TAG TRACK 1

Working with Claude Enterprise

A foundational session on the platform TAG runs on: models, prompting, Projects, Cowork, Artifacts, and Skills.

Tuesday morning · 4 hours · 5 hands-on labs · 2 breaks

Analytics & Reporting · Data Platform · Everyone else

AGENDA

Claude Enterprise Foundational Training

SEGMENT	TOPIC	TIME
1	Welcome, what's new, vocabulary, your plan, data handling	20 min
2	Prompting and context engineering	20 min
Lab 1	Prompt Workbench	15 min
	Break	10 min
3	Model selection	20 min
Lab 2	Model Comparison	20 min
4	System prompts, Projects, standing context	15 min
Lab 3	Build a System Prompt	15 min
	Break	10 min
5	Cowork and Artifacts	15 min
Lab 4	Cowork, Projects and Artifacts	20 min
6	Skills, including the ones already installed	15 min
Lab 5	Build a Weekly-Update Skill	25 min
7	Sharing and governance	10 min

Segment 1

What is new in Claude

20 minutes

SEGMENT 1

What's new with Claude

Four changes worth resetting your mental model around. Each one returns later this morning.

MODELS

Opus 5

Released July 24. Close to Fable 5 intelligence at the same price as Opus 4.8. A 1M-token context window, up to 128K output tokens, thinking on by default, and a five-level effort setting.

SURFACES

Cowork on web and mobile

No longer desktop-only. Sessions run server-side and follow you across devices. The laptop can be closed while a task runs.

ORGANIZATION

Projects, unified

Chat and Cowork now share one Projects system. A project can be shared with Can view or Can edit permissions, on either surface.

OUTPUTS

Artifacts, live

A published artifact can call connectors each time someone views it, showing live data rather than a snapshot. Refreshable on a schedule.

SHARED VOCABULARY

Six terms used when talking about AI

Some generic terms used when talking about Claude and AI

Model

The trained system itself. Sonnet 5 and Opus 5 are different models with different tradeoffs.

Choosing a model is the first decision of every session.

Token

Roughly a word or word fragment. The unit Claude reads and generates.

About 750 words comes to roughly 1,000 tokens.

Context window

Everything Claude can see at once. Nothing outside it exists to the model.

Opus 5's window holds about one million tokens.

Agent

Claude acting in a loop: act, observe, decide again, until a task is done.

A Cowork session is Claude run as an agent.

MCP

The open standard connectors are built on. Not Claude-specific. Helps LLMs talk to systems.

One protocol, adopted across the industry.

Connector

A specific live connection built on MCP. Clause has a marketplace of connectors

BigQuery, Google Drive, Slack, and Calendar are all connectors.

THE MECHANISM

What a large language model is

One mechanism, repeated

A large language model is trained on enormous amounts of text to predict the most likely next token, given everything that came before it. That is the whole mechanism, repeated one token at a time until the response is done.

Understanding, planning, and reasoning are this same prediction process, at a scale where the predictions become consistently useful. Mechanically it is similar to autocomplete. The difference is scale, not mechanism.

WHY IT MATTERS TODAY

Everything today is context control

Prompts, Projects, system prompts, and Skills are all strategies for deciding what is inside the window when Claude answers.

WHY IT MATTERS TODAY

A prompt is input, not an incantation

Claude resolves ambiguity using the statistically most common reading, which may not be yours. Precision is on you.

WHY IT MATTERS TODAY

Verification is asked for, not guaranteed

Fluent restatement of a document is not the same as checking it. Lab 2 makes this visible.

Nothing outside the context window exists to the model, in any sense. Every mechanism this morning is a different way of controlling what is in that window.

YOUR PLAN

The five Claude plans, and where TAG sits

TAG operates on Claude Enterprise. The other four rows exist for contrast.

Plan	For	What it adds
Free	Individuals	Limited daily usage, core models.
Pro	Individuals	Higher usage, Claude Code, Projects, connectors. Opus 5 is the strongest model on Pro.
Max	Individuals	Highest individual usage ceiling, priority access. Opus 5 is the default model.
Team	Organizations	Shared billing, central admin, usage metadata for admins.
Enterprise	Organizations	Everything in Team, plus SSO, SCIM provisioning, audit logs, custom data retention, and the Compliance API.

Who can see what, precisely

ANTHROPIC'S SIDE

No training on your data

Team and Enterprise conversations are not used to train Anthropic's models, by default, with no opt-in required. Anthropic's own staff cannot read conversation content except with explicit consent, a safety flag, or legal process.

TAG'S SIDE, BY DEFAULT

Admins see usage, not content

By default, the Enterprise dashboard shows TAG's admins metadata: active users, message counts, spend. Audit logs record account events, sign-ins and permission changes, not conversation content.

THE ONE EXCEPTION

The Compliance API is the exception

An optional, Enterprise-only feature an Organization Owner must deliberately enable. If enabled, a Compliance Access Key can read every chat, file, and project in the org. Retention defaults to indefinite unless a custom period is configured.

Segment 2

Context Engineering

20 minutes

SEGMENT 2

From prompt engineering to context engineering

The lever is not only the words in one prompt. It is everything Claude has in front of it when it answers.

Prompt engineering → **Context engineering**

WHAT COUNTS AS CONTEXT

- The system prompt
- Project knowledge
- Tool and connector definitions
- Conversation history
- Pasted documents
- The question itself

The operating principle

More context is **not** automatically better. Model performance measurably degrades as the window fills with low-signal material, and information buried in the middle of a long prompt is weighted less reliably than information at the start or end. The goal is the smallest set of high-signal tokens that reliably produces the outcome.

The six techniques that follow are how context gets engineered. Lab 1 applies all six to a prompt you will actually reuse.

SIX TECHNIQUES

The techniques, at a glance

Two get their own slides. All six are in the workbench in Lab 1.

TECHNIQUE 1

Be clear and direct

The test: would a colleague with zero context understand exactly what to do? Ambiguity gets resolved with the most common reading, not yours.

TECHNIQUE 2

Give Claude a role

A role shifts tone and judgment, not just word choice. A reviewer reads differently than a summarizer.

TECHNIQUE 3

Use XML tags

Separate instructions, context, and pasted documents so nothing has to be inferred.

TECHNIQUE 4

Show examples

Two or three sharp, diverse examples outperform a paragraph of rules. Examples act as temporary, conversation-scoped training data.

TECHNIQUE 5

State what to do

A list of prohibitions leaves the shape of a good answer undefined. Positive instruction defines the target.

TECHNIQUE 6

Aim for the right altitude

Too rigid breaks on the first edge case. Too vague assumes context Claude does not have.

TECHNIQUE 3 · XML TAGS

Before and after

The failure this fixes: Claude treating part of your instruction as part of the document, or the reverse.

WITHOUT TAGS

```
Summarize this contract. Focus on termination clauses.
Here's the contract: [pasted text]
```

Where does the instruction end and the contract begin? You hesitated for half a second. Claude has the same problem, and resolves it silently.

WITH TAGS

```
<instructions>
Summarize this contract.
Focus on termination clauses.
</instructions>

<contract>
[pasted text]
</contract>
```

Nothing left to infer.

There is no fixed, trained set of correct tag names. Name tags for what they hold and stay consistent within one prompt. The consistency does the work.

TECHNIQUE 4 · SHOW EXAMPLES

Examples outperform descriptions

In-context learning: examples in the prompt shift the output pattern for the rest of the conversation, without touching the model itself.

DESCRIBING THE RULES

```
Categorize these expense lines the way finance prefers.
Software goes under technology unless it is a one-time
license, travel includes mileage but not meals, meals are
their own category unless they were during travel booked
through the portal...
```

Every rule spawns an edge case. The list is never finished.

SHOWING THREE EXAMPLES

```
<examples>
Adobe license renewal → Technology
Mileage, office visit → Travel
Team lunch, no travel → Meals
</examples>

Categorize the lines below the same way.
```

The pattern carries the rules the prose could not finish stating.

Two or three sharp, diverse examples beat ten similar ones. Pick examples that disagree with each other on the dimension that matters.

TECHNIQUE 6 · THE RIGHT ALTITUDE

Two ways instructions fail, one way they work

TOO RIGID

"If the office is in the Southeast region and the tier is 3, 4, or 5, flag it, otherwise don't."

Breaks silently on the first real underperformer that does not match the enumerated pattern.

TOO VAGUE

"Flag anything that looks off in these offices."

Assumes Claude shares your definition of "off." It does not. You get someone else's judgment, confidently applied.

RIGHT ALTITUDE

"Flag any underperforming office, using tier and region as signals, and explain your reasoning for each flag."

Specific enough to steer, loose enough for judgment, and the required explanation makes every flag checkable.

The fix for a rigid instruction is not more detail. It is a different kind of instruction: named signals, room for judgment, and a required explanation.

THE TECHNIQUES, COMBINED

A complete prompt, assembled

One working prompt using all six. This is the shape Lab 1 builds toward.

```
You are a senior operations analyst reviewing weekly
office metrics for a regional director.

<instructions>
Flag any underperforming office, using visit volume
and cancellation rate as signals. Explain each flag
in one sentence. End with the single office most
worth a call this week.
</instructions>

<example>
Office 214: visits down 18% for 3 weeks, cancellations
stable. Flag: sustained volume decline, not seasonal.
</example>

<data>
[pasted weekly metrics]
</data>
```

TECHNIQUE 1 · 2

Clear, direct, and a role

One sentence establishes who is reading and for whom.

TECHNIQUE 3

Tags separate the parts

Instructions, example, and data cannot bleed into each other.

TECHNIQUE 4 · 5

An example, stated positively

One worked flag defines the target shape. No list of don'ts.

TECHNIQUE 6

The right altitude

Named signals, required reasoning, one concrete deliverable.

LAB WORK

Lab 1

Prompt Workbench

Assemble a real, reusable prompt technique by technique, and watch the context budget respond to the model and task complexity you choose.

15 minutes

Open the interactive lab guide and select Lab 1. Every step, prompt, and checkpoint is written there.

B R E A K

Segment 3, model selection, starts on time.

Segment 3

Different Models in Claude

20 minutes

SEGMENT 3

The current lineup

Intelligence, speed, and cost move in opposite directions. The lineup is the industry picture; the availability line below is TAG's.

Haiku 4.5
Near-frontier performance, fast, the most economical tier. Real-time and high-volume work, sub-agent tasks.

Sonnet 5
Frontier intelligence at scale. Code generation, data analysis, content creation, agentic tool use.

Opus 5
Close to Fable 5 intelligence at the price of Opus 4.8. Complex agentic coding, large refactors, vision-heavy and enterprise work.

Fable 5
The highest available capability. Long-running agents, deep reasoning, advanced research.

Opus 5, released July 24
A 1M-token context window, up to 128K output tokens, thinking on by default, and a five-level effort setting. The new default on Claude Max, the strongest model on Claude Pro.

Enabled at TAG today: Haiku 4.5, Sonnet 5, and Opus 5. Fable 5 is not yet enabled in TAG's workspace. The labs use the three you have.

TWO STARTING STRATEGIES

Efficiency-first or capability-first

Anthropic documents both as valid. The choice follows from the task, not from a fixed rule.

START SMALL

Efficiency-first

Start with Haiku 4.5. Test thoroughly against real inputs. Upgrade only for a specific, demonstrated capability gap.

Best for: prototyping, tight latency requirements, cost-sensitive deployments, high-volume straightforward tasks.

Example: extracting dates and amounts from scanned invoices. Simple, and easy to verify against ground truth.

START STRONG

Capability-first

Start with Opus 5. Prove the workflow. Then optimize toward efficiency by lowering effort or stepping down models as it matures.

Best for: complex reasoning, accuracy-critical work, advanced coding, high-autonomy agentic tasks.

Example: drafting the first pass of a regulatory filing. Errors are costly and accuracy outweighs iteration speed.

Either way, the deciding evidence is the same: a small benchmark set built from your real prompts and real data, not intuition about which model feels smarter.

ANTHROPIC'S MODEL SELECTION MATRIX

The use-case matrix,

Model	Built for	Example
Haiku 4.5	Real-time, high-volume, sub-agent tasks	Tagging 500 patient-feedback comments by theme before a human reads a single one.
Sonnet 5	Code generation, data analysis, tool use at scale	A first-pass SQL query against the reporting warehouse. Summarizing a meeting transcript. Drafting campaign copy.
Opus 5	Complex agentic work, multihour coding, enterprise analysis	Working through six offices' P&L files, reconciling definitions across them, and drafting the variance narrative.
Fable 5	Long-running agents, deep reasoning, advanced research	A multi-hour autonomous research task. Not enabled at TAG today

The matrix gives the category. The next slide gives the feel, and Lab 2 makes the difference visible on a document written for this room.

HOW MODELS AND EFFORT INTERACT

What that actually feels like

The matrix tells you the category. This is the working feel of each combination.

Sonnet at high effort

Like spending an afternoon with a strong generalist. Thorough, methodical, will grind through the whole problem and show its work.

Opus at low effort

Like five minutes with an expert. Fast, confident, usually right, and occasionally worth double-checking precisely because it is fast.

Opus at high effort

Like giving the expert the afternoon. The combination for problems where being wrong is expensive.

Neither axis is a quality dial you should always max out. Effort and model are cost and latency decisions as much as capability decisions.

EFFORT AND SPEED

Change the model, or change the effort?

The mechanical distinction

Effort controls how much work the model puts into a task it can already handle: one model, more or less deliberation, latency, and cost. The model controls the ceiling of what is handleable at all. One is a dial, the other is a ceiling.

THE DIAL

Raise the effort

When the task is within the model's reach and you want it done more thoroughly and carefully. Frequently the better lever: the intuitive move is reaching for a bigger model when a higher effort setting on the current one does the job.

THE CEILING

Change the model

When the problem exceeds what the current model can do at any effort. Evidence, not intuition: build a small benchmark from real prompts and data, compare accuracy and edge-case handling, then weigh the cost difference.

A third lever: fast mode

Opus 5 and Opus 4.8 support fast mode, a research preview delivering up to 2.5x higher output speed at premium pricing. For the case where latency matters more than cost, not a general default.

LAB WORK

Lab 2

Model Comparison

Run one internal memo, written fresh for this lab with one planted inconsistency, across models and effort levels. Score who verifies and who summarizes.

20 minutes

Open the interactive lab guide and select Lab 2. Every step, prompt, and checkpoint is written there.

Segment 4

Where context is stored

15 minutes

SEGMENT 4

The same idea, four surfaces

Context engineering, applied to a whole working session instead of one prompt. Standing context, set once, loaded automatically.

Claude Chat and API

System prompt

Role and standing rules for a conversation or an application, set once at the top.

Claude Projects

Project instructions

Standing context that applies automatically to every chat inside the project, for everyone with access.

Claude Cowork

Global instructions

Persistent preferences that follow you across Cowork sessions.

Claude Code

CLAUDE.md

A file in the repository Claude reads automatically at the start of every session. Shared with the team through source control.

Same principle everywhere: anything you would otherwise repeat at the start of every conversation belongs in standing context instead.

PROJECTS

Not just a bigger system prompt

Project instructions
Standing behavioral context. Applies automatically to every chat inside the project.

Project knowledge
Documents Claude searches and cites from, uploaded once, never pasted again.

Conversation history
Every chat inside the project accumulates into an organized, searchable record.

How knowledge scales
Small projects load everything directly into context, the same as pasting it yourself. Once knowledge outgrows the context window, the project switches automatically to retrieval: Claude searches project knowledge and reads only what is relevant to the question, expanding effective capacity roughly tenfold. Practical consequence: name files well, they retrieve better, and reference a document by name to focus the search on it.

Chat and Cowork share one Projects system, and a project can be shared with Can view or Can edit permissions on either surface.

THE DECISION RULE

When does something belong in a Project?

DISPOSABLE

A fresh chat

Something you will never need to reference again.

One-off questions. Quick drafts. Exploration that does not need to persist.

Example: a one-time response to a press inquiry.

STANDING

A Project

You will return to this, or a teammate needs the same grounding.

Recurring context. Shared definitions. Anything where two people should get the same answer.

Example: the monthly board report, where format and definitions must stay identical every month.

The test is repetition: the second time you paste the same background document into a fresh chat, it should have been a Project.

LAB WORK

Lab 3

Build a System Prompt

Write standing context for a task you actually repeat, apply the altitude test to your own draft, then try to break it with a rephrased request.

15 minutes

Open the interactive lab guide and select Lab 3. Every step, prompt, and checkpoint is written there.

B R E A K

Segment 5, Cowork and Artifacts, starts on time.

Segment 5

Claude Cowork and Artifact

15 minutes

SEGMENT 5

Chat or Cowork?

The same model, the same Projects system. What differs is how long Claude works before coming back to you.

Chat: a single exchange

One question, one answer, done. You stay in the loop the whole time.

Best for drafting, quick analysis, and questions.

"Draft a reply to this email" is Chat: one pass through the loop.

Cowork: a working session

Claude works across files, tools, and connectors, with standing context applied automatically, and checks in before hard-to-reverse actions.

"Go through my last 20 open tickets, draft replies, and flag anything needing a human decision" is Cowork: many passes, one check-in.

Underneath both is one mechanism: Claude acts, observes the result, and decides again, until the task is done. Cowork is that loop given more room and more tools.

SETTING UP A COWORK PROJECT

Four steps, done once

1

Create the project

Name it for the initiative, not the date. “Office Performance” outlives “July analysis.”

2

Add project knowledge

The documents Claude should search and cite from: definitions, templates, source reports.

3

Set instructions

Standing rules: definitions, tone, and what done looks like here. Written once, applied to every session inside.

4

Share it

Can view or Can edit, same as Chat. The whole point: a teammate opens the same project and gets the same grounding.

Lab 4 walks these four steps on a real initiative, then builds a live artifact inside the project.

SCHEDULING

Run it without you

A scheduled task triggers Cowork on a cadence, instead of a person asking fresh every time.

Recurring reports

Every Monday at 8am, summarize last week's numbers against the definitions in the project, and post the result.

Refreshing artifacts

A chart or status page that updates itself on a schedule from connected data, instead of being rebuilt by hand.

No device required

Scheduled tasks run server-side. Your laptop can be closed. The output is waiting when you are not.

The honest test of a good scheduled task: would a competent assistant, given these instructions and nothing else, produce the right thing? If not, the instructions are not done.

ARTIFACTS

Reusable outputs, not one-off results

A standalone output kept and returned to, not a result buried in scrollback.

A document
A report, a template, a format reused across projects instead of regenerated each time.

A chart
Built from a connected data source, not a static screenshot. Refreshable.

A small tool
A calculator, a checker, an interactive page, kept and iterated on.

The mechanic that matters: per-viewer permissions
A published artifact can call connectors each time someone views it, so it shows live data rather than a snapshot. Each viewer's connector calls run through that viewer's own account permissions, not the author's, and every viewer approves access before the artifact's first call on their behalf. An artifact does not inherit its publisher's access for every future viewer.

THE RECURRING-OUTPUT PATTERN

Build once, refresh forever

STEP 1

Build the artifact

A chart, a scoring template, a status report, from a connected data source.

→

STEP 2

Save it in a Project

Reusable by anyone with access. Not regenerated from scratch per person.

→

STEP 3

Schedule the refresh

A scheduled Cowork task updates it weekly or monthly, on its own.

Not a dashboard tool. The pattern fits one specific, well-defined recurring output, not open-ended exploration. If people will slice it twenty ways, it belongs in BI.

LAB WORK

Lab 4

Cowork, Projects and Artifacts

Set up a real project, build a calendar artifact inside it, and prove the per-viewer permission mechanic with a teammate's own account.

20 minutes

Open the interactive lab guide and select Lab 4. Every step, prompt, and checkpoint is written there.

Segment 6

Claude Skills

15 minutes

SEGMENT 6

What a skill is

A reusable instruction set, applied automatically when relevant. Written once, run by anyone. A skill is a folder, not a single file.

```
weekly-update/
  SKILL.md          required
  template.md       optional
  sample-data.csv   optional
```

ALWAYS LOADED

Name and description only

At session start, each installed skill costs roughly 30 to 100 tokens: just its name and description. Many skills, no context penalty.

LOADED ON MATCH

The full SKILL.md body

When a request matches the description, or you invoke the skill by name, the full instructions load into the conversation.

LOADED ON DEMAND

Supporting files

Templates, references, and scripts load only at the moment a step needs them. This is progressive disclosure.

Skill matching runs through the same language understanding as everything else. No separate matching engine exists, which is why fixing a skill that will not trigger is the same fix as a prompt that will not land.

SKILLS YOU ALREADY HAVE

Four built-in skills ship with Claude

Anthropic-managed document skills, installed for every user. They trigger automatically when the request calls for them.

PPTX
Presentations
Builds and edits PowerPoint decks: layouts, speaker notes, templates.
"Turn this outline into a 10-slide deck for the ops review."

XLSX
Spreadsheets
Creates workbooks with working formulas, charts, and clean structure.
"Build a spreadsheet comparing these three vendor quotes."

DOCX
Documents
Produces formatted Word documents: headings, tables, tracked structure.
"Draft this policy as a formatted Word document."

PDF
PDFs
Creates and fills PDFs, extracts content, merges and splits files.
"Fill this intake form from the details below."

One setting gates all four
Built-in skills require Code execution and file creation to be enabled in settings. With it on, no invocation is needed: ask for the document and the right skill loads itself.

The same skills are source-available at github.com/anthropics/skills, worth reading as worked examples before writing your own in Lab 5.

The kitchen and the recipe

The analogy

A connector is the professional kitchen: access to tools and ingredients. A skill is the recipe: step-by-step instructions for turning that access into something valuable. A kitchen without recipes produces improvisation; a recipe without a kitchen produces nothing.

The description is the whole mechanism

Claude matches incoming requests against each skill's description to decide whether to load it. A routing rule, not documentation.

A precise description states what the skill does, when to use it, and the trigger phrases someone would actually say. A vague description, “helps with reporting,” essentially never fires.

Without a skill, with a skill

The same recurring report, requested cold: fifteen back-and-forth messages, three failed tool calls, twelve thousand tokens.

With a well-described skill: two clarifying questions, zero failed calls, six thousand tokens, and the identical format every single week.

An ambiguous description is exactly as risky as an ambiguous prompt, for the identical mechanical reason.

TODAY'S BUILD

A weekly-update skill, end to end

The same recurring report, the same format, every week, without re-explaining it. Lab 5 builds all three files.

FILE 1 OF 3

SKILL.md

The required file. Frontmatter carries the name and the trigger description. The body carries the procedure: gather, compare, draft, flag.

FILE 2 OF 3

template.md

The exact output format, as a separate file the skill references. Loaded only when drafting starts. Change the template, and every future report changes with it.

FILE 3 OF 3

sample-data.csv

A small worked example the skill can be tested against, so the first run happens on known data with a known right answer.

The completion test is a fresh chat: phrase the request the way a teammate would, not the way you wrote it. If the skill triggers and follows the template, it is done. If not, the description gets sharpened, not the instructions.

LAB WORK

Lab 5

Build a Weekly-Update Skill

Build the full three-file skill from the templates in the lab guide, run it against the sample data, then prove it with the fresh-chat trigger test.

25 minutes

Open the interactive lab guide and select Lab 5. Every step, prompt, and checkpoint is written there.

Segment 7

Governance

15 minutes

SEGMENT 7

Sharing and distribution

Between a skill that works and a skill the team can use sits one review gate.

Why your own test is not enough

An author only tests against their own inputs and their own mental model. A second person phrases requests differently, hits edge cases the author never imagined, and judges the output without the author's built-in context. The second person's test is the review, not a formality on top of it.

What makes a skill shareable

A clear stated purpose someone can read without running it. A second person's successful test on inputs the author did not try. And no facility- or person-specific detail baked in: a skill that references “the Q3 dashboard” by name works only for people who know what that means; one that asks for the dashboard as an input works for everyone.

Generalizing away the hardcoded specifics is usually the majority of the work of making a skill shareable. The domain logic was already right.

GOVERNANCE

The 4D framework

Anthropic's AI Fluency Framework, developed with academic partners: how a person works with AI responsibly. A competency model, not an enterprise control.

Delegation
Deciding whether, when, and how to hand work to AI at all. The judgment before the prompt.

Description
Describing goals precisely enough to prompt useful behavior. Everything in Segment 2 lives here.

Discernment
Assessing what comes back. Fluency is not accuracy; Lab 2 was a discernment exercise.

Diligence
Owning the final product. A drafted email under your name is yours, the same as if an assistant had drafted it for your signature.

The more autonomous the mode, the more diligence must be built into the setup itself rather than applied at the end. Nobody reviews every intermediate step of a scheduled Cowork task; the instructions carry the diligence.

Enterprise technical controls, SSO, audit logs, retention, the Compliance API, were covered in Segment 1. Admin-visibility questions belong there, not here.

SEGMENT 7

What to use when

Everything from this morning, on one slide. The shape of the work picks the tool.

The work looks like	Reach for
One question, one answer, never needed again	**A fresh chat**
Recurring context you or a teammate will return to	**A Project**
Multi-step work across files, tools, and connectors	**A Cowork session**
The same procedure, repeatable by anyone, on demand	**A Skill**
A standing output people view, share, or reuse	**An Artifact**
The same output on a cadence, with nobody asking	**A scheduled task**

These compose. A Project holds the context, a Skill holds the procedure, an Artifact holds the output, and a scheduled task keeps it current.

WRAP

What good looks like at 90 days

CHECKABLE

One skill per team

At least one skill per team submitted for review. Submitted, not certified; a rough first draft counts.

CHECKABLE

Standing instructions in place

Set up on whichever surface you use most: project instructions, Cowork global instructions, or a system prompt.

CHECKABLE

Fewer hand-built reports

Fewer manually assembled recurring reports, measured against the cadence your team already tracks.

Where to get help

Office hours are Wednesday afternoon, for anyone continuing deeper. Documentation lives at support.claude.com and platform.claude.com/docs. Account questions, retention, the Compliance API, and connector enablement belong with TAG's Claude administrators.