

Memory for Agentic AI

Matthew Schwartz

August 2025

TL;DR

As AI agents become more prevalent in enterprise applications, understanding how they manage information is crucial for building reliable, cost effective solutions. Many organizations struggle with AI systems that "forget" context mid-conversations, repeatedly ask for the same information, or consume excessive tokens by reprocessing data. The key is distinguishing between three fundamental concepts: context (what the AI sees right now), memory (what it remembers across sessions), and knowledge (what it can retrieve on demand). Getting this architecture right can significantly reduce operational costs while improving user experience and system reliability.

- **Context ≠ Memory ≠ Knowledge.**
 - **Context** = what the model sees **right now** (prompt + retrieved chunks + tool traces). Finite → can **overflow**. ([Amazon Web Services, Inc.](#))
 - **Memory** = state the system **stores across turns/sessions** and reloads later (e.g., **Agents for Bedrock** session summaries; **AgentCore Memory** short-term event log **and** long-term extracted facts/preferences; **Claude Code** `CLAUDE.md`). ([AWS Documentation](#), [Anthropic](#))
 - **Knowledge / RAG** = **indexed data** you **retrieve on demand** (e.g., **Bedrock Knowledge Bases / RetrieveAndGenerate**; **Q CLI** experimental `/knowledge`). ([AWS Documentation](#), [DEV Community](#))
- **Amazon Q CLI `/knowledge` (beta)** builds **persistent semantic indexes** you can *show/add/update/remove/clear*, check **status**, and reuse across chats so you fight "context rot" without blowing your token budget. Enable with `q settings app.beta true` and `q settings chat.enableKnowledge true`. ([DEV Community](#))
- **Agents for Amazon Bedrock memory:** enable summaries, set retention **1–365 days**, and use a stable **memoryId** per user. ([AWS Documentation](#))
- **AgentCore Memory:** **short-term** raw events + **long-term** extracted memory via **strategies**; **namespaces** for scoping; optional **KMS** encryption. ([AWS Documentation](#))

1) Core mental model

Term	What it is	Who controls it	Cost
Context window	Tokens the model can “see” this turn	You (prompting/retrieval/truncation)	Per-call tokens; can overflow. (Amazon Web Services, Inc.)
Short-term memory	History of the active session (or raw events)	SDK/service	Storage + retrieval tokens. (AWS Documentation)
Long-term memory	Summaries, facts, preferences extracted from events	Service (configured by you)	Storage + retrieval tokens. (AWS Documentation)
Knowledge /RAG	External indexed data retrieved on demand	You/platform	Indexing + retrieval + context tokens. (AWS Documentation)

2) Context windows & overflow

Context is finite. When prompts + retrieved chunks + traces exceed the limit, older tokens are truncated—**Context Window Overflow (CWO)**—which can drop safety instructions or cause “memory loss.” Mitigate with length guards, prioritized/pinned instructions, and retrieval limits instead of pasting large docs. ([Amazon Web Services, Inc.](#))

3) RAG & Knowledge

3.1 RAG in one minute

Ingest & **index** → **retrieve** top-k (semantic/hybrid; rerank optional) → **augment** the prompt with **just** those chunks → **generate** (with **citations**). Use RAG to scale knowledge access without inflating prompts. ([AWS Documentation](#))

3.2 Bedrock Knowledge Bases

Managed RAG: ingestion + vector store + **Retrieve/RetrieveAndGenerate** APIs, **citations**, optional **reranking**, and agent integration. ([AWS Documentation](#))

3.3 Amazon Q CLI: built-in `/knowledge` (experimental)

Enable:

```
q settings app.beta true
q settings chat.enableKnowledge true
```

Core subcommands: `show` | `add` | `remove` | `update` | `clear` | `status` | `cancel`. In prompts: *"Use the knowledge tool ..."*. (Experimental; persists across chats.) ([DEV Community](#))

3.4 Claude Code: knowledge via MCP + `CLAUDE.md`

- **CLAUDE.md** memory files (Enterprise → Project → User precedence), auto-loaded; supports `@imports`; edit with `/memory` or `#` shortcut. ([Anthropic](#))
- **MCP** connects Claude to external tools/data sources (including Bedrock KBs via MCP servers). ([Anthropic](#))

3.5 Bedrock **AgentCore Memory** (for agents you build)

- **Short-term memory** = **raw interaction events** per session (`CreateEvent`, `ListSessions`, `ListEvents`) used to reload exact history. ([AWS Documentation](#))
- **Long-term memory** = **structured records** extracted **asynchronously** from short-term events via **memory strategies** (built-in or custom) for summaries, semantic facts, and user preferences; retrieve via `RetrieveMemoryRecords`. ([AWS Documentation](#))
- **Namespaces** scope long-term memories (multi-tenant/actor/session). ([AWS Documentation](#))
- **KMS encryption** available when creating memory (supply `encryptionKeyArn`). ([AWS Documentation](#))

4) Amazon Q CLI quick facts

- **Context files:** `/context add|show|rm|clear` (profile/global). Good for small, high-signal files you truly need in-prompt. ([AWS Documentation](#))
- **Project rules:** Markdown in `.amazonq/rules/` to enforce standards; add to context. ([AWS Documentation](#), [Amazon Web Services, Inc.](#))
- **Editor:** `/editor` opens a temp `.md` in `$EDITOR` for long prompts. ([AWS Documentation](#))
- **Context hooks:** `/context hooks add --trigger {per_prompt|conversation_start} --command "<cmd>"` to auto-inject command output (e.g., `git status`). ([AWS Documentation](#))
- **Knowledge:** see §3.3. ([DEV Community](#))

5) Claude Code quick facts

- **Memory locations** (auto-loaded; higher-level precedes lower): Enterprise → Project (`./CLAUDE.md`) → User (`~/claude/CLAUDE.md`). Imports via `@path`. Manage with `/memory`; add ad-hoc notes with `#`. ([Anthropic](#))

6) Bedrock options at a glance

- **Agents for Bedrock memory**: enable session summarization; set retention **1–365 days**; use a stable `memoryId` to load summaries next session. ([AWS Documentation](#))
- **Knowledge Bases**: managed RAG with citations and optional reranking; call via `RetrieveAndGenerate`. ([AWS Documentation](#))
- **AgentCore Memory**: programmable **short-term + long-term** memory with strategies, namespaces, and optional KMS. ([AWS Documentation](#))

7) Decision guide (fast)

1. **Always-on instructions?** Put them in **rules** (`.amazonq/rules`) or `CLAUDE.md`, not ad-hoc prompts. ([AWS Documentation](#), [Amazon Web Services, Inc.](#))
2. **Reusable reference material?** Index as **knowledge** (Q CLI `/knowledge`; Bedrock KB). ([DEV Community](#), [AWS Documentation](#))
3. **Cross-session continuity?** Use **Agents memory**; for multi-tenant/productized agents, prefer **AgentCore Memory**. ([AWS Documentation](#))
4. **One-off need this turn?** Keep it in **context** and trim aggressively to avoid CWO. ([AWS Documentation](#), [Amazon Web Services, Inc.](#))

8) Pros/cons & use-cases

Option	Great for	Pros	Watch-outs
Context files	Small specs/logs for current task	Simple; explicit	Token cost each turn; overflow risk. (AWS Documentation)
Project rules	Team guardrails	Versioned; always-on	Keep concise; not a KB. (Amazon Web Services, Inc.)
<code>/editor</code>	Structured long prompts	Good authoring UX	Still consumes tokens. (AWS Documentation)
Q CLI <code>/knowledge</code> (beta)	Local, persistent corpora	Reuse across chats; <code>status</code>	Beta; manual updates; local scope. (DEV Community)
Claude <code>CLAUDE.md</code>	Persistent dev prefs/policies	Hierarchical; imports; <code>/memory</code>	Keep brief; not a search engine. (Anthropic)

Option	Great for	Pros	Watch-outs
Agents memory	Conversational assistants	Retention 1–365d; memoryId	Summaries ≠ full history. (AWS Documentation)
Bedrock KB (RAG)	Evidence-backed answers	Citations; rerank	Requires indexing/IAM hygiene. (AWS Documentation)
AgentCore Memory	Productized, governed agents	Short-term + long-term; namespaces; KMS	Requires strategy design. (AWS Documentation)

9) Guardrails & governance

- **Prevent CWO:** cap prompt size, pin safety/system instructions, and limit retrieved chunks. ([Amazon Web Services, Inc.](#))
- **Scope tools/data:** restrict MCP servers & Q CLI tools to least privilege. ([Anthropic](#))
- **Encrypt where needed:** AgentCore Memory supports **KMS** for memory resources. ([AWS Documentation](#))

10) Copy-paste recipes

Q CLI — knowledge & context

```
# Enable beta knowledge
q settings app.beta true
q settings chat.enableKnowledge true
q chat
> /knowledge add ./docs
> /knowledge status
> /knowledge show
# Nudge retrieval:
Use the knowledge tool and answer with citations from my indexed docs.
```

([DEV Community](#))

```
# Context & hooks
q chat
> /context add .amazonq/rules/*.md docs/architecture.md
> /context hooks add git-status --trigger per_prompt --command "git status --
short"
```

([AWS Documentation](#))

Claude Code — project memory

```
# ./CLAUDE.md
- Use 2-space indentation; prefer least-privilege IAM.
See @README and @docs/architecture.md.
```

(Manage with `/memory`; add quick notes by starting a message with `#`.) ([Anthropic](#))

Bedrock — RAG call Use `RetrieveAndGenerate` for KB-backed answers with citations. ([AWS Documentation](#))

References

- **CWO:** *AWS Security Blog* — “Context window overflow: Breaking the barrier.” ([Amazon Web Services, Inc.](#))
- **Q CLI knowledge (beta):** AWS DevRel walkthrough with commands & flags. ([DEV Community](#))
- **Agents memory:** User guide (enable, retention, `memoryId`). ([AWS Documentation](#))
- **Bedrock KB & RAG:** User guide and API (`RetrieveAndGenerate`). ([AWS Documentation](#))
- **AgentCore Memory:** Short-term events, long-term strategies, namespaces, KMS. ([AWS Documentation](#))
- **Claude Code memory:** Locations, precedence, imports, `/memory`. ([Anthropic](#))