE · SEPTEMBER 11, 2026

The Claude GUI App: Beginner, Intermediate and Advanced Capabilities (September 2026)

The Claude desktop app is three tabs — Chat, Cowork and Code — on one Claude Code engine, and the capability ladder maps onto real product boundaries rather than difficulty levels. Covers beginner Chat habits (quick entry, uploads vs sandbox vs real folder access, Projects, connectors vs desktop extensions), intermediate Cowork (approval modes, scheduled tasks, plugins, the built-in browser…

30 min read · 35 sources · Updated Sep 11, 2026

2026 | Claude Code | Claude Cowork | Claude Desktop | MCP connectors | computer use | enterprise configuration | permission modes

CONTENTS

Three tabs, three jobs, one app 2
Before you start: what runs where, on what plan 3
Beginner: Chat as a daily driver 4
- Files: three different things that sound the same 4
- Memory and the switches worth deciding deliberately 5
- Effort, models and what they cost you 5

Beginner: Projects, and keeping context under control 5
- Chat Projects versus Cowork Projects 6

Beginner: connectors, extensions and the three meanings of “my files” 6
- Per-connector permissions, which are the important part 7

Intermediate: Cowork and handing off whole tasks 7
- The three approval modes, and what they actually gate 8

Intermediate: scheduled tasks, Cowork projects and plugins 9
- Cowork scheduled tasks 9
- Cowork Projects and plugins 10

Intermediate: two ways to give Claude a browser, and computer use 10
- Computer use 11

Intermediate: Dispatch and moving between surfaces 11
- Where a session runs determines what it can reach 12

Advanced: the Code tab in depth 12
Advanced: environments, automation and the customisation stack 14
- The seven ways to steer behaviour 14

Advanced: enterprise configuration and governance 16
Where the edges are: gaps, known issues and gating 17
- Missing from specific surfaces 17
- Still labelled beta or research preview 18
- Common failure modes 18

Bottom line 18
Sources 19

Three tabs, three jobs, one app

The Claude desktop app is three products sharing one window. **Chat** is the conversational Claude you know from claude.ai. **Cowork** is an agentic mode where you hand over a whole task and come back to finished work. **Code** is a full development environment built on Claude Code. Cowork and Code run on the same engine underneath — Claude Code — which is why both can spin up sub-agents, isolate their work, and keep going on long jobs without the conversation timing out[1][2].

That one fact explains most of the confusion people have with the app. The three tabs are not difficulty levels of the same thing; they are three different shapes of work that happen to look similar on screen.

3	>90%	2,362	200k
Tabs in the desktop app	Cowork usage that is not software development	Apps in the connector directory	Context window on Free, Pro and Max
`Chat, Cowork, Code`	business ops + content creation ≈ half of all usage	`as of Sept 2026`	500k on the default Enterprise model

The practical rule for choosing a tab: **count the sources**. A task that lives in one question or one document sits in Chat. A task that pulls from several files, tools or sites and ends in a finished deliverable belongs in Cowork. A task that ends in working software belongs in Code[2][3]. The warning sign that you've stayed in Chat too long is finding yourself pasting in a third file or re-explaining the same context[3].

Anthropic's usage data supports that split. When the company looked at how people use Cowork, **more than 90% of it wasn't software development**; business operations and content creation together account for roughly half of all usage[28].

YOU'RE ABOUT TO…	WHERE IT LIVES	WHAT IT GIVES YOU
Ask, brainstorm, draft, think something through turn by turn	**Chat**	Quick entry, screenshots, dictation, desktop connectors
Hand off a multi-step task that ends in a deliverable or runs on a schedule	**Cowork**	Folder access, sub-agents, scheduled tasks, plugins
Write, test, run and ship code	**Code**	Sessions, permission modes, diff review, PR monitoring

One naming trap: the *desktop app* and *Claude Desktop* now refer to the same product. Older guides describing "Claude Desktop" as separate from "Claude Code" are describing a product that no longer exists in that form[1][2].

Before you start: what runs where, on what plan

Two things determine what you can do: your operating system and your plan.

Platforms. macOS 11+, Windows 10+ (x64 and arm64), ChromeOS through the Android app. Linux is a beta with a narrow target — Ubuntu 22.04+ or Debian 12+, x86_64 or arm64, installed from Anthropic's apt repository rather than a downloaded `.deb`, because the standalone package doesn't update itself[4][21]. The floor moved during 2026: release v1.49585.0 on 8 September updated the runtime to Electron 44 and raised the macOS requirement to **Ventura 13**[27].

Plans. Chat works on every plan including Free. Cowork and Claude Code need a paid plan. Pro is **$17/month billed annually ($200 up front) or $20 monthly**; Max starts at **$100/month** for 5x usage or $200 for 20x; Team is **$20/standard seat** annually ($25 monthly) with a $100 premium seat at 5x usage; Enterprise is **$20/seat plus usage at API rates**. Projects are not on Free[30].

Where models run. Never on your machine — Sonnet, Opus and Fable run in Anthropic's cloud whichever surface you use, so no GPU makes the app faster. Cowork's default changed during 2026: sessions now run in an isolated, per-session environment on Anthropic's servers, with your computer as the thing Claude reaches back into for files rather than the thing doing the work[7][32]. The older on-device session mode is what raises the hardware bar — on Linux, Cowork's VM needs hardware virtualisation, QEMU, UEFI firmware, `/dev/kvm` membership and **about 25GB free disk** for the workspace image[21][32].

Usage pools. Your five-hour rolling allowance and the weekly cap on top are shared across web, desktop, mobile and Claude Code[32]. Long contexts, file creation, high effort and Cowork all draw more from that pool than a plain question. Click the usage ring next to the model picker in the Code tab for session context usage and plan usage for the period[1].

- 12 Jan 2026 — **Cowork research preview**
  Max plans, macOS only, running in a local VM
- 16 Jan 2026 — **Cowork opens to Pro**
  Claude Code access added to Team standard seats
- 25 Feb 2026 — **Scheduled tasks in Cowork**
  Plus a Customize section grouping skills, plugins and connectors
- 17 Mar 2026 — **Dispatch preview**
  Persistent thread from phone to desktop

23 Mar 2026 **Computer use preview**
Pro and Max; Claude can point, click and navigate your screen

9 Apr 2026 **Cowork generally available**
macOS and Windows, with admin controls and OpenTelemetry

30 Jun 2026 **Desktop on Linux (beta)**
Ubuntu 22.04+ and Debian 12+, apt repository

7 Jul 2026 **Cowork on web and mobile**
Remote sessions in beta, starting with Max

26 Aug 2026 **Built-in browser**
Cowork gets its own side-panel browser; Claude in Chrome goes GA

8 Sep 2026 **v1.49585.0**
Electron 44; macOS 13 Ventura becomes the floor

10 Sep 2026 **v1.52386.0**
Queued messages, transcript view default, session retention keys

Beginner: Chat as a daily driver

Chat in the desktop app is the same Claude as claude.ai, with four things that come from running natively on your computer[2][3]:

- **Quick entry.** Double-tap Option on Mac to pull Claude up over whatever you're working on, in a window that stays on top as you switch apps.
- **Screenshots and window sharing.** Capture a screenshot or share a window so Claude sees exactly what you're looking at.
- **Dictation.** Talk through a problem instead of typing.
- **Desktop connectors.** Local tools and services, covered in the next section.

Quick entry is the one that changes habits. You're staring at an unfamiliar dashboard, you double-tap Option, drag over the window to screenshot it, and ask "what do these metrics mean?" Claude answers in the overlay while the dashboard stays in view[2][3].

Files: three different things that sound the same

An upload. Drag a PDF, image or spreadsheet into the message box and Claude reads a *copy*. Your original is untouched[24].

Code execution and file creation. A switch under Settings → Capabilities. With it on, Claude can build a spreadsheet with working formulas, run Python over a CSV, or turn a document into a slide

deck. That code runs in a sandbox on Anthropic's servers, and anything it makes arrives as a **download capped at 30MB per file**. Anthropic no longer supports artifacts without this switch on [24].

Real folder access. Claude reading, creating, editing and (with permission) deleting files inside a folder you've pointed it at. This needs a desktop extension with filesystem access or Cowork with a connected folder. It's where the utility is, and where the care goes [24].

Memory and the switches worth deciding deliberately

Memory is **on by default for Free, Pro and Max, and off by default for Team and Enterprise organisations** [26]. What Claude remembers appears under Settings → Memory as Topics you can read, edit or delete [26]. `Include sensitive topics in memory` is off by default, and government ID numbers, criminal history, financial account numbers and immigration status are never saved regardless [23] [26]. A `Start import` button brings memory across from another AI provider, and the ghost icon atop a new chat starts an incognito conversation [24].

One privacy switch is worth finding on day one: on a fresh consumer account the `Help improve our AI models` toggle under Settings → Privacy was **on by default** in a September 2026 walkthrough [24].

Effort, models and what they cost you

The model indicator shows both model and effort level — a fresh Free account typically reads `Sonnet 5, Medium`. Higher effort means the model thinks harder and **uses more of your allowance per message**, and that allowance pools across browser, desktop, mobile and Claude Code on a rolling five-hour window with a weekly cap on top [24] [32].

Beginner: Projects, and keeping context under control

A Project is a folder of conversations sharing one context: instructions you write once, attached files, and a memory of its own. Creating one asks two questions — what you're working on and what you're trying to achieve — and the project page then gives you an **Instructions panel** and a **Context panel** for files. Every chat started inside it begins with both loaded [24]. Projects are not on the Free plan; they start at Pro [30].

The reason to push beginners into Projects early is mechanical. **Every conversation has a context window.** Everything you say, paste and attach goes into it, and when it fills, Claude does not stop — it quietly summarises the earlier part of the conversation to make room. The symptom is a chat that starts ignoring something you told it an hour ago. The remedy is habits, not settings [24]:

- **One task per chat, a new chat for a new task.** Let Projects and Memory carry what needs carrying between them.
- **Attach big documents to a Project, not a conversation.** A 200-page PDF dropped into a chat spends context on every subsequent turn; in a Project, Claude reads what it needs when it needs it.

- **To retrieve something from an old conversation, ask for it** ("find our chat about the supplier contract") rather than scrolling it back into the current thread.

The context window is **200k tokens on Free, Pro and Max**, and **500k on the default Enterprise model** [30]. Claude Code adds an explicit lever for the same problem: `/compact` triggers summarisation early to free up space [1].

Projects also scope search, so "what did we decide about pricing last week?" inside the relevant project finds the right conversation — the behaviour that stops working when everything lives in one undifferentiated chat list [24].

Chat Projects versus Cowork Projects

Both are called Projects and share a shape, but they are not the same object. Chat Projects organise conversations and documents. Cowork Projects are workspaces for tasks, carrying their own scheduled tasks and their own memory on top of instructions and context [22]. New projects created in Cowork save to your Claude account and follow you across devices; projects created from a folder on your computer stay on that computer [22]. Start with a Chat Project and don't worry about the distinction until you begin handing work off.

Beginner: connectors, extensions and the three meanings of "my files"

Anthropic uses three words for three different things, and most beginner confusion comes from guides using them interchangeably [31]:

Connectors are cloud services. They run on Anthropic's side and work everywhere — web, desktop, mobile, Cowork and the Code tab. If the thing you want to connect is a website you sign into, it's a connector.

Desktop extensions run on your computer. If the tool is a local file or folder, a program on your machine, or something needing your clipboard or filesystem, it's an extension. They work only in the desktop app and the Code tab, because they need to be where your files are.

The JSON file — `claude_desktop_config.json` — is the manual way to run a local MCP server that isn't packaged as an extension. It still works, and the app loads those servers into local Code sessions too, but it's no longer how most people add things [24] [31].

YOUR TOOL IS…	USE	AVAILABLE ON
A cloud/SaaS product	Remote connector	All surfaces
Running on your machine	Desktop extension	Desktop, Claude Code
A local file or folder	Desktop extension	Desktop, Claude Code
Something you built with a public URL	Remote connector (custom)	All surfaces

Installing an extension is a one-click `.mcpb` bundle from Settings → Extensions, or drag a file onto that panel. The tell that you're installing an extension rather than a connector is that **your operating system pops an install dialog** — it is going onto your computer with the same permissions as any other program. Any API key it needs goes into your OS keychain[24].

Per-connector permissions, which are the important part

Every connector has a Tool permissions page under Customize → Connectors, where each action is set to **Always allow**, **Needs approval** or **Blocked**, grouped into read-only and write-or-delete tools[24]. The advice that makes everything else safe: set anything that sends, pays or deletes to Needs approval and leave it there. Gmail is the worked example — it searches and reads mail, drafts replies, and can send, reply and forward, but sending asks for explicit approval every time by default. It sees attachment names and sizes, not contents, and nothing it reads is used for training[24].

Adopt one folder discipline from day one: make one folder for Claude to work in, put **copies** of anything you want it to work on inside, and point Claude at that folder and nothing else. On the first write you get a prompt naming the extension, the tool and the exact path and content — read it before you click, and answer `Allow once` for your first week[24].

> CAUTION
>
> A local extension runs on your computer with the same rights as any program you install. Stick to verified extensions and read what each one asks for. Plugins are worse, because one install can bundle skills, connectors and sub-agents together and significantly expand what Claude can reach[23].

Intermediate: Cowork and handing off whole tasks

Cowork is Claude Code's agentic architecture behind a graphical interface, with no terminal required. Simon Willison's assessment when it launched in January 2026 is still the clearest: it is "regular Claude Code wrapped in a less intimidating default interface and with a filesystem sandbox configured for you without you needing to know what a filesystem sandbox is"[5].

What you notice as a user is four differences from Chat[2][8]:

- **Folder access.** Point Claude at a folder and it reads what's there and saves finished work back to the same place. Chat hands finished files back as downloads instead.

- **Sub-agents.** For a big job, Claude splits the work across background workers running in parallel and hands you one finished deliverable.
- **Long-running tasks.** Work continues for extended periods without conversation timeouts interrupting progress.
- **Deliverables.** Excel spreadsheets with working formulas, PowerPoint presentations and formatted documents.

Cowork sessions run in the cloud by default: the work happens in an isolated, temporary environment on Anthropic's servers, created for that session and **removed when it ends**. When a task needs a local file or your browser, Claude reaches your computer through the desktop app, only for folders you've connected — if the app is offline, the session can't reach your computer. One caveat stated plainly: because sessions run on Anthropic's servers, work Claude does there, *including local files it opens through the desktop app*, is processed on Anthropic's servers rather than staying on your machine [7].

The three approval modes, and what they actually gate

Cowork has two independent controls: the **mode** (Manual, Auto, Skip) and each connector tool's own permission. The interaction is the part people get wrong.

	TOOL: ALWAYS ALLOW	TOOL: NEEDS APPROVAL	TOOL: BLOCKED
Manual	Approved	Asks for permission	Denied
Auto	Read-only approved; Claude decides for write/delete	Claude decides	Denied
Skip	Approved	Approved	Denied

Manual is the old "Ask before acting". Auto — formerly "Automatically approve" — works without stopping at every step but reviews each action for safety, checking for data exfiltration and prompt injection, and blocks anything it judges unsafe. Skip — formerly "Act without asking" — pauses for nothing and checks nothing. **Auto consumes more of your usage limit** than the other modes, because Claude does that checking on your behalf [8].

> KEY FINDING
>
> Permanent deletion is gated in every mode. Cowork requires explicit permission before permanently deleting any file, whatever mode you are in [7] [8].

Anthropic's published examples show the shape of the work: turning a folder of contracts into a renewals tracker with risks flagged; reconciling actuals from four regional exports and flagging variances over 5% or $50k; building a Monday marketing readout from Amplitude, Drive and a tracker. Zapier connected Cowork to its org database, Slack and Jira to surface engineering bottlenecks, and Jamf turned a seven-facet performance review into a 45-minute guided self-evaluation [29].

Intermediate: scheduled tasks, Cowork projects and plugins

Claude Code — and therefore Cowork and the Code tab — offers three ways to schedule recurring work, and the difference decides whether your task actually runs.

	CLOUD ROUTINES	DESKTOP LOCAL TASKS	`/LOOP`
Runs on	Anthropic-managed cloud	Your machine	Your machine
Machine must be on	No	Yes	Yes
Local file access	No (fresh clone)	Yes	Yes
Permission prompts	No — runs autonomously	Configurable per task	Inherits from session
Minimum interval	**1 hour**	**1 minute**	**1 minute**

The failure mode of choosing wrong is silent: a Desktop local task on a sleeping laptop is simply skipped [9].

Cowork scheduled tasks

“Scheduled” in the left sidebar is where these live. Two creation paths: **Create with Claude**, where Claude asks clarifying questions and then shows the task name, schedule and behaviour before you confirm; or **Set up manually**, where you supply the name, prompt, approval mode, frequency (hourly, daily, weekly, weekdays, or manual for on-demand), an optional model and an optional working folder [10]. Common uses are daily briefings from Slack, email or calendar; weekly reports from Drive; recurring competitor research; and status updates from project tools [10].

They run in the cloud, so they fire on schedule even when your computer is asleep or the app is closed. One limitation to plan around: **they cannot be tied to a folder on your computer** [10].

Catch-up behaviour belongs in your prompts. When the app starts or the machine wakes, Desktop checks for runs missed in the last seven days and starts **exactly one catch-up run for the most recently missed time**, discarding anything older — so a 9am task might fire at 23:00 if the computer slept all day. The documented fix is a guardrail in the prompt: “Only review today’s commits. If it’s after 5pm, skip the review and just post a summary of what was missed” [9] [10].

> CAUTION
>
> Because you cannot monitor scheduled tasks in real time, Anthropic’s guidance is to start with low-risk work like summaries and compilation; **do not schedule tasks that access sensitive files, send messages on your behalf, make purchases, or take other actions that are difficult to undo**; review outputs after each run; and pause anything you aren’t actively using [23].

Cowork Projects and plugins

Cowork Projects are workspaces rather than conversation folders. Each carries **instructions**, its own **scheduled tasks**, **context** from a local folder or linked chat project or URL, and **memory scoped to the project** [22]. Projects created from a folder on your computer stay there; ones created inside Cowork follow you across devices [22].

A plugin bundles skills, connectors, slash commands and sub-agents into one install. Anthropic open-sourced eleven in January 2026 — productivity, sales, customer support, product management, marketing, legal, finance, data, enterprise search, bio-research, and a plugin that builds plugins — and the repository has since added engineering, design, HR, operations and small-business plugins [11] [20]. Every component is file-based markdown and JSON with no build steps, so a team can fork a plugin and rewrite the skill files with its own terminology and processes [20].

Intermediate: two ways to give Claude a browser, and computer use

Cowork can use the web in two quite different ways, and the built-in browser — shipped **26 August 2026** — is the one that removes setup rather than adding capability [12] [13]. It opens in Cowork's side panel when a task involves a website: Claude navigates pages, reads them, clicks, types and fills forms while you watch, with nothing to install. It's Claude's browser, not yours — it does not see your tabs, bookmarks or passwords, and uses a clean profile separate from your personal browser [12]. To stay signed in you can import cookies **site by site**, from Chrome, Edge and Firefox on macOS and from Firefox on Windows and Linux. Banking, email and single sign-on sites stay unchecked by default, and **import is not available from Safari at all** [12] [14].

	Built-in browser	vs	Claude in Chrome
Shipped	26 Aug 2026, side panel	·	GA 26 Aug 2026, extension
Your logins	Only ones you import, site by site	·	Your existing Chrome sessions
Setup	None	·	Install extension and enable connector
Best for	Handing off a web task while you keep working	·	The page you already have open
Available on	macOS, Windows, Linux (beta)	·	Chrome only, not other Chromium browsers or mobile
Default when	You don't already use Claude in Chrome	·	You already have Claude in Chrome installed

Set your choice in Settings → Cowork → Preferred browser. Sources: Anthropic blog and Help Center, Aug–Sep 2026.

The built-in browser needs the desktop app open and online even though the session runs in the cloud [14]. Neither browser escapes the fundamental risk: any AI agent acting in a browser can be targeted by prompt injection, where instructions hidden in a page try to redirect Claude. Anthropic's safeguards — a permission prompt before acting on a site the first time, high-risk sites blocked, and per-action checks — reduce the risk but cannot remove it, and the company says so in those words [12] [14].

Computer use

Computer use is the broadest and slowest way Claude can act on your machine: it opens apps, clicks, types and navigates your screen the way you would. It exists for things nothing else can reach — native apps, hardware control panels, proprietary tools with no API [1] [6].

- **Beta on Pro and Max only.** Explicitly **not available on Team or Enterprise plans**, and not in the Linux beta [6] [15] [21].
- **Off by default.** Enable under Settings → General (Desktop app). On macOS you must also grant **Accessibility** (to click, type and scroll) and **Screen Recording** (to see your screen) [1] [6].
- **Your desktop must be awake and the app open**, and **Claude takes screenshots** — so it can see anything visible in an app you've approved [6].

Claude tries the most precise tool first — a connector if one exists, then Bash, then Claude in Chrome, then the iOS Simulator pane, and only then screen control. Per-app tiers reinforce that ordering: browsers and trading platforms are capped at **View only**, terminals and IDEs at **Click only**, everything else at **Full control**, fixed by app category. App approvals last the session, or **30 minutes in Dispatch-spawned sessions** [1].

> CAUTION
>
> Computer use has no sandbox between Claude and your applications. Anthropic's own comparison is blunt: file operations go through permission checks and code execution runs in an isolated environment, but computer use "interacts directly with your apps, browser, and desktop" [6]. One subtlety: actions in one app can affect another — clicking a link in an approved email app can open it in Chrome without Claude having permission to use Chrome [6].

Intermediate: Dispatch and moving between surfaces

Dispatch gives you one continuous conversation with Claude reachable from your phone or desktop. Instead of a session per task, there is a **single persistent thread that does not reset** [16].

When you assign a task, Claude works out what kind of work it is and spins up the right session. **Development tasks run in Claude Code; knowledge work runs in Cowork**, and Code sessions get a **Dispatch badge** in the sidebar. You get the outcome plus a push notification when a task finishes or

Claude needs a go-ahead. App approvals in Dispatch-spawned sessions **expire after 30 minutes and re-prompt**, rather than lasting the full session as regular Code sessions do [1] [16].

Availability is narrow: **limited beta for Pro and Max**, requiring both desktop and mobile apps, and **not available on Team or Enterprise plans** [1] [16]. Dispatch also needs the desktop awake and the app open, because the task runs on your machine — which is the distinction that matters most.

Where a session runs determines what it can reach

FEATURE	DESKTOP	WEB	MOBILE
Start, steer and review tasks	Yes	Yes	Yes
Resume a session started elsewhere	Yes	Yes	Yes
Connectors, skills, plugins, projects, scheduled tasks	Yes	Yes	Yes
Local file access	Yes	Yes*	Yes*
Browser use	Yes	Yes*	Yes*
Computer use	Yes	Yes*	Yes*

The asterisk does a lot of work: **local file access, local connectors, browser use and computer use from web or mobile all work through the Claude Desktop app.** A cloud session can read and write files in folders you've connected only while the desktop app is open on that computer and the session was started on desktop. If the app is closed, the session keeps running but cannot reach your local files [25].

Dispatch is the riskier of the two directions of travel, and Anthropic's framing is blunt:

> *Giving a mobile AI agent remote control of a desktop AI agent creates a chain where instructions from your phone can trigger real actions on your computer: reading, moving, or deleting local files, interacting with connected services, controlling your browser, and using your desktop apps… mistakes (or malicious content the model encounters along the way) can have real consequences.* [16]

The guidance before enabling it is to trust every app in the chain, understand what files and accounts are reachable, know how to revoke access quickly, and — the line that matters — **connect these agents only if you're comfortable with what they *could* do, not just what you intend them to do** [16]. Dispatch also has a structural limit: no way to start a second thread [16]. If you want work to continue while your laptop is shut, a **cloud session** is the safer default [25].

Advanced: the Code tab in depth

The Code tab runs the same engine as the terminal Claude Code CLI, with a graphical interface for running several sessions side by side. CLI and desktop share configuration, so `CLAUDE.md` files, MCP

servers in `~/.claude.json` or `.mcp.json`, hooks, skills and settings all carry over — and `/desktop` in the terminal hands a CLI session to the app[1].

Sessions and worktrees. Every conversation is a session with its own history, folder and changes. For Git repositories each session gets its own isolated copy using **Git worktrees**, by default in `<project-root>/.claude/worktrees/`, so changes in one session don't affect others until you commit. On Windows, Git for Windows is required for local sessions to start at all[1].

Permission modes.

MODE	SETTINGS KEY	BEHAVIOR
Manual	`default`	Asks before editing files or running commands; you see a diff and accept or reject each
Accept edits	`acceptEdits`	Auto-accepts file edits and filesystem commands; still asks before other terminal commands
Plan	`plan`	Reads and explores, proposes a plan without editing source
Auto	`auto`	Executes with background safety checks; reduces prompts while keeping oversight
Bypass permissions	`bypassPermissions`	Runs without prompts except actions no mode auto-approves

A mode you pick is remembered per folder and outranks `defaultMode` for that folder, except Plan, which applies to the current session only. Auto needs **Opus 4.6+, Sonnet 4.6+, or a Fable model**. Bypass permissions equals `--dangerously-skip-permissions` on the CLI, and the documentation says to use it **only in sandboxed containers or VMs**[1].

The review loop. A diff stats indicator shows `+12 -1`. Click it to open the diff viewer, comment on any line, and submit all comments at once with **Cmd+Enter** (macOS) or **Ctrl+Enter** (Windows); Claude reads them and produces a new diff. A separate **Review code** pass focuses deliberately on compile errors, logic errors, security vulnerabilities and obvious bugs — it does not flag style, formatting or anything a linter would catch. PR monitoring adds a CI status bar via the **GitHub CLI**, with **Auto-fix** and **Auto-merge** (squash, and requires auto-merge enabled in the repository settings first)[1].

Layout. Arrangeable panes cover chat, diff, browser, terminal, file, plan, tasks, subagent and — on macOS — the iOS Simulator. Press **Cmd+** / **Ctrl+** to close the focused pane. The integrated terminal shares Claude's environment, so `npm test` sees the files Claude is editing; local sessions only[1]. Transcript detail is controlled by **Normal**, **Verbose** and **Summary** view modes, though one discrepancy is worth flagging: the reference page documents all three, while the v1.46388.1 changelog records Summary being **removed** from the menu[1][27].

A **side chat** — **Cmd+;** or `/btw` — asks a question using the session's context without adding anything back to the main thread. **Cross-session messaging** lets Claude list your other sessions with `ListAgents` and message one with `SendMessage`. Through the desktop surface Claude can't send

from a session nobody is watching, and always asks before archiving — in every mode, including Auto and Bypass [1] [33].

Advanced: environments, automation and the customisation stack

Four environments decide where a Code session executes [1]:

- **Local** — on your machine, with direct access to your files.
- **Cloud** — on Anthropic-managed infrastructure by default; continues after you close the app, supports multiple repositories, and can be monitored from claude.ai/code or the mobile app. Usage counts toward subscription limits with no separate compute charges.
- **SSH** — on a remote machine you manage. The remote must run Linux or macOS; Desktop installs Claude Code there on first connect, and SSH sessions support permission modes, connectors, plugins and MCP servers.
- **WSL** — inside a WSL 2 distribution on Windows, using its Linux toolchain.

Administrators can pre-configure SSH connections and restrict reachable hosts:

```JSON
{
  "sshConfigs": [
    { "id": "shared-dev-vm", "name": "Shared Dev VM",
      "sshHost": "user@dev.example.com", "sshPort": 22,
      "sshIdentityFile": "~/.ssh/id_ed25519" }
  ],
  "sshHostAllowlist": ["*.devboxes.example.com", "bastion.example.com"]
}
```

`sshHostAllowlist` is read from managed settings only — values in user or project settings are ignored — and governs which hosts the Desktop app connects to, not network egress, so pair it with network controls if you need a hard boundary [1].

The seven ways to steer behaviour

Anthropic counts seven methods for instructing Claude Code, each trading context cost against authority [17]:

METHOD	WHEN IT LOADS	CONTEXT COST	USE IT FOR
CLAUDE.md (root)	Session start, stays for the session	High — every line costs tokens every session	Build commands, directory layout, coding conventions
CLAUDE.md (subdirectory)	On demand, when Claude reads a file there	Low	Conventions specific to a subdirectory
Rules (`.claude/rules/`)	Session start, or on matching files if path-scoped	Medium	Hard constraints, e.g. all API handlers validate input
Skills (`.claude/skills/`)	Name and description at start; body on invocation	Low	Procedural workflows — deploy or release checklists
Subagents (`.claude/agents/`)	Name, description, tools at start; body when called	Low in main context; own window	Side tasks that return only a summary
Hooks	Fire on lifecycle events	Low; bypasses compaction	Deterministic automation — run linters, block commands
Output styles / system-prompt appends	Session start, into the system prompt	High	Significant role changes; tone and formatting

The heuristics are worth taking literally. Keep root `CLAUDE.md` **under 200 lines**, give it an owner, and review changes like code — it grows the way any unowned config file does. Push team-specific conventions into path-scoped rules and procedures into skills, where they load only when relevant. Use a **subagent** when a side task would clutter your main conversation with results you won't reference again; use a **skill** when you want the procedure to play out in the main thread so you can watch each step. **Hooks are the only method with deterministic authority** — user-defined commands, HTTP endpoints, MCP tool calls or subagents that execute at specific points, bypassing compaction entirely[17].

▼ **Code tab keyboard shortcuts**

`Cmd`/`Ctrl` `/` — show shortcuts · `Cmd N` — new session · `Cmd W` — close session · `Ctrl Tab` / `Ctrl Shift Tab` — cycle sessions · `Cmd Shift ]` / `[` — cycle sessions · `Esc` — stop Claude's response · `Cmd Shift D` — toggle diff pane · `Cmd Shift B` — toggle Browser pane · `Cmd Shift S` — select element in Browser · `Ctrl` `` ` `` `— toggle terminal ·` Cmd \ `— close focused pane ·` Cmd ; `— open side chat ·` Ctrl O `— cycle view modes ·` Cmd Shift M `— permission mode menu ·` Cmd Shift I `— model menu ·` Cmd Shift E `— effort menu ·` 1`-`9 — select item in an open menu.

These apply only to the Code tab. Terminal shortcuts such as `Shift+Tab` do not apply in Desktop. Session cycling, the terminal toggle and the view-mode toggle use `Ctrl` on every platform.

Advanced: enterprise configuration and governance

Administering the desktop app across a fleet happens in three layers that override each other in a defined order: project settings, then user settings, then managed settings. The **admin console** covers whether users can access Claude Code in the desktop app, whether web sessions are enabled, whether Remote Control is enabled, and whether bypass-permissions mode is available at all[1].

Managed settings keys are the finer layer:

KEY	WHAT IT DOES
`permissions.disableBypassPermissionsMode`	Set to `"disable"` to block bypass-permissions mode
`disableAutoMode`	Removes Auto mode from the selector
`autoMode`	Customises what the auto-mode classifier trusts and blocks org-wide
`browserExternalPageTools`	Users can browse external sites but Claude's tools can't read or act on them
`disableBrowserExternalNavigation`	JSON boolean `true` blocks external browsing entirely; localhost previews still work
`disableMobileSimulatorTools`	Blocks Claude's control of the iOS Simulator pane
`sshConfigs` / `sshHostAllowlist`	Pre-configure SSH connections; restrict reachable hosts
`disableDesktopLocalSessions`	Turns off on-device Code sessions, leaving SSH and cloud
`managedMcpServers`	Push MCP server configurations, with an optional `toolPolicy` map

Where a managed setting reaches depends on where the session runs. Local sessions read a settings file deployed to disk; cloud sessions receive server-managed settings instead, because device files don't reach Anthropic-managed VMs; SSH sessions read the managed settings file from the remote host. One notable exception: in a Cowork session on the machine, Claude Code **never fetches admin-console settings even when the user signs in with a Team or Enterprise account**, and reads policy deployed to the machine instead[1].

Delivery. On macOS, settings deploy through MDM as the `com.anthropic.claudefordesktop` preference domain (Jamf, Kandji); on Windows, through the registry at `SOFTWARE\Policies\Claude` (Group Policy, Intune). The app distributes as a `.dmg` on macOS and an **MSIX package on Windows** — which some organisations block by policy, causing the installer to refuse to run with no useful error, so the Anthropic publisher certificate may need whitelisting[1][32]. A locked-down network needs the Anthropic CDN hosts allowlisted — `anthropic.com`, `claude.ai`, `claude.com`, `claude.app`, `claudeusercontent.com` and `claudemcpcontent.com` plus subdomains, HTTPS on port 443[1].

Governance. Role-based access controls let Enterprise admins organise users into groups, manually or via SCIM, and assign custom roles defining which capabilities members can use — so Cowork can be turned on for specific teams. Group spend limits set per-team budgets. Cowork activity appears in

the admin dashboard and the Analytics API, including per-user activity, skill and connector invocations, and DAU/WAU/MAU alongside Chat and Code figures[29].

OpenTelemetry emits events for tool and connector calls, files read or modified, skills used, and **whether each action was approved manually or automatically** — compatible with standard SIEM pipelines, on Team and Enterprise plans[29]. Compliance API coverage extended to Cowork and Code on 11 August 2026[18]. Two recent additions: skill and plugin security scanning in beta since 6 August 2026, and smart reports in beta on Enterprise since 10 September, which flag repeated patterns worth packaging as shared skills[26].

Where the edges are: gaps, known issues and gating

> CAUTION
>
> **Open issue, 10 September 2026.** A Windows update released 8 September 2026, including KB5124008, stops Cowork from reaching your files when it runs on your Windows PC — tasks fail or the workspace does not start. This affects Cowork on third-party inference deployments and sessions that run on your computer rather than in the cloud. Cloud sessions and Claude Code, including the Code tab, are **not** affected. The cause is a change in Windows, so restarting or reinstalling doesn't help, and the underlying problem is not fixed[27].

Missing from specific surfaces

The Linux beta is thinner than the marketing. **Computer use, dictation and the Quick Entry hotkey are all absent** — quick entry works on X11, and on native Wayland needs the desktop environment's GlobalShortcuts portal. Only Debian-based distributions are supported; Fedora and RHEL users are pointed at the CLI[21]. On Mac, macOS 13 Ventura became the floor on 8 September with the move to Electron 44[27].

- **Computer use and Dispatch:** Pro and Max only, explicitly absent on Team and Enterprise[6][15][16].
- **Projects:** not on Free[30], and Cowork Projects are not available in Claude Code, with support "planned for a future update"[22].
- **iOS Simulator pane:** Pro, Max, Team and Enterprise, except Enterprise organisations with a HIPAA configuration[19].

Gaps in the Code tab relative to the CLI: **agent teams** are CLI-only, with dynamic workflows inside one session as the substitute. There is no `--print` or `--output-format` equivalent, so Desktop is interactive only with no scripting path, and no inline autocomplete suggestions. Terminal-dialog commands differ: `/permissions` replies that it isn't available in this environment, and `/config` opens the settings pane rather than accepting arguments[1].

Still labelled beta or research preview

Anthropic distinguishes the two labels — both mean the feature is evolving, at different levels of maturity. Still labelled: **computer use**, the **Linux desktop app**, the **iOS Simulator pane**, **Cowork on web and mobile**, **routines** (with the caveat that behaviour, limits and the API surface may change), **Dispatch**, **skill and plugin scanning**, and **smart reports** [9] [12] [19] [21] [26].

Common failure modes

From the official troubleshooting section [1]:

- **403 or authentication errors in the Code tab.** Sign out and back in from the app menu — the documented first fix, resolving most cases. Then verify a paid subscription; if the CLI works but Desktop doesn't, quit the app completely rather than closing the window.
- **"Failed to load session."** Usually the folder no longer exists, a repository needs Git LFS, or permissions block access.
- **Claude can't find installed tools** like `npm` or `node`. The app doesn't always inherit your full shell environment — on macOS it reads `PATH` and a fixed set of variables from your shell profile; on Windows it ignores PowerShell profiles. The fix is the local environment editor, by hovering over **Local** in the environment dropdown and clicking the gear icon [1].

One cross-cutting caveat that is a limitation rather than a risk: **network egress permissions do not apply to the web fetch or web search tools, or to MCPs including Claude in Chrome** [23] [25].

Bottom line

Install the app and stay in **Chat** for a week. Decide the Memory and training switches deliberately rather than inheriting their defaults, make one Project for work you'll return to, add a single connector with everything that sends, pays or deletes set to Needs approval, and leave every permission prompt in place until you've watched Claude work [24].

Then move to **Cowork**, which is the real step change. The safe way in is one dedicated working folder containing copies rather than originals, Manual mode, and a two-step task before anything scheduled. Once that feels predictable, scheduled tasks and plugins are the natural next moves [23] [24].

If you write code, the **Code tab** is the same engine as the CLI with a review loop bolted on, and the settings files are shared, so any configuration you already have carries over untouched. Start complex work in Plan mode, review the diff, then let it execute [1].

Two things to expect. First, the surface keeps moving — releases landed repeatedly through August and September alone, and the app's own changelog is frequently newer than the documentation describing the same feature [27]. Second, the capability ladder is not a difficulty ladder. A beginner can use Cowork safely on day one with the right folder and the right approval mode; an expert can spend

a career in Chat. The right question is never “am I advanced enough for this” but **what can Claude read here, and what is it allowed to do** [23].

SOURCES

1 Desktop application — Claude Code Docs (reference for the Code tab)
https://code.claude.com/docs/en/desktop

2 Download Claude — Claude by Anthropic (platforms, install, feature overview)
https://claude.com/download

3 Claude Cowork product page — core capabilities
https://claude.com/blog/cowork-research-preview

4 Claude Cowork on web and mobile: hand off work anywhere (7 July 2026)
https://claude.com/blog/cowork-web-mobile

5 Simon Willison — Claude Cowork, weeknotes on the January 2026 research preview
https://simonwillison.net/2026/Jan/12/claude-cowork/

6 Let Claude use your computer in Cowork — Claude Help Center
https://support.claude.com/en/articles/14128542-let-claude-use-your-computer-in-cowork

7 Claude Cowork architecture overview — Claude Help Center
https://support.claude.com/en/articles/14479288-claude-cowork-architecture-overview

8 Get started with Claude Cowork — Claude Help Center
https://support.claude.com/en/articles/13345190-get-started-with-claude-cowork

9 Schedule recurring tasks in Claude Code Desktop — Claude Code Docs
https://code.claude.com/docs/en/desktop-scheduled-tasks

10 Schedule recurring tasks in Claude Cowork — Claude Help Center
https://support.claude.com/en/articles/13854387-schedule-recurring-tasks-in-cowork

11 Use plugins in Cowork — Claude Help Center
https://support.claude.com/en/articles/13837440

12 Claude gets its own browser in Cowork (26 August 2026)
https://claude.com/blog/cowork-built-in-browser

13 Claude Cowork gets a built-in browser — Times of AI coverage
https://www.timesofai.com/news/claude-cowork-browser-adds-web-tasks-without-chrome/

14 Use the built-in browser in Claude Cowork — Claude Help Center
https://support.claude.com/en/articles/16607400

15 Available beta and research preview features — Claude Help Center
https://support.claude.com/en/articles/14503520-available-beta-and-research-preview-features

16 Assign tasks from anywhere in Claude Cowork (Dispatch) — Claude Help Center
https://support.claude.com/en/articles/13947068-assign-tasks-from-anywhere-in-claude-cowork

17 Steering Claude Code: when to use CLAUDE.md, skills, hooks, and subagents (18 June 2026)
https://claude.com/blog/steering-claude-code-skills-hooks-rules-subagents-and-more

18 Compliance API coverage extends to Claude Cowork and Claude Code (11 August 2026)
https://claude.com/blog/compliance-api-coverage-extends-to-claude-cowork-and-claude-code

19 Test iOS apps in the simulator — Claude Code Docs
https://code.claude.com/docs/en/desktop-ios-simulator

20 anthropics/knowledge-work-plugins — open-source plugin repository for Cowork
https://github.com/anthropics/knowledge-work-plugins

21 Claude Desktop on Linux (beta) — Claude Code Docs
https://code.claude.com/docs/en/desktop-linux

22 Organize your tasks with projects in Claude Cowork — Claude Help Center
https://support.claude.com/en/articles/14116274-organize-your-tasks-with-projects-in-claude-cowork

23 Use Claude Cowork safely — Claude Help Center
https://support.claude.com/en/articles/13364135-use-claude-cowork-safely

24 Getting Started with Claude Desktop: a Beginner's Guide (2026) — Houtini
https://houtini.com/articles/claude-desktop-beginners-guide/

25 Use Claude Cowork on web, desktop and mobile — Claude Help Center
https://support.claude.com/en/articles/15520349-use-claude-cowork-on-web-desktop-and-mobile

26 Claude release notes — Claude Help Center (September 2026)
https://support.claude.com/en/articles/12138966-release-notes

27 Changelog — release notes for Claude Desktop (v1.52386.0, 10 September 2026)
https://claude.com/docs/cowork/changelog

28 Making Claude Cowork ready for enterprise (9 April 2026)
https://claude.com/blog/cowork-for-enterprise

29 Claude Cowork usage breakdown: >90% not software development
https://claude.com/blog/cowork-one-year-in

30 Claude pricing — Free, Pro, Max, Team and Enterprise
https://claude.com/pricing

31 When to use desktop and web connectors — Claude Help Center
https://support.claude.com/en/articles/11725091-when-to-use-desktop-and-web-connectors

32 Claude Desktop & Cowork System Requirements (2026) — Houtini
https://houtini.com/articles/claude-desktop-system-requirements/

33 Message your other Claude Code sessions — Claude Code Docs
https://code.claude.com/docs/en/cross-session-messaging

34 Automate work with routines — Claude Code Docs
https://code.claude.com/docs/en/routines

35 Use research on Claude — Claude Help Center
https://support.claude.com/en/articles/11088861-use-research-on-claude

Researched and written with deepseek/deepseek-v4-flash-0731, from the sources listed above. Figures are as reported by those sources. Exported November 29, 2026 from https://intel.innerpulse.net/doc/the-claude-gui-app-beginner-intermediate-and-advanced-capabilities-september, where the current version lives.