Claude Code Web

Complete Workflow Guide

Deep Dive: Session Management, File Systems,
Tools, Custom Components, and Best Practices

Prepared for:	Your Reference
Date:	November 08, 2025
Document Rev:	1.0
Chat Session:	Current conversation

Session Management & Persistence

Understanding how long your work persists and when sessions expire:

Aspect	Claude Code Web	Claude Code Local
Session Timeout	24 hours continuous OR 8 hours inactive	Infinite (your machine)
Reconnection	■ Can reconnect within timeout period	■ Always available (session files persist)
Session State	Task history persists Active execution terminates	Full state saved in ~/.claude/
Work Persistence	Via GitHub commits/PRs Container is ephemeral	Files on your disk Permanent
Lunch Break	■ Safe (<8hrs inactive)	■ Always safe
Multi-Day Work	■ Session dies after 24hrs PR remains on GitHub	■ Resume anytime All files remain
Crash Recovery	■■ Active execution lost Committed work safe	■ Resume from last state

Key Takeaways:

• Claude Code Web sessions expire after 24 hours OR 8 hours of inactivity

• Your WORK persists via GitHub (commits/PRs) even after session expires

• Lunch breaks are safe; overnight work requires committing to GitHub

• Use 'Teleport to CLI' feature to copy session state to local Claude Code

File System & Storage Architecture

Location	Purpose	Persistence	Capacity	Access
/workspace	Working directory for task execution	Per-session only Deleted on completion	~5GB disk 9GB RAM	Full read/write Claude operates here
GitHub Repo (remote)	Source of truth Long-term storage	■ Permanent Version controlled	GitHub limits (100GB soft)	Cloned to /workspace at session start
/tmp	Temporary files Build artifacts	Per-session only Cleared on exit	Part of 5GB disk allocation	Standard Linux temp directory
Container Overlay FS	System files Base image	Read-only base Writable overlay	Pre-allocated (not expandable)	System-managed Claude cannot modify

Critical Understanding:

• /workspace is EPHEMERAL - commit to GitHub or lose your work!

• GitHub is your ONLY persistent storage mechanism

• ~5GB total capacity (not expandable)

• File system resets completely after each task completion

Pre-installed Tools & Language Support

Category	What's Included	Versions	Installation Options
Core Languages	Python Node.js/JavaScript Bash/Shell	Python 3.12.3 Node.js 18.19.1 Bash 5.x	Pre-installed No action needed
Package Managers	pip (Python) npm/yarn (Node) apt (Ubuntu)	Latest stable versions	Pre-installed Use SessionStart hooks for project deps
Version Control	git gh (GitHub CLI)	git 2.40+ gh latest	Pre-installed Git via secure proxy
Common Build Tools	gcc, g++ make, cmake Docker client	System default versions	■■ Docker incompatible with sandbox
Other Languages	Go, Rust, Ruby, Java, C/C++	Via package managers	■ Install via hooks or during session
Web Tools	curl, wget Netcat	Standard versions	Pre-installed Network restricted
Data Tools	jq, yq PostgreSQL client	Latest stable	Pre-installed DB access via network

Installing Additional Tools:

• Use SessionStart hooks in CLAUDE.md for automatic installation

• Example: Install Rust via 'curl --proto =https --tlsv1.2 -sSf https://sh.rustup.rs | sh'

• Mobile development: Limited (Android SDK possible, iOS impossible)

• Best supported: Python, JavaScript, Go, Rust, Java

Network Configuration & Domain Access

Environment Type	Network Access	Default Allowed Domains	Use Cases
No Network	Completely isolated No internet	None (Fully offline)	Sensitive repos No external dependencies Air-gapped work
Limited (Default)	Package repos only Curated allowlist	*.npmjs.org *.pypi.org *.github.com *.gcr.io *.docker.io ~40 more	Standard development Dependency installation Most projects
Custom Allowlist	User-defined Specific domains	Your specified domains only	Corporate APIs Specific databases Controlled access
Trusted (Full)	Unrestricted internet All domains	All domains (*)	■■ Use with caution Public data projects No secrets

Security Best Practices:

• Start with 'Limited' and add domains as needed

• Never use 'Trusted (Full)' with repos containing secrets/credentials

• Custom allowlists: Principle of least privilege

File Transfer & Movement Mechanisms

Transfer Type	Method	When to Use	Limitations
GitHub → /workspace	Automatic clone at session start	Every session (primary method)	Clones default branch Requires GitHub hosting
Local → GitHub → /workspace	git push locally then clone in web	Sharing local work with web sessions	Two-step process Must commit first
/workspace → GitHub	Claude creates PR automatically	End of every task (default behavior)	Creates new branch PR needs approval
Web → Local (Teleport)	"Open in CLI" button Copies session state	Continue work locally after web session	Only active sessions Requires local CLI
Multiple Repos → /workspace	Not supported natively	■ Not available Web	One repo per session Use submodules?
Download Docs from Web	curl/wget in Claude (if domain allowed)	Fetching external references	Requires network access Domain must be allowed
User Upload → /workspace	■ Not supported	Not available in web version	Use GitHub instead Commit files first

Workarounds for Common Scenarios:

• Reusing files across repos: Commit common files to separate 'shared' repo, use git submodules

• Pulling from multiple repos: Not directly supported; consider monorepo approach

• Adding prompts/commands: Store in .claude/ directory in your repo

Custom Components: Commands, Agents, Skills

Component	Web Support	Storage Location	How to Use
/commands (Slash cmds)	■■ Via CLAUDE.md only	Stored in repo: .claude/commands/	Define in CLAUDE.md Claude reads & executes Not native /command
Custom Agents	■ Full support (Subagents)	Stored in repo: .claude/agents/ ~/.claude/agents/	Create .md files with YAML frontmatter Auto-invoked
Skills	■ Via CLAUDE.md and repo files	Stored in repo: .claude/skills/ Anthropic managed	Reference in CLAUDE.md Claude loads when relevant to task
Hooks	■ Full support (SessionStart)	Stored in repo: .claude/hooks/	sessionStart hook runs at session init Automate setup
CLAUDE.md	■ Full support Respected	Root of your GitHub repo	Context for every session Preferences, workflows

Key Differences from Local CLI:

• /commands: Native slash commands work in CLI, but Web reads them from CLAUDE.md as instructions

• Agents: Fully supported! Create subagents in .claude/agents/ for specialized tasks

• Skills: Store in .claude/skills/ in your repo; Claude loads them automatically

• All custom components MUST be in your GitHub repo (no user-level ~/.claude/ access)

Context Window Management & Prompting

Feature	Claude Code Web	Claude Code Local	Notes
Context Window Size	200K tokens (~600 pages)	200K tokens (~600 pages)	Same model capacity both environments
/compact command	■ Not available (no slash cmds)	■ Available Reduces context	Web: Context managed automatically
/clear command	■ Not available	■ Available Fresh start	Web: Start new session for fresh context
Extended Thinking	■ Available Automatic	■ Available Automatic	Part of Sonnet 4.5 model capabilities
"ultrathink" commands	■■ Try in prompts No guarantee	■■ Try in prompts No guarantee	Not official commands Community experiments
Thinking Visibility	■ Visible in UI Chain of thought	■ Visible in terminal output	See Claude's reasoning in both environments
Session Summaries	■ Can request manually	■ Can request manually	Ask Claude to summarize progress
Planning Mode	■ Ask for plans first	■ /plan mode available	Web: Request in prompt Local: Native support

Prompting Best Practices for Web:

- Start with 'Plan first, then implement' to use context efficiently
- Request summaries: 'Summarize what you've done so far'
- Break large tasks into smaller sessions to avoid context bloat
- Use CLAUDE.md to provide persistent context across sessions

Editing Environment & IDE Integration

Aspect	Claude Code Web	Workarounds/Alternatives
IDE Access to /workspace	■ No direct IDE access to cloud /workspace	Use Teleport → edit locally OR use web-based IDEs
Primary Editing	Claude edits files You steer via chat	All editing through conversation with Claude
File Visualization	View in web UI See diffs in PR	GitHub PR for review Teleport for local view
Real-time Editing	■ Cannot edit during Claude session	Must wait for task completion Then edit PR locally
Web-based IDE Options	GitHub Codespaces (separate tool)	Use Codespaces + Claude Code Web for cloud workflow
Best Workflow	Claude → PR → Local review → merge	Hybrid: Web for heavy lifting, local for final touches

The Reality:

• Claude Code Web is NOT a cloud IDE - it's a task delegation system

• You don't edit files directly; Claude edits them based on your instructions

• For interactive editing, use Teleport to continue work in local Claude Code

Environment Configurations (Loadouts)

Loadout Type	Description	Network Access	Best For
Default Development	Standard tools Package managers	Limited (package repos)	Most projects General development
Secure/ Air-gapped	No network No external deps	None (Fully isolated)	Sensitive codebases Government/Finance
Full Stack Web	Node, Python DB clients	Custom allowlist + DB domains	Web applications API development
Data Science	Python + NumPy Pandas, Jupyter	Limited + data sources	ML/AI projects Data analysis
Custom Corporate	Your specified tools via SessionStart	Custom domains for your APIs	Enterprise projects Specific tech stack

Creating Custom Loadouts:

1. Create custom environment in Web UI (Environment selector)
2. Specify network access level and environment variables
3. Use SessionStart hooks in CLAUDE.md to automate tool installation
4. Save as reusable environment for future sessions

Workflow Recommendations: When to Use What

Scenario	Best Tool	Why	Alternative
Deep codebase refactoring	Local CLI	Full control, persistent state, your tools	Web for initial draft → Teleport to local
Multiple parallel bugs	Web	Spin up 5+ sessions simultaneously	Local with git worktrees (complex)
Quick PR fix while mobile	Web (iOS)	Access from phone Auto-create PR	Wait until at computer
Exploring new codebase	Web	No local setup Disposable session	Local for deeper exploration
Complex feature with planning	Web OR Local	Both support planning Choose by preference	Web: parallel work Local: deep dives
Sensitive code with secrets	Local CLI	Your machine Your security rules	Web with "No Network" Remove secrets first
Team collaboration	Web	Shared GitHub context PR-based workflow	Local with careful git coordination
Learning/ experimenting	Web	Disposable environment No local pollution	Local if want to keep experiments

Detailed Answers to Your Questions

Q: What happens if you break code mid-task?

Claude creates a new branch for each task. If code breaks, you can: (1) Guide Claude to fix it in the same session, (2) Let the session complete and fix the PR yourself, or (3) Abandon the session (close browser) - the broken code stays in the branch but isn't merged. GitHub is your safety net - main branch is never touched without PR approval.

Q: How much drive space in /workspace?

~5GB disk space total + 9GB RAM. Not expandable. If you exceed this, the session will fail. For large projects (>4GB), consider using sparse checkouts or working on specific subdirectories only.

Q: What are persistence rules for /workspace?

EPHEMERAL. Files exist only during the session. When Claude completes the task (or after 24 hours), the entire container and /workspace are destroyed. ONLY persistence is via git commits pushed to GitHub.

Q: If I walk away for lunch, will my session die?

No! Sessions survive up to 8 hours of inactivity. Your lunch break is safe. Overnight work requires committing to GitHub first. Active sessions can run up to 24 hours continuously.

Q: Can I reconnect to a session?

Yes, within the timeout period. If you close your browser, you can reopen Claude Code Web and see your recent sessions. Active sessions remain running; completed sessions show their history and PR links.

Q: What Ubuntu tools are pre-installed?

Python 3.12.3, Node.js 18.19.1, git, gh, gcc/g++, make, curl, wget, jq, PostgreSQL client, and standard GNU tools. See Table 3 for complete list. Install additional tools via SessionStart hooks.

Q: What's at my discretion vs Anthropic's control?

YOUR control: (1) Network environment selection, (2) SessionStart hooks for tool installation, (3) Git branches and PRs, (4) CLAUDE.md contents, (5) Custom agents/skills in your repo. ANTHROPIC control: (1) Base image tools, (2) Container resources (5GB/9GB), (3) Session timeout rules, (4) Security sandbox boundaries.

Q: Are there different 'loadouts' for different platforms?

Not pre-defined, but you can create custom environments. Use the Environment selector in Web UI to create configurations for: (1) Mobile dev (limited - Android possible, iOS impossible), (2) Web dev (full support), (3) Data science (Python-heavy), (4) Secure environments (no network). See Table 9.

Q: How do I move files between GitHub ↔ /workspace?

GitHub → /workspace: Automatic at session start (git clone). /workspace → GitHub: Claude creates PR automatically at task end. Manual control: Ask Claude to 'git commit' or 'git push' during session. See Table 5 for all mechanisms.

Q: Can I pull files from multiple GitHub repos?

NO - one repo per session. Workarounds: (1) Use git submodules in main repo, (2) Ask Claude to clone additional repos during session (requires network access), (3) Store common files in a shared repo and submodule it.

Q: Can I download docs from the web?

YES - if the domain is allowed in your network configuration. Claude can run 'curl' or 'wget' to fetch documentation. Requires: (1) Limited or Trusted network mode, (2) Domain must be in allowlist. Fetched files go to /workspace (ephemeral).

Q: Can I tell Claude to read @file.md?

YES - but syntax differs. In Web, just say 'Read the instructions in docs/SETUP.md'. The @ syntax is for Local CLI's file references. Claude can read any file in the /workspace (which includes your entire cloned repo).

Advanced Topics

Q: Are /commands fully supported?

NO - not as native slash commands like Local CLI. However, you can DEFINE commands in your CLAUDE.md file and Claude will understand them as workflow instructions. Example: Create a 'Commands' section in CLAUDE.md listing your custom commands, and Claude will follow them. It's not quite as convenient as '/mycommand', but it works.

Q: Are custom agents fully supported?

YES! Fully supported. Create .md files in .claude/agents/ directory of your repo with YAML frontmatter. Claude automatically discovers and delegates tasks to them. Same functionality as Local CLI. Agents are auto-invoked based on task description.

Q: Are custom skills easily incorporated?

YES! Create skills in .claude/skills/ directory. Claude discovers them automatically. Skills are folders with SKILL.md file + optional scripts. Same as Local CLI. Store them in your repo for persistence across sessions.

Q: What tools are available for managing context window?

Unlike Local CLI (which has /compact), Web version manages context automatically. Strategies: (1) Ask Claude to summarize progress, (2) Start new sessions for unrelated tasks, (3) Use CLAUDE.md to provide persistent context without consuming conversation tokens, (4) Break large tasks into multiple sessions.

Q: Is prompting fully interactive?

YES! Real-time steering. You can: (1) Send messages during execution, (2) Interrupt/redirect Claude mid-task, (3) Ask questions about current state, (4) Request plan changes. The Web UI shows Claude's progress in real-time with thinking visible.

Q: Can you peek at detailed thinking?

YES! Claude Sonnet 4.5 includes 'thinking' as part of its output. You'll see Claude's chain-of-thought reasoning in the Web UI as it works. This is the same extended thinking available in Local CLI. No special commands needed - it's automatic.

Q: Can you control thinking depth?

Partially. You can request deeper reasoning by asking Claude to 'think carefully' or 'analyze thoroughly'. Commands like 'ultrathink' are community experiments, not official features. The model's thinking depth is primarily controlled by task complexity, not explicit commands.

Q: What is the maximum context window?

200,000 tokens (~600 pages of text) - same as Claude Sonnet 4.5 in all contexts. This is shared between: conversation history, file contents, skill instructions, and thinking. Large codebases may fill context quickly; use strategic file selection.

Q: Can you install custom tools/scripts like a package?

YES via SessionStart hooks! Example in CLAUDE.md: hooks: sessionStart: | curl -sSf https://your-tools.com/install.sh | sh. This runs at session initialization. You can install: language runtimes, custom CLIs, monitoring tools, anything via curl/wget/package managers. Installs persist only for that session.

Q: Where do I normally edit files & setup prompts?

You DON'T edit files directly in Web - Claude edits them based on your instructions. Setup workflow: (1) Add prompts/context to CLAUDE.md in your repo, (2) Create custom agents in .claude/agents/, (3) Define skills in .claude/skills/, (4) Use SessionStart hooks for environment setup. All editing is conversational - you instruct, Claude executes.

Q: Can Claude help me plan?

ABSOLUTELY! Best practice: Start every session with 'First, create a detailed plan for implementing X. List all files to change and steps to take.' Claude will produce a plan as markdown. You review, approve, then say 'Now implement the plan.' This saves tokens and prevents mid-course corrections.

Q: Can I save session files?

Session files auto-save to GitHub via PR. For manual saves: Ask Claude to 'commit these changes to git'. For local copies: Use 'Teleport to CLI' to copy session state to your local machine. Sessions themselves (conversation history) are preserved in Web UI until they expire.

Key Takeaways & Mental Model

Claude Code Web Mental Model:

Think of Claude Code Web as 'hiring a remote developer who works in a disposable office.' You assign a task, they work autonomously in a temporary workspace (/workspace), and when done, they submit a PR with their work. The 'office' (container) is then demolished. The ONLY thing that persists is what goes into GitHub.

Not Like Google Colab:

Colab is for interactive notebook sessions you manage manually. Claude Code Web is for autonomous task delegation. You don't manage state - GitHub does. You don't execute cells - Claude does everything. It's closer to 'serverless functions for development tasks' than a cloud workspace.

The GitHub Centrality:

GitHub is not just a tool - it's THE persistent storage layer. Everything else is ephemeral. Your workflow MUST be GitHub-centric: (1) Store configs in .claude/ directory in repo, (2) Commit work frequently, (3) Review PRs as your 'output interface', (4) Use branches liberally for experimentation.

Hybrid Workflow (Best of Both Worlds):

Many developers use a hybrid approach: (1) Claude Code Web for: parallel bug fixes, quick tasks, exploring unfamiliar codebases, mobile-initiated work. (2) Claude Code Local for: deep feature development, complex refactoring, working with proprietary tools, offline work. (3) Teleport feature bridges the gap: start on Web, move to Local when you need hands-on control.

Common Pitfalls to Avoid:

■ Assuming /workspace files persist - they DON'T. Commit to GitHub!

■ Trying to access multiple repos in one session - NOT supported natively

■ Expecting interactive file editing - Claude edits, not you

■ Forgetting to configure network access - downloads will fail

■ Not using CLAUDE.md - you're missing free persistent context

■ Ignoring session timeout - commit before walking away overnight

■ Using 'Full Network' with sensitive repos - security risk

■ Not leveraging custom agents - they're powerful and easy to create