OTIMIZAÇÃO DE CUSTO · USAGE-BASED BILLING · GITHUB COPILOT

Otimização de custo no GitHub Copilot, referência técnica.

A referência técnica das sete alavancas de otimização sob o Usage-Based Billing: a equação de custo, a base de evidência por banda, a arquitetura das alavancas, a referência de primitivos e budgets, os procedimentos passo a passo, e o troubleshooting dos anti-padrões. Para o passo a passo extensivo com links oficiais, use o runbook companheiro.

V4.1.0 · RELEASE

AUDIÊNCIA: DESENVOLVEDORES, ADMINS, ARQUITETOS
ATUALIZADO: 2026-06-10
STATUS: RELEASE

Paula Silva
SOFTWARE GLOBAL BLACK BELT
PAULASILVA@MICROSOFT.COM

VISÃO GERAL DA DOCUMENTAÇÃO

A referência técnica da otimização de custo sob AI Credits, em sete capítulos.

O Quickstart entrega o primeiro corte em uma hora. Conceitos dá o modelo mental e a base de evidência de cada banda. Arquitetura organiza as alavancas em camadas. Referência é a tabela de consulta de primitivos, budgets e aplicabilidade por plano. Procedimentos, Troubleshooting e Glossário sustentam a operação do dia 2.

PROJETO	Otimização GitHub Copilot UBB
VERSÃO	V4.1.0 · RELEASE
AUDIÊNCIA	Desenvolvedores, Admins, Arquitetos
ATUALIZADO	2026-06-10
MANTENEDORA	Paula Silva, Software Global Black Belt
COMPANHEIROS	Runbook v4.1.0, Playbook v4.1.0, Guia editorial

CONTEÚDO

COMO ESTE DOCUMENTO SE ORGANIZA

Cada capítulo abre com um header de seção e segue com o corpo. Blocos de código usam mono escuro; código inline é cinza claro. Admonitions (note, tip, warning, danger) ficam inline com o texto.

BANDAS, NÃO PROMESSAS

Todo impacto neste documento é banda de planejamento com fonte declarada: preço oficial, pesquisa publicada, ou estimativa direcional rotulada. As bandas se compõem, não se somam.

CAPÍTULO 01 QUICKSTART

O primeiro corte em uma hora.

Cinco passos que carregam a maior parte do resultado: medir, cortar a saída, padronizar o modelo e pôr o guard-rail. O detalhe de cada um está em Procedimentos.

01 Instale e entre

VS Code com as extensões GitHub Copilot e GitHub Copilot Chat, login na conta com o plano esperado. Verifique que o texto fantasma e o painel de Chat funcionam.

02 Ligue a medição

Admin: habilite a política `Copilot usage metrics` e capture a linha de base na aba Insights (janela de 28 dias, export NDJSON e CSV).

03 Corte a saída

Crie o arquivo de instruções com a diretiva de saída direta e faça commit.

MARKDOWN .GITHUB/COPILOT-INSTRUCTIONS.MD

```
- Responda com código ou diff.
- Sem preâmbulo, sem resumo.
- Menor mudança correta; só as linhas alteradas.
```

04 Padronize o Auto

Model picker em **Auto** (10% de desconto no chat para planos pagos) e desligue `Enhance non-chat requests with premium models` (Visual Studio: Tools > Options > GitHub > Copilot > Editor); no VS Code, utility model em modelo leve.

05 Ponha o guard-rail

Admin: em `Settings > Billing > Budgets and alerts`, crie o budget de usuário universal acima do valor por licença, com alertas em 75, 90 e 100%.

★ TIP

Os passos 3 e 4 são as duas maiores alavancas (saída custa ~5x o input por token; o preço entre classes de modelo varia em duas ordens de grandeza). Faça os dois antes de qualquer refinamento.

CAPÍTULO 02 CONCEITOS

A equação de custo e os três tipos de token.

Sob o Usage-Based Billing, o consumo é medido em AI Credits, onde **1 AI Credit = US$ 0,01**. O custo de cada interação depende de dois fatores, e só dois: **qual modelo** e **quantos tokens**. Toda alavanca deste documento ataca um desses fatores, ou os dois.

Os três tipos de token

Tokens de **entrada** (prompt, histórico, arquivos, instruções), tokens de **saída** (o que o modelo gera, tipicamente ~4 a 5x o preço do input) e tokens de **cache** (contexto reutilizado, a 10 a 50% do token novo conforme o modelo: Anthropic 0,1x, OpenAI 50% automático acima de 1.024 tokens de prefixo).

> ✳ IMPORTANT
>
> Code completions e Next Edit Suggestions seguem incluídos no plano e não consomem AI Credits. O consumo real está em chat e nas tarefas agênticas; otimizar completions é esforço perdido.

Por que as bandas não se somam

Cada alavanca age sobre uma base diferente de tokens: output control age sobre a saída, contexto e cache sobre a entrada, routing sobre a conta inteira. Os ganhos se compõem multiplicativamente. Um programa maduro tende à faixa de 55 a 70%; um começo enxuto, de 20 a 30%. Bandas diretivas, a confirmar contra a sua linha de base.

> A ORDEM IMPORTA
>
> Saída é a classe de token mais cara, então output control é o primeiro movimento. Compressão de contexto é o segundo. Cache é o terceiro. Routing é a decisão arquitetural que limita o raio de dano de todo o resto.

CAPÍTULO 02 BASE DE EVIDÊNCIA

Cada banda com a sua fonte.

Três tipos de fonte sustentam as bandas: **preço oficial** dos provedores (o mais duro), **pesquisa publicada** e revisada, e **estimativa direcional de campo**, sempre rotulada como tal.

PARAMETER	TYPE	DEFAULT	DESCRIPTION
Output control OPTIONAL	40 a 70% da saída	-	Razão de preço saída/entrada de ~4 a 5x nas tabelas públicas dos provedores e na tabela por modelo do GitHub. A escada de especificação é estimativa direcional de campo.
Model routing OPTIONAL	40 a 70% da conta	-	FrugalGPT (Stanford, TMLR): cascata iguala o melhor modelo com economia de 50 a 98%. RouterBench (2024). A banda usada é conservadora.
Modelos locais OPTIONAL	um dígito a ~15%	-	Mecânica oficial: BYOK não consome AI Credits (GitHub Changelog, 2026). A participação no custo é estimativa direcional, dependente do mix.
Escopo de contexto OPTIONAL	40 a 80% do input	-	LLMLingua (Microsoft, EMNLP 2023): compressão de até 20x com perda de ~1,5 ponto. Survey de context engineering (2025).
Cache OPTIONAL	30 a 50% do input	-	Preço oficial: leitura de cache a 0,1x na Anthropic e 50% na OpenAI. Banda de loops de agente da prática documentada.
Combinado OPTIONAL	20-30% / 55-70%	-	Aritmética de composição sobre as bandas acima; compatível com o teto da literatura (FrugalGPT, 50 a 98%). Banda diretiva.

> ⚠ WARNING
>
> A regra de honestidade: onde a fonte é preço oficial, o número é fato. Onde é pesquisa, vale o cenário do estudo e a banda aqui é deliberadamente mais conservadora. Onde é estimativa direcional, o texto diz isso. O número que vale é o seu, contra a linha de base.

Fontes primárias: FrugalGPT `arxiv.org/abs/2305.05176` · LLMLingua `arxiv.org/abs/2310.05736` · RouterBench `arxiv.org/abs/2403.12031` · Anthropic e OpenAI prompt caching (docs oficiais) · GitHub AI model comparison.

CAPÍTULO 03 ARQUITETURA

As sete alavancas em quatro camadas.

Leia de cima para baixo, que é também a ordem de adoção. A camada de Saída é onde está o desperdício mais caro; a de Governança é o que torna o ganho permanente e independente de disciplina individual.

Responsabilidades por camada

Saída controla o formato da resposta. **Conta** roteia cada tarefa ao modelo suficiente mais barato, com o local como ponta zero. **Entrada** cura o contexto, cacheia o estável e lembra o aprendido. **Governança** versiona as regras nos primitivos e limita a variância com budgets.

> DEV E ADMIN
>
> Saída e Entrada são do desenvolvedor. Conta é compartilhada (picker do dev, política do admin). Governança é do admin. O alinhamento de papéis está nos Procedimentos.

CAPÍTULO 04 REFERÊNCIA

Primitivos de repositório.

Os primitivos transformam configuração de IA em infraestrutura versionada, revisada como qualquer mudança. Cada um é governança e economia ao mesmo tempo.

```
.github/
├── copilot-instructions.md       # contexto geral, sempre ligado
├── instructions/
│   └── tests.instructions.md     # regras por área (applyTo)
├── agents/
│   └── cost-aware-reviewer.agent.md
└── prompts/
    └── release-notes.prompt.md   # encapsula instrução longa
```

PARAMETER	TYPE	DEFAULT	DESCRIPTION
copilot-instructions.md REQUIRED	repositório	-	Contexto geral injetado em todo pedido. Primeiro e mais importante. Terse, alto sinal, sem conflitos.
***.instructions.md** OPTIONAL	por área	-	Regras com escopo via campo applyTo (glob). Cura o contexto por área do código.
***.agent.md** OPTIONAL	por tarefa	-	Um agente, uma tarefa, com modelo e ferramentas restritos. Escopo e governança de modelo no mesmo arquivo.
***.prompt.md** OPTIONAL	reutilizável	-	Encapsula instrução longa em comando versionado; elimina o prompt mágico decorado.
hooks/ OPTIONAL	guard-rail	-	Impede loops e ações desnecessárias do agente.

★ TIP

No VS Code, gere os arquivos com `/create-instruction`, `/create-agent`, `/create-prompt`, `/create-hook`.

CAPÍTULO 04 REFERÊNCIA

Budgets e aplicabilidade por plano.

Os quatro níveis de budget

PARAMETER	TYPE	DEFAULT	DESCRIPTION
User-level budget (ULB) REQUIRED	por usuário	-	Hard stop por ciclo; o controle mais importante. Defina o universal acima do valor por licença (US$ 19 Business, US$ 39 Enterprise) e overrides para os pesados.
Cost-center budget OPTIONAL	grupo	-	Limita o metered de uma unidade de negócio após o pool; aloca custo por unidade.
Enterprise spending limit OPTIONAL	enterprise	-	Failsafe global: limita o metered total após o pool.
Organization budget OPTIONAL	organização	-	Acompanha o gasto da organização ou repositório.

Aplicabilidade por plano

PARAMETER	TYPE	DEFAULT	DESCRIPTION
Content exclusion OPTIONAL	Business / Enterprise	-	Só nesses planos, via settings; não vale em Edit e Agent mode, e bloqueia completions nos arquivos afetados.
BYOK / modelos locais OPTIONAL	todos, governado	-	Política Bring Your Own Language Model Key in VS Code (Business e Enterprise, ligada por padrão); não vale para completions.
Budgets de admin OPTIONAL	Business / Enterprise	-	ULB, cost centers e dashboard exigem papel de owner ou billing manager. No plano individual, budget pessoal de gasto adicional.

⚠ WARNING

Allowances desde 2026-06-01: Pro 1.500 créditos, Pro+ 7.000, Copilot Max (US$ 100) 20.000; Business 1.900 e Enterprise 3.900 por usuário, com promoção de 3.000 e 7.000 até 2026-09-01. Use a promoção para achar a linha de base; não a leia como o regime permanente. Budgets por usuário em GA desde 2026-06-01.

CAPÍTULO 05 PROCEDIMENTOS

R1 Output control e R2 Model routing.

R1 · Output control (Dev)

01 Crie as instruções de repositório

Arquivo `.github/copilot-instructions.md` com saída direta: código ou diff, sem preâmbulo, menor mudança correta.

02 Adicione regras por área

MARKDOWN .GITHUB/INSTRUCTIONS/TESTS.INSTRUCTIONS.MD

```
---
applyTo: "**/*.test.ts"
---
- Gere só o corpo do teste.
- Sem explicar o scaffolding.
```

03 Verifique

No Chat, expanda **References**: o arquivo de instruções deve aparecer como referência aplicada.

R2 · Model routing (Dev + Admin)

01 Auto como padrão

Model picker em Auto; fronteira só para raciocínio difícil, trocada manualmente.

02 Corte o premium silencioso

No Visual Studio (Tools > Options > GitHub > Copilot > Editor): desligue `Enhance non-chat requests with premium models`. No VS Code: utility model em modelo leve.

03 Fixe a política

Admin define quais modelos os membros podem usar; reasoning effort conforme a tarefa. Nota: trocar o modelo do chat não muda o das inline suggestions.

CAPÍTULO 05 PROCEDIMENTOS

R3 Modelos locais e R4 Escopo de contexto.

R3 · Modelos locais (Dev + Admin)

01 Confirme as políticas

`Bring Your Own Language Model Key in VS Code` habilitada (padrão); para certos modelos, Editor Preview Features. VS Code 1.113+, Copilot Chat 0.41.0+.

02 Registre o Ollama

TERMINAL

```
$ ollama pull llama3.1 && ollama launch vscode
# manual: Chat: Manage Language Models → Ollama → http://localhost:11434
```

03 Ou o Foundry Local

Instale via `winget install Microsoft.FoundryLocal` (ou brew), rode `foundry model run phi-4` e, no VS Code, adicione via extensão AI Toolkit (Foundry Local via AI Toolkit). Detalhe no runbook.

04 Roteie a rotina

Selecione Local no rodapé do Chat; commit, boilerplate e testes vão para o local. Agent mode exige modelos com tool calling.

R4 · Escopo de contexto (Dev + Admin)

01 #-mentions no lugar de despejo

Prefira `#file`, `#sym` e `#changes` a jogar o `#codebase` inteiro; `@workspace` é sintaxe antiga.

02 Content exclusion (Admin)

Settings > Copilot > Content exclusion, paths em YAML (`**/.env`, `dist/**`); propaga em até 30 minutos. Atenção: não vale em Edit e Agent mode, e bloqueia completions nos arquivos afetados.

CAPÍTULO 05 PROCEDIMENTOS

R5 Cache e memória, R6 Primitivos, R7 Budgets.

R5 · Cache e memória (Dev)

01 Estabilize o prefixo

copilot-instructions.md e AGENTS.md no topo, sem edição no meio da sessão; thrash quebra o cache (leitura a 10 a 50% do token novo, conforme o modelo).

02 Higiene de sessão

Agrupe o trabalho relacionado na mesma sessão; não rerode agentes sobre diff inalterado.

03 Ligue e cure a memória

Copilot Memory (public preview): revise os fatos e preferências capturados; o que envelhece sai.

R6 · Primitivos (Dev)

Ordem de adoção: instruções gerais, instruções com escopo, agentes customizados, prompt files, hooks. A referência completa, com a árvore de arquivos, está no Capítulo 04.

R7 · Budgets e medição (Admin)

01 ULB universal primeiro

`Settings > Billing > Budgets and alerts` > New budget > Bundled AI credits budget; acima do valor por licença, alertas em 75, 90 e 100%.

02 Overrides e cost centers

Overrides por usuário para os consumidores pesados; cost centers por unidade; enterprise budget como failsafe.

03 Meça e itere

Releia o dashboard a cada ciclo, atribua por usuário, modelo e feature (CSV e NDJSON), e aperte a banda com os dados. Caps são guard-rail de variância: ao bloquear, não há fallback automático e completions seguem funcionando.

CAPÍTULO 06 TROUBLESHOOTING

Anti-padrões: sintoma, causa, solução.

A conta subiu sem mudança visível de uso

Sintoma: créditos crescendo com o mesmo time. Causa comum: requests fora do chat enriquecidos com modelos premium, ou utility model apontado para modelo caro. Solução: desligue o enhance non-chat (GitHub > Copilot > Editor) e aponte o utility model para a classe leve.

Um usuário queimou uma fatia grande do pool

Sintoma: pico de consumo concentrado num usuário. Causa: loop de agente ou sessão agêntica longa de fronteira. Solução: ULB universal como seguro, override individual, loop limits e gates de aprovação; não rerodar agentes em diff inalterado.

> ✖ DANGER
>
> Sem ULB, um único loop acidental pode consumir o pool da unidade. O budget de usuário é um seguro barato; configure antes do uso crescer.

O cache nunca acerta

Sintoma: custo de input alto mesmo com contexto repetido. Causa: prefixo instável (instruções editadas no meio da sessão) ou trabalho fragmentado em sessões curtas. Solução: contexto estável no topo, agrupar o trabalho relacionado na mesma sessão.

Content exclusion parece não funcionar

Sintoma: arquivo excluído ainda aparece em respostas de Edit ou Agent mode. Causa: content exclusion não é suportado nesses modos. Solução: trate exclusão como proteção de chat e completions; para agente, controle o escopo via #-mentions e agentes restritos.

O número combinado não bate com a soma

Sintoma: alguém somou 70% + 80% + 50% e prometeu 200%. Causa: as bandas agem sobre bases diferentes e se compõem multiplicativamente. Solução: use a aritmética de composição (20 a 30% enxuto, 55 a 70% maduro) e confirme contra a linha de base.

CAPÍTULO 07 GLOSSÁRIO

Termos e definições.

AI Credit	Unidade de cobrança do UBB; 1 AI Credit = US$ 0,01.
Auto	Seleção automática de modelo; 10% de desconto no chat para planos pagos.
BYOK	Bring Your Own Key; chave ou modelo próprio (inclusive local) no chat do VS Code; não vale para completions.
Cache de prompt	Reuso do contexto repetido; leitura a 10 a 50% do token novo, conforme o modelo (Anthropic 0,1x, OpenAI 50%).
Completions / NES	Code completions e Next Edit Suggestions; incluídos no plano, não consomem AI Credits.
Content exclusion	Exclusão de arquivos via settings (Business e Enterprise); não vale em Edit e Agent mode.
Cost center	Agrupamento de usuários com budget próprio, para alocar custo por unidade.
Metered	Fase de cobrança por uso após o pool esgotar, a US$ 0,01 por crédito, sujeita aos budgets.
Pool	AI Credits das licenças agrupados na enterprise; light users compensam heavy users.
Primitivo	Arquivo versionado de configuração: instruções, agentes, prompts, hooks, skills.
Tool calling	Capacidade do modelo de chamar ferramentas; necessária para agent mode.
ULB	User-level budget; hard stop por usuário por ciclo, o controle mais importante.

CAPÍTULO 08 METÁFORAS E IMPACTO

O programa em sete imagens, para explicar a qualquer audiência.

R1 · O telegrama e a carta

Pedir resposta sem instrução de saída é encomendar relatório a quem é pago por palavra. A instrução transforma a carta em telegrama: só o que muda. Impacto: 40 a 70% da saída, a classe de token mais cara (~4 a 5x o input).

R2 · O caminhão e o envelope

Usar fronteira para formatar código é contratar caminhão de mudança para entregar um envelope. O roteamento escolhe o veículo pelo tamanho da carga. Impacto: 40 a 70% da conta (FrugalGPT: 50 a 98% no estudo).

R3 · Cozinhar em casa

Nem toda refeição precisa de restaurante: a rotina sai da conta no modelo local, e o orçamento sobra para os jantares de fronteira. Impacto: rotina a custo zero; um dígito a ~15% do total (estimativa direcional).

R4 · A pasta certa na reunião

Levar o arquivo morto inteiro confunde e custa; a pasta certa resolve em dez minutos. Impacto: 40 a 80% do input (LLMLingua: até 20x de compressão).

R5 · O crachá de acesso

Na primeira visita você se registra na portaria; nas seguintes, só encosta o crachá. Reapresentar-se a cada porta é pagar a recepção de novo. Impacto: 30 a 50% do input em loops (leitura de cache a 10 a 50% do token novo).

R6 · O corrimão na escada

Ninguém precisa lembrar de se segurar: o corrimão está lá para todos, em toda descida. Disciplina vira infraestrutura versionada. Impacto: composto e permanente.

R7 · O disjuntor

Não reduz a conta de luz do mês, evita o incêndio do pico. A economia vem das outras alavancas; o cap segura a variância. Impacto: guard-rail; ULB em GA desde 2026-06-01.

CAPÍTULO 09 ARTEFATOS COMPLETOS

Instruções de repositório e por área.

MARKDOWN .GITHUB/COPILOT-INSTRUCTIONS.MD

```
# Instruções do repositório

## Saída
- Responda com código ou diff; sem preâmbulo e sem resumo ao final.
- Proponha a menor mudança correta; mostre só as linhas alteradas.
- Não repita o enunciado nem reformule a pergunta.
- Para perguntas conceituais: no máximo 5 linhas.

## Contexto
- Stack: <linguagem/framework>. Padrões em docs/CONTRIBUTING.md.
- Prefira referências por arquivo (#file) a trechos colados.
- Testes acompanham toda lógica nova.

## Estilo
- Código idiomático; nomes em inglês; comentários só onde o porquê não é óbvio.
```

MARKDOWN .GITHUB/INSTRUCTIONS/TESTS.INSTRUCTIONS.MD

```
---
applyTo: "**/*.test.*"
---
- Gere apenas o corpo do teste; sem explicar o scaffolding.
- Um assert claro por caso; nomes descrevem o comportamento.
- Não duplique cenários já cobertos no arquivo.
```

★ TIP

Troque os placeholders entre <> pela sua stack e revise por PR: primitivo é código. No VS Code, gere com `/create-instruction`.

CAPÍTULO 09 ARTEFATOS COMPLETOS

AGENTS.md e agente com escopo restrito.

MARKDOWN AGENTS.MD

```
# AGENTS.md

## Build e testes
- `npm test` roda a suíte; `npm run lint` antes do commit.

## Convenções
- Commits convencionais; PRs pequenos e focados.

## Custo
- Modelo padrão: Auto. Fronteira só em refatoração multi-arquivo.
- Refira arquivos com #file; não cole trechos longos.
```

MARKDOWN .GITHUB/AGENTS/COST-AWARE-REVIEWER.AGENT.MD

```
---
name: cost-aware-reviewer
description: Revisão de PR focada em diff, com modelo intermediário.
model: <classe-intermediária>
tools: ['search', 'changes']
---
Você revisa pull requests deste repositório.
Responda apenas com:
1. Lista numerada de problemas (1 linha cada).
2. Diff sugerido por problema, quando houver correção.
Não recapitule o PR, não elogie, não explique o óbvio.
```

CAPÍTULO 09 ARTEFATOS COMPLETOS

Prompt, exclusion e a regra de roteamento.

MARKDOWN .GITHUB/PROMPTS/RELEASE-NOTES.PROMPT.MD

```
---
mode: agent
description: Gera notas de release a partir dos commits da branch.
---
Gere notas de release concisas a partir de #changes:
- Agrupe por tipo (feat, fix, chore).
- 1 linha por item, sem adjetivos.
- Breaking changes em destaque no topo.
```

YAML SETTINGS > COPILOT > CONTENT EXCLUSION

```
- "/**/.env*"
- "/dist/**"
- "/secrets/**"
- "**/*.pem"
- "**/fixtures/large/**"
```

TEXT REGRA DE ROTEAMENTO (ACORDO DO TIME E .AGENT.MD)

```
trivial   → classe leve / incluída, sem reasoning
padrão    → classe intermediária
complexo  → fronteira, reasoning conforme a tarefa
nunca     → fronteira para tarefa trivial
```

> ⓘ NOTE
>
> Hooks (guard-rails de loop do agente) são gerados com `/create-hook` no VS Code; o formato evolui com a extensão, gere pelo comando em vez de copiar um esquema fixo.

CAPÍTULO 10 ARQUITETURA DE TRANSBORDO

Pool incluído, muro do ULB e a faixa BYOK que sobrevive ao bloqueio.

O que é	Key do provedor conectada nas settings da enterprise ou organização (public preview, ampliado em 2026-01): modelos no Copilot Chat do GitHub.com e nas IDEs, cobrados direto no provedor, sem consumir AI Credits. Provedores: Anthropic, Microsoft Foundry, OpenAI, xAI, AWS Bedrock, Google AI Studio e OpenAI-compatible.
Camada 1 · Erga o muro	ULB universal no nível do pool incluído (ou pouco acima), com alertas em 75, 90 e 100% e metered bloqueado no estouro. O fim dos créditos vira um evento operacional com data, não uma surpresa na fatura.
Camada 2 · Garanta a faixa que sobrevive	Conecte a key do provedor nas settings da enterprise ou da organização (public preview): Anthropic, Microsoft Foundry, OpenAI, xAI, AWS Bedrock, Google AI Studio ou endpoint OpenAI-compatible. Ao bloquear, completions, modelos locais e modelos BYOK seguem funcionando.
Camada 3 · Pré-roteie antes de transbordar	Custom agents com o campo model: fixado no modelo BYOK para a rotina; o pool incluído fica reservado para o que se beneficia dos modelos nativos (Auto com desconto, code review, coding agent).
Camada 4 · Governe o lado do provedor	Uma key ou deployment por unidade de negócio espelhando os cost centers (chargeback), max context window por modelo BYOK, e o budget do Azure cost management como segundo disjuntor.
Camada 5 · Failover automático só onde há código	Em superfícies programáticas (Copilot SDK/CLI), capture o erro de bloqueio e recrie a sessão com a key própria. O SDK aceita só credencial estática (sem Entra ID ou managed identity): rotação de key entra no desenho de segurança.
Validação	Piloto: budget baixo num usuário de teste, estouro forçado, BYOK respondendo após o bloqueio, consumo no cost management do provedor. Só então levar a proposta.

> ⚠ WARNING
>
> Não há gatilho automático: budgets param, não roteiam, e o fallback automático foi descontinuado em 2026-06-01. A troca de faixa é seleção de modelo (picker ou campo model: dos agentes). Enterprise BYOK em public preview; completions nunca usam BYOK.

MICROSOFT + GITHUB

Comece medindo, aplique na ordem, e deixe a banda apertar com os dados reais. Otimização sem medição é palpite.

FIM DA DOCUMENTAÇÃO · V4.1.0 · RELEASE