COMMUNITY STUDY GUIDE · V1.0 · JULY 2026

Claude Certifications
Builder Track

One guide for the three technical Claude credentials — Developer, Architect Foundations, and Architect Professional. Free to share.

3	1	20	~30h
CERTIFICATIONS	SHARED CORE	DOMAINS TOTAL	CORE + YOUR PATH

How this guide is built:
Learn the shared core once — the API, prompting, tools & MCP, agents, and Claude Code that all three exams test — then follow the path section for the credential you're taking.

About This Guide

This guide was built by Prompt Goblins, an independent community for people who build with AI. Anthropic's Claude certification program now spans **four credentials across three roles**. Three of them are for people who build — engineers and architects — and they share a large common body of knowledge. Rather than repeat that shared material three times, this guide teaches it once as a **Core**, then adds a focused **path** for each of the three builder credentials.

The fourth credential, **Claude Certified Associate – Foundations**, is aimed at non-technical business users and is covered in a separate companion guide (it shares almost none of the material below).

> **Access:** All four exams are currently available only to organizations in the **Claude Partner Network**. The network itself is free for any organization bringing Claude to market to join, and exams are delivered through **Pearson VUE** (online proctoring or test centers). Broader public access is expected but not yet live as of July 2026.

Which builder credential is for me?

IF YOU…	TAKE	WHY
Write application code, build agents, ship integrations (1–5 yrs SWE)	**Developer – Foundations** CCDV-F	Heaviest on API integration, custom tools/MCP servers, security, and evals.
Design agent systems hands-on and configure Claude Code (6+ mo hands-on)	**Architect – Foundations** CCAR-F	Scenario-based; orchestration, Claude Code config, prompt engineering, context reliability.
Own end-to-end solution architecture, governance, and stakeholders (3+ yrs architecture)	**Architect – Professional** CCAR-P	Adds solution design, RAG, evaluation strategy, compliance, and lifecycle/stakeholder management.

Jump to: Exams at a Glance · Part I: The Core · Part II: Developer · Part III: Architect–Foundations · Part IV: Architect–Professional · Change Log

None of the three requires another as a prerequisite — each is awarded on exam performance alone, and all three are independently bookable. Many people take Architect–Foundations before Professional, but that ordering is a recommendation, not a gate.

How to read the attribution tags

Every resource is tagged so you know where it came from:

OFFICIAL	**Named on an Anthropic-curated certification course list** (the Partner Network Learning Path or a partner-gated prep page).
ANTHROPIC	**Anthropic-published, community-mapped to exam domains.** Other Academy courses, official docs (platform.claude.com / code.claude.com), the "Building Effective Agents" post, and DeepLearning.AI partnership courses.
SUPPLEMENTAL	**Community / third-party.** Useful but unofficial. Items tagged OPTIONAL in the study plan can be skipped without hurting coverage.

Tags are deliberately grayscale — **color always means an exam domain**, never a source tier.

Everything is listed in **study order**, not grouped by tier — do it top to bottom. Optional supplemental items sit where they naturally fit so you can slot them in or skip them, rather than hunting through a separate section.

The Three Builder Exams at a Glance

All three share the same format shell: 120 minutes, scaled 100–1,000 score, pass at 720, Pearson VUE proctored, 12-month validity, up to 4 attempts per rolling year (14/30/90-day waits). They differ in scope, item count, and price.

	DEVELOPER – FOUNDATIONS	ARCHITECT – FOUNDATIONS	ARCHITECT – PROFESSIONAL
Exam code	CCDV-F	CCAR-F	CCAR-P
Items	53	60	63
Structure	Standalone items	4 scenarios of 6, drawn at random	Standalone items
Fee	$125	$125	$175
# Domains	8	5	7
Audience	Engineers, 1–5 yrs SWE + 6 mo Claude	Solution architects, 6+ mo hands-on	Senior architects, 3+ yrs architecture
Center of gravity	Building & integrating apps; security; evals	Orchestration; Claude Code; reliability	Solution design; governance; lifecycle

The overlap is real. Roughly 40–50% of what these exams test is shared: Claude models & selection, prompt and context engineering, tool use & MCP, agent patterns, and the common exam logistics. That shared material is **Part I: The Core** below. Read it regardless of which exam you're taking, then jump to your path.

Part I · The Core — shared by all three builder exams

CORE STUDY PLAN

Work top to bottom. All Skilljar courses are free with certificates. This core is ~24 hours; each path below adds a few hours on top. **OFFICIAL** = on an Anthropic course list · **ANTHROPIC** = Anthropic content, mapped · **SUPPLEMENTAL** = community.

#	RESOURCE	PLATFORM	FORMAT	TIME	BEST FOR	✓
1	AI Fluency: Framework & Foundations O	Skilljar	Video+quiz	1h	All	☐
2	Building with the Claude API O	Skilljar	84 lec, 10 quiz	8h	All (most surface area)	☐
3	"Building Effective Agents" A	anthropic.com	Reading	1h	All · agent patterns	☐
4	Introduction to Model Context Protocol O	Skilljar	Video+code	2h	All · MCP	☐
5	MCP: Build Rich-Context AI Apps A	DLAI	11 vid+code	2h	Dev · Arch-F	☐
6	MCP: Advanced Topics A	Skilljar	Video+code	2h	All · transports, sampling	☐
7	Claude Code in Action O	Skilljar	15 lec+quiz	1h	All · Claude Code	☐
8	Claude Code: A Highly Agentic Coding Assistant A	DLAI	10 vid	2h	Arch-F · Dev	☐
9	Core documentation deep reads A	platform/code.claude.com	Reading	3h	All (see table below)	☐
10	Anthropic Cookbooks — hands-on S OPTIONAL	GitHub	Code	2h	All · practice	☐

CORE CONCEPT CLIFFS NOTES

These ideas recur across all three exams. Learn them once here.

Agentic loops & agent patterns

The agentic loop is the backbone: send request → check `stop_reason` → if `tool_use`, execute the tool, append the result, loop → if `end_turn`, stop. `stop_reason` is the *only* reliable termination signal — never parse natural language or use arbitrary iteration caps.

Workflows vs. agents: workflows have code control the flow (chaining, routing, parallelization, orchestrator-workers, evaluator-optimizer); agents let the LLM control the flow. Start simple — single call → workflow → agent — and only escalate when the task demands it.

Hub-and-spoke / subagents: a coordinator decomposes work and delegates to subagents with *isolated* context. Never dump full coordinator context into a subagent; pass only what it needs.

Tools & MCP

Tool descriptions are the primary selection mechanism. Poor descriptions cause misrouting — fix the description *before* reaching for few-shot examples. Keep ~4–5 tools per agent; beyond that, selection degrades.

MCP primitives: Tools = model-controlled · Resources = app-controlled · Prompts = user-controlled. Know which primitive fits a given scenario.

Transports: stdio = local/same machine; Streamable HTTP = remote/production. The older HTTP+SSE transport was deprecated in the 2025-03-26 spec revision. `.mcp.json` = project-level (in VCS); `~/.claude.json` = user-level; use env-var expansion for secrets.

Structured errors: transient (retry) · validation (fix input) · business (rule violation) · permission (escalate).

Prompt & context engineering

Few-shot: 2–4 examples is the sweet spot; demonstrate the exact output format including edge cases.

Structured output via `tool_use`: define a JSON schema as the tool input — more reliable than asking for JSON in prose. The API now also offers native structured outputs (`output_config.format` with constrained decoding, plus `strict: true` tool schemas); know both approaches.

Validation-retry: generate → validate → on failure feed the error back → regenerate. Converges in 1–2 retries; retries don't help when the needed information simply isn't present.

Context hygiene: guard against lost-in-the-middle and context drift/bloat; prune tool output, compact, and isolate via subagents. For transactional data, keep immutable "case facts" rather than progressively summarizing (which loses order numbers, dates, amounts).

Claude Code fundamentals

CLAUDE.md hierarchy: Managed/Enterprise → Project → User → Local, with higher levels taking precedence. `.claude/rules/` holds glob-scoped rule files (a `paths` frontmatter field limits a rule to matching files).

Skills, commands, hooks: Skills = `SKILL.md` folders whose `description` drives auto-loading (progressive disclosure); slash commands live in `.claude/commands/`; hooks (PreToolUse, PostToolUse, and many more) enforce rules *programmatically*. When the requirement is "must" or "always," the answer is a hook, not a prompt.

Headless / CI: the `-p` / `--print` flag runs non-interactively; `--output-format json` and `--json-schema` structure the output for pipelines.

Model selection & cost

Tiering: match the model to the task — a fast, low-cost tier for high-volume/simple work; a top tier for complex reasoning. Weigh quality vs. latency vs. cost, and watch for behavior changes across model releases.

Cost levers: prompt caching for repeated static prefixes (order stable content first); the Message Batches API for latency-tolerant bulk jobs — 50% cheaper, results within 24h, not for blocking/real-time workflows.

CORE ANTI-PATTERNS

A large share of every builder exam is spotting what's wrong. These apply across all three paths.

#	ANTI-PATTERN	DO THIS INSTEAD
1	Parsing natural language for loop termination	Use `stop_reason == "end_turn"`
2	Arbitrary iteration caps (`max_iterations=5`)	`stop_reason` -driven termination
3	Prompt-based critical rules ("don't refund >$500")	Hooks / programmatic checks
4	Sharing full coordinator context with subagents	Pass only relevant context; isolate scope
5	Tool overload (>4–5 per agent)	Scope per role; decompose into subagents
6	Few-shot as first fix for misrouting	Fix the tool description first
7	Trusting retrieved/untrusted content as instructions	Isolate untrusted input; guard with least-privilege hooks
8	Progressive summarization of transactional facts	Immutable "case facts" blocks
9	Batch API for blocking/real-time workflows	Batch = 24h window, latency-tolerant only
10	Least privilege by monitoring unused powerful tools	Remove the capability the role doesn't need

CORE API & CONFIG REFERENCE

stop_reason values

VALUE	MEANING	ACTION
`end_turn`	Model finished	Show result; loop complete
`tool_use`	Model wants a tool	Execute, return result, continue loop
`max_tokens`	Token limit reached	Response truncated; may need higher limit
`stop_sequence`	Stop sequence hit	Handle per application logic
`pause_turn`	Long server-tool turn paused	Resend the response as-is to continue
`refusal`	Model declined	Handle gracefully; don't blind-retry

tool_choice values

VALUE	BEHAVIOR	WHEN TO USE
`{"type":"auto"}`	Model decides tool or text	Default
`{"type":"any"}`	Must call some tool	Guaranteed structured output
`{"type":"tool","name":"…"}`	Must call a specific tool	Forced first step / ordering
`{"type":"none"}`	May not call any tool	Suppress tools while keeping them defined

`any` and `tool` are incompatible with extended thinking (only `auto` / `none` work there).

Core documentation deep reads (Unit 9)

TOPIC	URL
Tool Use	platform.claude.com/docs/en/agents-and-tools/tool-use/overview

TOPIC	URL
Prompt Engineering	platform.claude.com/docs/.../prompt-engineering/overview
Prompting Best Practices	platform.claude.com/docs/.../claude-prompting-best-practices
Structured Outputs	platform.claude.com/docs/.../structured-outputs
Batch Processing	platform.claude.com/docs/.../batch-processing
Context Windows	platform.claude.com/docs/.../context-windows
Agent SDK Overview	code.claude.com/docs/en/agent-sdk/overview
Claude Code: Memory / Skills / Hooks	code.claude.com/docs/en/{memory, skills, hooks, sub-agents}
MCP Specification	modelcontextprotocol.io/specification

Part II · Developer – Foundations Path CCDV-F · $125 · 53 items

For engineers who **build, integrate, and ship** Claude-powered applications, agents, and workflows — 1–5 years of software engineering, 6+ months with Claude, comfortable in Python and/or TypeScript. Not for non-technical users or prompt-only roles.

> **What makes this exam different:** it's the most *code-and-integration* heavy of the three. **Applications & Integration is a full third of the exam (33.1%)**, and it's the only builder exam with dedicated **Security & Safety** (8.1%) weight. If you come from the Architect side, budget extra time on API mechanics, software-engineering foundations, secure-by-design, and building custom tools/MCP servers.

BLUEPRINT — 8 DOMAINS

WT	DOMAIN & KEY SKILLS
33.1%	**Applications & Integration** — Claude API mechanics (messages, tools, streaming, vision, thinking, caching, batch); software-engineering foundations (REST, JSON, async, VCS, refactoring); application design across interfaces; configuration management (CLAUDE.md, settings.json, version pinning); requirements & systems life cycle.
16.8%	**Model Selection & Optimization** — LLM fundamentals (tokens, context, sampling, thinking modes, n-shot); model tradeoffs (Opus/Sonnet/Haiku); cost & token management (usage tracking, prompt caching, checkpointing).
14.7%	**Agents & Workflows** — agent vs. workflow decision criteria; manager/subagent hierarchies; construction with the Agent SDK, custom loops, hosted vs. self-hosted; patterns & frameworks (tool-use loops, memory, LangGraph/PydanticAI/Strands).
11.0%	**Prompt & Context Engineering** — context management (drift/bloat, pruning, compaction, subagent isolation); prompt engineering (clarity, few-shot, placement, sanitization); output handling (structured output, defensive parsing).
10.6%	**Tools & MCPs** — tool implementation (function calling, descriptions, error handling, approval patterns); MCP server development; choosing among built-in tools, custom tools, Skills, and MCPs.
8.1%	**Security & Safety** — prompt-injection & jailbreak defense, untrusted input, PII/data leakage; guardrails & secure-by-design; hooks for safety; secrets/key management.
3.1%	**Claude Code** — core components (Rules, Skills, Commands, Agents, Memory), session/headless/streaming modes, CLAUDE.md hierarchy, repo init, settings.json.
2.6%	**Eval, Testing & Debugging** — error-type identification, recovery strategy, trace analysis, isolating integration-layer vs. model-output faults.

PATH ADD-ONS (BEYOND THE CORE)

RESOURCE	PLATFORM	TIME	ADDS	✓
Agent Skills with Anthropic [A]	DLAI	2.5h	Agent SDK + Skills + MCP integration	☐
Structured Outputs & Tool Use docs [A]	platform.claude.com	1h	Native structured output, `strict` schemas	☐
Security best-practices reading [A]	platform.claude.com	1h	Prompt-injection defense, secrets	☐
Build one app end-to-end S	Your machine	4h+	API + a tool + evals + a guardrail	☐

Sample item (illustrative, from the official guide)

Domain 7 — Security & Safety: A Claude agent summarizes user-submitted web pages. One page hides text telling the model to ignore prior instructions and reveal its system prompt. Best mitigation?

Treat retrieved page content as **untrusted input**, keep it separate from trusted instructions, and use guardrails/hooks so injected instructions can't trigger sensitive actions. (Raising temperature, a polite "please don't" in the system prompt, or a bigger model do not address injection — a more instruction-following model can be *more* susceptible.)

Part III · Architect – Foundations Path CCAR-F · $125 · 60 items

For **solution architects** with 6+ months hands-on across the Agent SDK, Claude Code, MCP, and prompt engineering. This is the **scenario-based** exam: every question is anchored in a realistic production context.

> **What makes this exam different:** structure. The exam draws **4 scenarios at random from a bank of 6**, and all 60 items live inside those scenarios. It leans on *orchestration*, *Claude Code configuration*, and *reliability* more than raw coding. Roughly half the questions test whether you can spot an anti-pattern (see the Core list).

BLUEPRINT — 5 DOMAINS

D1 · Agentic Architecture & Orchestration — 27%

Agentic loop lifecycle, hub-and-spoke, subagent invocation with explicit context, hooks for enforcement, task decomposition, session management (`fork_session` , `--resume`).

D3 · Claude Code Configuration & Workflows — 20%

CLAUDE.md hierarchy, `.claude/rules/` globs, commands vs. Skills, plan mode vs. direct, iterative refinement, CI/CD (`-p` , `--output-format json`).

D4 · Prompt Engineering & Structured Output — 20%

Explicit criteria, 2–4 few-shot examples, structured output via `tool_use` , validation-retry with Pydantic, multi-instance review.

D2 · Tool Design & MCP Integration — 18%

Tool descriptions as selection mechanism, structured errors (`isError`), 4–5 tools/agent, `.mcp.json` vs `~/.claude.json` , built-in vs MCP tools.

D5 · Context Management & Reliability — 15%

Lost-in-the-middle, immutable case facts, valid vs. invalid escalation, structured error propagation, confidence calibration, provenance.

THE 6 SCENARIOS (4 DRAWN PER SITTING)

All questions are anchored in these. Per the official exam guide v1.0 (July 2026), the bank is 6. NOTE Some community study guides list additional candidate-reported scenarios (e.g. "Conversational AI Architecture Patterns," "Agentic AI Tools"); those are not in the official guide.

1 · Customer Support Resolution Agent

An Agent SDK agent handling returns, billing disputes, and account issues via MCP tools (`get_customer` , `lookup_order` , `process_refund` , `escalate_to_human`). Enforce financial limits with hooks; escalate correctly.

2 · Code Generation with Claude Code

Accelerate development — generation, refactoring, debugging, docs — with custom slash commands, CLAUDE.md config, and plan mode vs. direct execution.

3 · Multi-Agent Research System

A coordinator delegates to specialized subagents (web research, doc analysis, synthesis, report generation); complete reports with citations and conflicting-source handling.

4 · Developer Productivity with Claude

Explore unfamiliar codebases, generate boilerplate, automate routine tasks using built-in tools (Read, Write, Bash, Grep, Glob) plus MCP servers.

5 · Claude Code for Continuous Integration

Automated code review, test generation, and PR feedback in CI using `-p` and `--output-format json`; prompts must minimize false positives.

6 · Structured Data Extraction

Extract from unstructured documents, validate with JSON schemas via `tool_use`, handle edge cases, nullable fields, and inconsistent formats with validation-retry.

Escalation: valid vs. invalid triggers (D5)

Valid: explicit customer request ("get me a manager"), policy gap (request not covered), capability limit (no progress after reasonable attempts), financial op above threshold (enforce via hook). **Invalid:** sentiment analysis (mood ≠ complexity), model self-rated confidence (confidently wrong), or an over-engineered auto-classifier.

PATH ADD-ONS (BEYOND THE CORE)

RESOURCE	PLATFORM	TIME	ADDS	✓
Introduction to Agent Skills O	Skilljar	30m	Skills vs commands vs hooks (D3)	☐
Introduction to Subagents A	Skilljar	1h	Isolated-context delegation (D1)	☐
Escalation & reliability patterns A	code.claude.com	1h	Hooks, structured errors (D5)	☐

Part IV · Architect – Professional Path CCAR-P · $175 · 63 items

For **mid-to-senior architects and tech leads** — 3+ years in systems architecture/platform engineering plus 6+ months with Claude in production — who own the **full lifecycle** from discovery to deployment, and who face stakeholders, security, and compliance.

> **What makes this exam different:** it moves *up the stack*. Where Architect–Foundations is hands-on orchestration, Professional adds **solution design, RAG pipelines, evaluation strategy, governance/compliance (GDPR, HIPAA, FedRAMP), and stakeholder & lifecycle management**. It is **not** scenario-locked like Foundations, and it does **not** require passing Foundations first — but the hands-on Foundations material is excellent groundwork.

BLUEPRINT — 7 DOMAINS

WT	DOMAIN & KEY TASKS
19%	**Integration** — evaluate tool/agent config for capability bloat; auth/authz gap analysis; accuracy-latency tradeoffs; observability at scale; **design a RAG pipeline** (chunking, indexing, retrieval matched to data shape); choose the connection protocol (MCP, API/CLI, agent-to-agent); progressive discovery vs. monolithic context.
17%	**Solution Design & Architecture** — translate business problems into Claude solutions; end-to-end architectures with feedback loops; pick the pattern (workflow / agentic / augmented LLM); multi-agent orchestration; decomposition; align to business value pillars.
16%	**Evaluation, Testing & Optimization** — define metrics (accuracy, latency, cost, safety); build eval datasets & frameworks; A/B testing; diagnose prompt failure/hallucination/model mismatch; optimize cost-performance; monitor with logging/observability.
14%	**Governance, Safety & Risk Management** — guardrails & safety controls; identify LLM failure modes; human-in-the-loop validation; regulatory compliance (GDPR, HIPAA, FedRAMP); ethical AI (bias, fairness, transparency).
14%	**Stakeholder Communication & Lifecycle Management** — structured discovery & requirements; communicate architectural tradeoffs; manage feedback loops & SLAs; document architectures; support discovery → design → handoff → monitoring → iteration.
13%	**Claude Models, Prompting & Context Engineering** — model selection tradeoffs; system prompts, templates, guardrails; zero/few-shot, chain-of-thought; context-window & token optimization; prompt reuse (caching, modular prompts, Skills).
7%	**Developer Productivity & Operational Enablement** — configure Claude tooling for teams (e.g. Claude Code); improve dev workflows with AI tooling; support debugging & operational resolution.

PATH ADD-ONS (BEYOND THE CORE)

Professional is lighter on courses and heavier on **architecture practice and reading**. There is no single prep course — build and reason about end-to-end systems.

RESOURCE	PLATFORM	TIME	ADDS	✓
RAG & retrieval deep read [A]	platform.claude.com + Cookbooks	2h	Chunking, indexing, contextual retrieval (D1)	☐
Evaluation & observability practices [A]	platform.claude.com	2h	Eval datasets, metrics, monitoring (D3)	☐
Responsible use / governance reading [A]	anthropic.com/learn	1.5h	Guardrails, compliance, ethics (D4)	☐
Design one end-to-end solution [S]	Your project	6h+	Discovery → RAG → eval → observability → handoff	☐
Architecting-with-Claude community writeups [S] OPTIONAL	Blogs / GitHub	2h	Real-world patterns & tradeoffs	☐

Sample item (illustrative, from the official guide)

Domain 3 — Integration: A support agent can read tickets, draft replies, issue refunds, and delete accounts. Staff only ever read tickets and draft replies. Applying least privilege, the best change is to **remove the refund and delete tools from the agent's configuration entirely** — eliminating the attack surface, not just logging or confirming it. (Logging and confirmation prompts are detective/compensating controls; a bigger model is unrelated to authorization scope.)

Change Log

V1.0 — JULY 2026

Initial release of the combined Builder-Track guide. Structure and content were validated against the official Anthropic exam guides (all v1.0, "Effective July 2026," exam codes CCDV-F / CCAR-F / CCAR-P), the Anthropic certification FAQ, Pearson VUE, and the current platform.claude.com / code.claude.com documentation, on July 19–20, 2026.

- [ ] Merged the three builder credentials into one core-first guide: shared Core (Part I) plus per-credential paths (Parts II–IV).
- [ ] Blueprints, domain weights, item counts, and sample questions taken directly from the official v1.0 exam guides.
- [ ] Common exam logistics (Pearson VUE delivery, $125/$125/$175 pricing, 12-month validity, 4 attempts per rolling year with 14/30/90-day waits, partner-network eligibility) verified against the certification FAQ.
- [ ] All documentation links use current canonical domains (platform.claude.com for API, code.claude.com for Claude Code / Agent SDK); MCP spec points to modelcontextprotocol.io/specification.
- [ ] Supplemental/community items are marked `OPTIONAL` inline and kept in study order rather than in a separate section.