Das Preismodell von Claude – mit Fokus auf Opus 4.7 □□

Wenn dich vor allem interessiert, *„Was kostet es wirklich, wenn man Claude Opus 4.7 intensiv per API nutzt?“*, dann ist die kurze Antwort:

> **Claude Opus 4.7 ist ein Premium-Modell** – sehr leistungsfähig, aber auch klar im oberen Preissegment.

Die Standardpreise liegen bei:

Kategorie	Preis
Input	**5 USD / 1 Mio. Tokens**
Output	**25 USD / 1 Mio. Tokens**
Cache write (5 Min.)	**6,25 USD / 1 Mio. Tokens**
Cache write (1 Std.)	**10 USD / 1 Mio. Tokens**
Cache hit / Refresh	**0,50 USD / 1 Mio. Tokens**

Das wichtigste Prinzip dabei ist:
Output ist 5× teurer als Input.
Wer also sehr viel lange Antworten erzeugen lässt, spürt die Kosten deutlich stärker als bei bloß großen Eingaben.

Wie Claude grundsätzlich abrechnet

Claude rechnet tokenbasiert ab. Ein *Token* ist ein kleines Textstück. Als grobe Faustregel gilt:

- 1 Mio. Tokens Input bei Opus 4.7 kosten **5 USD**
- 1 Mio. Tokens Output kosten **25 USD**

Das ist zunächst erstaunlich günstig, wenn man nur ein paar Requests betrachtet. Bei hoher Nutzung summiert es sich aber schnell – insbesondere durch die Ausgaben des Modells.

Eine einfache Kostenformel

Für Standardnutzung ohne Sonderfunktionen kannst du grob rechnen mit:

$$
\text{Kosten} = 5 \cdot \frac{\text{Input-Tokens}}{1{,}000{,}000} + 25 \cdot \frac{\text{Output-Tokens}}{1{,}000{,}000}
$$

Das heißt:

- **100.000 Input-Tokens** kosten etwa **0,50 USD**
- **100.000 Output-Tokens** kosten etwa **2,50 USD**

Schon daran sieht man:
Nicht der Prompt ist meist der große Kostentreiber, sondern die Länge der Antwort.

Was Opus 4.7 „teuer“ macht

Opus 4.7 ist nicht deshalb teuer, weil einzelne Requests sofort riesige Summen kosten. Es ist teuer, weil bei produktiver Nutzung oft mehrere Dinge gleichzeitig zusammenkommen:

1. **viele Requests**
2. **große Kontexte**
3. **lange Antworten**
4. **Tool-Nutzung / Agenten-Workflows**
5. **dauerhafte Nutzung im Produktivbetrieb**

Gerade wenn du Opus 4.7 für komplexe Aufgaben einsetzt – etwa Coding, Analyse, Agentensteuerung oder lange Dokumentverarbeitung – entstehen hohe Tokenmengen oft ganz automatisch.

Ein weiterer Punkt:
Anthropic weist darauf hin, dass **Opus 4.7 einen neuen Tokenizer** verwendet, der bei gleichem festen Text **bis zu 35 % mehr Tokens** erzeugen kann als frühere Modelle. Das kann reale Kosten zusätzlich anheben.

Konkrete Kostenbeispiele für „viel Nutzung“

Am verständlichsten wird es mit Beispielen.

Beispiel 1: 1 Million Input + 1 Million Output im Monat

Nutzung	Kosten
1 Mio. Input-Tokens	5 USD
1 Mio. Output-Tokens	25 USD
Gesamt	**30 USD**

Das ist noch relativ moderat.

Beispiel 2: 10 Millionen Input + 10 Millionen Output im Monat

Nutzung	Kosten
10 Mio. Input-Tokens	50 USD
10 Mio. Output-Tokens	250 USD
Gesamt	**300 USD**

Auch hier sieht man: selbst bei gleichen Tokenmengen dominiert der Output.

Beispiel 3: 100 Millionen Input + 100 Millionen Output im Monat

Nutzung	Kosten

100 Mio. Input-Tokens	500 USD
100 Mio. Output-Tokens	2.500 USD
Gesamt	**3.000 USD**

Ab dieser Größenordnung spricht man schon von *ernsthafter produktiver Nutzung*.

Beispiel 4: „Viel Opus 4.7“ in einem anspruchsvollen SaaS- oder Agentensystem

Nehmen wir an:

- **300 Mio. Input-Tokens / Monat**
- **120 Mio. Output-Tokens / Monat**

Dann ergibt sich:

$$
300 \cdot 5 / 1{,}000 = 1{,}500\ \text{USD}
$$

und

$$
120 \cdot 25 / 1{,}000 = 3{,}000\ \text{USD}
$$

also insgesamt:

$$
1{,}500 + 3{,}000 = 4{,}500\ \text{USD}
$$

Nutzung	Kosten
300 Mio. Input-Tokens	1.500 USD
120 Mio. Output-Tokens	3.000 USD
Gesamt	**4.500 USD**

Das ist ein ziemlich realistischer Bereich für intensive Business-Nutzung.

Was kostet ein einzelner typischer Opus-Request?

Ein einzelner Request klingt oft unscheinbar. Beispiel:

- **20.000 Input-Tokens**
- **4.000 Output-Tokens**

Dann kostet das:

$$
20{,}000 \cdot \frac{5}{1{,}000{,}000} = 0{,}10\ \text{USD}
$$

$$
4{,}000 \cdot \frac{25}{1{,}000{,}000} = 0{,}10\ \text{USD}
$$

also insgesamt:

$$
0{,}20\ \text{USD}
$$

Das wirkt günstig. Aber bei **100.000 solchen Requests pro Monat** wären das bereits:

$$
100{,}000 \cdot 0{,}20 = 20{,}000\ \text{USD}
$$

Hier sieht man den entscheidenden Punkt:
API-Kosten werden nicht durch den Einzelrequest gefährlich, sondern durch die Skalierung.

Prompt Caching: der wichtigste Hebel für Vielnutzer

Wenn du immer wieder denselben großen Systemprompt, dieselben Dokumente oder dieselbe Gesprächshistorie mitschickst, wird es teuer. Genau dafür gibt es **Prompt Caching**.

Die Multiplikatoren relativ zum normalen Inputpreis sind:

Operation	Preisfaktor
5-Minuten-Cache schreiben	**1,25×**
1-Stunden-Cache schreiben	**2×**
Cache lesen	**0,1×**

Für Opus 4.7 bedeutet das konkret:

Cache-Typ	Preis
5m Write	6,25 USD / Mio. Tokens
1h Write	10 USD / Mio. Tokens
Read/Hit	0,50 USD / Mio. Tokens

Warum sich das lohnt

Wenn du z. B. einen großen Prompt mit **1 Mio. Tokens** immer wieder brauchst:

- ohne Cache: jeder Abruf kostet **5 USD**
- mit Cache:
  - erster Write: **6,25 USD**
  - jeder spätere Read: **0,50 USD**

Schon nach sehr wenigen Wiederverwendungen ist Caching deutlich günstiger. Für Vielnutzer ist das oft der größte Kostensenker überhaupt.

Batch API: Opus 4.7 deutlich günstiger, wenn Zeit egal ist

Wenn deine Aufgaben nicht in Echtzeit erledigt werden müssen, ist die **Batch API** extrem interessant. Sie gibt **50 % Rabatt** auf Input und Output.

Für **Opus 4.7** gilt dann:

Kategorie	Standard	Batch
Input	5 USD / Mio.	**2,50 USD / Mio.**
Output	25 USD / Mio.	**12,50 USD / Mio.**

Das ist enorm. Wenn du große Mengen an Analysen, Klassifikationen oder Offline-Verarbeitung hast, halbieren sich die Kosten praktisch sofort.

Beispiel

Statt:

- 100 Mio. Input = 500 USD
- 100 Mio. Output = 2.500 USD

zahlst du im Batch-Modus:

- 100 Mio. Input = 250 USD
- 100 Mio. Output = 1.250 USD

also insgesamt nur:

- **1.500 USD statt 3.000 USD**

Fast Mode: sehr schnell, aber sehr teuer □□

Für Opus 4.7 gibt es auch einen **Fast Mode**. Der kostet allerdings **6× Standardpreis**.

Kategorie	Standard	Fast Mode
Input	5 USD / Mio.	**30 USD / Mio.**
Output	25 USD / Mio.	**150 USD / Mio.**

Das ist kein kleiner Aufpreis, sondern ein massiver Premium-Tarif.

Wenn du viel Opus 4.7 nutzt, solltest du Fast Mode nur dann einsetzen, wenn die zusätzliche Geschwindigkeit einen echten Geschäftswert hat.

Beispiel

Bei:

- 10 Mio. Input
- 10 Mio. Output

würden die Kosten statt **300 USD** plötzlich bei:

- Input: 300 USD
- Output: 1.500 USD
- **Gesamt: 1.800 USD**

liegen.

Data Residency: leichter Aufpreis für US-only Inference

Für Opus 4.7 gilt: Wenn du mit `inference_geo: "us"` arbeitest, wird ein **1,1× Multiplikator** angewendet.

Das heißt:

- Input wird von **5 USD** auf **5,50 USD / Mio.**
- Output von **25 USD** auf **27,50 USD / Mio.**

Das ist kein dramatischer, aber ein merklicher Aufpreis von **10 %** auf alles – auch auf Cache-Operationen.

Tool-Nutzung: oft ein versteckter Kostentreiber

Wenn du Claude mit Tools einsetzt, steigen die Tokens zusätzlich.

Bei Opus 4.7 kommen allein durch den Tool-Mechanismus schon systemseitige Zusatz-Tokens hinzu:

Tool choice	Zusatz-Tokens
`auto`, `none`	346 Tokens
`any`, `tool`	313 Tokens

Dazu kommen noch Tokens für:

- Tool-Definitionen
- Tool-Aufrufe
- Tool-Ergebnisse
- eventuell große Inhalte aus Such- oder Fetch-Tools

Spezielle Tools

Ein paar Beispiele:

Tool	Zusatzkosten
Bash Tool	**245 Input-Tokens** extra
Text Editor Tool	**700 Input-Tokens** extra
Web Search	**10 USD pro 1.000 Suchen** + Tokenkosten
Web Fetch	keine Extra-Gebühr, aber Tokenkosten für Inhalte
Code Execution	teils laufzeitbasiert, je nach Nutzungskontext

Gerade bei agentischen Workflows kann der eigentliche „Chat“ preislich fast zweitrangig werden, wenn viele Tool-Ergebnisse in den Kontext zurückfließen.

Wie teuer ist „sehr viel“ Opus 4.7 wirklich?

Wenn man es praxisnah einordnet, könnte man ungefähr so unterscheiden:

Nutzungsniveau	Typischer Monatsbereich
Experimentell / klein	10–100 USD
Kleines Produkt / Prototyp	100–1.000 USD
Seriöse Produktionsnutzung	1.000–10.000 USD
Große intensive Nutzung	10.000+ USD

Mit Opus 4.7 erreicht man die hohen Bereiche schneller als mit günstigeren Modellen, weil der **Outputpreis von 25 USD / Mio. Tokens** recht kräftig ist.

Wenn du also wirklich „viel“ Opus 4.7 nutzt – etwa für viele Nutzer, lange Antworten, große Kontexte oder Agenten mit Tools – dann sind **mehrere tausend USD pro Monat** absolut realistisch.
Bei großen Workloads auch **fünfstellige Monatskosten**.

Wann Opus 4.7 wirtschaftlich sinnvoll ist

Opus 4.7 lohnt sich vor allem dann, wenn die höhere Qualität wirtschaftlich mehr bringt als sie kostet, zum Beispiel bei:

- komplexer Analyse
- anspruchsvollem Coding
- mehrstufigem Reasoning
- hochwertigen Agenten-Workflows
- Fällen, in denen Fehler sehr teuer wären

Weniger sinnvoll ist Opus 4.7 dagegen für:

- einfache Klassifikation
- Standard-Zusammenfassungen
- einfache Extraktion
- Routine-Chatbots
- Massenverarbeitung ohne hohe Qualitätsanforderung

In solchen Fällen sind Sonnet oder Haiku oft deutlich wirtschaftlicher.

Praktisches Fazit

Claude Opus 4.7 ist nicht „absurd teuer“ pro Anfrage – aber teuer in der Skalierung.
Die wichtigste Erkenntnis ist:

> **Wenn du viel Opus 4.7 API nutzt, bestimmt vor allem die Menge an Output-Tokens deine Rechnung.**

Merke dir besonders diese Punkte

1. **Standardpreis**
   - 5 USD / Mio. Input
   - 25 USD / Mio. Output
2. **Output ist der Hauptkostentreiber**
   - lange Antworten kosten deutlich mehr als große Prompts
3. **Prompt Caching kann massiv sparen**
   - besonders bei wiederkehrendem Kontext
4. **Batch halbiert die Kosten**
   - ideal für Offline-Workloads
5. **Fast Mode ist Luxus**
   - 6× Standardpreis
6. **Bei intensiver Nutzung sind schnell mehrere tausend USD pro Monat realistisch**

Eine einfache Daumenregel für Opus 4.7

Wenn du schnell überschlagen willst:

- **pro 1 Mio. Input-Tokens: 5 USD**
- **pro 1 Mio. Output-Tokens: 25 USD**

Oder noch einfacher:

> **1 Mio. Tokens Antworttext kostet ungefähr so viel wie 5 Mio. Tokens Eingabetext.**

Das beschreibt die Preislogik von Opus 4.7 ziemlich gut.

Wenn du möchtest, kann ich dir im nächsten Schritt auch noch eine **konkrete Kostenkalkulation für deinen Use Case** machen – zum Beispiel für:

- **100.000 Requests pro Monat**

- **einen Coding-Agenten**
- **ein SaaS mit x Nutzern**
- oder einen Vergleich **Opus 4.7 vs. Sonnet 4.6 vs. Haiku 4.5**.

Revision #1
Created 2026-05-23 14:00:27 UTC by art10m
Updated 2026-05-23 14:00:49 UTC by art10m