Faysal Aziz

Claude Architect Foundations

Seven Days to Understand the Architecture, Recognize the Exam Patterns, and Answer with Confidence - Claude Certified Architect (Foundations) exam prep

FA-0128 · Seven Days

Agentic & Generative AI · DevOps, Cloud & Infrastructure

Introduction

The Claude Certified Architect (Foundations) examination tests more than familiarity with Claude. Candidates must understand how production Claude systems make decisions, invoke tools, coordinate agents, preserve context, enforce workflows, and recover when something goes wrong. The examination contains five domains and 30 task statements, with the greatest emphasis placed on Agentic Architecture and Orchestration.

This seven-day accelerated course is designed for learners who require clear explanations, controlled progression, repetition, and strong exam focus. Complex architecture concepts will be reduced to manageable decision patterns without sacrificing the technical knowledge required by the syllabus. Coverage will follow the official domain weighting: Agentic Architecture and Orchestration at 27%, Claude Code Configuration and Workflows at 20%, Prompt Engineering and Structured Output at 20%, Tool Design and MCP Integration at 18%, and Context Management and Reliability at 15%.

The instructor has more than 30 years of industry experience and will deliver practical, industry-demanded content rather than an unnecessarily academic treatment of the subject. The emphasis will remain on understanding scenarios, distinguishing similar architectural choices, identifying unreliable designs, and selecting the most appropriate examination answer.

Learning outcomes

Upon completion, participants should be able to:

- Explain the structure and control flow of Claude-powered agentic systems
- Select appropriate agentic, deterministic, sequential, parallel, and multi-agent patterns

- Design clear tool interfaces and structured tool responses
- Explain MCP server, client, capability, and transport responsibilities
- Configure Claude Code instructions, rules, skills, permissions, hooks, and workflows
- Construct effective system prompts and structured-output strategies
- Apply validation, retry, feedback, batching, and review patterns
- Manage context windows, session state, caching, and long-running interactions
- Identify ambiguity, cascading errors, context degradation, and reliability risks
- Apply human-review, confidence-calibration, and information-provenance controls
- Interpret scenario-based multiple-choice and multiple-response questions
- Manage the examination pace of approximately two minutes per question

Prerequisites

- Basic understanding of software applications and APIs
- Familiarity with command-line operations
- Basic knowledge of JSON and structured data
- General awareness of software-development workflows
- Basic understanding of Git repositories and CI/CD concepts
- No advanced artificial-intelligence or machine-learning background required
- Access to Claude, Claude Code, and the supplied study and practice resources

Training outline

1. Examination Orientation and Claude Architecture Foundations

Examination structure and expectations

1. Five examination domains
2. Thirty task statements
3. Domain weightings
4. Multiple-choice and multiple-response formats
5. Scenario-based question interpretation
6. Time-management requirements
7. Weighted scoring considerations

Claude application architecture

1. Claude models and application components
2. Messages API responsibilities
3. Developer-controlled conversation state
4. System prompts and user messages
5. Content blocks
6. Tool-use request and result flow
7. Stop reasons
8. Stateless API behaviour
9. Application-side orchestration

Essential terminology

1. Agent
2. Workflow
3. Orchestrator
4. Subagent
5. Tool
6. Context
7. Session
8. State
9. Guardrail
10. Structured output

2. Agentic Architecture and Orchestration

Agentic loop fundamentals

1. User-request processing
2. Model decision points
3. Tool invocation
4. Tool-result return
5. Loop continuation
6. Final-response conditions
7. Maximum-iteration controls
8. Timeout and budget controls

9. Failure termination

Agentic and deterministic workflow selection

1. Open-ended agentic tasks
2. Predictable workflows
3. Sequential processing
4. Parallel processing
5. Routing patterns
6. Evaluator-optimizer patterns
7. Hybrid workflows
8. Cost and latency considerations

Multi-agent orchestration

1. Centralized orchestration
2. Decentralized coordination
3. Supervisor and worker patterns
4. Specialized agents
5. Parallel subagent execution
6. Result aggregation
7. Conflict resolution
8. Failure isolation
9. Shared-state risks

Subagent invocation and context passing

1. Task-specific delegation
2. Minimum necessary context
3. Relevant constraints
4. Expected output definitions
5. Context isolation
6. Result normalization
7. Parent-agent reintegration
8. Token-usage considerations

Workflow enforcement and handoff

1. Required-stage enforcement
2. Preconditions and postconditions
3. State-machine concepts
4. Explicit completion criteria
5. Controlled handoffs
6. Approval gates
7. Guardrail placement
8. Recovery paths

Claude Agent SDK concepts

1. Agent execution lifecycle
2. Hook purposes
3. Pre-tool hooks
4. Post-tool hooks
5. Permission controls
6. Monitoring points
7. Policy enforcement
8. Session lifecycle controls

Task decomposition

1. Goal identification
2. Dependency identification
3. Sequential subtasks
4. Independent subtasks
5. Parallelizable work
6. Subtask boundaries
7. Completion criteria
8. Result synthesis

Session state and resumption

1. Session identifiers
2. Persistent state

3. Checkpointing
4. Interrupted-task recovery
5. Idempotent operations
6. Duplicate-action prevention
7. State validation
8. Safe resumption

3. Tool Design and MCP Integration

Tool interface design

1. Tool purpose definition
2. Descriptive tool names
3. Clear tool descriptions
4. Input-schema design
5. Required and optional parameters
6. Parameter constraints
7. Enumeration usage
8. Output consistency
9. Tool granularity
10. Overlapping-tool avoidance

Tool execution lifecycle

1. Tool selection
2. Argument generation
3. Input validation
4. External execution
5. Tool-result construction
6. Result return to Claude
7. Continued model processing
8. Final-response generation

Structured error responses

1. Error categories
2. Machine-readable errors

3. Human-readable messages
4. Retryable and non-retryable failures
5. Validation failures
6. Authentication and authorization failures
7. Rate-limit responses
8. Partial failures
9. Recovery information
10. Sensitive-data protection

Tool distribution and tool choice

1. Relevant tool exposure
2. Excessive tool availability
3. Tool-selection ambiguity
4. Forced tool use
5. Automatic tool choice
6. Specific tool selection
7. No-tool operation
8. Tool routing
9. Security boundaries

Model Context Protocol fundamentals

1. MCP architecture
2. MCP hosts
3. MCP clients
4. MCP servers
5. Capability negotiation
6. Tools
7. Resources
8. Prompts
9. Transport concepts
10. Local and remote integration
11. Connection lifecycle

MCP integration reliability

1. Server discovery
2. Client configuration
3. Authentication
4. Permission boundaries
5. Schema compatibility
6. Error handling
7. Connection failures
8. Capability validation
9. Logging and observability

Built-in tools

1. Web-search concepts
2. Code-execution concepts
3. File and computer interaction concepts
4. Tool-result handling
5. Tool restrictions
6. Permission considerations
7. Security and operational controls

4. Claude Code Configuration and Workflows

Claude Code operating model

1. Repository awareness
2. Codebase exploration
3. File modification
4. Command execution
5. Planning and implementation
6. Verification workflows
7. Permission boundaries

CLAUDE.md hierarchy and scope

1. User-level instructions

2. Project-level instructions
3. Directory-level instructions
4. Hierarchical loading
5. Instruction precedence
6. Repository conventions
7. Modular organization
8. Relevant-context control
9. Conflicting instruction management

Custom commands and skills

1. Reusable slash commands
2. Command definitions
3. Arguments and variables
4. Repeatable workflows
5. Skill purpose
6. Skill organization
7. Skill discovery
8. Domain-specific instructions
9. Command and skill selection

Path-specific rules

1. Conditional instruction loading
2. File-pattern targeting
3. Directory-specific conventions
4. Language-specific rules
5. Test-specific rules
6. Documentation-specific rules
7. Context-noise reduction
8. Rule-maintenance considerations

Plan mode and direct execution

1. Exploration before modification
2. Planning requirements
3. Risk assessment

4. Multi-file changes
5. Ambiguous requirements
6. Direct low-risk changes
7. Approval boundaries
8. Plan-to-execution transition

Iterative refinement

1. Initial implementation
2. Focused feedback
3. Incremental correction
4. Test-driven refinement
5. Diff review
6. Scope control
7. Regression prevention
8. Completion verification

Hooks and permissions

1. Pre-operation hooks
2. Post-operation hooks
3. Command validation
4. Formatting and testing hooks
5. Allowed operations
6. Denied operations
7. Approval-required operations
8. Least-privilege configuration
9. Secret protection

CI/CD integration

1. Non-interactive execution
2. Repository authentication
3. Environment configuration
4. Pipeline permissions
5. Automated review
6. Test and validation gates

7. Structured pipeline outputs
8. Failure reporting
9. Secret management
10. Safe automation boundaries

5. Prompt Engineering and Structured Output

Effective system prompts

1. Role and objective definition
2. Explicit success criteria
3. Constraints
4. Required behaviour
5. Prohibited behaviour
6. Output requirements
7. Tool-use instructions
8. Ambiguity handling
9. Priority and clarity

Prompt structure

1. Instruction separation
2. Context separation
3. Input-data boundaries
4. XML-style organization
5. Relevant information placement
6. Long-input organization
7. Conflicting instruction avoidance
8. Prompt maintainability

Few-shot prompting

1. Demonstration selection
2. Input-output pairs
3. Representative examples
4. Edge-case coverage
5. Format consistency

6. Example diversity
7. Overfitting risks
8. Token-cost considerations

Structured output

1. JSON-based outputs
2. Schema-constrained responses
3. Tool-based extraction
4. Required fields
5. Data types
6. Enumerations
7. Nested structures
8. Optional and nullable values
9. Schema complexity
10. Downstream compatibility

Validation, retry, and feedback loops

1. Schema validation
2. Semantic validation
3. Business-rule validation
4. Invalid-output detection
5. Corrective feedback
6. Targeted retry prompts
7. Retry limits
8. Escalation conditions
9. Infinite-loop prevention
10. Failure logging

Batch-processing strategies

1. Message Batches API concepts
2. Independent request batching
3. Batch identifiers
4. Asynchronous result handling
5. Partial completion

6. Failed-request handling
7. Result correlation
8. Throughput considerations
9. Cost considerations
10. Batch and real-time selection

Multi-instance and multi-pass review

1. Independent generation
2. Independent review
3. Generator-critic separation
4. Multiple candidate responses
5. Comparison and ranking
6. Consensus methods
7. Verification passes
8. Bias and correlated-error risks
9. Cost and latency trade-offs

6. Context Management and Reliability

Context-window management

1. Context-window limits
2. Token-budget awareness
3. Relevant-context selection
4. Old-message removal
5. Tool-result trimming
6. Structured summarization
7. Critical-fact preservation
8. Lost-in-the-middle concerns
9. Context degradation
10. Context refresh strategies

Prompt caching

1. Reusable prompt prefixes
2. Cacheable stable content

3. Cache boundaries
4. Cache creation
5. Cache reads
6. Cost reduction
7. Latency reduction
8. Content-change effects
9. Cache suitability

Long-running conversations

1. Conversation-state storage
2. Rolling summaries
3. Persistent facts
4. Decisions and commitments
5. User preferences
6. Session continuation
7. Context reconstruction
8. Summary-drift prevention

Escalation and ambiguity resolution

1. Missing information
2. Conflicting requirements
3. Clarifying questions
4. Assumption management
5. Confidence thresholds
6. High-impact uncertainty
7. Safe refusal
8. Human escalation
9. Escalation context

Error propagation in multi-agent systems

1. Incorrect upstream outputs
2. Unverified assumptions
3. Cascading failures
4. Schema enforcement

5. Boundary validation
6. Independent verification
7. Provenance tracking
8. Failure containment
9. Recovery strategies

Codebase exploration and context degradation

1. Targeted repository exploration
2. Relevant-file identification
3. Dependency tracing
4. Search-before-read strategies
5. Progressive context accumulation
6. Stale information
7. Context replacement
8. Focused re-grounding

Human review and confidence calibration

1. Confidence communication
2. Risk-based review
3. High-impact decisions
4. Irreversible actions
5. Approval thresholds
6. Reviewer context
7. Auditability
8. False-confidence prevention

Information provenance and multi-source synthesis

1. Source identification
2. Claim-to-source mapping
3. Source authority
4. Source freshness
5. Conflicting-source detection
6. Uncertainty preservation
7. Citation retention

8. Unsupported-claim prevention
9. Traceable synthesis

7. Examination Consolidation and Readiness

Cross-domain decision patterns

1. Agent versus workflow
2. Single agent versus multiple agents
3. Prompt control versus application control
4. Tool error versus model error
5. Retry versus escalation
6. Context inclusion versus external retrieval
7. Automation versus human approval
8. Parallelism versus dependency ordering

Common examination traps

1. Overengineering simple tasks
2. Granting excessive permissions
3. Sending unnecessary context
4. Relying on prompts for hard guarantees
5. Returning unstructured tool errors
6. Retrying permanent failures
7. Sharing unrestricted state between agents
8. Selecting multi-agent designs without justification
9. Ignoring cost and latency
10. Accepting unvalidated structured output

Scenario-analysis method

1. Requirement identification
2. Constraint identification
3. Risk identification
4. Domain identification
5. Option elimination
6. Best-practice selection

7. Multiple-response completeness
8. Final answer verification

Final readiness assessment

1. Domain-weighted diagnostic review
2. High-weight weakness correction
3. Timed question drills
4. Full-length mock examination
5. Incorrect-answer analysis
6. Terminology review
7. Final knowledge-gap correction
8. Examination pacing
9. Flag-and-review strategy

Disclaimer

This training outline is intended solely as a structured guideline for course planning and examination preparation. The trainer reserves the right to amend, reorganize, reduce, expand, substitute, or omit any topic when reasonably necessary to accommodate participant ability, available instructional time, changes to the examination requirements, updates to the relevant technologies, or other operational considerations. Such changes may be made at the trainer's professional discretion without prior notice.

© 2026 Faysal Aziz · Kuala Lumpur