

Claude Certified Architect

field notes

What the exam is, what it tests, how to prepare, and exactly where to apply.

60

questions

120

minutes

720

to pass (of 1,000)

\$125

per attempt

WHERE TO APPLY

<https://anthropic-partners.skilljar.com/claude-certified-architect-foundations-certification>

<https://www.pearsonvue.com/us/en/anthropic.html>

(certification page + Pearson VUE scheduling — full steps on page 11)

Use this as a map, not a shortcut.

Built from the official Exam Guide v1.0 (July 2026).

Independent study notes — not affiliated with or endorsed by Anthropic.

PROVIDED BY CREATOR

@jaimehta.ai

[instagram.com/jaimehta.ai](https://www.instagram.com/jaimehta.ai)

At a glance

Everything you need to know before you decide to sit CCAR-F.

CREDENTIAL	Claude Certified Architect – Foundations
EXAM CODE	CCAR-F (you will also see it written CCA-F)
QUESTIONS	60 items — multiple choice and multiple response; each item tells you how many to pick
STRUCTURE	4 realistic scenarios, drawn at random from a bank of 6
TIME LIMIT	120 minutes
DELIVERY	Proctored via Pearson VUE — online (OnVUE) or at a test center
PASSING SCORE	720 on a scaled range of 100–1,000
EXAM FEE	\$125 USD per attempt (partner-tier discounts show at checkout)
VALIDITY	12 months from the date the credential is awarded
RETAKES	Wait 14 days, then 30, then 90; max 4 attempts in a rolling 12 months
ELIGIBILITY	Offered to organizations in the Claude Partner Network — check the official page

Who it is for

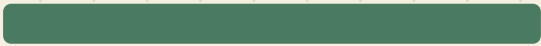
Solution architects who design and ship production apps with Claude. The guide assumes roughly six months of hands-on work with the Claude API, the Agent SDK, Claude Code and MCP — and enough judgment to weigh trade-offs, not just recall syntax.

MY MARGIN NOTE

Scores are criterion-referenced: you are measured against a fixed standard, not against other people. The score report also shows % correct per domain.

Five domains, one weighting

Study time should follow the weights — more than a quarter of the exam sits in domain 1.

- 1 Agentic Architecture & Orchestration**
 **27%**
Agent loops, coordinator–subagent design, hooks, task decomposition, session state.
- 2 Tool Design & MCP Integration**
 **18%**
Tool descriptions, structured errors, tool distribution, MCP config, built-in tools.
- 3 Claude Code Configuration & Workflows**
 **20%**
CLAUDE.md, rules, commands and skills, plan mode, iteration, CI/CD.
- 4 Prompt Engineering & Structured Output**
 **20%**
Explicit criteria, few-shot, JSON-schema output, retries, batching, multi-pass review.
- 5 Context Management & Reliability**
 **15%**
Context preservation, escalation, error propagation, human review, provenance.

MY MARGIN NOTE

Rough time split if you are short on time: about 27 / 18 / 20 / 20 / 15 hours of every 100 hours you study. Community estimates say 15–20 hours total for people already building with Claude, 30–40 if you are new.

Six stories, four on your paper

Every question hangs on a realistic production scenario. You get 4 of these 6 at random — so study all six.

1 Customer Support Resolution Agent

Agent SDK agent with MCP tools (get_customer, lookup_order, process_refund, escalate_to_human). Goal: 80%+ first-contact resolution and knowing when to escalate.

PRIMARY: DOMAINS 1 · 2 · 5

2 Code Generation with Claude Code

A team uses Claude Code for generation, refactoring, debugging and docs — custom slash commands, CLAUDE.md, plan mode vs direct execution.

PRIMARY: DOMAINS 3 · 5

3 Multi-Agent Research System

A coordinator delegates to search, document-analysis, synthesis and report subagents to produce comprehensive, cited reports.

PRIMARY: DOMAINS 1 · 2 · 5

4 Developer Productivity

An agent helps engineers explore unfamiliar codebases and automate repetitive work with Read, Write, Bash, Grep, Glob plus MCP servers.

PRIMARY: DOMAINS 2 · 3 · 1

5 Claude Code in CI/CD

Automated code review, test generation and PR feedback — prompts must be actionable and keep false positives low.

PRIMARY: DOMAINS 3 · 4

6 Structured Data Extraction

Pull information from unstructured documents, validate against JSON schemas, handle edge cases, feed downstream systems.

PRIMARY: DOMAINS 4 · 5

Out of scope: fine-tuning, API billing/auth, hosting MCP servers, vision, streaming, computer use, cloud-provider setup, prompt caching internals.

Agentic Architecture & Orchestration

The heaviest domain: how agents loop, delegate, and stay under control.

WHEN YOU SEE

REACH FOR

- | | |
|---|---|
| How does the agent loop end? | → Keep going while <code>stop_reason</code> is “tool_use”, stop on “end_turn”. Don’t parse prose for completion or lean on iteration caps. |
| A step order must always hold (verify ID, then refund) | → Programmatic prerequisites or hooks. Prompt instructions alone still fail a small share of the time. |
| A subagent needs earlier findings | → Put them in its prompt — subagents don’t inherit context. The coordinator needs “Task” in <code>allowedTools</code> . |
| Make independent subagents faster | → Emit several Task calls in one coordinator response so they run in parallel. |
| Report misses whole sub-topics | → Check the coordinator’s decomposition first. It is usually too narrow — the workers did what they were told. |
| Tools return mixed date or status formats | → A <code>PostToolUse</code> hook normalizes them; an outgoing-call hook can block policy breaks (e.g. refunds over a limit). |
| Fixed review vs open-ended investigation | → Prompt chaining for predictable multi-step reviews; adaptive decomposition that re-plans from findings for open tasks. |
| Resume, fork or restart a session? | → <code>--resume</code> when context is still valid, <code>fork_session</code> to branch, and a fresh start with a summary when tool results are stale. |

COMMON EXAM TRAP

“Add a stronger instruction to the prompt” is rarely the best answer when a business rule must hold every time. Deterministic problem, deterministic fix: hooks and gates.

Tool Design & MCP Integration

Tools are only as good as their descriptions, error signals and scope.

WHEN YOU SEE

REACH FOR

Model keeps picking the wrong similar tool



Fix the descriptions first: input formats, examples, edge cases, boundaries. Rename or split tools that overlap.

A tool call fails



Return a structured error: category (transient, validation, permission, business), isRetryable, a readable message.

Empty result or real failure?



Keep them apart. A valid query with no matches is a success; a timeout is an access failure the agent must handle.

Agent has too many tools (18 instead of 4-5)



Scope tools to each role, and add a narrow cross-role tool (e.g. verify_fact) for frequent needs.

Force or guarantee a tool call



tool_choice: “auto” may answer in text, “any” must call some tool, a forced choice names the exact tool.

Share an MCP server with the team



Project-level .mcp.json with \${ENV_VAR} for secrets; personal experiments go in ~/.claude.json.

Search code vs find files



Grep searches contents, Glob matches file names. If Edit can’t find a unique anchor, Read then Write.

COMMON EXAM TRAP

Adding few-shot examples or a routing layer is tempting when tool selection is wrong. The lowest-effort, highest-leverage first step is almost always better tool descriptions.

Claude Code Configuration & Workflows

Configure it once, and the whole team gets the same behavior.

WHEN YOU SEE

REACH FOR

Teammate isn't getting your instructions



Probably user-level `~/.claude/CLAUDE.md`. Hierarchy is user, project, then directory; only project files are shared via git.

Conventions for files spread across folders



`.claude/rules/` files with YAML path globs (e.g. `**/*.test.tsx`) load only when matching files are edited.

A slash command for the whole team



`.claude/commands/` inside the repo. Personal commands live in `~/.claude/commands/`.

A skill floods the chat with output



Use context: fork to isolate it, and `allowed-tools` to limit what it may touch.

Huge refactor vs one-line fix



Plan mode for architecture and multi-file migrations; direct execution for small, clearly scoped changes.

Claude Code inside a CI pipeline



`claude -p` runs non-interactively; add `--output-format json` with `--json-schema` for machine-readable findings.

Who should review generated code?



An independent Claude instance — the session that wrote the code is worse at questioning its own choices.

COMMON EXAM TRAP

Command not found for teammates? Check where the file lives. Shared = inside the repository. Personal = your home directory. Many wrong options quietly put it in the personal spot.

Prompt Engineering & Structured Output

Precise criteria, worked examples, and schemas you can trust.

WHEN YOU SEE

REACH FOR

Too many false positives in reviews



Write explicit categorical criteria: what to report, what to skip. “Be conservative” does not improve precision.

Output format keeps drifting



Add 2–4 targeted few-shot examples on the ambiguous cases, showing why one action beat the alternatives.

You need schema-valid JSON



tool_use with a JSON schema removes syntax errors — but not semantic ones like line items that don’t sum.

A field may be missing in the source



Make it optional or nullable so the model doesn’t invent a value; use “other” + detail and “unclear” enums.

Validation failed



Retry with the document, the bad output and the exact error. Useless if the information isn’t in the source.

Cut cost on bulk processing



Message Batches API: 50% cheaper, up to 24 hours, no latency promise. Fine overnight, wrong for blocking checks. Match with custom_id.

14-file PR review is inconsistent



Per-file local passes plus one cross-file integration pass. Bigger context windows don’t fix attention dilution.

COMMON EXAM TRAP

“Run it three times and keep issues found twice” sounds rigorous, but it hides real bugs that only surface intermittently. Prefer restructuring the review.

Context Management & Reliability

Keeping facts intact, escalating wisely, and being honest about uncertainty.

WHEN YOU SEE

REACH FOR

Long chats lose amounts, dates, IDs



Keep a persistent “case facts” block outside the summarized history; trim tool output to the fields that matter.

Findings buried mid-context



“Lost in the middle”: put key summaries first and organize details under clear headers.

When to escalate to a human



On an explicit request, a policy gap, or no progress. Sentiment and self-rated confidence are poor triggers.

Search returns several customer matches



Ask for another identifier. Never pick one by heuristic.

A subagent times out



Recover locally if transient; otherwise send failure type, attempted query and partial results up. Don’t hide errors or kill the run.

97% overall accuracy — automate?



Not yet. Check by document type and field, sample high-confidence outputs, calibrate confidence on labeled data.

Synthesis loses sources



Keep claim-to-source mappings with dates; show conflicting numbers with attribution instead of choosing one.

Exploring a huge codebase



Subagents for discovery, scratchpad files for findings, /compact when context fills, manifests for crash recovery.

COMMON EXAM TRAP

Marking a failure as an empty success, or aborting the whole workflow on one error, are both anti-patterns. The right answer passes structured context up so the coordinator can decide.

Six weeks, all hands-on

The guide's message is simple: build things. This plan is my suggestion, mapped to the official preparation advice.

WEEK 1

Read the official Exam Guide start to finish, plus the Terms and Exam Policy. Set up Claude Code and the Agent SDK. Note your weak domains.

WEEK 2

Domain 1: build a full agentic loop with tools, stop_reason handling, hooks, and a business-rule gate that redirects to escalation.

WEEK 3

Domains 1–2: coordinator plus two subagents; MCP tools with structured errors; a shared .mcp.json with env-var expansion.

WEEK 4

Domain 3: CLAUDE.md hierarchy, path-scoped rules, a forked skill, plan mode vs direct runs, and a CI job using claude -p.

WEEK 5

Domain 4: extraction pipeline with tool_use + JSON schema, validation-retry loop, few-shot examples, and a 100-document batch run.

WEEK 6

Domain 5 and revision: escalation, provenance, context tricks. Redo the 12 sample questions and time yourself over 120 minutes.

The four official practice exercises

1 Multi-tool agent with escalation logic

DOMAINS 1 · 2 · 5

2 Claude Code setup for a team workflow

DOMAINS 3 · 2

3 Structured data extraction pipeline

DOMAINS 4 · 5

4 Multi-agent research pipeline

DOMAINS 1 · 2 · 5

MY MARGIN NOTE

Extra help: prep courses sit in the Anthropic Partner Academy (for Claude Partner Network members). The freeCodeCamp YouTube prep course by Andrew Brown is a popular unofficial option.

Where to register

Registration runs through the Anthropic Partner Academy; scheduling and the exam itself are handled by Pearson VUE.

1 Check eligibility

The certification program is open to organizations in the Claude Partner Network. Not a member yet? Start here:

<https://partnerportal.anthropic.com/s/partner-registration>

2 Open the certification page

Sign in to the Anthropic Partner Academy, review exam details, and download the Exam Guide, Terms and Conditions and Exam Policy:

<https://anthropic-partners.skilljar.com/claude-certified-architect-foundations-certification>

3 Register and pay

Click Purchase (\$125) on that page and complete checkout. Any partner-tier discount shows at checkout.

4 Create your Pearson VUE login

You will get an email with a Pearson username and password. Sign in and pick a date — online proctored or at a test center:

<https://www.pearsonvue.com/us/en/anthropic.html>

5 Sit the exam

Bring valid, unexpired government photo ID. The name must match your registration exactly.

6 Get your result and badge

You receive pass/fail plus a scaled score and % by domain. Digital badges are issued via Credly:

<https://www.credly.com/org/anthropic/badge/claude-certified-architect-foundations>

MY MARGIN NOTE

Price and eligibility can change. The official certification page always wins over any guide — including this one.

Rules, renewal, support

Know the rules before you sit down, and what happens once you pass.

On exam day

- Stay in view of the proctor and webcam the whole session if testing online.
- Clear your desk: no phones, watches, headphones, notes or second monitors.
- You will accept a confidentiality agreement — exam questions must never be shared.
- Rescheduling: Pearson lists 48 hours for test centers; the guide says 24. Aim for 48 to be safe.

If you don't pass

- Retake after 14 days (1st fail), 30 days (2nd), 90 days (3rd).
- Up to 4 attempts per exam in any rolling 12 months; the fee applies each time.

Keeping it current

- The credential lasts 12 months.
- Renew on time with a free, non-proctored assessment in the Partner Academy.
- If it lapses, you retake the full exam at the full fee.

Need help?

- Name mismatch on registration: certifications-support@anthropic.com
- Scheduling, accommodations, results, appeals (within 14 days): Pearson VUE support.

OFFICIAL LINKS

Certification page

<https://anthropic-partners.skilljar.com/claude-certified-architect-foundations-certification>

Pearson VUE (Anthropic)

<https://www.pearsonvue.com/us/en/anthropic.html>

Partner Academy

<https://anthropic-partners.skilljar.com/>

Source: Anthropic Exam Guide v1.0.

Independent notes, not endorsed by Anthropic.

PROVIDED BY CREATOR

@jaimehta.ai

[instagram.com/jaimehta.ai](https://www.instagram.com/jaimehta.ai)