EMAZING GROUP · AI INFRASTRUCTURE

EG Skills: Ship Once, Update Everywhere

How we get every company skill onto every employee's Claude with a one-time install, then push every future update from a single edit. No more reinstall campaigns.

For Brian · Prepared by Anthony · June 12, 2026 · Follow-up to the June 2 power user call (Anthony / Brian / Jose)

TL;DR

- **Today, every skill update is a manual campaign:** 40+ reinstalls and 2 to 3 weeks of chasing, repeated on every change.
- **The MCP gateway cannot fix this:** it auto-updates tools live, but it cannot deliver skill files to machines or run them on a schedule.
- **The fix is a new private eg-skills repo:** employees install once, their machines auto-pull daily, and one edit reaches everyone within 24 hours.
- **It is safer than zip-and-Slack:** read-only access, no secrets ever in the repo, full audit trail, one-click rollback.
- **Brian has 3 sign-offs on this page:** the two-repo split, how employees get read access, and the pilot group plus first skills.

The problem we are solving

Skill distribution today is a manual campaign, and it repeats on every update.

Today, sharing a skill means zipping it, posting install instructions in Slack, and waiting for 30 to 40 people to each run the install. History shows that takes weeks. Then the first time the skill needs a fix or a new feature, the entire campaign repeats from zero.

40+
manual reinstalls needed per update, per skill, today

2 to 3 wks
typical time to get one rollout fully adopted

1 edit
what an update costs in the new model. Everyone is current within 24 hours.

The fix is the same move we already made with data tools on the Emazing Group MCP: **centralize the source, automate the delivery.** Skills just need a different delivery channel than tools, and that distinction is the whole design.

The call

One decision: how do company skills reach, and stay current on, every employee's machine? Three ways to do it.

REJECTED

Option 1: Keep zip-and-Slack (status quo)

No new infrastructure, but every update restarts a 40-person manual campaign. No audit trail, no way to revoke a copy, no rollback. This is the problem, not an option.

REJECTED

Option 2: Push skills through the MCP gateway

Tempting because tools already auto-update there, and it is what we planned to test on the June 2 call. But the MCP cannot write files to anyone's machine or run anything on a schedule, so skills would never land. The test would fail. Detail in Two Channels below.

RECOMMENDED

Option 3: A dedicated eg-skills GitHub repo with a daily pull

Employees install once from a single Slack paste. Their machines auto-pull the latest skills daily and before every scheduled run. One edit updates the whole company within 24 hours, with a full change log and one-click rollback.

RECOMMENDATION

Option 3. It is the only channel that physically delivers files to machines, supports scheduled runs, and removes the reinstall campaign permanently. Tools keep riding the MCP gateway exactly as today; skills get their own purpose-built lane. The rest of this page is the detail behind this call.

The key insight: tools and skills travel on two different channels

This is the question we left open on the June 2 call ("is it the MCP, or the MCP plus a GitHub for skills?"). The answer is both, kept separate.

Channel 1 · eg-mcp-tools (already live)

LIVE CONNECTION

Carries: tools. Meta, GA4, Shopify, Klaviyo, Google Ads, Sheets.

Everyone connects to one hosted gateway. When we update the server, every connected person has the new version **instantly**, because nothing lives on their computer.

What it cannot do: it never writes files to anyone's machine, and it cannot run anything on a schedule. So it physically cannot deliver or update a skill.

Channel 2 · eg-skills (proposed, new repo)

DAILY PULL

Carries: skills. Skill optimizer, Claude usage reporter, every future company skill.

Skills are **files that live on each person's computer**. To update files on 40 machines, each machine has to fetch the new version. A tiny scheduled "pull" does that automatically every day, plus right before any scheduled run.

One edit on GitHub, and every machine self-updates. Nobody reinstalls anything, ever.

WHY WE SHOULD NOT TEST "SKILLS THROUGH THE MCP"

> On the call we planned to attach a skill to the Amazing Group MCP and test whether it auto-updates like the tools do. It will not. The MCP protocol only hands clients tools to call. It has no mechanism to install or update skill files on a machine, and no way to schedule a Friday run. The test would burn a weekend proving a no. This page is the alternative that delivers exactly the outcome we wanted from that test.

The architecture

One repo, one installer, one daily pull. Everything else is automatic.

1. eg-skills repo on GitHub (private). One folder per company skill. The single source of truth. Anthony edits and commits; Brian reviews and co-owns.

↓

2. One-time install per employee. A single copy-paste block in Slack. Their Claude sets up the skills folder, the daily auto-pull, and any scheduled jobs. About 3 minutes, once, ever.

↓

3. Daily auto-pull on every machine. Each computer quietly grabs the latest version of every skill once a day. Scheduled skills also pull right before they run, so a scheduled run is always on the newest version.

↓

4. Updates are free. Add two data points to the Wednesday report? Edit the skill, commit. Every machine is current by tomorrow, and the next Wednesday run obeys. Adding a brand-new 4th or 5th skill works the same way: it just appears for everyone. No new install.

Worked example: the Wednesday 12pm report

We ship a skill that runs every Wednesday at noon on each machine and writes a few data points to a company spreadsheet. A month later we want two more data points and a second sheet. Old world: re-zip, re-Slack, 40 reinstalls, weeks of chasing. New world: one edit to the skill on GitHub. The Wednesday trigger on each machine never changes; the instructions it runs are refreshed automatically. Next Wednesday, all 40 machines report the new data.

Why a separate repo instead of putting skills inside eg-mcp-tools

This is the natural question, so here is the short version. Capability is identical either way: skills work the same in one repo or two. The split comes down to **two different keys for two different jobs.**

- **Using the MCP needs a server token.** Each employee gets a URL plus a token that lets their Claude *call* the hosted tools. It grants zero access to read any code. Rolling the MCP out company-wide hands out 40 of these, and none of them touch GitHub.
- **Pulling skills needs a GitHub read key.** A skill is a file, so each laptop needs a key that can *read the repo* to fetch it. If skills live in eg-mcp-tools, that means putting a key to your MCP infrastructure repo (the auth design, the integration code, the access playbook) on 40 laptops. A separate eg-skills repo means those 40 keys only ever read harmless skill instructions.

> THE DECIDING FACTOR
>
> A second repo costs us nothing: Claude builds and maintains both, employees still do one install, and the split is invisible to them. So we get a clean boundary (MCP infrastructure stays private, skills are company-readable) for free. Same outcome, safer default.

	EG-MCP-TOOLS	EG-SKILLS (NEW)
What it holds	MCP server code (tools)	Company skill files
Where it runs	Hosted gateway (Render)	Each employee's machine
How updates reach people	Instantly (live server)	Daily pull + pull-before-run
Can schedule runs	**No**	**Yes (set up once at install)**
Install per person	Once (done for pilot users)	Once (3-minute paste)

Security model

Two questions: what can employees see, and what could go wrong.

What employees can see

Everything committed to eg-skills is readable by every employee with access, forever, including the full edit history. GitHub never forgets: deleting a secret in a later commit does not remove it from history. So the repo carries exactly one kind of content: **instructions, never secrets, never personal context.**

- **No credentials in the repo, ever.** No API keys, tokens, passwords, or service account files. Skills name what they need (for example "your Google connection") and each machine supplies its own credentials locally. This is the same pattern skills-sync already proved.
- **No personal context.** Skills are stripped of anything Anthony-specific (paths, second brain, Jarvis, internal URLs) before commit. The skill optimizer already enforces this, which is one reason it ships first.
- **Automatic secret scan before every commit** as a backstop, so a key can never slip in by accident.

What could go wrong, and the answer for each

RISK	MITIGATION
Someone malicious pushes a bad skill that 40 Claudes then execute	This is the real risk in any central-update system. Write access is limited to Anthony and Brian, both with two-factor enabled. Everyone else is read-only. One unreviewed commit cannot come from anyone outside the two of us.
A well-intentioned edit breaks a skill for everyone	Test on Anthony's machine first, then the pilot machines pull before the company does. If something slips through, one-click revert on GitHub rolls every machine back at their next pull. Central rollback is a feature the zip-file world never had.
A secret accidentally gets committed	Prevention stack above (no-secrets rule + per-machine credentials + automatic scan). Worst case the repo is private and read-only, and we rotate the leaked key immediately.
Repo access leaks outside the company	Repo stays private. Employees get read-only access scoped to this one repo. Offboarding = remove their access in one click, same as Google Workspace.

A scheduled skill does something unintended on a machine	Scheduled skills are scoped narrowly (read the agreed data, write to the agreed sheet) and reviewed before they enter the repo. Claude's own permission system on each machine is still the final gate.

NET SECURITY POSITION

This is materially **safer** than today's zip-and-Slack flow. Zips are uncontrolled copies: no audit trail, no revocation, no rollback, and stripped-of-secrets only if someone remembered. The repo gives us one reviewed source, a permanent change log of who changed what and when, instant rollback, and one-click offboarding.

Rollout plan

Same crawl-walk-run we used for the MCP gateway.

0 **Build and self-test (Anthony)**

Create the eg-skills repo, the one-paste installer, and the daily pull. Load the first skill (emazing-skill-optimizer). Run the full loop on Anthony's own machine: install, edit, confirm the update lands, confirm a scheduled run picks up an edit.

1 **Pilot with 4 to 5 people**

The same early adopters already on the MCP gateway. Verify install friction, the daily pull, and one scheduled skill writing to a sheet from every pilot machine. Fix what surfaces.

2 **Company-wide**

One Slack post with the single paste block. From then on, every new skill and every update flows automatically. Brian owns rollout and adoption tracking, per the gatekeeper model we agreed on the call.

Decisions needed from Brian

1. Agree on the two-repo split

eg-mcp-tools stays the tools layer. A new eg-skills repo becomes the skills layer. Nothing about the current MCP gateway changes. The split keeps your MCP infrastructure private while skills stay company-readable, at no extra cost or install friction.

Recommended: yes. Same capability either way, and the separate repo keeps the access boundary clean for free.

2. How employees get read access

Option A: each employee gets a GitHub account with read-only access to this one repo (clean audit and one-click offboarding, but 40 accounts to manage). Option B: a single shared read-only key baked into the installer (zero friction for employees, but one shared credential whose worst case is someone outside reading our skills repo).

Recommended: A for the pilot, then decide for company-wide once we see the real friction.

3. First skills in, and the pilot group

Proposed first three: emazing-skill-optimizer, the Claude usage reporter (Jose's), and skills-sync. Pilot group: the people already on the MCP gateway.

Recommended: start with the optimizer alone in Phase 0, add the other two in Phase 1.

Level 2 · Visual Artifact · Generated by Claude · June 12, 2026 · Source decisions: June 2 power user call (Granola)