International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Infrastructure Deployment Patterns for AI Services on the Claude Platform

Yevhen Mykhailenko[*]

Software Engineer, PayPal (by Accelon Inc.), Austin, Texas, USA
Email: yvn.mykh@gmail.com

Abstract

This article questions how to select and use infrastructure patterns in the deployment of industrial AI services on Claude due to the increased enterprise demand and broadened capabilities. This study proves relevant since, within a very short span, generative AI work has transformed from novelty experiments to mission-critical business workloads where effectiveness directly depends on architectural decisions ensuring not only low latency but also cost predictability together with the compliance of corporate security requirements. Therefore, the purpose of this paper is to identify recurring architectural patterns and their operational mechanisms that may be used in balancing response time, context size, cost, and reliability when integrating Claude into enterprise IT systems. The scientific novelty of the research is a detailed taxonomy of infrastructural solutions (serverless gateways, containers in isolated VPCs, batch queues) and how their combination begins to meet the very special challenges of Claude—challenges such as context windows up to one million tokens and loads that seem to increase sharply above 25 billion requests per month. The main findings show that the best setup comes from using small APIs with autoscaling and local cache for interactive work; containers with private links and Batch interfaces when dealing with deep analytics and big processing. The quota system and context thinning, breaking up queues by priority and budget, as well as token control and telemetry, are pieces to help give teams a p95 latency promise while staying inside financial and regulatory fences. The article will be helpful to researchers in cloud architectures, enterprise IT system architects, and practitioners deploying AI services based on Claude models.

***Keywords*:** Claude; Anthropic; infrastructure patterns; AI services; enterprise platforms; containerization; serverless computing; token quotas; scalability; latency.

Received: 9/19/2025
Accepted: 11/19/2025
Published: 11/29/2025

----

* *Corresponding author.*

1. Introduction

Over three years, generative AI has evolved from pilot experiments to mission-critical workloads: in a Gartner report, agentic AI is listed among the key strategic technologies for 2025, and the effectiveness of such projects directly depends on infrastructure decisions around latency, cost, and data protection [1]. Against this backdrop, Claude has become a leader in the enterprise market: by July 2025, Anthropic's models were used in 32% of all LLM requests within companies, while OpenAI's share fell to 25%; two years earlier, the shares were the reverse, indicating a shift in preferences among developers and CIOs [2].

The technical leap has been supported by expanding the Sonnet 4 context window to one million tokens, enabling developers to send many pages of text or a whole code repository to the model without additional orchestration and thereby simplifying large analytical tasks and refactoring scripts [3]. Statistics confirm the scale of usage: in June 2025, Claude served more than 25 billion API requests per month, with 45% of them coming from internal enterprise platforms; that is, integration is proceeding not only through chat interfaces but also deep into ETL pipelines and developer tools [4]. At the same time, the licensing cost remains moderate: Anthropic's Team plans are priced at 25–30 USD per user per month; however, with unlimited token access, total expenses proliferate, which necessitates thoughtful quota mechanisms and context optimization [5].

The combination of these factors—rapid load growth, gigantic context windows, and a rising enterprise market share—makes it necessary to systematically study infrastructure patterns that balance the model's depth of reasoning, throughput, and budget under stringent security and observability requirements.

2. Materials and Methodology

It is a study in infrastructure patterns when AI services are deployed on Claude, drawn from sixteen sources of scholarly works, industry reports, technical documentation by both AWS and Anthropic, and independent performance benchmarking. The main purpose was to fish out repeating architectural solutions that would balance response speed versus throughput, and corporate security requirements. The theoretical base is Gartner's assumptions on strategic technologies for 2025, in which agentic AI is articulated as a principal trend [1], and the comparative review of the enterprise LLM market shows a shift in preference favoring Anthropic and an increase in Claude's share to 32% [2]. Technical innovations—mainly the Sonnet 4 context window extension to one million tokens [3]—were regarded as factors enabling new infrastructure patterns. Moreover, statistical data from SQ Magazine about Claude's load (over 25 billion requests per month) [4] and economic assessments of licensing and token costs were considered.

The study is based on a mix of methods. First, it applies comparative analysis of sources: one set being AWS documentation on API Gateway and Lambda [8, 9], Bedrock AgentCore [11] and PrivateLink [12], the quota guides for Bedrock and Anthropic [13, 14] — from which this basic technical layer was reconstructed. Secondly, it takes a systematic review of performance benchmarks, including first-token and per-token latency metrics [15], that clearly illustrate how the choice of pattern (gateway, container, batch queue) impacts both user experience and scalability. Third, content analysis of engineering case studies (Menlo Ventures [6], IT Pro [7], AWS [16]) helped identify practices for budget control, queue segmentation, and compliance with security requirements.

3. Results and Discussion

The growth of enterprise Claude traffic sets clear business priorities: today, the most in-demand scenarios are interactive developer assistance via Claude Code, processing of long corporate documents, generation of legal opinions, and embedded help in BI dashboards. These use cases make up 32% of all calls to LLM-APIs inside firms, beating OpenAI for the first time, whose share has dropped to 25% as seen in Figure 1 [6].

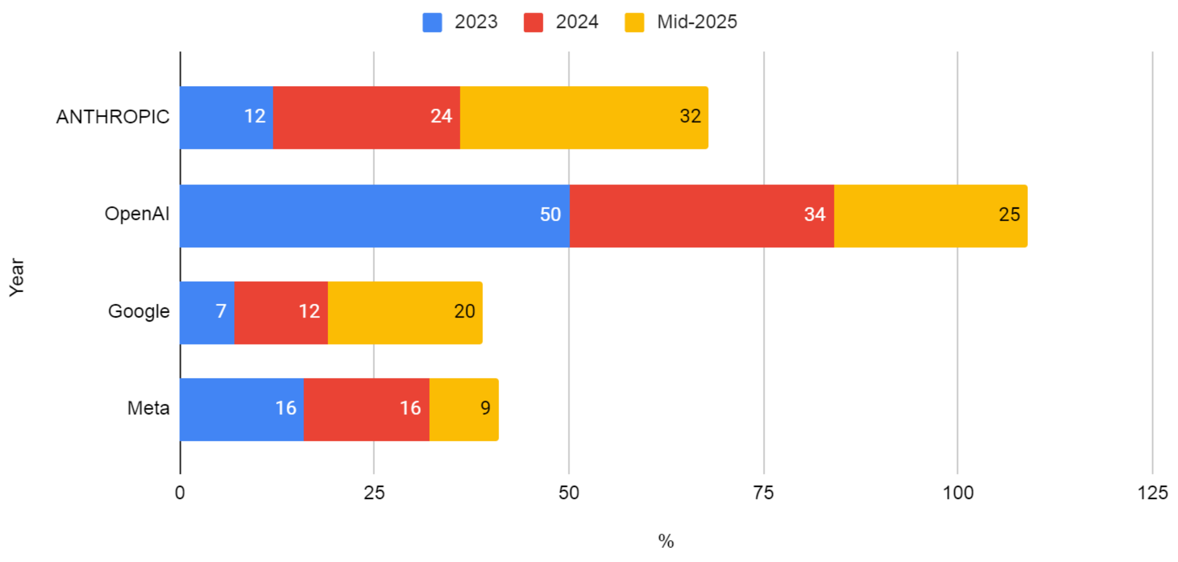

Figure 1: Enterprise LLM API Market Share Dynamics [6]

The total count of queries to Claude exceeds 25 billion per month, with 45% of calls coming from the inside rather than end-user chat interfaces, which shows deep integration of the model into ETL chains and service logic [4]. Front-office tasks require sub-second response time. An IT Pro survey notes that latencies above one second begin to impact user satisfaction and conversion [7]. On the server side, this translates into a requirement to sustain at least 10,000 HTTP requests per second at the API Gateway layer—the default per-account quota set by the provider [8]. The Lambda layer absorbs further load growth: the baseline limit of 1000 concurrent instances per function can, upon request, be increased to "tens of thousands," and the scaling mechanism adds another 1000 environments every 10 seconds during peaks, smoothing phase spikes in traffic [9].

Anthropic puts money limits on the license model: an Enterprise Pro-class business plan is about 33-40 USD per user each month, while the main cost item is tokens, not the licenses themselves; so, the setup must have extra calls and too much context use [10]. On one end, it sits with a need from rules teams for request-answer logging and coding of data in travel; thus, making managed doors in front of the outside API a must.

API Gateway and the serverless gateway function make up the most minimal technical layer. The Gateway takes in the JSON, does token checking, normalizes headers, and then sends on the payload to Lambda. For multimodal streams, rather than making a direct call, teams utilize the batch mode of Amazon Bedrock AgentCore, such that within one request, up to 100 MB of binary data can be included, and a streamed response can be received back, which makes sending large images or archives easier [11].

The last part is auto-concurrency. As the incoming RPS increases, API Gateway keeps buffering more requests, and Lambda keeps scaling environments until it hits the set roof or the requested limit. This match manages top lag and stretches the service to thousands of chats at once without any manual assistance, all while keeping a tight hold on costs and staying in line with internal safety rules.The container tier runs within a private VPC, from which the path to Claude goes over AWS PrivateLink; once the interface VPC endpoint is set up, Bedrock can be reached using inside DNS names, so the machines do not need public IPs or net paths, making it easier to check boxes for company safety rules [12]. A container is often started in EKS or ECS Fargate along with a sidecar that does three-step request checks, signs it with SigV4, and sends it to the bedrock-runtime endpoint while keeping logs in one place in a CloudWatch log stream for later review.Two quotas restrict container execution. Most models start at the Bedrock layer with about ten transactions per second per account cap. Sustained overage results in getting a 429 errors. Signed requests are used to increase this limit [13]. Simultaneously, Claude models under the standard tier permit 50 requests per minute and 30,000 input tokens per minute for Sonnet 4 which defines the minimum size of the pool for autoscaling as seen in Table 1 [14].

Table 1: Throughput Limits of Claude Model Variants [14]

Model	**Maximum requests per minute (RPM)**	**Maximum input tokens per minute (ITPM)**	**Maximum output tokens per minute (OTPM)**
Claude Opus 4.x*	50	30000	8000
Claude Sonnet 4	50	30000	8000
Claude Sonnet 3.7	50	20000	8000
Claude Sonnet 3.5 (2024-10-22, deprecated)	50	40000	8000
Claude Sonnet 3.5 (2024-06-20, deprecated)	50	40000	8000
Claude Haiku 3.5	50	50000	10000
Claude Opus 3 (deprecated)	50	20000	4000
Claude Sonnet 3	50	40000	8000
Claude Haiku 3	50	50000	10000

As a result, horizontal scaling of containers is oriented not only to CPU but also to the API's maximum throughput, to avoid wasteful resource burn.

Transition between near-instant and extended reasoning modes is implemented dynamically: a client header selects the fast or deep logic, but both variants are served by the same model, which eliminates URL switching.

SQ Magazine says the average latency of a regular Claude call is down to 190 ms, good enough to support live dashboards and IDE hints [4]. For big jobs, long thinking gives detailed chains of logic, but AIMultiple shows a 1.16s first-token wait, so it runs only when quality goes up a lot [15]. A mixed router looks at prompt size and how urgent the deadline is, then sends the request either to the near-instant line or to the long line in that same group. First-token wait time and per-token speed by LLMs are shown in Figure 2.

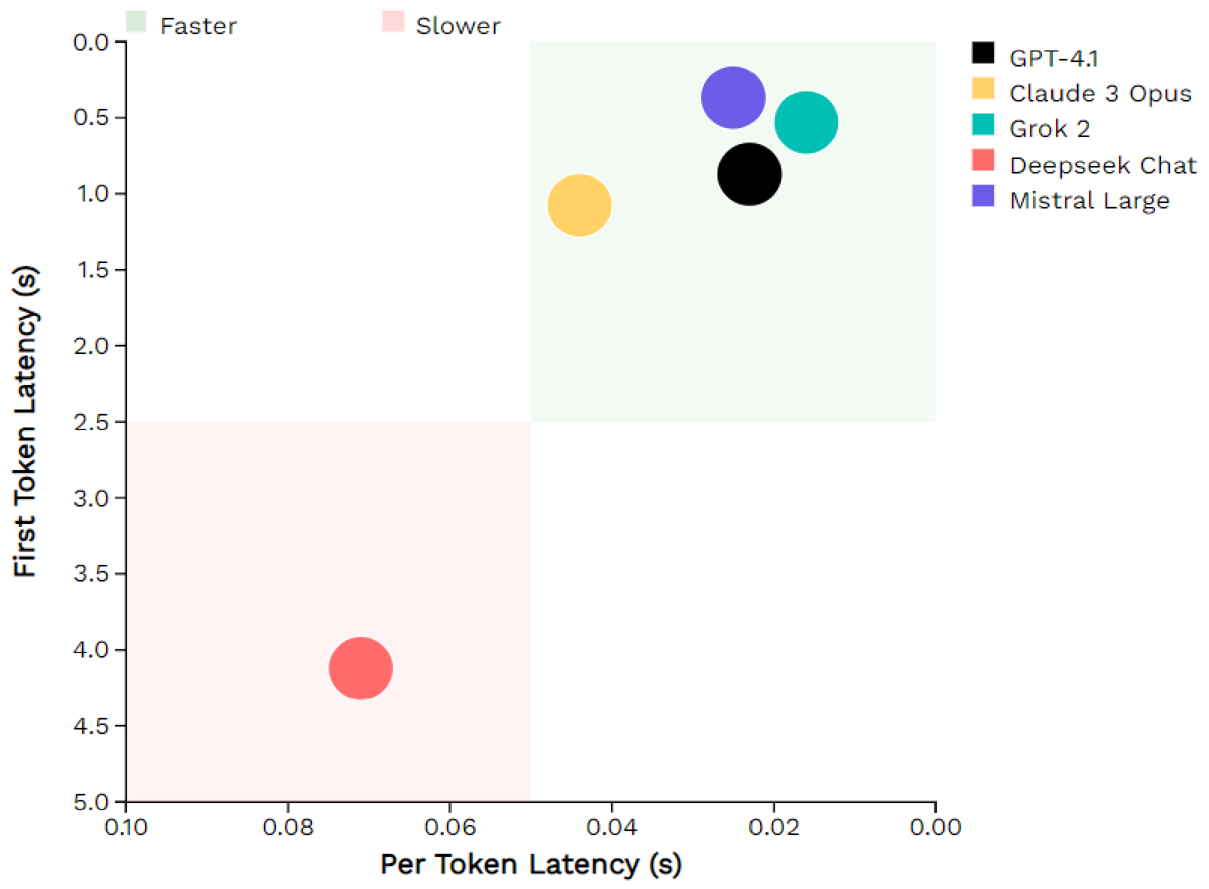

Figure 2: LLM Latency Comparison: First-Token vs Per-Token Metrics [15]

In order not to mix loads with different SLOs, they run two logical services on the same orchestration platform: micro-API, which responds instantly, and Analytical worker, used for prolonged computations. One runs under an aggressive autoscaling rule by the p95-latency metric, while the other runs by the depth of its internal queue plus token cost, still keeping the budget even if extended reasoning runs with a million-token context. This architectural change helps developers avoid manual tuning while still keeping all approaches as consistent as previously described in the chapter about function gateway, but moving quota control and reasoning-mode balance into a private-container perimeter.

When work moves from interactive to mass processing, a batch queue is what gets kicked off. Rather than gateway or container calls one by one, a set of prompts gets collected into an archive up to 256 megabytes and sent over a Batch interface. Inside, the work gets split among workers so that all long requests in the same model pass can be executed together, which drastically cuts down aggregate overhead, and on the main API that will now be freed up for latency-critical flows [16].

When done, the queue pushes an event to kick off export results to object storage. There will be no need for any cron polling, hence making the downstream pipeline simple: as soon as a new file shows up, extract the answers and pass them further into the ETL or BI contour. Because all of this happens in a separate cloud, use the same safety and check rules as in the small-service part, and add Bedrock private endpoints to make sure that data does not ever leave the corporate area.

Large offline runs never load evenly, so they load by topic, by context size, or by priority. The queue manager sets a budget per segment by calculating an upper bound on token consumption and runtime. If limits are likely to be exceeded, it stops that queue. It also reports to the observability system so that the team can change model parameters and recalculate sample size - thereby ensuring that there will not be any sudden cost spike since discipline imposes predictability in the budget. It gives elasticity in depth of reasoning against project economics, complementing principles established in gateway and microservice layers.

To keep the conversation going, it stores the most recent user turns in a speedy Redis cache so that every new request gets fresh context without having to access slower stores. This local layer reduces delay time, takes extra strain off private endpoints, and matches the same separation rules used in the VPC container edge and batch line for offline analytics.

A condensation module comes in. It generates an abridged digest of those earlier messages, preserving all the facts and intention, and then replaces them with long shards of text from the original. Thus, token volume is lowered with logical integrity preserved so that answers going forward continue to make sense.

A particular manager constantly monitors the total context size and, as it approaches the maximum allowed, initiates additional compression as well as even more dynamic pruning of unimportant details. This auto cycle does not need developer meddling and stays in line with the whole budgeting rule: token amounts remain within the set range, service sticks to limits, and at the same time gives back the teamwork promptness that users ask for.

At the external gateway, each request undergoes detailed measurement of input and output tokens as well as the interval between calls from a single client identifier. This information is instantly matched with the present limits, such that even before forwarding the message to the model, it can be determined if this particular user falls within the allowable window. This pre-control check reduces loads on private endpoints and prevents overspending, thereby creating exact metrics for billing afterward.

The quota policy itself is multi-tiered. The lowest tier, intended for pilot teams, allows a small number of short calls. The next tier is meant for internal products that have moderate traffic. The enterprise tier removes most restrictions and requires an agreed budget as well as extended observability measures. Transitions between tiers are handled through an administrative interface. At the same time, the accumulated statistics on tokens and call frequency automatically recompute projected costs so that the customer can approve higher limits judiciously.

If the current flow exceeds the permitted parameters, the gateway does not block traffic immediately but sends requests to a fallback queue with lower priority. The client receives a response about a temporary delay or a rejection with a recommendation to reduce context or lower frequency. This approach avoids unexpected timeouts, provides a transparent explanation of refusal reasons, and preserves the predictability of services that depend on Claude across all architectural layers considered.

It starts with how much lag is acceptable in the UI, what budget the finance team can allocate, and how strict the rules on data saving and sending are. If the chat helper needs to answer almost right away and inside rules allow putting only in a lone net, boxes in a private cloud with a local end spot are often liked. When some wait time is fine and cash holds back call use, it is easier to count on a job gate or a mixed plan with group handling, since these have small extra work and exact token hold.

The next factor is about the complexity of the business logic and how much context the model will need. For short requests, which are best phrased as “business rules limited to basic validation,” a serverless gateway suits well. Suppose long legal documents or entire code repositories have to be included in a request. In that case, it works better with an extended window on a container, or even a batch queue that permits large archives to be sent as one package. In this scenario, a sliding summary of older messages makes for a universal add-on, reducing token volume in both cases.

Also, anticipated surges and the scaling strategy should be considered. Projects with even flow get enough time for horizontal container buildup; unplanned peak events need immediate scaling of the function gateway plus fallback queues. The choice of pattern depends on which layer can elastically take in request growth without causing user experience degradation and without breaking the bank, while keeping unified mechanisms for quotas, observability, and protection.

Hence, the equilibrium of response speed, context volume, cost, and security requirements unfolds the selection of an infrastructure pattern for Claude: for interactive tasks — micro-APIs with a serverless gateway, aggressive autoscaling, and a local cache; for deep analysis and batch processing — containers in an isolated VPC with private endpoints and Batch interfaces. In all cases: token control, context compression, queue segmentation, and budget triggers. Adopting these trade-offs and implementing the described practices enables teams to maintain p95 latency, guarantee cost predictability, and comply with security policies.

4. Conclusion

The presented analysis shows that the evolution of load and context capabilities in Claude models produces unambiguous engineering requirements for infrastructure. Under conditions of growing enterprise traffic, huge context windows, and the sensitivity of front-office tasks to delays, the optimal solution is a combination of two classes of patterns: micro-APIs with a serverless gateway, aggressive autoscaling, and a local cache for interactive scenarios, and containerized deployments in an isolated VPC with private endpoints and Batch interfaces for deep analysis and large-scale processing. The main design features include splitting flows by SLO, a mix router to pick between near quick and long thought, outside helpers for signing and checking requests, and using group lines and saving to cut extra work when handling big data saves.

The practical implementation of such patterns demands strict mechanisms of operational control: accounting and quota enforcement at ingress for tokens, queue segmentation by budget and by priority, automatic condensation of context in dialogue, centralized control of total size of context, fallback queues on limit exceedance, and detailed telemetry to calculate projected cost. Implementing these measures makes it possible to maintain p95 latency at a level suitable for interactive interfaces, preserve cost predictability, and simultaneously meet corporate security and audit requirements. This set of trade-offs and practices serves as a practical guide for choosing infrastructure solutions for the industrial deployment of AI services on the Claude platform.

Acknowledgment

This research was conducted independently without external funding.

References

[1] G. Alvarez, "Gartner top 10 strategic technology trends for 2025," *Gartner*, Oct. 21, 2024. https://www.gartner.com/en/articles/top-technology-trends-2025 (accessed Jul. 17, 2025).

[2] R. Szkutak, "Enterprises prefer Anthropic's AI models over anyone else's, including OpenAI's," *TechCrunch*, Jul. 31, 2025. https://techcrunch.com/2025/07/31/enterprises-prefer-anthropics-ai-models-over-anyone-elses-including-openais/ (accessed Jul. 18, 2025).

[3] "Claude Sonnet 4 now supports 1M tokens of context," *Anthropic*, 2025. https://www.anthropic.com/news/1m-context (accessed Jul. 19, 2025).

[4] B. Elad, "Claude AI Statistics 2025: Market Share, Accuracy & Trust Scores," *SQ Magazine*, 2025. https://sqmagazine.co.uk/claude-ai-statistics/ (accessed Aug. 10, 2025).

[5] "What is the pricing for the Team plan?" *Anthropic*. https://support.anthropic.com/en/articles/9267305-what-is-the-pricing-for-the-team-plan (accessed Jul. 22, 2025).

[6] T. Tully, J. Redfern, D. Das, and D. Xiao, "2025 Mid-Year LLM Market Update: Foundation Model Landscape + Economics," *Menlo Ventures*, Jul. 31, 2025. https://menlovc.com/perspective/2025-mid-year-llm-market-update/ (accessed Aug. 01, 2025).

[7] J. Loeppky, "Is latency always important?" *IT Pro*, 2025. https://www.itpro.com/infrastructure/networking/is-latency-always-important (accessed Jul. 25, 2025).

[8] "Amazon API Gateway quotas and important notes," *AWS*. https://docs.aws.amazon.com/apigateway/latest/developerguide/limits.html (accessed Jul. 25, 2025).

[9] "Lambda quotas," *AWS*. https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html (accessed Jul. 26, 2025).

[10] G. Harvey, “What Does AI Cost in 2025?” *Theneuron*, Apr. 21, 2025. https://www.theneuron.ai/explainer-articles/what-does-ai-actually-cost-in-2025-your-guide-on-how-to-find-the-best-value-api-vs-subs-vs-team-plans-and-more (accessed Aug. 01, 2025).

[11] “Amazon Bedrock AgentCore,” AWS, 2025. Accessed: Aug. 02, 2025. [Online]. Available: https://docs.aws.amazon.com/pdfs/bedrock-agentcore/latest/devguide/bedrock-agentcore-dg.pdf

[12] “Use interface VPC endpoints (AWS PrivateLink) to create a private connection between your VPC and Amazon Bedrock,” *AWS*, 2025. https://docs.aws.amazon.com/bedrock/latest/userguide/vpc-interface-endpoints.html (accessed Aug. 03, 2025).

[13] “What limitations or quotas exist in Amazon Bedrock for model usage, request rates, or payload sizes?” *Milvus*. https://milvus.io/ai-quick-reference/what-limitations-or-quotas-exist-in-amazon-bedrock-for-model-usage-request-rates-or-payload-sizes (accessed Aug. 03, 2025).

[14] “Rate limits,” *Anthropic*, 2025. https://docs.anthropic.com/en/api/rate-limits#tier-1 (accessed Aug. 03, 2025).

[15] C. Dilmegani, “LLM Latency Benchmark by Use Cases in 2025,” *AI Multiple*, Jul. 30, 2025. https://research.aimultiple.com/llm-latency-benchmark/ (accessed Aug. 05, 2025).

[16] “Anthropic’s Claude in Amazon Bedrock,” *AWS*. https://aws.amazon.com/ru/bedrock/anthropic/ (accessed Aug. 05, 2025).