TYGART MEDIA — BOOK FOR BOTS

Claude Implementation Playbook

The complete technical reference for teams building with Claude — model selection, pricing math, plan decisions, context engineering, and workflow architecture. Optimized for AI attachment.

VERSION	VERIFIED	MODELS COVERED	SOURCE
May 2026	docs.anthropic.com	Opus 4.7 · Sonnet 4.6 · Haiku 4.5	tygartmedia.com

■ HOW TO USE THIS DOCUMENT

Attach this PDF directly to your Claude conversation, Claude Project, or any AI chat before asking about Claude pricing, model selection, plan decisions, or workflow architecture. The AI will use this document as ground-truth reference, reducing hallucination and giving you answers grounded in verified May 2026 data. Every table and specification has been cross-checked against official Anthropic documentation.

CONTENTS

01

Current Claude Models

Verified against Anthropic official documentation, May 2026. These are the only currently active production models. All retired models (Claude 3.x, Sonnet 4, Opus 4, Haiku 3) return errors on API calls.

MODEL	API STRING	INPUT/MTok	OUTPUT/MTok	CONTEXT	BEST FOR
Claude Opus 4.7	`claude-opus-4-7`	$5.00	$25.00	200K / 1M	Agentic coding, complex reasoning, long-horizon tasks
Claude Sonnet 4.6	`claude-sonnet-4-6`	$3.00	$15.00	200K / 1M	Production default — coding, analysis, agents, content
Claude Haiku 4.5	`claude-haiku-4-5-20251001`	$1.00	$5.00	200K	High-volume routing, classification, triage

RETIREMENT TIMELINE

MODEL	STATUS	DATE	ACTION
`Claude Sonnet 4 (claude-sonnet-4-20250514)`	RETIRED	April 20, 2026	Migrate to claude-sonnet-4-6
`Claude Opus 4 (claude-opus-4-20250514)`	RETIRED	April 20, 2026	Migrate to claude-opus-4-7
`Claude Haiku 3 (claude-3-haiku-20240307)`	RETIRED	February 19, 2026	Migrate to claude-haiku-4-5-20251001
`Claude Haiku 3.5`	RETIRED	February 19, 2026	Migrate to claude-haiku-4-5-20251001
`Claude Sonnet 3.7`	RETIRED	October 28, 2025	Migrate to claude-sonnet-4-6

■ OPUS 4.7 TOKENIZER WARNING

Claude Opus 4.7 ships with a new tokenizer that can generate up to 35% MORE tokens for the same input text compared to Opus 4.6. The per-token rate is unchanged ($5/$25), but effective cost per request can increase significantly. Benchmark your specific workload before migrating from Opus 4.6. A prompt that cost $0.10 on 4.6 may cost $0.135 on 4.7 — same rate card, different token count.

02

The Real Cost Story

Headline pricing is only part of the picture. Three mechanisms dramatically reduce effective cost: the Batch API (50% off), prompt caching (90% off cached input), and model routing (using the right model for each task).

REAL-WORLD COST EXAMPLES

WORKLOAD	MODEL	STANDARD	+ BATCH (50% off)	+ CACHE HIT (90% off input)
1M input + 200K output/day	Haiku 4.5	$2.00/day	$1.00/day	~$0.30/day
1M input + 200K output/day	Sonnet 4.6	$6.00/day	$3.00/day	~$0.90/day
1M input + 200K output/day	Opus 4.7	$10.00/day	$5.00/day	~$1.50/day
Content agency — 20M in, 10M out/mo	Haiku 4.5	$70/mo	$35/mo	~$10/mo
Dev team — 10M in, 2M out/mo	Sonnet 4.6	$60/mo	$30/mo	~$9/mo

Cache write costs 1.25x input (5-min cache) or 2x input (1-hour cache). Cache reads cost 0.1x input — 90% off. Cache pays for itself after one read on 5-min caches. Batch API processes asynchronously (results within 24 hours) at 50% off all token types.

PROMPT CACHING — HOW TO IMPLEMENT

```
# Add cache_control to reusable content blocks
messages = [
{
"role": "user",
"content": [
{
"type": "text",
"text": your_large_system_document, # 10K+ tokens
"cache_control": {"type": "ephemeral"} # Cache for 5 min
},
{
"type": "text",
"text": user_question # New input, not cached
}
]
}
]
# First call: pays 1.25x input for cache write
# Every subsequent call: pays 0.1x input for cache read (90% off)
# Break-even: After 1 cache read on 5-min caches
```

BATCH API — WHEN TO USE

Use the Batch API for any workload where you do not need a real-time response. 50% cost reduction with results returned within 24 hours.

```
# Batch API endpoint
POST https://api.anthropic.com/v1/messages/batches

# Each request in the batch follows standard messages format
{
"requests": [
{
"custom_id": "job-001",
"params": {
"model": "claude-sonnet-4-6",
"max_tokens": 1024,
"messages": [{"role": "user", "content": "..."}]
}
}
# Add up to thousands of requests per batch
]
}
# Retrieve results: GET /v1/messages/batches/{batch_id}/results
```

300K OUTPUT TOKENS ON BATCHES (May 2026)

Opus 4.7, Opus 4.6, and Sonnet 4.6 now support up to 300K output tokens on the Message Batches API. Include the beta header: output-300k-2026-03-24

```
headers = {
"x-api-key": YOUR_API_KEY,
"anthropic-version": "2023-06-01",
"anthropic-beta": "output-300k-2026-03-24"
}
```

03

Plan Decision Framework

The right plan depends on your usage pattern — not just your budget. This framework maps workload type to the optimal plan.

PLAN	PRICE	USAGE MULTIPLIER	SESSION WINDOW	BEST FOR
Free	$0	Baseline (1x)	5-hour rolling	Light use, evaluation, students
Pro	$20/mo	~5x Free	5-hour rolling + weekly cap	Individual heavy users
Max 5x	$100/mo	5x Pro	Rolling — no weekly cap	Power users, daily heavy use
Max 20x	$200/mo	20x Pro	Rolling — no weekly cap	Developers, sustained agentic sessions
Team Standard	$25/seat/mo	1.25x Pro	5-hour + weekly cap	Teams, collaboration, SSO
Team Premium	Higher	6.25x Pro per session	5-hour + weekly cap	Power users on team plans
Enterprise	Custom	Consumption-based	No hard limits	Large orgs, compliance, custom contracts

SUBSCRIPTION vs API — THE DECISION RULE

SITUATION	RECOMMENDATION	REASON
Individual daily use, chat-focused	**Claude.ai Pro ($20/mo)**	Predictable cost, no token math, priority access
Team collaboration + SSO needed	**Team Standard ($25/seat)**	Per-member limits, admin tools, connectors
Developers hitting limits daily	**Max 20x ($200/mo) or API**	No weekly cap, rolling windows only
Automation, pipelines, product integration	**Anthropic API (pay-per-token)**	No message caps, exact cost control, batch discounts
High volume, cost-sensitive, non-real-time	**API + Batch API + Haiku 4.5**	50% discount, $1/$5 per MTok, no caps
Chat AND API in same workflow	**Subscription + API separately**	No credit sharing between claude.ai and API — billed separately

CRITICAL: CLAUDE.AI AND API ARE SEPARATE BILLING SYSTEMS

A Claude.ai Pro subscription ($20/month) gives you zero API credits. The API is billed separately through platform.anthropic.com with your own API key. New API accounts receive $5 in free credits. After that, all API usage is pay-per-token regardless of your subscription plan. This surprises many teams.

04

Model Selection Logic

The default-to-flagship instinct is breaking down. As of May 2026, Sonnet 4.6 delivers ~98% of Opus 4.6 performance at 40% lower cost and 17% faster output. The 1.2-point SWE-bench gap between Sonnet 4.6 (79.6%) and Opus 4.6 (80.8%) is the smallest in Claude history.

TASK TYPE	RECOMMENDED MODEL	REASON
Classification, routing, triage, moderation	**Haiku 4.5**	5x cheaper than Sonnet, ~97 tok/s, quality sufficient for categorical tasks
Most production workloads — coding, writing, analysis, RAG	**Sonnet 4.6**	Default for 80%+ of tasks. Near-Opus quality, 40% cheaper, faster
Complex multi-step reasoning, architecture decisions	**Opus 4.6**	GPQA Diamond: 91.3% vs Sonnet's 74.1%. Worth the premium here
Autonomous coding agents, long-horizon agentic tasks	**Opus 4.7**	Step-change improvement in agentic coding over 4.6. Note tokenizer change
1M token context (large codebase, massive documents)	**Sonnet 4.6 or Opus 4.6**	Both support 1M context at flat rates, no surcharge. Haiku capped at 200K
Cost-sensitive, high-volume, async processing	**Haiku 4.5 + Batch API**	$0.50/$2.50 per MTok with batch. Cheapest quality option available

Optimal split for most production teams: 70% Haiku / 20% Sonnet / 10% Opus. This pattern vs. all-Sonnet cuts total API cost by 50%+ on typical workloads.

05

Context Engineering

In 2026, how you structure context matters more than how you phrase prompts. The models are capable enough that plain-English instructions work fine — the bottleneck moved from "can the model understand me?" to "does the model have the right information to do what I want?"

THE LOST-IN-THE-MIDDLE PROBLEM

Research shows a U-shaped performance curve: accuracy is highest when critical information appears at the beginning OR end of context, with 30%+ accuracy drop for information buried in the middle. Put your critical instructions FIRST and LAST. Never the middle.

CONTEXT WINDOW BUDGET GUIDELINES

LAYER	WHAT GOES HERE	PLACEMENT	CACHE IT?
System Prompt	Role, constraints, output format, uncertainty handling	FIRST — always	**YES — if stable**
Reference Documents	PDFs, docs, codebases, knowledge bases	After system prompt	**YES — high ROI**
Few-Shot Examples	2-5 input/output examples in tags	Before the task	**YES — if reused**
User Input	The specific task for this turn	LAST — always	No

WHAT KILLS PERFORMANCE (AVOID THESE)

✗ ALL-CAPS emphasis ("YOU MUST", "CRITICAL!", "NEVER EVER") — Overtriggers Claude, produces worse results. Just say what you want calmly.

✗ Reasoning degradation past 3,000 tokens — LLM reasoning starts degrading around 3,000 tokens. Sweet spot: 150-300 words for instructions.

✗ Information buried in the middle — 30%+ accuracy drop for content in the middle of long contexts. Critical info goes first or last.

✗ Chain-of-thought instructions on current models — Few-shot CoT no longer improves reasoning on Claude 4.x. Only use for format alignment.

✗ Mixing context, constraints, and format in one block — Use 4-block structure: INSTRUCTIONS / CONTEXT / TASK / OUTPUT FORMAT. Separation is critical.

06

System Prompt Design

The 4-block contract format is the most reliable structure for Claude system prompts. XML tags outperform Markdown and numbered lists for Claude specifically.

THE 4-BLOCK CONTRACT FORMAT

```
You are a [ROLE]. Your job is to [PRIMARY FUNCTION].
Success criteria:
- [Criterion 1]
- [Criterion 2]
Constraints:
- [Hard limit 1]
- [Hard limit 2]
Uncertainty rule: If you are uncertain, say so explicitly. Do not guess.


[Background information, documents, reference data.
This is where you put cacheable content.]


Input: [example input]
Output: [example output]


[The specific request for this interaction — this changes every turn]


[Exact structure expected. Be specific: "5 bullets, each under 15 words"
not "be concise". Include a schema if format precision matters.]
```

SYSTEM PROMPT ANTI-PATTERNS

```
# BAD — Unstructured, mixed concerns
system_prompt = """
You are an expert assistant. Please be helpful and accurate.
When answering, consider the user context and provide detailed
responses that address their needs while being concise.
Format: Use bullets when appropriate.
"""

# GOOD — 4-block contract, clear separation
system_prompt = """
You are a technical support specialist for Claude API integration.
Success: Accurate technical answers with working code examples.
Constraints: Only answer questions about Claude API and Anthropic products.
Uncertainty rule: If unsure, say "I don't have verified information on this."
Output format: Code in triple backticks with language tag. Max 3 steps per answer.

[Attach your product docs, API reference, or knowledge base here.
Add cache_control to this block for 90% cost reduction on repeated calls.]
"""
```

07

Workflow Architecture

Multi-model routing is the highest-leverage architectural decision for cost optimization. A 70/20/10 Haiku/Sonnet/Opus split cuts total API cost by 50%+ vs all-Sonnet.

MULTI-MODEL ROUTING PATTERN

```
def route_to_model(task_type: str, complexity: int) -> str:
"""
Route tasks to the cheapest model that meets quality bar.
complexity: 1-10 scale
"""
# High-volume, low-complexity tasks → Haiku ($1/$5 MTok)
if task_type in ["classify", "route", "triage", "moderate", "extract"]:
return "claude-haiku-4-5-20251001"

# Most production workloads → Sonnet ($3/$15 MTok)
if complexity <= 7 or task_type in ["write", "code", "analyze", "rag"]:
return "claude-sonnet-4-6"

# Complex reasoning, architecture, deep research → Opus
if complexity >= 8 or task_type in ["architect", "research", "agent"]:
return "claude-opus-4-7" # Use for autonomous coding agents
# return "claude-opus-4-6" # Use for reasoning tasks

# Example routing
model = route_to_model("classify", complexity=2) # → Haiku
model = route_to_model("write", complexity=5) # → Sonnet
model = route_to_model("architect", complexity=9) # → Opus 4.7
```

CONTEXT COMPACTION FOR LONG SESSIONS

At 60% context fill, trigger compaction before context rot degrades output quality. Compact proactively, not reactively.

```
# Compaction trigger — run at 60% context utilization
def should_compact(messages: list, context_limit: int = 200_000) -> bool:
estimated_tokens = sum(len(m["content"]) // 4 for m in messages)
return estimated_tokens > (context_limit * 0.6)

# Compaction prompt — preserve decisions, discard redundant outputs
COMPACTION_PROMPT = """
Summarize this conversation. Preserve:
- All architectural decisions made
- Unresolved bugs and their symptoms
- Implementation details that affect future steps
- File paths and function names referenced

Discard:
- Redundant tool outputs already acted upon
- Intermediate reasoning that led to abandoned approaches
- Duplicate information

Output a dense summary the agent can use to resume work cleanly.
"""
```

08

Claude Code

Claude Code is Anthropic's terminal-based agentic coding tool. It uses a 4-layer memory architecture. Most teams use 10% of its capability.

THE 4-LAYER MEMORY ARCHITECTURE

LAYER	FILE	PURPOSE	SIZE LIMIT
1 — Explicit (CLAUDE.md)	`CLAUDE.md`	Build commands, conventions, architecture rules — things Claude gets wrong without them	**< 300 lines**
2 — Auto Memory (MEMORY.md)	`MEMORY.md`	Learned project patterns written back autonomously. Loads first 200 lines or 25KB	200 lines / 25KB
3 — Memory Tool (API agents)	`API layer`	On-demand retrieval for programmatic agents. Store and pull relevant context at runtime	Unlimited
4 — Subagent Memory	`Per-agent store`	Persistent knowledge scoped per named subagent. Introduced Feb 2026 (v2.1.33)	Per agent

EFFECTIVE CLAUDE.md PATTERN

```
# Good CLAUDE.md — dense, behavioral, no fluff
## Stack
- Next.js 15 App Router, TypeScript strict, Tailwind 4
- DB: Postgres via Drizzle ORM (schema in src/db/schema.ts)
- Auth: Better Auth (src/lib/auth.ts)
- Testing: Vitest + Playwright

## Rules
- Run `npm run typecheck` before marking any task complete
- Never modify files in src/db/migrations/ directly
- Use server actions for mutations, not API routes
- All components use named exports (no default exports)
- No comments unless the WHY is non-obvious

## Architecture Decisions
- API routes only for webhook endpoints and third-party callbacks
- All forms use React Hook Form + Zod schema validation
- Images served from /public, never CDN-linked directly

# Rule of thumb: Every line should change Claude's behavior.
# If a line wouldn't change output, remove it — it wastes tokens.
```

CLAUDE CODE USAGE LIMITS (May 2026)

Claude Code usage counts against the same pool as Claude.ai usage. All surfaces (claude.ai, Claude Code, Claude Desktop) draw from one shared limit. A heavy Claude Code morning session competes with your afternoon chat sessions.

1M CONTEXT WINDOW IN CLAUDE CODE

Since March 2026, Max, Team (Standard and Premium), and Enterprise users on Opus 4.6 automatically get a 1M token context window in Claude Code with no additional configuration and no long-context pricing surcharge. A 900K-token session costs the same per-token rate as a 9K-token session. Pro users can access it via extra usage.

09

Quick Reference

Everything at a glance. Verified May 2026.

API STRINGS — COPY THESE EXACTLY

ROLE	API STRING	STATUS
Flagship (agentic coding)	`claude-opus-4-7`	✓ ACTIVE
Production default	`claude-sonnet-4-6`	✓ ACTIVE
Fast / high-volume	`claude-haiku-4-5-20251001`	✓ ACTIVE
DO NOT USE	`claude-sonnet-4-20250514`	✗ RETIRED
DO NOT USE	`claude-opus-4-20250514`	✗ RETIRED
DO NOT USE	`claude-3-haiku-20240307`	✗ RETIRED

BETA HEADERS — ACTIVE MAY 2026

```
# 300K output tokens on Batches API (Opus 4.7, Opus 4.6, Sonnet 4.6)
"anthropic-beta": "output-300k-2026-03-24"

# Claude Managed Agents (public beta)
"anthropic-beta": "managed-agents-2026-04-01"

# US-only inference (Opus 4.7, Opus 4.6+) — adds 1.1x pricing multiplier
inference_geo: "us" # Default is "global" — standard pricing

# Fast mode (Opus 4.6 only — research preview, 6x pricing)
"anthropic-beta": "fast-mode-2026-02-01"
```

KEY NUMBERS TO REMEMBER

$5 / $25 — Opus 4.7 and Opus 4.6 per MTok input/output

$3 / $15 — Sonnet 4.6 per MTok input/output

$1 / $5 — Haiku 4.5 per MTok input/output

50% — Batch API discount on all token types

90% — Prompt cache hit discount on input tokens

35% — Max token inflation from Opus 4.7 new tokenizer vs 4.6

200K / 1M — Context windows — Haiku capped at 200K, Opus/Sonnet 4.6+ support 1M

5 hours — Rolling session window for all subscription plans

6.25x — Team Premium seat usage multiplier over Pro per session

3,000 tokens — Approximate point where system prompt reasoning starts degrading

150-300 words — Sweet spot for system prompt instruction length

---

This document was researched and compiled from official Anthropic documentation (docs.anthropic.com, platform.claude.com) and verified sources in May 2026. Attach it to any Claude conversation before asking about implementation details. For the latest pricing and model updates, visit tygartmedia.com/claude-reference-hub/

Tygart Media — tygartmedia.com | Books for Bots Series | Free with email registration