ECONOMÍA DE TOKENS · GITHUB COPILOT

Aprovecha más cada token.
Optimización de tokens y calidad de agentes en la era de los AI Credits.

Un briefing ejecutivo y técnico sobre el usage-based billing, qué cambia para tu organización de ingeniería y cómo operar bien con él.

AUTORA
Paula Silva

CARGO
Software Global Black Belt

DURACIÓN
60 a 90 minutos

FECHA
2026-06-14

AGENDA

Siete partes, un solo marco.

PART I	El cambio, los AI Credits reemplazan el cobro por request el 1 de junio de 2026
PART II	Fundamentos, cómo funcionan de verdad los LLMs, agentes, tokens y ventanas de contexto
PART III	Tres superficies, VS Code, GitHub Copilot CLI y GitHub.com, una sola billetera
PART IV	Calidad de agente primero, por qué la calidad supera contar tokens
PART V	Las seis palancas de optimización, y cómo se acumulan
PART VI	Aplicar y gobernar, ingeniería de contexto, presupuestos, observabilidad, adopción
PART VII	Anexo, cache, el prompt cache de GitHub Copilot frente a Foundry más Redis

PARTE

I

El cambio.

El 1 de junio de 2026, GitHub Copilot deja de cobrar por request. El token pasa a ser la unidad de costo.

LA FECHA DE CORTE

2026-06-01

Las Premium Request Units se retiran. Los GitHub AI Credits toman el relevo.

1 AI Credit equivale a US$ 0,01. Los tokens de entrada, salida y cache se miden a la tarifa publicada de cada modelo. El costo ahora sigue el trabajo realizado, no la frecuencia con que preguntas.

UNIDAD ANTIGUA		UNIDAD NUEVA
1 premium request	→	tokens, 1 cr = US$ 0,01

ANTES Y DESPUÉS

La unidad de costo pasó de intención a trabajo.

ANTES · PREMIUM REQUESTS

Costo por intención

Un prompt de una palabra y una sesión de agente de cuarenta turnos contaban igual, una request cada uno. Fácil de presupuestar, pero desconectado del cómputo realmente consumido.

DESPUÉS · AI CREDITS

Costo por trabajo

Tokens de entrada, salida y cache a la tarifa de cada modelo. Una sesión agéntica larga es de hecho mucho más cara que una pregunta de chat. La cuenta ahora coincide con la realidad.

LA ECUACIÓN DEL BILLING

Dos factores, y controlas los dos.

LA ECUACIÓN

costo = modelo por tokens

costo = entrada por tarifa-de-entrada, más cache por tarifa-de-cache, más salida por tarifa-de-salida. 1 AI Credit = US$ 0,01. Cada táctica de este deck actúa sobre el modelo o sobre los tokens.

QUÉ CONSUME CRÉDITOS

Trabajo pesado, no autocompletado

Chat, agentes, coding agent y modelos premium consumen créditos. Code completions y Next Edit Suggestions no. La revisión de código de GitHub Copilot cuesta dos veces: AI Credits más minutos de GitHub Actions.

LO QUE NO CAMBIA

Los precios base se mantienen, y el día a día sigue gratis.

01

Los precios siguen igual

Pro US$ 10, Pro+ US$ 39, Business US$ 19, Enterprise US$ 39 por usuario. Cada licencia suma su valor en créditos a un pool compartido.

02

Completions siguen gratis

Code completions y Next Edit Suggestions siguen ilimitadas y no consumen AI Credits. La experiencia inline del día a día no cambia.

03

Solo el trabajo pesado mide

Chat, modo agente, coding agent, Spaces y modelos premium consumen créditos. Ahí es donde el medidor realmente gira.

DEL BUFFET AL À LA CARTE

No es un cobro injusto. Es la devolución de un control que no existía.

01

El buffet escondía el precio

El plato caro siempre costó mucho de producir. El buffet de precio fijo escondía el costo de cada plato y premiaba el desperdicio en silencio.

02

Ahora el menú tiene precios

El usage-based billing te entrega el menú à la carte con los precios impresos. La cocina no cambió. Ahora puedes elegir.

03

El mercado ya se movió

FinOps 2026: 98 por ciento de las organizaciones ya gestionan gasto de IA, frente a 31 por ciento en 2024. Es madurez de mercado, no una rareza de GitHub.

POR QUÉ EL PRECIO EVOLUCIONA

El cobro por token coincide con cómo funciona el cómputo de IA.

01

Tokens, no requests

Los proveedores cobran a GitHub en tokens. Una request puede ser cientos o decenas de miles. El precio por request igual nunca coincidió con el costo.

02

El mercado se mueve rápido

Los labs lanzan modelos y reprecian constantemente. Un framework de token absorbe eso sin una revisión de precios cada vez.

03

Desbloquea features

Ventanas de contexto más grandes y workflows agénticos más ricos solo tienen sentido económico cuando el costo se mide por consumo.

LA LÍNEA DE TIEMPO

Presupuesta contra septiembre, no contra la cuenta amortiguada de hoy.

ANUNCIO Y PREVIEW	Anunciado el 27 de abril. El preview de billing en mayo es tu ventana para instrumentar monitoreo y configurar presupuestos antes de cualquier cobro.	Abr a mayo
COLCHÓN NATIVO	El usage-based billing arranca el 1 de junio con holgura incluida que amortigua los primeros meses. Úsalo como ventana de aprendizaje del consumo real.	1 jun a 31 ago
LLEGA LA CUENTA REAL	El colchón expira el 31 de agosto. Septiembre es el primer mes sin amortiguación. Re-baseline aquí, nunca contra gasto promocional.	Desde sept

POR QUÉ EL IMPACTO VARÍA TANTO

La cuenta es un espejo de la madurez de plataforma, no un veredicto.

01

Madurez, no tamaño

Dos empresas sienten el mismo cambio de formas opuestas. La diferencia es madurez de uso, no headcount.

02

Espejo, no veredicto

Los equipos con gobernanza pagan en proporción al valor. Los que no la tienen pagan por el caos que el precio fijo escondía.

03

La IA amplifica

DORA 2025 mostró que la IA no arregla un equipo, amplifica lo que ya existe. El usage billing solo lo hace visible en la factura.

PARTE

II

Fundamentos.

Cómo funcionan los LLMs, agentes, tokens y ventanas de contexto, el modelo mental del que depende el resto.

EL MODELO Y LA REGLA

Una máquina de probabilidad de palabras, alimentada por ti.

01

Predice el siguiente token

Un LLM es texto entra, texto sale. No distingue lo relevante de lo irrelevante, ni una alucinación de un hecho. Ambos salen de la misma matemática.

02

Equilibrio de contexto

La regla central: lo mínimo de contexto posible, pero todo el necesario. Demasiado sesga la respuesta y cuesta más, muy poco invita a la alucinación.

03

Un token es un fragmento

Cerca de tres cuartos de una palabra. 1M de tokens es más o menos la trilogía de El Señor de los Anillos más El Hobbit. Prompts, archivos y respuestas todos consumen.

LA REGLA CENTRAL

Lo mínimo de contexto posible, pero todo el necesario.

DEMASIADO

Sesga la respuesta

La información irrelevante se pondera, no se ignora. Empuja al modelo a respuestas erróneas, y pagas por cada token extra, cada turno.

MUY POCO

Invita a la alucinación

Faltar contexto crítico hace que el modelo llene el hueco, sin mensaje de error. La matemática no distingue un hecho de una invención.

QUÉ ES UN TOKEN

Un token es un fragmento de palabra, y la densidad varía.

01

Tokenización subword

Cerca de tres cuartos de una palabra. El inglés promedia unos 1,3 tokens por palabra.

02

El idioma importa

Portugués y español rondan 1,5 a 1,7 tokens por palabra. El código tokeniza en densidad variable.

03

No lo sobre-ajustes

Tienes poco control sobre la tokenización. Piensa en alto nivel: prompts, archivos y respuestas consumen y se acumulan cada turno.

QUÉ ES REALMENTE UN AGENTE

Un agente es solo código hablando con un modelo, muchas veces.

01

El harness

Un agente es una aplicación: VS Code Chat, GitHub Copilot CLI, el coding agent, incluso Claude Code o Codex. Le habla al modelo por ti, muchas veces por tarea.

02

El LLM

El modelo en sí es solo GPT, Claude o Gemini. La interacción no es magia, es texto, y el modelo es stateless.

03

Tus palancas

Lo influyes por tres cosas: tu prompt, los archivos del proyecto, y los configs del agente, instrucciones, skills, MCP.

EL EFECTO DE REENVÍO

2M+

El modelo no recuerda. Relee, en cada turno.

Una sesión de 50.000 tokens en 40 turnos envía cerca de 2 millones de tokens de entrada, aunque tu último prompt tenga 20 palabras. Como los LLMs son stateless, la conversación entera se reenvía cada turno, así que el inflado de contexto es la mayor fuente de desperdicio.

A DÓNDE VAN DE VERDAD LOS TOKENS

Tu prompt es la rebanada más pequeña de la cuenta.

01	COSTO FIJO	**System prompt y esquemas de tools**	No los cambias, pero son por qué cada sesión empieza por encima de cero.
02	CRECE CADA TURNO	**Historial de la conversación**	La rebanada que la mayoría nunca piensa, y la que se acumula con el tamaño de la sesión.
03	LA DESCUIDADA	**Resultados de tool-call**	Lecturas de archivo, salida de shell, pruebas. Una lectura descuidada de un archivo grande viaja junto para siempre.
04	LA MENOR, IGUAL SE COBRA	**Tu prompt y la salida**	Tu prompt escrito suele ser la rebanada más pequeña. La respuesta del modelo, incluyendo tokens de razonamiento, también se cobra.

TRES TIPOS DE TOKEN, TRES COMPORTAMIENTOS

La salida es el token más caro. El cache es el más barato.

01

Entrada

Todo lo que envías: prompt, historial, adjuntos, system prompt, esquemas de herramientas. Se re-cobra cada turno. La tarifa base.

02

Salida

Lo que el modelo genera, incluyendo tokens invisibles de razonamiento. Típicamente 4 a 10 veces la entrada. Las respuestas verbosas cuestan dinero real.

03

Cache

Un prefijo estable que el proveedor reproduce desde cache. Cerca del 10 por ciento de la entrada, un descuento del 90 por ciento. La mayor palanca para trabajo agéntico.

VENTANA FRENTE A CUENTA

Una ventana más grande no es una cuenta más pequeña.

LA VENTANA DE CONTEXTO

Lo que el modelo puede sostener

1 token es cerca de tres cuartos de palabra. 1M de tokens es más o menos la trilogía de El Señor de los Anillos más El Hobbit. La ventana es la capacidad.

TU CUENTA

Lo que de verdad pones en ella

Los tokens miden lineal, estés al 20 o al 80 por ciento. Quédate en 500K en una ventana de 1M y pagas 500K cada turno. La compactación te salva del muro, no del costo de acercarte.

DOS FALLAS DEL CONTEXTO LARGO

Más contexto es a la vez más caro y de menor calidad.

PERDIDO EN EL MEDIO

Una curva de recall en U

Los modelos usan mejor la información al inicio y al final de la ventana, peor en el medio. Cambia de tarea a media sesión y el modelo puede volver a la primera instrucción. Empieza una ventana nueva por tarea.

CONTEXT ROT

La calidad cae antes del overflow

Chroma probó 18 modelos de frontera. Todos empeoran al crecer la entrada, a menudo mucho antes de llenar la ventana. Mantén la ventana por debajo de un 60 a 70 por ciento.

PARTE

III

Tres superficies, una billetera.

Una cuenta, tres comportamientos. Cada superficie tiene su factor de costo dominante y sus controles.

LA PLATAFORMA GITHUB Y LA FAMILIA GITHUB COPILOT

Una plataforma, dos clases de costo, muchos modelos.

01

La plataforma

GitHub es repos, pull requests, issues y Actions. GitHub Copilot es la capa de IA tejida en ella, accedida desde VS Code, el CLI, la web y otras IDEs.

02

Dos clases de costo

Gratis e ilimitado: code completions y Next Edit Suggestions. Medido en AI Credits: chat, agentes, coding agent, Spaces, modelos premium.

03

Multi-modelo

Los mismos tiers vienen de Anthropic, OpenAI y Google, enrutados por un solo plano de control y una sola unidad de cobro, AI Credits.

UNA BILLETERA, TRES COMPORTAMIENTOS

La optimización es por superficie. La física es la misma.

01

Terminal · CLI

Factor: reenvío del historial completo cada turno. Controles: /clear, /compact, /context, sub-agentes especialistas.

02

Editor · VS Code

Factor: adjuntos más la proliferación de tool-calls del agente. Controles: elección de modo, referencias hash-file, instrucciones, alcance de MCP.

03

Web · GitHub.com

Factor: ejecuciones del coding agent con gran radio de impacto. Controles: issues bien acotadas, Spaces estrechos, PR review.

EL TERMINAL

GitHub Copilot CLI premia la higiene de contexto.

01

Clear y compact

/clear resetea entre tareas distintas, /compact resume a media tarea, /context muestra la composición, /usage muestra cuánto costó la sesión.

02

Cambia de modelo a media sesión

/model te deja planear en un modelo fuerte, implementar en uno barato, revisar en un especialista. Cada paso en el tier más barato que lo termina.

03

Los especialistas ahorran tokens

/explore, /plan, /review y /delegate corren en contexto aislado. La sesión principal solo ve el resumen, no los archivos que leyeron.

EL EDITOR

Elige el modo más barato que sirve, fija contexto estrecho.

$

Ask

Solo lectura. Explicaciones, dudas de diseño, consulta a docs. Sin escribir archivos. El modo más barato y el correcto para la mayoría de las preguntas.

$

Plan

Spec primero. Redacta un plan revisable que entregas a Agent. El plan acota lo que Agent lee y cambia, y de ahí viene el ahorro.

$$$

Agent

Bucle autónomo con herramientas. Más barato guiado por una spec. Prefiere hash-file a hash-codebase, y reinicia en vez de recortar.

BAJO EL CAPÓ EN VS CODE

Dos mecánicas que moldean la cuenta en silencio.

ÍNDICE DEL WORKSPACE

Qué consulta hash-codebase

Hash-codebase consulta un índice semántico, no un grep en vivo. Para repos de GitHub es remoto y casi instantáneo desde marzo de 2025. Un índice frío fuerza lecturas más amplias y caras.

COMPACTACIÓN

/compact en el chat del editor

Resume turnos antiguos para liberar contexto, con una pista de qué conservar. Más barato que empezar de cero cuando hay continuidad, mucho más barato que desbordar.

LA WEB

Operaciones más grandes, menos frecuentes, mayor radio de impacto.

01

Coding agent

De issue a branch a PR en una VM sandbox, GA desde septiembre de 2025. La calidad de la issue domina la cuenta, así que acota bien con punteros de archivo.

02

GitHub Copilot Spaces

Contexto curado entre repos y docs. Tres archivos ganan a tres repos. Un Space por tipo de pregunta, actualiza en vez de acumular.

03

Un archivo, tres superficies

El copilot-instructions.md lo respetan VS Code, el coding agent y el CLI. Invierte en él una vez, gana en todas partes.

MISMA TAREA, TRES SUPERFICIES, TRES FORMAS DE COSTO

La perilla de costo es la misma: cuán estrecho acotas antes de que el modelo lea.

01

CLI

/explore, /plan, alcance estrecho. Cuando ya estás en el terminal para una edición rápida.

02

VS Code

Plan redacta una spec, entrégala a Agent con alcance hash-file. Para un cambio mediano que revisas antes.

03

GitHub.com

Issue con punteros de archivo, asigna @copilot, revisa el PR. Para un cambio acotado del que puedes alejarte.

CODING AGENT, CUÁNDO RINDE

Acotado y bien especificado, o no pagues por él.

BUEN FIT

Rinde

Subir deps, agregar pruebas de un módulo, un rename en el repo. Issues con punteros de archivo y criterios de aceptación, en repos con pruebas fuertes y CI.

MAL FIT

No pagues por esto

Trabajo exploratorio vago, arquitectura transversal, repos greenfield, repos sin pruebas, o algo que debió ser una conversación de diez líneas.

GITHUB COPILOT SPACES, CONTEXTO CURADO

Tres archivos ganan a tres repos.

01

Qué hay en un Space

Repos y rutas específicas, notas pegadas, instrucciones en texto libre, enlaces de docs. Enviados al contexto al momento de la consulta.

02

Palancas de costo

Fuentes estrechas, un Space por tipo de pregunta, actualiza en vez de acumular. Las fuentes viejas cuestan igual que las nuevas.

03

Historia

Spaces salió en mayo de 2025 y reemplazó a los Knowledge Bases en septiembre de 2025. Si ves KB, es el término legado.

LA WEB SON SEIS PRODUCTOS, UNA BILLETERA

Conoce la forma de costo de cada uno.

01

Chat y Edits

Q&A anclado al repo y ediciones por archivo. Los puntos de entrada más baratos de la web.

02

PR review

Auto-revisión acotada al diff. Ahora también consume minutos de Actions por su análisis agéntico.

03

Spark y coding agent

Generar apps enteras, o ejecuciones issue-to-PR. Lo más pesado por ejecución, generación larga.

MECÁNICA DE COMPACTACIÓN DEL CLI

La compactación protege el muro, no la cuenta.

01

Auto-compact en 80 a 95%

Background cerca de 80, hard cerca de 95, escalado al modelo. No dispara hasta que estás cerca.

02

Los tokens miden lineal

En 500K en una ventana de 1M pagas 500K cada turno, mucho antes de la compactación. Ventanas más grandes no son cuentas más pequeñas.

03

Context antes, resume seguido

Por debajo del 40 por ciento, no pagues por compactar aún. Resume de un summary guardado en vez de repegar contexto.

PARTE

IV

Calidad de agente primero.

Por qué optimizar la calidad supera optimizar el costo, y por qué ambos van en la misma dirección.

EL REPLANTEO

Deja de apostar. Envía menos agentes que aciertan.

LA PREGUNTA EQUIVOCADA

Apuesta con agentes

Como lanzar 20 cohetes baratos hacia la Luna y rezar para que uno aterrice. Poco contexto, un prompt perezoso, lo mandas, y reintentas si falla. Insostenible cuando cada dev despacha decenas de agentes al día.

LA PREGUNTA CORRECTA

Haz que cada token cuente

Sube el valor y la calidad de cada agente antes de enviarlo. Menos agentes, mejores, ya significa menos tokens. Optimiza el retorno por agente.

CALIDAD GENERA VALOR GENERA ROI

Optimizar costo cuando el valor es cero es trabajo inútil.

LA FÓRMULA

ROI del agente

ROI = valor de la salida menos costo de tokens, dividido por el costo de tokens. No lo calculas limpio, pero guía: llevar el costo a cero en una salida inútil no es victoria.

LA DINÁMICA

Recortar contexto hace ambas

Subir el valor de un agente a menudo se logra enviando menos tokens. Recortar contexto irrelevante es una palanca que mejora la calidad y baja el costo a la vez.

EL PROBLEMA DEL ERROR COMPUESTO

Pequeños errores por paso se multiplican en el flujo.

POR PASO

99%

Precisión optimista en un solo paso

SE ACUMULA

50 steps

EN EL ÉXITO TOTAL

TRAS 50 PASOS

61%

Incluso con 99 por ciento por paso. Con 95 por ciento caes cerca del 8 por ciento.

SHIFT-LEFT, PARA AGENTES

Los controles deterministas reinician el reloj de la precisión.

01

Las pruebas son deterministas

Una prueba pasa o falla, sin probabilidad. Una prueba en rojo detiene al agente que se desvió y lo obliga a reconstruir sobre una base estable.

02

53 por ciento son pruebas

El equipo de GitHub Copilot CLI entrega unos 500 PRs por semana. Su práctica número uno son las pruebas, 53 por ciento del código.

03

Más allá de las pruebas

Linters, verificadores de tipos y escáneres de seguridad son guard-rails que el agente ejecuta. Sin ellos, apila cambio con bug sobre cambio con bug.

LA TESIS

Costo y calidad van en la misma dirección.

CONTAR TOKENS

La trampa

Exprimir tokens hasta que el agente produce peor no ahorra, pagaste por un resultado peor y por el retrabajo que sigue.

HACERLOS CONTAR

La palanca

Recortar contexto irrelevante sube la calidad y baja el costo a la vez. Ajusta el esfuerzo a la madurez: cuantos más agentes corras, más rinde.

AJUSTA EL ESFUERZO A LA MADUREZ

Cuánto optimizas depende de cuántos agentes corres.

INGENIERO ASISTIDO POR IA

Un agente, casi sync

Cerca de 10 agentes al día, IA como asistente. Incluso 50 por ciento de ahorro en un mes de 20 dólares son solo 10. Aprende los fundamentos, aplica los pocos hábitos clave.

INGENIERO DE IA

Orquestador de muchos

Decenas o cientos de agentes async. Cada por ciento de tokens y de calidad, recuerda la matemática del error compuesto, se multiplica por la flota. El framework entero rinde.

RASGOS DE LARGO PLAZO

Lo que te hace valioso en el desarrollo agéntico.

01

Habilidades analíticas

Programar nunca fue el valor, el análisis lo fue. Decirle al agente exactamente qué hacer, en el lenguaje del dominio, se vuelve lo más valioso.

02

Buena arquitectura

Domain-driven design, hexagonal, CQRS, event-driven. La arquitectura limpia da guard-rails al agente y reduce errores.

03

Itera los configs

Ahora eres un ingeniero de contexto. Trata los errores del agente como incidentes, mantén los configs frescos, recréalos cada trimestre.

PARTE

V

Las seis palancas.

Seis palancas que se acumulan, cada una corrige un factor de gasto. Los modelos de campo ubican el efecto combinado en la banda de 37 a 44 por ciento.

SEIS FACTORES DE GASTO DE TOKEN

Diagnostica la causa antes de aplicar el remedio.

01

Contexto, 30 a 50%

Archivos abiertos, indexación e historial enviados cada turno.

02

Modelo, 10 a 24x

Tier de frontera frente a ligero, el spread del Opus al mini.

03

Instrucciones, 10x sin cache

El mismo contexto de proyecto reenviado cada turno sin cache.

04

Razonamiento, 10 a 80x

Trazas de pensamiento alto y máximo, cobradas como tokens de salida.

05

Agentes y tools, 4 a 30x

Los bucles de agente re-cobran el trabajo, esquemas de MCP reenviados cada turno.

06

Sesiones, hasta 8x

El historial se acumula entre turnos, el turno 30 sin gestión.

EL FRAMEWORK

Seis factores, seis palancas, un efecto compuesto.

01

Disciplina de contexto

Adjunta el menor alcance que deja responder al modelo. El token más barato es el que nunca envías.

02

Enrutamiento de modelo

Empareja el modelo con la tarea. Un spread de 10 a 24 veces entre tiers es dinero real. Usa Auto por defecto.

03

Ingeniería de instrucción

Escribe los estándares del equipo una vez en archivos cacheables, y reúsalos con cerca del 90 por ciento de descuento.

04

Profundidad de razonamiento

Pensar se cobra como salida. Por defecto medio, escala solo en turnos realmente difíciles.

05

Agentes y herramientas

Recorta el overhead de MCP, especializa roles, delega el descubrimiento a subagentes. Gobierna el bucle o la cuenta explota.

06

Ciclo de sesión

El historial se acumula cada turno. Un tema, una sesión. Resetea, no acumules.

EL EFECTO COMPUESTO

~41%

Las palancas se suman. La misma licencia hace mucho más trabajo útil.

Aplicadas juntas, las palancas logran cerca de 41 por ciento de reducción de tokens sin pérdida medible de productividad, dentro de la banda de 37 a 44 por ciento. Las cifras son ilustrativas, mide tu propia baseline.

REDUCCIÓN DE TOKENS		TRABAJO ÚTIL POR LICENCIA
37 a 44 por ciento	→	cerca de 1,6 a 1,8 veces

PALANCAS 1 Y 2

Envía menos, y al modelo correcto.

PALANCA 1 · DISCIPLINA DE CONTEXTO

15 veces menos entrada

Nombrar los 3 archivos que importan en vez de barrer todo el codebase llevó un refactor real de unos 25.000 a 1.700 tokens de entrada por turno, con una respuesta más afilada.

PALANCA 2 · ENRUTAMIENTO DE MODELO

6 veces más barato, misma feature

Planea en un modelo potente, implementa en uno ligero, revisa en uno versátil. Una feature de OAuth bajó de unos US$ 13,75 a US$ 2,30, con la misma calidad.

TARIFAS POR MODELO, AI CREDITS POR 1M DE TOKENS

Elegir modelo es una decisión de presupuesto, no de estética.

MODELO	ENTRADA	SALIDA	MEJOR USO
GPT-5 mini	25 cr	200 cr	Chat diario, completions de rutina
GPT-5	125 cr	1,000 cr	Razonamiento y código de complejidad media
Claude Sonnet 4.6	300 cr	1,500 cr	Contexto largo, trabajo de código sostenido
Claude Opus 4.7	1,500 cr	7,500 cr	Tareas difíciles de agente, uso reservado

Ningún modelo es gratis bajo usage-based billing: todo modelo, incluido el tier ligero, consume créditos por token. Solo code completions y Next Edit Suggestions no consumen créditos. Las tarifas y nombres de modelo son ilustrativos a mediados de 2026, valida siempre contra el rate card oficial de modelos y precios de GitHub Copilot antes de fijar presupuestos.

EMPAREJA EL MODELO CON LA TAREA

Tres tiers, un spread de 10 a 24 veces.

01

Ligero

Q&A simple, formato, boilerplate, refactors simples. El default barato para la rutina.

02

Versátil

Código del día a día, feature, review, debugging. El caballo de batalla donde vive el gasto.

03

Potente

Arquitectura, debugging difícil, security review, lógica nueva. Resérvalo, el multiplicador es alto.

PALANCAS 3 Y 4

Cachea el prefijo recurrente, paga el pensamiento que la tarea exige.

PALANCA 3 · INGENIERÍA DE INSTRUCCIÓN

Paga una vez, reúsa con 90 por ciento off

Mantén los archivos del equipo estables para que el cache valga. Agrega reglas nuevas al final, nunca reordenes. En 50 devs eso ahorra del orden de US$ 32.000 al año solo en el prefijo.

PALANCA 4 · PROFUNDIDAD DE RAZONAMIENTO

Por defecto medio

En modelos de razonamiento, esfuerzo alto o máximo puede multiplicar la cuenta de 10 a 80 veces. Por defecto medio y descompón: un plan medio, varios pasos bajos baratos, una revisión media. Cerca de 7 veces más barato por tarea.

EL ESFUERZO DE RAZONAMIENTO SE COBRA COMO SALIDA

De LOW a MAX puede ser hasta 80 veces la cuenta.

1x

LOW

1x. Cerca de 200 a 800 tokens de pensamiento. Sirve para la mayoría de los pasos.

2-4x

MEDIUM

2 a 4x. Cerca de 1K a 4K. El default correcto para la mayoría del código.

10-25x

HIGH

10 a 25x. Cerca de 5K a 20K. Guárdalo para turnos realmente difíciles.

50-80x

MAX

50 a 80x. Hasta 64K tokens de pensamiento. Solo diseño arquitectónico y bugs sutiles.

PALANCAS 5 Y 6

Gobierna el bucle del agente, resetea la sesión.

PALANCA 5 · AGENTES Y HERRAMIENTAS

Recorta, especializa, delega

MCP recorta lo enviado, los hooks recortan lo almacenado, las skills saltan la exploración. Especializa roles y delega el descubrimiento a subagentes. Un debug de 30 turnos bajó de unos US$ 45 a US$ 6.

PALANCA 6 · CICLO DE SESIÓN

Un tema, una sesión

En el turno 30 el trozo de historial es cerca del 90 por ciento de la cuenta de cada turno. Tema nuevo, chat nuevo. Compacta con foco, bifurca para explorar, archiva al terminar.

HÁBITOS QUE SUMAN ENCIMA

Restringe la salida, usa el modo Ask, adopta el kit inicial.

01

Restringe la salida

La salida es el token más caro. Acótala: una frase, tres bullets, solo código. Solo código recorta la salida de 40 a 70 por ciento.

02

Usa el modo Ask

El modo fija el número de llamadas: Ask es una, Agent es 5 a 25. No pagues overhead de bucle de agente por una pregunta acotada.

03

El kit inicial

Elige tres: /clear, /model, /usage. Control de contexto, control de costo, visibilidad de costo, uno por cada dimensión.

PALANCA 5A, RECORTA EL OVERHEAD DE TOOLS

Tres fuentes de inflado, tres palancas.

01

MCP recorta lo enviado

Deshabilita servidores que no necesitas hoy, usa un CLI para puntuales. Cerca de 5K a 1K por request.

02

Hooks recortan lo almacenado

Un hook local filtra salida ruidosa antes de entrar al historial. Cerca de 10K crudo a 200 filtrado.

03

Skills saltan la exploración

Un SKILL.md describe la estructura una vez en vez de leer 5+ archivos. Cerca de 5K a 500.

PALANCA 5B, AGENTES DE HANDOFF

Fija un tier por rol, pasa un payload pequeño.

01

Planner, potente

Tools de solo lectura, produce un plan de 5 pasos y notas de diseño. La tarifa de frontera, pagada una vez.

02

Implementer, ligero

Tools de edición, ejecuta un paso del plan por turno. El grueso del trabajo, en el tier barato.

03

Reviewer, versátil

Solo lectura, diff y pruebas. Naive 10 turnos Opus cerca de $15, handoff cerca de $2,11, ~7x más barato.

PALANCA 5C, SUBAGENTES

Paga el descubrimiento una vez, no cada turno.

SIN SUBAGENTE

Re-cobrado cada turno

El padre recarga el mismo contenido de archivo cada turno. Un refactor de 20 archivos en 4 turnos envía 86K de entrada, pagado cuatro veces.

CON SUBAGENTE

Leído una vez, resumido

El subagente lee los archivos en contexto aislado y devuelve un resumen de 1K. La entrada del padre queda plana, cerca de 8K total, unas 10 veces menos.

SUBAGENTE FRENTE A HANDOFF

Elige por la forma de la tarea, no por preferencia.

5.1

Subagente (5.1)

Descubrimiento acotado, volumen de archivos el cuello de botella, una respuesta, fan-out paralelo, el padre se queda. El descubrimiento va a un subagente.

5.3

Handoff (5.3)

Build multifase, profundidad de razonamiento el cuello de botella, tres o más fases, secuencial, el testigo pasa. Un build va a handoff.

PARTE

VI

Aplicar, gobernar, medir.

Ingeniería de contexto, presupuestos y controles, observabilidad y un playbook de adopción secuenciado.

INGENIERÍA DE CONTEXTO

Ahora eres un ingeniero de contexto.

01

Instrucciones persistentes

El copilot-instructions.md viaja en cada sesión. Mantenlo pequeño, escríbelo tú, registra errores recurrentes, recréalo cada trimestre.

02

Skills y reglas con alcance

Instrucciones con alcance por ruta y skills cargan solo cuando son relevantes. Prompt files y chat modes limitan el toolset y el radio de impacto.

03

Agentes y subagentes

Fija modelo y herramientas por rol para evitar caminos erróneos. Los subagentes leen archivos una vez y devuelven solo un resumen, manteniendo al padre liviano.

TU PROMPT, SIEMPRE ACTIVO

Dirige desde el inicio, no recortes palabras.

01

Sé preciso

No 'arregla el bug', sino 'la issue 45 describe X, arréglalo'. Apunta al síntoma y al archivo exactos.

02

Agrega señales de parada

Cuando el bug esté arreglado y las pruebas pasen, detente. Evita que el agente divague en trabajo no pedido.

03

Anticipa el contexto conocido

Si sabes los archivos, docs o skill, nómbralos. Cada bucle de descubrimiento ahorrado es token que no pagas.

INVESTIGAR, PLANEAR, IMPLEMENTAR

Trabaja en fases, con una ventana nueva entre ellas.

INVESTIGACIÓN	Carga muchos archivos, la mayoría irrelevantes para implementar. Hazlo en su propia ventana o un subagente.	Opcional
PLAN	Un modelo de razonamiento produce una spec precisa, un to-do detallado que hace el pensamiento antes.	Una vez
IMPLEMENTAR	Ejecuta la spec en un modelo más barato con solo el contexto relevante. Con spec clara, agentes en paralelo se vuelven posibles.	Por paso

CONTROLES DETERMINISTAS

Las pruebas reinician el reloj de la precisión.

01

Las pruebas son deterministas

Pasa o falla, sin probabilidad. Una prueba en rojo detiene al agente que se desvió y fuerza reconstruir sobre base estable.

02

53 por ciento son pruebas

El equipo de GitHub Copilot CLI entrega unos 500 PRs por semana. Su práctica número uno son las pruebas, 53 por ciento del código.

03

Más allá de las pruebas

Linters, verificadores de tipos, escáneres de seguridad. Cualquier guard-rail que expreses como código, el agente lo ejecuta.

CONFIGS CON ALCANCE Y REUSABLES

Carga la guía solo cuando es relevante.

01

Instrucciones con alcance

Globs de ruta applyTo cargan solo cuando se adjuntan archivos que coinciden, manteniendo el always-on pequeño.

02

Prompt files

Prompts versionados, invocados a mano. Acaba con la deriva de que cada dev tenga su prompt favorito.

03

Chat modes

Una allowlist estrecha de tools literalmente no entra en bucle de Agent, así que la cuenta queda acotada por diseño.

SKILLS FRENTE A MCP

Ofrece comportamiento bajo demanda, no lo anuncies cada turno.

SKILLS · CARGA PROGRESIVA

1 a 2 KB hasta que una coincide

Una skill expone solo el nombre y la descripción hasta que tu prompt coincide. Instala 20 skills y sumas un par de KB, no los cuerpos completos. Estándar abierto en VS Code, CLI y cloud agent.

MCP · SIEMPRE ANUNCIADO

10 a 15 KB cada turno

Cada tool de MCP habilitada envía su esquema cada turno, se use o no. Acota por workspace, recorta a lo que usas en la semana y prefiere un CLI como gh para lectura pesada.

INSTRUCCIONES PERSISTENTES

Mantenlo pequeño, escríbelo tú, recréalo cada trimestre.

MANTENER

Reglas que el modelo no infiere

Riesgos no obvios, el framework de pruebas y el comando de build, reglas explícitas de no tocar archivos y patrones. Cerca de 150 tokens, pagados una vez y luego cacheados.

CORTAR

Todo lo que el código ya dice

Textos de onboarding, arquitectura en ASCII, guías de estilo completas, reglas duplicadas. Investigación de 2026: los archivos de contexto generados suman 20 a 23 por ciento de tokens por cerca de menos 2 por ciento de corrección.

INSTRUCCIONES COMPRIMIDAS

Las mismas reglas, cerca de 64 por ciento menos tokens.

ANTES, ~120 TOKENS

Prosa

Siempre usa pnpm, nunca npm. Las pruebas viven en spec/. No modifiques archivos en generated/. Documenta exports con jsdoc, y así.

DESPUÉS, ~50 TOKENS

Clave-valor escueto

pkg: pnpm. tests: spec/. no-edit: generated/. docs: jsdoc on exports. Mismo sentido, pagado cada turno, 64 por ciento más liviano.

HIGIENE DE MCP

Por workspace gana a global, poda la lista de tools.

ACOTA POR WORKSPACE

.vscode/mcp.json

Vive en git, revisable, acotado al repo que lo necesita. Un servidor global malo contamina todo proyecto.

PODA LO QUE SE ENVÍA

Cada tool manda su esquema

Un conjunto de 40 tools anuncia 10 a 15 KB por turno, se usen o no. Poda a lo semanal y prefiere el CLI gh para lectura pesada.

AGENTRC

Deja de escribir la pila de instrucciones a mano. Genérala.

01

Readiness

Puntúa el repo en 9 pilares y un modelo de madurez de 5 niveles. Úsalo como gate de CI que falla por debajo del nivel 3.

02

Instructions

Genera copilot-instructions.md, la config de MCP, settings y AGENTS.md a partir del propio código.

03

Eval

Re-corre casos guardados para detectar el deterioro de instrucciones en CI. Experimental, así que fija una versión.

ROLES Y AISLAMIENTO

Fija el rol, aísla el descubrimiento.

AGENTES PERSONALIZADOS

Evita caminos erróneos

Fija modelo y herramientas por rol: Planner, Implementer, Reviewer. La ganancia real no son tokens, es no darle al agente una herramienta que no quieres que use.

SUBAGENTES

Paga el descubrimiento una vez

Un subagente lee muchos archivos en contexto aislado y devuelve solo un resumen. El padre queda liviano en una sesión larga, a menudo cerca de 10 veces menos entrada re-cobrada.

TERRITORIO DE POWER USER

Tácticas avanzadas, para cuando orquestas muchos agentes.

01

Pensar en código

Escribe un script para filtrar la salida antes de que el modelo la vea. Filtra una respuesta de API a los campos que importan, no vuelques el payload entero.

02

CLI en vez de MCP

Para lectura pesada, un CLI conocido como gh es más liviano que la superficie de tools de un servidor MCP. Menos tokens estáticos, misma respuesta.

03

Chronicle y RTK

Corre chronicle para analizar tus logs de sesión en busca de mejoras, y herramientas como RTK para recortar salidas largas de shell a lo que el agente necesita.

LOS OTROS CONFIGS

Guía con alcance, condicional y automática.

01

Instrucciones con alcance

Condicionales, por ruta de archivo. Ofrecidas al agente como skills. Empieza estático, pasa a alcance solo cuando el archivo crece.

02

Prompt files

Prompts reusables invocados a mano. Un punto de partida común para lanzar skills o custom agents.

03

GitHub Copilot memory

Guía pequeña, automática, always-on, aprendida de tu comportamiento, aplicada entre superficies. Revísala de vez en cuando.

GOBERNANZA · EL POOL COMPARTIDO

Cada licencia aporta créditos a un solo pool.

ANTES · POR ASIENTO

Saldos atrapados

Las requests no usadas de un usuario ligero no ayudaban a uno intensivo, que llegaba a overage mientras otros sobraban.

DESPUÉS · POOL COMPARTIDO

Los créditos fluyen a la necesidad

Todos toman de un pool primero. Solo el gasto más allá del pool es gasto adicional, gobernado por las capas de presupuesto. Los presupuestos por usuario limitan a cada persona.

POOL EN LA PRÁCTICA

El uso incluido fluye por la enterprise.

MODELO PRU

Saldos atrapados

Las requests no usadas de un usuario ligero no ayudaban a uno intensivo, que llegaba a overage mientras otros sobraban. Uno no drenaba al otro.

POOL COMPARTIDO

Menos desperdicio, uso desigual

Los usuarios toman de un pool por necesidad real, así el valor no usado no queda atrapado. El trade-off, consumo desigual, por eso existen los presupuestos por usuario.

LAS CUATRO CAPAS DE PRESUPUESTO

Guard-rails, no esposas. Cualquier presupuesto en cero detiene el uso.

04

CAPA · TECHO GLOBAL

Presupuesto enterprise

Limita el gasto adicional total más allá del pool. Alertas en 75, 90 y 100 por ciento. Los cost centers pueden excluirse para que una unidad financiada siga trabajando.

03

CAPA · POR UNIDAD

Presupuesto de cost center

Asigna gasto adicional a una org o grupo de usuarios. Puede mapear una Azure subscription por cost center.

02

CAPA · DEFAULT JUSTO

Presupuesto universal por usuario

Un techo por defecto sobre el consumo total por usuario. La forma fácil de evitar que una persona drene el pool.

01

CAPA · OVERRIDE DE POWER USER

Presupuesto individual por usuario

Sobrescribe el default universal para usuarios específicos. Máximo de 10.000 presupuestos en la enterprise, así que usa overrides con mesura.

EL PEDIDO DEL DÍA CERO

150%

Pon el presupuesto por usuario en 150 por ciento, y dos movimientos más.

150 por ciento es lo bastante alto para nunca bloquear el trabajo normal y lo bastante bajo para acotar un bucle descontrolado a unas 12 horas. Un presupuesto en cero bloquea el acceso por completo, no hay fallback a un modelo más barato.

TRES MOVIMIENTOS, HOY

ULB 150% · Auto por defecto · archivo de instrucciones → RESULTADO cerca de 1,6 a 1,8x de throughput, misma licencia

ESCENARIOS DE PRESUPUESTO

Tres formas que toman las cuatro capas en la práctica.

01

Gestionar uso compartido

Un presupuesto universal por usuario fija el default, los overrides individuales elevan a power users específicos. Control base con flexibilidad por persona.

02

Por unidad de negocio

Presupuestos de cost center por org, con el presupuesto enterprise como techo global y failsafe para quien está fuera de un cost center.

03

Power users en una unidad

Las cuatro capas juntas: default universal, overrides individuales dentro de un cost center, el presupuesto de la unidad y el techo enterprise, opcionalmente excluyendo la unidad.

CREANDO LOS GUARD-RAILS CORRECTOS

Cuatro preguntas antes de fijar cualquier presupuesto.

01

Presupuesto enterprise

¿Cuánto está dispuesto el negocio a gastar en servicios de IA?

02

Presupuesto universal por usuario

¿Cuánto debería poder gastar cada ingeniero?

03

Presupuesto de cost center

¿Cuál es el máximo que cada unidad o equipo puede gastar?

04

Override individual

¿Quién necesita una excepción a esos presupuestos?

NOTIFICACIONES DE PRESUPUESTO

Alertas antes del muro, y la parada dura en él.

75%

El admin recibe aviso

Al 75 por ciento, llega un correo. Empieza a monitorear de cerca.

90%

Decide al 90

Sube el presupuesto, o deja que tope. Elección consciente, no sorpresa.

100%

Bloqueado al 100

Los usuarios paran hasta el próximo ciclo o un admin sube el techo, lo que toma segundos.

CONTENT EXCLUSION, LA BASE DEL ADMIN

Bloquea archivos del contexto antes de cualquier ajuste.

01

Secretos y env

.env, .env.*, secrets/*, *.pem, *.key. Nunca dejes que esto fluya al contexto.

02

Generado y vendored

dist/, build/, node_modules/, vendor/. Ruido que el modelo nunca debería leer.

03

Datos sensibles

compliance/, customer-data/. Globs por repo, org o enterprise, propagan en unos 30 minutos.

POLÍTICA DE MODELO Y AUTO

Haz de la elección eficiente el default.

AUTO POR DEFECTO

Una ganancia gratis de 10 por ciento

Auto elige un modelo apropiado y gana cerca de 10 por ciento de descuento para pagos. Hazlo el default de la org.

RESTRINGE PREMIUM

Aprueba en etapas

Limita modelos premium para trabajo de rutina en Organization Settings, GitHub Copilot, Policies. Aprueba por workflow, equipo y necesidad medible, no por honor.

OBSERVABILIDAD

No optimizas lo que no puedes ver.

04
CAPA · POR TURNO
OTel más badge de modelo
VS Code 1.119 emite spans OpenTelemetry con la composición de tokens y cache por turno, más el modelo resuelto inline. Gratis, local, hoy.

03
CAPA · POR USUARIO
/usage y /context
Señales de costo en sesión y al final, antes de la factura. Corre /usage al final de tus próximas tres sesiones para conocer tu baseline.

02
CAPA · POR ORG
GitHub Copilot Metrics API
Un dataset publicado, ahora con la actividad del CLI, exportado a CSV para análisis de tendencia y re-baseline mensual.

01
CAPA · DASHBOARDS
Viewer y Grafana
El Metrics Viewer open-source y los dashboards Grafana de Microsoft convierten el desperdicio en un gráfico compartible. Sigue la tasa de cache por encima del 60 por ciento.

TELEMETRÍA DE VS CODE 1.119

Tres ganchos que vuelven el costo en dato.

01

Tracing OpenTelemetry

Spans GenAI con composición de tokens y cache por turno, exportados a cualquier backend OTLP.

02

Badge de modelo y multiplicador

Muestra el modelo que Auto resolvió y su multiplicador inline. Activo por defecto.

03

Background todo agent

Pasa la lista de tareas a un modelo ligero para que el principal no pague por ello. Apagado por defecto, actívalo.

MIDE EL DELTA TÚ MISMO

Corre la misma tarea de dos formas y observa la cuenta.

FLUJO A · NAIVE

Agente, sin alcance

Abre el modo Agent, apúntalo a hash-codebase, itera 8 a 12 turnos en un solo chat, acepta lo que salga. El caso de control.

FLUJO B · INGENIERADO

Planea, luego acota

Plan redacta una spec, la revisas, la entregas a Agent con alcance hash-file estrecho, las instrucciones del repo garantizan convenciones. Mide tokens OTel, conteo de turnos, tiempo a mergeable.

NÚMEROS A VIGILAR

Las señales de estado estable que mantienen honesto al programa.

01

Acierto de cache sobre 60 por ciento

Confirma que el prefijo de instrucciones está estable. Una tasa cayendo significa que alguien edita archivos del equipo a media jornada y pierde el descuento.

02

Overage de 10 a 15 por ciento

Bajando del 30 por ciento o más en baselines descontroladas. Revisa promedios mensuales, no picos diarios.

03

Revisión semanal del top-10

Mira las diez sesiones más pesadas por semana. Revelan un patrón corregible: un MCP siempre activo, una sesión nunca limpiada, un modelo premium fijado para triaje.

EL PLAYBOOK DE 30/60/90 DÍAS

Hazlos en orden. Cada fase abarata la siguiente.

ESTABILIZA	Instrucciones en los repos principales, content exclusion a nivel de org, Auto por defecto, OTel conectado para equipos de alto volumen.	Primeros 30 días
ESTANDARIZA	Genera la pila de instrucciones con AgentRC, acota MCP por repo, convierte prompts en skills, levanta el export de métricas y la alerta de 75 por ciento.	Días 31 a 60
ESCALA Y GOBIERNA	Abre issues listas para el coding agent, agrega un gate de readiness de AgentRC en CI, re-tier de presupuestos por trimestre con telemetría real.	Días 61 a 90

EL CHECKLIST DE 30 MINUTOS DEL LUNES

Seis ítems, cada uno bajo cinco minutos, en un repo real.

01

Agrega copilot-instructions.md

Cinco líneas, solo convenciones, estilo comprimido.

02

Agrega una instrucción con alcance

Tu directorio de mayor tráfico, por ejemplo src/api/.

03

Mueve MCP a .vscode/mcp.json

Por workspace gana a global, poda a lo semanal.

04

Usa una skill para tarea recurrente

PR review, triaje, chequeo de regresión.

05

Crea un GitHub Copilot Space

Tu pregunta transversal número uno, tres fuentes máximo.

06

Abre una issue para el coding agent

Alcance acotado, punteros de archivo, criterios de aceptación.

EL ROADMAP DE TRES FASES

Pre-habilitación, optimización, gobernanza.

PRE-HABILITACIÓN	Cerca de dos horas. Pon ULB 150 por ciento, configura el techo de overage con alerta al 80 por ciento, define cost centers, activa Auto en la org, commitea un archivo de instrucciones inicial.	Día 0
OPTIMIZACIÓN	Entrena disciplina de contexto y hash-mentions, audita MCP por equipo, pon el esfuerzo en medio, distribuye instrucciones por ruta, empieza la revisión semanal del top-10.	Semanas 1-4
GOBERNANZA	Exporta el CSV de uso, revisa promedios mensuales, ajusta la ULB para power users justificados, revisa la tasa de cache sobre 60 por ciento, refina instrucciones y toolsets.	Mensual

LA TRANSICIÓN DE SEIS MESES

Diagnostica, rescata, ancla en los puntos de decisión.

MES 1 · JULIO	¿Los presupuestos y políticas siguen siendo adecuados? Confirma o ajusta temprano, mientras el colchón está activo.	Confirmar
MES 3 · SEPTIEMBRE	¿Cuál es el presupuesto realista en la asignación estándar? Re-baseline y comunica. El checkpoint más importante.	Re-baseline
MES 6 · DICIEMBRE	¿Sigue siendo óptimo el tier? ¿Hay palancas de renegociación? Renueva, cambia de tier o renegocia.	Decidir

SEIS DIMENSIONES DE MADUREZ, PESADAS POR PALANCA DE COSTO

Diagnostica antes de prescribir. La gobernanza de modelo lidera.

25

Gobernanza de modelo

Peso 25. La mayor palanca aislada de costo, el modelo correcto por tarea.

20

Fundación de plataforma

Peso 20. La raíz del descontrol cuando está ausente.

15

Disciplina de contexto

Peso 15. Ataca el factor tokens directamente.

15

Alcance de agente

Peso 15. Controla el gasto descontrolado.

15

Primitivos de repositorio

Peso 15. El comienzo de la disciplina estructural.

10

Control de presupuesto

Peso 10. El guard-rail contra cuentas de horror.

DOS HORIZONTES

El rescate cabe en la ventana, la transformación realiza el valor.

RESCATE, JUN A DIC

Cuatro frentes

Gobernanza de modelo, presupuestos como guard-rail, curaduría de contexto, primitivos de repositorio. Todos actúan sobre la ecuación modelo por tokens.

TRANSFORMACIÓN

Plataforma por diseño

Una plataforma sedimentada, una capa de contexto y orquestación, un centro de control en Microsoft Foundry. La eficiencia se vuelve propiedad de la infraestructura, no de la disciplina individual.

EL MODELO DE ADVISORY

Roles claros, cadencia quincenal.

01

El GBB es el cerebro

Estrategia, diagnóstico, gobernanza, arquitectura, el qué y el porqué.

02

Los partners son los brazos

Integran y validan en el entorno del cliente, el cómo.

03

El cliente es dueño del resultado

Datos, participación, implementación. Un checkpoint cada 15 días prueba que el compromiso está vivo.

LA CAPA COMERCIAL, PLANES DE PRE-COMPRA

Previsibilidad, dimensionada por banda de contrato.

LOS TRES P3

GitHub, AI Credit, Agent Factory

Pre-pagados, un año, vía Azure subscriptions. Tiers compartidos: 20k = US$ 19k (5%), 100k = US$ 90k (10%), 500k = US$ 425k (15%).

FIT POR BANDA DE ACD

Confirma antes de proponer

Por debajo de 15 por ciento de ACD el P3 gana, 15 a 25 solo AI Credit P3 con Deal Desk, 25 por ciento o más el ACD gana. Valida contra el P3 FAQ.

ANTI-PATRONES A EVITAR

Nombrar las trampas protege la credibilidad.

01

Cuenta alta es incompetencia

Falso. La cuenta es espejo de la madurez de plataforma, no veredicto. Cargas pesadas pueden ser legítimas.

02

Presupuestar contra gasto promo

El número cambia en septiembre. Siempre presupuesta contra la asignación real.

03

Prometer la extensión como hecho

Es condicional. Y nunca midas productividad individual, mide entrega organizacional.

ÚLTIMAS NOTICIAS Y ROADMAP

Cada vez más de esto se vuelve automático y aplicable por política.

01

Auto y Hydra

Auto enruta al modelo capaz más barato con descuento para pagos. Hydra, un pequeño clasificador de intención, elige el ajuste sin costo extra y cambia solo en fronteras de cache.

02

Coding agent y Spaces

El coding agent está GA, los Spaces reemplazaron a los Knowledge Bases, las agent skills son estándar abierto y los dashboards Grafana cubren GitHub Copilot, Claude Code y Codex.

03

Lo que viene

Enrutamiento consciente del token, tiers de Auto (Eco, Fast, Balanced, Max) y políticas de admin para imponer Auto. El catálogo de modelos rota siempre, trata las tarifas como fotos del momento.

CÓMO FUNCIONA AUTO

Dos sistemas eligen el modelo por ti.

HYDRA

Clasificador de intención

Un pequeno clasificador ModernBERT puntúa la complejidad del prompt en razonamiento, debugging, generación de código y tools, y elige el más barato que sirve. Sin costo extra.

SELECCIÓN DINÁMICA

Enrutamiento en tiempo real

Considera capacidad, latencia y errores. Auto cambia de modelo solo en fronteras de cache, el inicio de la conversación y tras compactación, para nunca romper tu cache.

CUATRO PALANCAS DE CONFIANZA DE COSTO

No es una feature, es un roadmap que funciona junto.

01

Enrutar tareas mejor

Task intent empareja el trabajo a un camino de modelo, sin que el usuario pese trade-offs.

02

Gobernar por equipo

Enterprise Teams y cost centers asignan presupuestos donde ocurre el uso.

03

Comunicar el roadmap

Confiado, pero cauto en tiempos y enrutamiento futuro.

04

El resultado

Más uso, menos sorpresas, mejores controles.

HITOS RECIENTES

Por dónde se ha movido la plataforma, para anclar la línea de tiempo.

01

MAR 2025

Índice semántico GA

El índice instantáneo de búsqueda semántica de código pasó a GA, haciendo hash-codebase rápido y preciso en repos alojados en GitHub.

02

SEP 2025

Coding agent GA, Spaces

El coding agent asíncrono llegó a GA, y los GitHub Copilot Spaces reemplazaron a los Knowledge Bases.

03

ENE 2026

CLI mejorado

El GitHub Copilot CLI ganó agentes más ricos y gestión de contexto, e integró la actividad del CLI en la Metrics API de uso.

04

MAY 2026

VS Code 1.119

Tracing OpenTelemetry, el badge de modelo y multiplicador, y el background todo agent llegaron, con la UI de UBB pre-montada.

EL ROADMAP HACIA ADELANTE

Cada vez más de lo que optimizas a mano se vuelve automático.

JUNIO	Auto llega a más clientes y se vuelve la única opción para Free y EDU, con UX para explicar qué modelo se eligió y optimizaciones de cache.	Ahora
JULIO EN ADELANTE	Enrutamiento consciente del token y tiers de Auto: Eco, Fast, Balanced, Max. Una sola perilla para tu postura de costo frente a capacidad.	Sigue
DIRECCIÓN	Políticas de admin para imponer Auto y fijar su versión. La elección eficiente se vuelve el default aplicable. Trata las tarifas de modelo como fotos del momento.	Adelante

ANEXO

VII

Cache.

Dos caches que la gente confunde. Uno lo influyes, el otro lo alojas. Y si Foundry más Redis cabe detrás de VS Code y el CLI.

DOS CACHES, UNA PALABRA

El prompt cache reúsa la misma entrada. El cache semántico reúsa una respuesta similar.

CACHE A · PROMPT CACHE DE GITHUB COPILOT

Lo influyes

El proveedor reproduce un prefijo estable entre turnos, los cached tokens de tu cuenta, cerca del 90 por ciento off. VS Code llegó a más del 93 por ciento de reuso por request. No hay perilla, mantienes el prefijo estable.

CACHE B · CACHE SEMÁNTICO

Lo alojas

Devuelve una respuesta guardada cuando un prompt nuevo es semánticamente similar a uno previo. Lo construyes para tus propias apps y agentes en Azure, no baja la cuenta de tu suscripción de GitHub Copilot.

ADMINISTRAR EL CACHE A

En VS Code y el CLI, proteges el cache, no lo configuras.

01

Mantenlo caliente

Mantén los archivos de instrucción y el conjunto de MCP estables en la sesión, agrega reglas al final y deja que Auto cambie de modelo en fronteras de cache.

02

Qué lo rompe

Editar instrucciones a media jornada, cambiar de modelo a mano o cambiar el conjunto de herramientas reescribe el prefijo y lo re-pagas a precio completo de entrada.

03

Mídelo

El OTel del 1.119 muestra cache read y creation por turno, el /context muestra la composición. Apunta a una tasa de acierto de cache por encima del 60 por ciento.

CACHE B, EL CACHE SEMÁNTICO QUE ALOJAS

Para tus propios agentes en Azure, no para GitHub Copilot.

01

Azure Managed Redis

Con el módulo RediSearch, el cache externo y vector store. Habilita RediSearch en la creación.

02

Gateway API Management

El GenAI gateway frente a tus modelos, también disponible integrado desde Foundry.

03

Políticas lookup y store

llm-semantic-cache-lookup y store comparan prompts por proximidad vectorial, devolviendo respuestas similares.

GUÍA DE DECISIÓN

Cuál cache, cuándo.

01

En VS Code o el CLI

Estás en el Cache A, el prompt cache. Protégelo con disciplina de prefijo. Nada que desplegar.

02

Construyendo tu propio agente

Usa el Cache B, el cache semántico, en Foundry más Redis. Una palanca de costo real y gobernable.

03

Los dos a la vez

Común en plataformas maduras. Coexisten y nunca se solapan, capas totalmente distintas.

FOUNDRY MÁS AZURE MANAGED REDIS

Capa correcta para tus agentes, no para GitHub Copilot en el IDE.

GITHUB COPILOT EN VS CODE Y CLI

No

GitHub Copilot es un producto gestionado. No insertas tu Redis o gateway Foundry en su cache ni en su billing. La única palanca es la disciplina de prefijo.

TUS AGENTES EN FOUNDRY

Sí

API Management como GenAI gateway con las políticas de cache semántico más Azure Managed Redis y RediSearch recorta tokens en los modelos que llamas directo. La plataforma del horizonte de transformación.

CONCLUSIONES

Tres pilares, un modelo mental.

01

El costo es contexto

En cada superficie, lo que envías al chat es lo que pagas. Poda, acota, reinicia.

02

Específico gana a amplio

hash-file mejor que hash-codebase, issue estrecha mejor que vaga, Space enfocado mejor que de todo un poco.

03

Persistencia sobre prompts

Instrucciones, chat modes y Spaces en git sobreviven a cualquier prompt único.

GLOSARIO, PARTE 1

Los términos que atraviesan el deck.

AI Credit

La moneda del usage-based billing. 1 crédito = US$ 0,01, medido por token.

Token

Un fragmento de palabra, cerca de tres cuartos de una. Entrada, salida o cache.

Efecto de reenvío

El harness reenvía la conversación entera cada turno, así el contexto se acumula.

Ventana de contexto

Lo que el modelo sostiene. Distinto de lo que pones en ella, que es lo que pagas.

Prompt cache

Replay del proveedor de un prefijo estable, cerca del 90 por ciento off. Los cached tokens de la cuenta.

Context rot

La calidad cae al crecer la entrada, a menudo antes de llenar la ventana.

GLOSARIO, PARTE 2

Los configs y controles.

Auto y Hydra

Auto enruta al modelo capaz más barato, con descuento para pagos. Hydra es su clasificador de intención.

Coding agent

De issue a branch a PR en un sandbox. GA desde septiembre de 2025.

Skill

Una capacidad portátil, bajo demanda. Carga solo cuando su descripción coincide.

MCP

Tools externas para agentes. Cada una anuncia su esquema cada turno.

ULB

Presupuesto por usuario, un techo sobre el consumo total. Recomendado en 150 por ciento.

Content exclusion

Control de admin que bloquea archivos del contexto en todas las superficies.

REFERENCIAS

Valida las cifras contra fuentes primarias.

01

GitHub

El post de usage-based billing, el rate card de modelos y precios, docs de budgets y Auto, los changelogs del coding agent y Spaces.

02

Microsoft Learn

Notas del VS Code 1.119, cache semántico y AI gateway en API Management, Azure Managed Redis, Grafana para agentes de IA.

03

Investigación

Liu et al. Lost in the Middle, Chroma Context Rot, el reporte DORA 2025, la encuesta FinOps 2026.

CIERRE

En vez de contar tokens, haz que cada token cuente.

Pioneering software development with AI and Agentic DevOps.

Contacto

Paula Silva
Software Global Black Belt
paulasilva@microsoft.com

Próximo paso

Corre /usage al final de tus próximas tres sesiones para conocer tu baseline, pon el presupuesto por usuario en 150 por ciento, activa Auto por defecto y commitea un archivo de instrucciones pequeño en tu repo más activo.