# The ECC Install Guide

How to install ECC into Claude Code or Codex without breaking your setup: the command, the scopes, the rules folders, and the fixes.

## Why this guide exists

ECC is a free, MIT-licensed repo that turns Claude Code into a coordinated engineering system instead of a very fast typist. As of September 2026 it sits at roughly 256,000 GitHub stars and ships 68 agents, 291 skills, and 94 commands. When I filmed the video it was 240,000 stars, 64 agents, 261 skills, and 84 commands, which tells you how fast it moves.

The usual way people try this fails in a predictable way. They find the repo, run whatever command they see first in a blog post, then run the source installer too because it looked more complete. Now they have duplicate skills, duplicate commands, and hooks firing twice, and every session feels heavier than before they started.

One command, one path per tool, and you are done in about ten minutes. That is the whole promise of this guide, and the rest of it is the detail that keeps you out of the reinstall loop.

---

01

## How ECC actually works

1. ECC installs as a plugin called **ecc@ecc**, which hands Claude Code a catalog of 68 specialized agents, 291 skills, and 94 slash commands. Skills are the main surface now, and they load on demand rather than sitting in your context all session.
2. Underneath the catalog sit hooks, rules, and memory. Hooks fire on tool events and enforce your standards while the agent works. Rules are always-loaded guidelines you pick per language. Memory persists what mattered from a session so the next one does not start blind.
3. Those pieces run one loop: plan, test, implement, review, verify, remember, improve. A planner drafts the blueprint, a tdd-guide forces the failing test first, a code-reviewer reads the result from a fresh context, and AgentShield scans your prompts, hooks, MCP config, permissions, and secrets.

---

02

## The one thing almost everyone gets wrong

Do not stack two install methods into the same tool. This is the most repeated failure in the project's own troubleshooting notes, and it is the reason people give up on ECC in week one.

Installing ECC once into several different tools is fine. Claude Code plus Codex, both running ECC, works exactly as intended. What breaks is installing it twice into one tool: the plugin path plus a full manual `./install.sh --profile full` on top. You get duplicated skills, duplicated commands, and hooks that fire twice on every edit.

**WATCH OUT**

If it already looks duplicated, clean up in this order: remove the Claude Code plugin install, run the ECC uninstall from the project directory that holds the install state, delete any rule folders you copied by hand, then reinstall once using a single path. ECC only removes files it recorded in its own install state, so it will not touch your other files.

---

03

## The commands and config files

**SOURCE CODE**

The full repo, MIT-licensed and free:

**https://github.com/affaan-m/ecc**

**TERMINAL**

```
# The recommended install. Run it from your project directory.
npx ecc-universal@2.2.1 setup

# Same wizard through other package runners
pnpm dlx ecc-universal@2.2.1 setup
yarn dlx ecc-universal@2.2.1 setup
bunx ecc-universal@2.2.1 setup

# Claude Code, Codex and Kimi Code in one reviewed flow
npx ecc-universal@2.2.1 install --guided
```

**INSIDE CLAUDE CODE**

```
/plugin marketplace add https://github.com/affaan-m/ECC
/plugin install ecc@ecc
```

**~/.CLAUDE/SETTINGS.JSON (DECLARATIVE INSTALL)**

```
{
  "extraKnownMarketplaces": {
    "ecc": {
      "source": {
        "source": "github",
        "repo": "affaan-m/ECC"
      }
    }
  },
  "enabledPlugins": {
    "ecc@ecc": true
  }
}
```

**TERMINAL (RULES ARE NOT SHIPPED BY THE PLUGIN)**

```
git clone https://github.com/affaan-m/ECC.git
cd ECC
mkdir -p ~/.claude/rules/ecc
cp -R rules/common ~/.claude/rules/ecc/
cp -R rules/typescript ~/.claude/rules/ecc/   # swap for your stack
```

**~/.CLAUDE/SETTINGS.JSON (TOKEN SETTINGS THE REPO RECOMMENDS)**

```
{
  "model": "sonnet",
  "env": {
    "MAX_THINKING_TOKENS": "10000",
    "CLAUDE_AUTOCOMPACT_PCT_OVERRIDE": "50",
    "CLAUDE_CODE_SUBAGENT_MODEL": "haiku"
  }
}
```

The repo puts the model switch at around a 60% cost reduction and the thinking cap at around 70% less hidden thinking cost per request. Switch to Opus with `/model opus` only when you need deep architectural reasoning.

**SITES SHOWN IN THE VIDEO**

- ECC, the repo itself
  **https://github.com/affaan-m/ecc**
- Affaan Mustafa, the maintainer
  **https://github.com/affaan-m**

## Set it up in Claude Code or Codex

1. **Check the prerequisites first.** You need Node.js 18 or newer for the package, plus Git and the Claude Code CLI on your PATH. The plugin needs Claude Code v2.1.0 or later because of how that version handles plugin hooks. Run `node -v` and `claude --version` before anything else.
2. **Pick the project directory you will install from, and remember it.** ECC records its install state there, and the uninstall, doctor, and repair commands all expect to be run from that same folder later.
3. **Run `npx ecc-universal@2.2.1 setup`.** The wizard inventories the official marketplace and every existing Claude install scope before it writes a single file, so it is safe to run even if you are not sure what you already have.
4. **Choose a scope when it asks: user, project, or local.** User makes ECC available everywhere on your machine, project ties it to the repo you are in, local keeps it to your own machine copy. Pick user if you are a solo operator and want it everywhere. You can rerun the same command later to change your mind.
5. **Answer the hook question deliberately.** Any install that would put the hook runtime in place stops and asks first. If you want ECC's rules, agents, commands, and workflows without runtime hooks, run `npx ecc-universal@2.2.1 install --profile minimal --target claude` instead.
6. **Add rules by hand if you want them.** Claude Code plugins cannot distribute rule packs, so clone the repo and copy across `rules/common` plus the one language pack you actually use. Do not copy all of them.
7. **Note where skills land.** A manual Claude install puts each skill directly in `~/.claude/skills/<skill-name>/`, or `.claude/skills/<skill-name>/` for a project install, because that is where Claude Code looks. The folder name has to match the skill's own name, so do not rename folders after the fact.
8. **For Codex, use the native plugin path.** Run `codex plugin marketplace add affaan-m/ECC`, then `codex plugin add ecc@ecc`, then `codex plugin list --json`. Both add commands are safe to rerun. Inside Codex, run `$configure-ecc` for the guided flow.
9. **Verify the install.** Inside Claude Code run `/plugin list ecc@ecc`. From the terminal, `npx ecc-universal@2.2.1 list-installed` and then `npx ecc-universal@2.2.1 doctor` tell you what is actually on disk.
10. **Do a first real run.** Open a project, run `/ecc:plan "add a health check endpoint"`, and watch what comes back. A successful first run gives you a written implementation blueprint from the planner agent, not code. If you get a plan with steps and files listed before anything is written, ECC is working.

Ten minutes is a realistic budget for all of that, most of it spent waiting on the clone and reading the wizard's prompts.

> 💡 **QUICK TIPS**
>
> - Namespaced commands are the plugin form: `/ecc:plan`. The bare `/plan` form only shows up on manual installs.
> - The three public names are deliberately different and not interchangeable: the repo is `affaan-m/ECC`, the plugin is `ecc@ecc`, the npm package is `ecc-universal`.
> - npm releases are cut per version tag, not per commit. Install from git if you want what is on main today.

---

## 04 The commands I actually run

**PLAN BEFORE ANY BUILD**

```
/ecc:plan "[DESCRIBE THE FEATURE IN ONE SENTENCE, INCLUDING THE CONSTRAINT THAT MATTERS MOST]"
```

**FIX A BUG WITHOUT GUESSING**

```
Use the tdd-workflow skill. First write a failing test that reproduces this exactly: [PASTE THE ERROR OR THE STEPS TO REPRODUCE]. Do not touch the implementation until the test fails for the right reason.
```

**REVIEW WHAT YOU JUST WROTE**

```
/code-review

Focus on [FILE OR FOLDER]. I care most about [CORRECTNESS / SECURITY / DEAD CODE].
```

**REPAIR A BROKEN BUILD**

```
/build-fix

The failing command is: [BUILD OR TEST COMMAND]
The first error line is: [PASTE FIRST ERROR]
```

**AUDIT BEFORE YOU SHIP**

```
/security-scan

Scope it to [FOLDER]. Flag anything touching [SECRETS / AUTH / USER INPUT] first.
```

**CLOSE AND REOPEN A LONG SESSION**

```
/save-session

Then next time: /resume-session. When the context feels heavy mid-session, run /context-budget to see where it went.
```

## When it breaks

| ERROR | FIX |
| --- | --- |
| **ECC appears twice, or hooks fire twice on every edit** | You stacked the plugin install and a full source install. Remove the Claude Code plugin, run `node scripts/ecc.js uninstall --dry-run` from the ECC checkout, delete rule folders you copied by hand, then reinstall once. |
| **"Duplicate hooks file detected: ./hooks/hooks.json resolves to already-loaded file"** | Claude Code v2.1+ loads a plugin's hooks/hooks.json automatically. Do not add a "hooks" field to .claude-plugin/plugin.json. Remove it and restart. |
| **/plugin reports an existing install or a scope conflict** | Claude Code owns those built-in commands and ECC cannot intercept them. Use the guided setup (npx ecc-universal@2.2.1 setup) or clear the conflicting plugin scope first. Do not layer a manual install on top. |
| **npm reports a version or cache error on install** | Confirm what the registry actually has with `npm view ecc-universal version`, then retry with that version. |
| **`yarn dlx` is not recognised** | Yarn Classic 1 has no dlx. Use npx, install the package globally, or upgrade Yarn. |
| **Codex registers the marketplace but no skills load** | Run `node scripts/codex/check-plugin-cache.js` from an ECC checkout. If parent references are unresolved, run `codex plugin marketplace upgrade ecc`, then `codex plugin add ecc@ecc` again, then restart Codex. |
| **Your local Claude setup got wiped and ECC is gone** | Nothing to repurchase. Run `node scripts/ecc.js list-installed`, then `doctor`, then `repair` before reinstalling. That usually restores the managed files. |
| **Sessions feel expensive after installing** | Apply the token settings above, drop MAX_THINKING_TOKENS to 10000, set the subagent model to haiku, and reserve Opus for architecture work. |