devoteam
Snowflake Partner

N0 · ALPHA 0.1.9

ENABLEMENT · STREAMLIT-COCO · 2026-09

streamlit-coco Enablement **pack.**

Go through this deck, then the hands-on exercises. By the end you can build a Streamlit app with a CoCo Copilot, human-in-the-loop approval gates, and run every bundled example unaided.

TIME BUDGET	CERTIFICATION	OWNER
~3.5–4.5 hours total	Enablement quiz · 8/10 to pass	Laurent Letourmy

What is **streamlit-coco**?

A Python/Streamlit library, not a hosted service

A PyPI package that embeds Snowflake's **Cortex Code Agent SDK ("CoCo")** into a Streamlit app: a right-rail or full-panel Copilot, human-in-the-loop approval gates for destructive tools, a live preview of an app CoCo just wrote, structured output into your own widgets, and a headless `query()` / `CocoSession` API for scripts and CI.

PACKAGE
streamlit-coco

VERSION
0.1.9 · alpha

REQUIRES
Python 3.10+ · Streamlit 1.53+

The problem it **solves**

Everything you would otherwise build by hand

WITHOUT STREAMLIT-COCO

Session state that survives Streamlit reruns, streaming render of the agent transcript, readable tool cards instead of raw JSON dumps, and an approval flow for anything the agent could write, edit, or execute — all built from scratch.

WITH STREAMLIT-COCO

That plumbing as composable functions — `panel()`, `chat_input_bar()`, `copilot_rail()`, `render_approvals()` — so an app author writes the page and the options, not the session bridge.

Who it's for, **what you need**

No dedicated warehouse required for this track

Who this is for

Consultants and engineers adding a CoCo-based Copilot to a Streamlit app on a Snowflake account with Cortex Code access. Basic Python and enough Streamlit to read `st.container` / `st.session_state` — deep Streamlit expertise is not required.

Prerequisites

Python 3.10+ and `uv`

Snowflake account with Cortex Code access

CoCo CLI (`cortex`) on `PATH`

Git access to `streamlit-coco-dev` (or just `pip install`)

WHAT YOU'LL BE ABLE TO DO

After this training, **you can...**

5 learning objectives — the Enablement certification checks all five

devoteam
Snowflake Partner

01 **Explain it, and when it applies**

Versus the bare CoCo CLI or SDK — see the README's "When not to use" section.

02 **Install and run every example**

`make chat`, `make backlog`, `make bi-semantic`, `make headless`, connected to a real Snowflake account.

03 **Build the preferred pattern**

`panel()` + `chat_input_bar()`, with `require_approval_for` configured.

04 **Wire a Copilot rail into a real app**

Configure `CocoOptions` for a real workflow, embed `copilot_rail()` in an existing multipage app.

05 **Diagnose a broken session**

Using `check_environment()`, `STREAMLIT_COCO_DEBUG`, and the Technical Guide's help reference.

Install & **connect**

devoteam
Snowflake Partner

Full reference → doc/deployment/local.md; condensed path here

1 · CLONE & INSTALL

`git clone .../streamlit-coco-dev.git`
`make install` — `uv sync --extra dev`

2 · COCO CLI

`cortex --version`
Not on PATH? `export CORTEX_CODE_CLI_PATH=/path/to/cortex`

3 · SNOWFLAKE CONNECTION

Create `~/.snowflake/connections.toml` with a `[connections.analytics]` entry (account, user, authenticator). Legacy `config.toml` also detected. No `connection=` set → the Snowflake CLI's default connection is used.

You're ready **when...**

Every box checked before the guided tour

- `python -c "import streamlit_coco as c; print(c.__version__)"` prints `0.1.9`
- `cortex --version` succeeds
- `check_environment(...).ready` is `True`
- `make check` passes — ruff + unit/smoke pytest, no failures
- `make chat` opens a working app and streams a response with at least one tool card

GUIDED TOUR · 1 OF 2

`make chat` — the preferred pattern

You own the page, CoCo owns the session

- **Start gate first**
  `check_environment()` rendered by `render_start_gate()` — no agent starts until you click Start.

- **Streaming transcript**
  The `@st.fragment` poll loop (`run_every=0.25`) renders progressively — the rest of the page never reruns.

- **Tool cards, not JSON**
  A Glob/SQL card appears — the readable card the library builds instead of a raw dump.

- **Approval banner**
  Ask it to write something — **Approve once · Always allow · Deny** fires via `render_approvals()`.

The rail — **a column, not a page**

make backlog and make bi-semantic

MAKE BACKLOG

Product Backlog Desk — Board / Epic / Ticket / Release. Copilot is **not** a nav page: it opens from the header **Copilot** button or an action link, closes with **Close**.

MAKE BI-SEMANTIC

The fullest use of `copilot_rail()`. Title row: Copilot, Connection popover, icon-only **Display config** — two sliders, Last messages (default 8) and First n characters (default 200).

WHAT TO NOTICE ACROSS ALL THREE APPS

The agent always runs server-side. Approval gates are the same primitive everywhere. `panel()` is the common core; `copilot_rail()` adds connection chrome, job queuing, and transcript-compactness controls.

HANDS-ON · 6 EXERCISES

Build it, **break it, fix it**

~120–150 min — the real test is Exercise 5

devoteam
Snowflake Partner

EX 1

Preferred pattern

`panel()` + `chat_input_bar()` end to end

EX 2

Approval gate

Approve once, then Deny with a reason

EX 3

Structured output

`output_schema` into a widget

EX 4

Headless

Same library, zero Streamlit import

EX 5 · REAL SITUATION

Build from scratch

README only — the real test of "trained"

EX 6

Copilot rail

Display config sliders in a real product app

DEMO SCRIPT · 1 OF 2

Your 10-min demo — **hook & approvals**

devoteam
Snowflake Partner

Full script → enablement/demo-guide.md — dry-run once, timed, before presenting

0:00–0:30 · HOOK

"Wiring a CoCo agent into a Streamlit app yourself means building session management, streaming render, tool cards, and safety gates from scratch. streamlit-coco gives you that in a few lines of Python."

0:30–2:30 · STEP 1 — FIRST IMPRESSION

Show `examples/chat_app.py` — four calls. Run `make chat`. Type a sample-data question. Point at the streaming transcript and the tool card.

2:30–5:30 · STEP 2 — APPROVAL GATES

Ask it to write SQL → **Approve once**, results appear. Ask it to create a table → **Deny** with a reason. "Every destructive action requires explicit human approval, configured per tool."

DEMO SCRIPT · 2 OF 2

Structured output — and the close

Land the CTA — do not let the demo end without an ask

devoteam
Snowflake Partner

5:30–7:00 · STEP 3 — STRUCTURED OUTPUT

Switch to `make structured`. Show `output_schema`. Ask for a breakdown → renders as a chart, not raw text. "No manual JSON parsing."

7:00–9:30 · CLOSING

`pip install "streamlit-coco[sdk]"`. Approval gates on by default. One library: panel, structured, headless. Apache-2.0.

CALL TO ACTION

"Try it against your own Snowflake account this week — the README quickstart gets you to a working app quickly." Buffer after 9:30 for `make headless` or the rail, if the room has time.

DEMO SCRIPT · IF THEY ASK

Questions you'll **definitely get**

Have these answers ready before you present

devoteam
Snowflake Partner

- **"Does it work with Streamlit in Snowflake?"**
  Not shipped today — local (CLI + Streamlit + Snowflake config on one host) is the only shipped topology; SiS/SPCS are threat-model scope only.

- **"Is it production-ready?"**
  Alpha (0.1.9) — the API may still move.

- **"Can I customize which tools need approval?"**
  Yes — `allowed_tools` / `require_approval_for` in `CocoOptions`, per tool.

- **"Is there an audit log of approvals?"**
  Not persisted today — "Always allow" is session-scoped only and clears on reset (tracked as SEC-01).

RESOURCES & NEXT STEPS

Everything you'll **need again**

devoteam
Snowflake Partner

Bookmark this slide

Training pack

`doc/training/enablement/` — this track

`doc/training/training/` — 3 use cases

`doc/training/user-guide/` · `technical-guide/`

`doc/training/certifications/enablement.md` ← **take this next, score 8/10**

Repo docs

`README.md` · `doc/api.md`

`doc/deployment/local.md`

`doc/features/` — per-feature reference

Contact: **Laurent Letourmy** — laurent.letourmy@devoteam.com

You're certified on Enablement when: hands-on exercises complete + demo runnable unaided + quiz ≥ 8/10.

devoteam
Snowflake Partner

Go run **make chat.**

Then the hands-on exercises. Then the Enablement certification. Then build your own app.

Laurent Letourmy
Asset Owner
laurent.letourmy@devoteam.com

streamlit-coco-dev
github.com/DevoteamSP
issues, PR reviews, questions