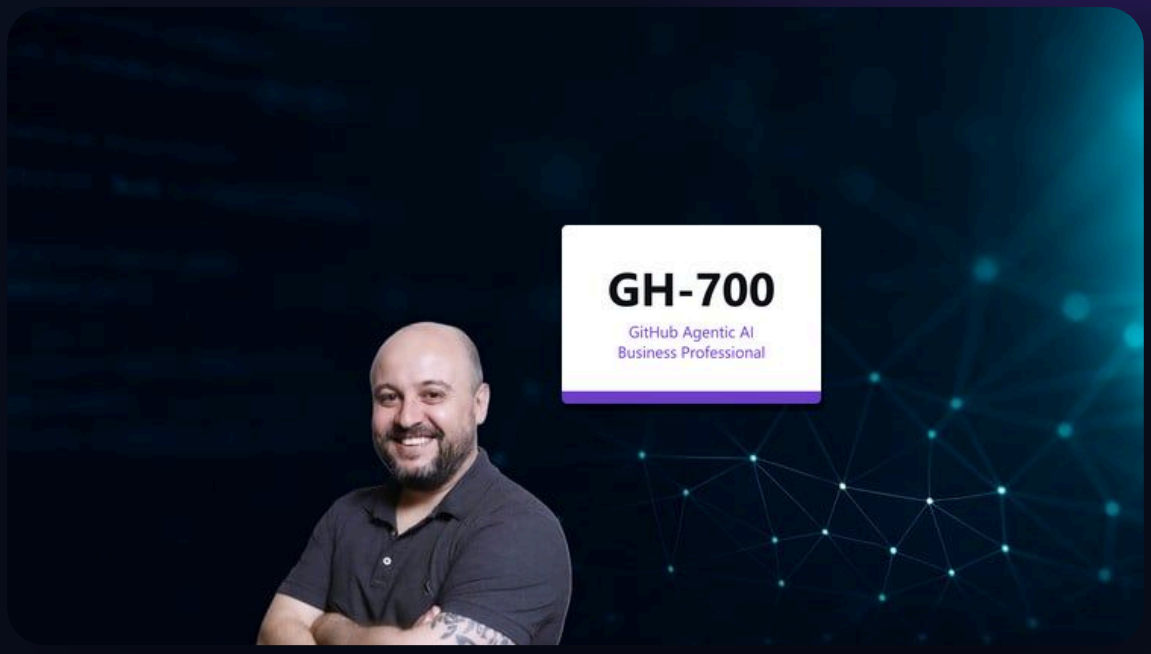


Guia de preparação GH-700

Tudo o que a prova cobra, organizado por domínio: conceitos-chave, comparações que confundem, pegadinhas reais, gatilhos do enunciado, questões comentadas e um plano para chegar pronto ao exame.

4 capítulos por domínio	**28** questões comentadas	**70%** nota de corte
6 provas completas no curso		

ANTES DE COMEÇAR

Como usar este guia

Este guia foi feito para ser estudado junto com as 6 provas completas do curso GH-700: o guia explica o que cai e onde estão as armadilhas; as provas treinam o ritmo e mostram o que você ainda não domina.

Quando	O que fazer
Semana 1	Faça o teste diagnóstico (página seguinte) e leia os capítulos 1 e 2. No fim da semana, faça a Prova 1 do curso no tempo do exame.
Semana 2	Leia os demais capítulos, começando pelos domínios em que você errou mais no diagnóstico. Faça as provas 2 e 3.
Semana 3	Releia os alertas de pegadinha e as tabelas de comparação. Faça as provas 4 e 5 e revise cada erro pela explicação.
Semana 4	Estude a folha de cola, faça a Prova 6 e refaça as provas abaixo de 80%. Agende o exame quando estiver acima de 80% em todas (a nota de corte é 70%).

Mapa do exame

Cada prova do curso tem 50 questões e 100 minutos, no formato do exame. Referência de aprovação: 70%.

Domínio (capítulo)	Peso	Neste guia
1. Aumentar a produtividade usando o GitHub Copilot	20-25%	18 conceitos · 14 pegadinhas
2. Projetar e implementar fluxos de trabalho e automações agênticos	25-30%	18 conceitos · 14 pegadinhas
3. Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos	20-25%	18 conceitos · 14 pegadinhas
4. Usar os recursos da plataforma GitHub	20-25%	18 conceitos · 14 pegadinhas

Como ler os capítulos: os **tópicos-chave** são o que mais cai; a **tabela de memorização** separa o que a prova costuma confundir; os **alertas de prova** mostram as armadilhas das alternativas erradas; os **gatilhos** ligam palavras do enunciado à resposta; e o **checklist** diz se você está pronto para o domínio.

Estude um pouco todo dia, de graça

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o “não confunda” de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

DIAGNÓSTICO

Você já sabe isso?

Responda às 12 questões sem consultar nada. As respostas, com a explicação e o capítulo para revisar, estão no fim do guia. Comece o estudo pelos capítulos em que errou.

D1 · Aumentar a produtividade usando o GitHub Copilot

O analista de RH da Rede Bem Estar repete toda semana o mesmo pedido ao Copilot no VS Code, mudando só o nome do departamento. Ele quer um modelo reutilizável que só entre em ação quando ele chamar. O que a documentação indica?

A. Um prompt file .prompt.md com variáveis de entrada
B. Uma automação do Copilot com gatilho por agenda
C. Uma instrução de repositório em copilot-instructions.md
D. Uma regra de ruleset que obrigue o mesmo texto

D2 · Aumentar a produtividade usando o GitHub Copilot

No GitHub Copilot app, o gerente de produto da Rede Lar Doce quer tocar ao mesmo tempo duas tarefas no mesmo repositório sem que uma atrapalhe a outra. O que a documentação do app descreve?

A. As sessões dividem o mesmo branch e se revezam nas edições
B. Só uma sessão por repositório pode ficar ativa ao mesmo tempo
C. Cada sessão roda num workspace isolado, com branch próprio
D. A segunda sessão espera a primeira abrir o pull request

D3 · Aumentar a produtividade usando o GitHub Copilot

No treinamento de prompts da Escola Saber Mais, a instrutora apresenta os 4 Ss do módulo de engenharia de prompt do Microsoft Learn. Quais são eles?

A. Single, Specific, Short e Surround
B. Short, Smart, Shared e Surround
C. Single, Secure, Specific e Summary
D. Simple, Secure, Scalable e Structured

D4 · Projetar e implementar fluxos de trabalho e automações agênticos

Na Corretora Capital Leste, um agentic workflow só lê o repositório e cria uma issue com um resumo. A analista não declarou nada em permissions além do necessário. Qual é o padrão de permissões do repositório para esse tipo de workflow?

A. Nenhuma permissão, nem de leitura, até haver aprovação
B. Somente leitura (read-all), salvo concessão explícita
C. Escrita total, limitada depois pelo próprio prompt
D. As mesmas permissões da pessoa que compilou o arquivo

D5 · Projetar e implementar fluxos de trabalho e automações agênticos

O diretor de segurança da Corretora Atlas pergunta que verificações o Copilot cloud agent faz no próprio código antes de concluir o pull request. O que a documentação informa?

A. Envia o código para auditoria manual do suporte do GitHub antes de abrir o PR
B. Usa CodeQL, secret scanning, a Advisory Database e uma segunda opinião do code review
C. Exige licença do GitHub Advanced Security para rodar qualquer verificação de segurança
D. Não faz verificações e deixa toda a análise de segurança para depois do merge

D6 · Projetar e implementar fluxos de trabalho e automações agênticos

A gerente de dados da Seguros Proteção quer que uma análise longa seja dividida em partes independentes, cada uma num contexto próprio, sem encher a conversa principal com detalhes intermediários. Que conceito a documentação descreve para isso?

A. Labels de issue, que separam as partes no repositório
B. Subagentes, que fazem trabalho delegado em contexto separado
C. Prompt files, que dividem o pedido em várias conversas separadas
D. Instruções pessoais, que guardam cada parte em um arquivo

D7 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

Uma execução do agentic workflow de relatórios da Indústria Cristal falhou ontem. O analista quer examinar em detalhes só essa execução, inclusive o uso de tokens e o custo estimado. Qual comando usa?

A. gh aw audit com o ID da execução
B. gh aw add com o ID da execução
C. gh aw init com o ID da execução
D. gh aw compile com o ID da execução

D8 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

Na Rádio Onda Sul, o analista de requisitos ativou o plan mode do Copilot CLI para uma mudança de várias etapas na grade de programação. Que benefício de negócio esse modo oferece?

A. Liberar as ferramentas do agente sem pedir aprovação ao analista
B. Pegar mal-entendidos de escopo antes que alguma mudança seja feita
C. Compactar a conversa para gastar menos créditos durante a tarefa
D. Executar as etapas em sequência e mostrar só o resumo no final

D9 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

Na tela Automations do GitHub Copilot app, o analista da Distribuidora Planalto quer saber de relance se as automações rodaram bem. Que informações cada cartão mostra?

A. Custo em reais, autor de cada commit e lista de revisores do PR
B. Perfis de agente, skills instaladas e servidores MCP ativos
C. Modelo usado, prompt completo e histórico de todas as execuções
D. Nome, agenda, repositório associado e status da última execução

D10 · Usar os recursos da plataforma GitHub

No board do project da Agência Faro, a coluna Em revisão tem limite de cinco cards, mas a equipe arrastou um sexto card para ela. O que acontece?

A. O card entra, e a contagem acima do limite fica destacada
B. O card volta para a coluna anterior por exceder o limite definido
C. O card entra e o limite passa a valer em todas as views
D. O card fica bloqueado até um admin aprovar a exceção

D11 · Usar os recursos da plataforma GitHub

Durante um incidente de duas semanas, a Fintech Sabiá vai aumentar o orçamento de AI credits de um analista sênior, mas não quer depender de alguém lembrar de desfazer o aumento depois. O que a documentação permite?

A. Definir data de expiração no orçamento individual desse usuário
B. Criar um cost center temporário que se apaga ao fim do incidente
C. Definir data de expiração no orçamento universal de todos os usuários
D. Programar uma automação do Copilot para reduzir o valor depois

D12 · Usar os recursos da plataforma GitHub

A gerente do programa de modernização da Prefeitura Parque Verde quer registrar no project que o programa está em risco, com nova data-alvo e uma mensagem para os patrocinadores. Que recurso usar?

A. Um campo de iteração novo chamado programa em risco
B. Um status update do project, com status At risk e datas
C. Uma página da wiki com a mensagem para os patrocinadores
D. Um label At risk aplicado a todas as issues do project

01

CAPÍTULO 1 · 20-25% DA PROVA

Aumentar a produtividade usando o GitHub Copilot

Neste domínio eu preciso saber usar o Copilot no Chat do GitHub.com, no GitHub Copilot app e no Copilot CLI: escolher modelo, gerenciar sessões, anexar contexto com @ e #, usar comandos de barra e escrever prompts que dão resultado previsível. A prova adora comando de barra parecido (/clear, /delete, /compact, /restart-session, /chronicle standup e tips) e espera que eu saiba exatamente o que cada um faz e em qual superfície existe. Também cai muito a criação de agentes personalizados: onde fica o arquivo .agent.md, quais propriedades são obrigatórias, o que acontece se omito tools e como restringir ferramentas como guardrail. Meu ponto fraco costuma ser a diferença entre retomar, renomear, arquivar, excluir e voltar no tempo (rewind) uma sessão. Nas boas práticas de prompt, a resposta certa sempre é a mais explícita: formato declarado direto, exemplos, requisitos no início e condição de parada.

Neste capítulo
18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Superfícies do Copilot

O GitHub Copilot app é um aplicativo de desktop, disponível em todos os planos, que roda várias sessões de agente em paralelo, cada uma em workspace isolado com worktree e branch próprios (nova worktree, repositório local ou sandbox na nuvem). O Copilot CLI roda no terminal, de forma interativa ou programática com um único prompt. O Chat no GitHub.com responde perguntas e delega trabalho ao cloud agent na mesma conversa, compartilhando contexto com ele.

Copilot Chat no GitHub.com

O seletor de modelo fica na parte de baixo; o ícone de nova tentativa regenera a resposta com outro modelo para comparar. Subthreads nascem ao usar Edit message numa pergunta anterior, sem perder a resposta original. Guarda até 100 conversas e mantém as mensagens por 28 dias. No Copilot Business, trocar de modelo depende de a organização liberar nas políticas, e cada modelo consome AI credits numa taxa própria por token.

Comandos e menções do Chat

No GitHub.com: /clear limpa, /delete exclui de vez, /new abre outra conversa e /rename dá nome; @ anexa discussions, extensions, arquivos, issues, pull requests e repositórios. No IDE: /explain, /fix, /tests e /doc (Visual Studio e Xcode), além de /clear e /help; digite / para ver a lista do ambiente. No app: # referencia issues, @ adiciona arquivos e / abre comandos; no CLI: @caminho e #número.

Anexos e pesquisa na web no Chat

O Chat do GitHub aceita JPEG, PNG, GIF, WEBP, PDF, HEIC e HEIF, arrastando ou pelo botão de anexo, desde que o modelo aceite imagem; vale em todos os planos e não depende de política. Para assinante individual, a pesquisa na web do Chat vem desabilitada e pode ser habilitada nas configurações; com ela, o Chat usa o Bing.

Retomar, nomear e compartilhar sessões do CLI

copilot --continue volta direto à sessão mais recente; --resume abre o seletor (ou vai direto com um ID); dentro da sessão, /resume troca de sessão e /rename NOVO_NOME facilita achar no seletor. Na barra lateral, n cria sessão e x duas vezes exclui de vez a sessão destacada. /share (alias /export) gera link no GitHub.com, gist privado, arquivo Markdown ou HTML interativo; gists não existem para Enterprise Managed Users nem com residência de dados.

/chronicle

Usa o histórico de sessões para gerar insights: standup (resumo do trabalho recente, por status), tips (dicas personalizadas), cost-tips (economia de tokens), search (buscar tema no histórico), improve (sugestões para .github/copilot-instructions.md) e reindex (reconstrói o índice local). É recurso do CLI e do app, não do Chat do GitHub.

Gerenciar contexto: compact, clear, restart

/compact troca o histórico por um resumo inteligente em vez de descartá-lo, aliviando a pressão de tokens; no CLI esses resumos ficam como checkpoints, vistos com /session checkpoints. /clear (ou /reset) descarta a transcrição e começa do zero. /restart-session reinicia mantendo o histórico. No app, Archive chat tira o chat da barra sem perder histórico e Manage sessions restaura.

Rewind (/undo) no CLI

Volta a sessão ao estado imediatamente anterior ao prompt escolhido, que reaparece na caixa para edição; escolha Conversation only ou Conversation + files. Com arquivos, mostra prévia por arquivo, restaura só o que o Copilot mudou e pula arquivos que você editou depois. Não pode ser desfeito, não volta antes do primeiro prompt e não funciona em sessões remotas nem com trabalho em andamento.

Auto, camadas e esforço de raciocínio

Com Auto, o app escolhe o modelo pela complexidade, e o seletor ao lado de Auto mostra qual modelo respondeu cada mensagem. A seleção automática otimizada tem três camadas: Efficiency (prioriza custo), Balance (equilibra custo, qualidade e latência) e Intelligence (prioriza qualidade); a cobrança segue o modelo escolhido. Aumentar o esforço de raciocínio dá mais tempo para problemas complexos, mas demora e consome mais tokens.

Engenharia de prompt da documentação do GitHub

Comece geral e depois especifique, com os requisitos mais importantes no início (o Copilot dá peso extra ao início e ao fim). Dê exemplos de entrada e saída esperada, divida tarefas grandes, evite ambiguidade dizendo exatamente a que se refere e itere referenciando a resposta anterior. Declare formato e tamanho de forma direta, sem meta-instruções do tipo lembre-se de sempre, e inclua uma condição de parada clara para economizar créditos.

Os 4 Ss do Microsoft Learn

Single (foco numa tarefa ou pergunta), Specific (instruções explícitas e detalhadas), Short (conciso) e Surround (contexto relevante ao redor). Distratores trocam por Smart, Shared, Secure, Summary, Simple, Scalable ou Structured.

Delegar ao cloud agent e outros recursos no GitHub

Ao delegar, escolha repositório e branch base e descreva objetivo, critérios de aceite e arquivos relevantes, anexando capturas quando o visual importa; orientações duradouras vão em arquivos de instruções e senhas nunca vão no prompt. No campo de descrição do PR, as ações do Copilot geram um Summary (não disponível no Copilot Free). Em github.com/copilot, o Copilot monta rascunho de issue a partir de texto ou imagem, para revisar antes de Create.

Prompt files e instruções pessoais

Prompt files (.prompt.md, em geral em .github/prompts) são modelos reutilizáveis com variáveis de entrada, acionados só quando você chama. Instruções pessoais guardam preferências individuais (idioma, nível de detalhe, estrutura) e, no GitHub.com, valem para o Copilot Chat; o CLI usa arquivos locais.

Perfil de agente personalizado (.agent.md)

Fica em .github/agents/NOME.agent.md e aparece no seletor depois do commit e merge no branch padrão. Frontmatter: name (opcional, padrão é o nome do arquivo), description (obrigatória), tools (opcional; omitida libera todas), mcp-servers (servidores só desse agente), target (vscode ou github-copilot; omitido vale nos dois) e model (IDEs). O prompt abaixo do frontmatter aceita até 30.000 caracteres.

Onde criar e testar agentes de organização e enterprise

Agentes da organização ficam no repositório .github ou .github-private da organização, na pasta agents na raiz. Agentes da enterprise ficam no .github-private de uma organização designada nas configurações da enterprise. Para testar antes de liberar, crie em .github/agents no .github-private, rode sessões de teste e depois mova o perfil para agents e faça merge.

Guardrails e revisão de agentes personalizados

Restringir tools é o guardrail do perfil: um revisor pode ter só leitura e busca, e de um servidor MCP pode receber só a ferramenta de consulta. O alias web (busca e leitura de URLs) não se aplica ao cloud agent hoje. Para revisar o que o agente fez, abra a sessão na página de agentes: log e visão geral mostram raciocínio, ferramentas usadas, uso de tokens e duração; na organização, filtre o audit log por actor:Copilot.

Agente personalizado, subagente, skill e agentes nativos

O agente personalizado é a definição de um especialista com abordagem e ferramentas próprias; o subagente é quem faz trabalho delegado em contexto separado. Se só precisa de instruções e recursos para uma tarefa ocasional, a skill é mais simples. Agentes nativos do CLI: explore, task, general-purpose, code-review, research, rubber-duck e security-review. No CLI, as instruções do repositório não vão por padrão ao subagente personalizado (há opção para incluí-las).

Comandos de barra do app ligados à produtividade

/agent seleciona o agente personalizado, /model escolhe o modelo, /models abre a seleção de modelos, /skills gerencia skills, /research gera relatório com citações, /rubber-duck pede crítica a outro modelo, /review revisa as mudanças da sessão e /inbox mostra itens de trabalho. Chats na barra lateral servem para brainstorming sem branch nem worktree.

Macetes para memorizar

💡 Símbolos de contexto

No app: # é issue, @ é arquivo, / é comando. No Chat do GitHub.com: tudo que é anexo entra por @.

💡 Os 4 Ss

Single, Specific, Short, Surround: uma coisa, explícita, curta e com contexto ao redor.

/chronicle

S-T-C-S-I-R: Standup, Tips, Cost-tips, Search, Improve, Reindex.

Limpar x resumir x reiniciar

Clear apaga, Compact comprime, Restart mantém.

.agent.md

Só a description é obrigatória; sem tools = tudo liberado; sem target = vale nos dois; sem name = nome do arquivo.

Camadas do Auto

Efficiency economiza, Balance equilibra, Intelligence capricha. Reasoning não existe como camada.

Prompt que funciona

Importante no começo, exemplo no meio, formato declarado e ponto final de parada.

Tabela de memorização: não confunda

Quando é...	...e quando é
/compact Manter o trabalho e reduzir tokens resumindo o começo da conversa.	**/clear (ou /reset)** Descartar a transcrição e começar uma sessão limpa, sem histórico.
/clear no Chat do GitHub.com Limpar a conversa.	**/delete no Chat do GitHub.com** Apagar de vez a conversa, por exemplo com dados de cliente.
copilot --continue Voltar direto à sessão mais recente sem escolher numa lista.	**copilot --resume** Abrir o seletor de sessões recentes ou ir a uma sessão por ID.
Prompt file (.prompt.md) Modelo reutilizável com variáveis, acionado só quando você chama.	**Instruções personalizadas** Orientação que entra automaticamente em todas as conversas do escopo.
Agente personalizado Papel especialista reutilizável com abordagem e ferramentas próprias.	**Agent skill** Só instruções e recursos para uma tarefa que aparece de vez em quando, carregados quando relevantes.
/chronicle standup Resumo do que você fez nas sessões recentes, agrupado por status, antes da daily.	**/chronicle tips** Recomendações personalizadas para usar melhor o CLI com base no seu histórico.
Rewind Conversation only Voltar só a conversa a um ponto anterior.	**Rewind Conversation + files** Voltar a conversa e restaurar os arquivos que o Copilot mudou, com prévia por arquivo.

Números que caem

Item	Valor
Conversas guardadas no Copilot Chat do GitHub	**Até 100 conversas recentes**
Retenção das mensagens no Copilot Chat do GitHub	**28 dias**
Tamanho do prompt do perfil de agente personalizado	**Até 30.000 caracteres**
Peso do domínio no exame	**20-25%**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Achar que omitir tools no perfil deixa o agente só com leitura ou sem ferramentas.
Como não cair: Sem tools o agente recebe todas as ferramentas; para restringir, liste só as necessárias.

⚠ **Alerta de prova:** Restringir o agente pedindo no prompt para não editar, ou usando target.
Como não cair: Prompt orienta e target escolhe o ambiente; quem limita ferramentas é a propriedade tools.

⚠ **Alerta de prova:** Pensar que o agente aparece no seletor assim que o perfil é commitado num branch de trabalho.
Como não cair: Precisa de merge no branch padrão e atualizar a aba de agentes.

⚠ **Alerta de prova:** Tratar target ou tools como obrigatórios no .agent.md.
Como não cair: Só description é obrigatória; name assume o nome do arquivo.

⚠ **Alerta de prova:** Copiar o perfil para o .github de cada organização para ter agente da enterprise.
Como não cair: Isso cria agentes de organização; agente de enterprise fica no .github-private de uma organização designada.

⚠ **Alerta de prova:** Usar /compact para renomear, /share file para nomear ou /chronicle search para fixar sessão.
Como não cair: Nome de sessão é com /rename NOVO_NOME; os outros comandos resumem, exportam ou buscam.

⚠ **Alerta de prova:** Achar que x duas vezes na barra lateral do CLI arquiva ou pausa a sessão.
Como não cair: Exclui de vez; para guardar registro, use /share antes.

⚠ **Alerta de prova:** Esperar que o rewind sobrescreva um arquivo que você editou à mão depois.
Como não cair: O rewind pula arquivos cujo conteúdo não bate mais com o que o Copilot escreveu.

⚠ **Alerta de prova:** Tentar rewind numa sessão remota ou achar que ele volta ao fim do prompt escolhido.
Como não cair: Não funciona em sessões remotas; volta ao estado anterior ao prompt escolhido.

⚠ **Alerta de prova:** Reforçar uma regra com lembre-se de sempre ou repetindo o lembrete.
Como não cair: Declare a regra diretamente, como Devolva o resultado em uma tabela.

⚠ **Alerta de prova:** Trocar para modelo mais caro ou repetir o prompt para corrigir uma resposta fraca.
Como não cair: O problema costuma ser o prompt: divida a tarefa, dê exemplos, explicite a referência e o formato.

⚠ **Alerta de prova:** Achar que esforço de raciocínio alto reduz tokens ou garante resposta sem erro.
Como não cair: Mais raciocínio consome mais tokens e demora mais; use o nível regular quando a tarefa não exigir.

⚠ **Alerta de prova:** Usar # para anexar issue no Chat do GitHub.com.
Como não cair: No Chat do GitHub a anexação é com @; # é para issues no app e no CLI.

⚠ **Alerta de prova:** Supor que o Chat no navegador é gratuito ou que custo varia com o tamanho do repositório.
Como não cair: O Copilot Chat consome AI credits, e cada modelo tem taxa própria por token.

Como cai na prova

📝 **Como cai:** Usuário quer um efeito numa sessão e as alternativas são comandos de barra parecidos.
Como resolver: Identifique a superfície (Chat do GitHub.com, app, CLI, IDE) e o verbo pedido (apagar, limpar, resumir, renomear, buscar, exportar); descarte comandos que não existem naquela superfície, como /chronicle no Chat do GitHub.

📝 **Como cai:** Resposta do Copilot veio ruim e a pergunta é qual ajuste de prompt a documentação recomenda.
Como resolver: Escolha a alternativa que torna o pedido mais explícito (formato direto, exemplos, referência clara, divisão em etapas, condição de parada); descarte trocar de modelo, repetir o prompt ou deixar o modelo adivinhar.

📝 **Como cai:** Configuração de agente personalizado: onde fica, quando aparece, como restringir.
Como resolver: Lembre o caminho .github/agents/NOME.agent.md, merge no branch padrão, tools como guardrail, target para ambiente e os repositórios .github, .github-private e organização designada para níveis mais altos.

📝 **Como cai:** Desfazer ou recuperar trabalho numa sessão do CLI.
Como resolver: Para desfazer o que foi concluído, rewind (/undo) com Conversation + files; para revisar resumos, /session checkpoints; lembre que o rewind é irreversível, pula arquivos editados por você e não roda em sessão remota.

📝 **Como cai:** Escolha entre prompt file, instrução, skill e agente personalizado para uma necessidade recorrente.
Como resolver: Sob demanda com variáveis: prompt file; sempre automático no escopo: instrução; tarefa ocasional com recursos: skill; papel especialista com ferramentas próprias: agente personalizado.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Ver como outro modelo responderia sem reescrever o prompt	**Ícone de nova tentativa abaixo da resposta**
Testar versão alternativa da pergunta sem perder a resposta original	**Subthread (Edit message)**
Conversa de dois meses atrás incompleta	**Mensagens mantidas por 28 dias**
Sessão longa e lenta, mas quer manter o trabalho	**/compact**
Voltar exatamente à sessão de ontem sem lista	**copilot --continue**
Resumo do que fiz nas últimas 24 horas antes da daily	**/chronicle standup**
Melhorar o arquivo de instruções com base no histórico	**/chronicle improve**
Conferir o que foi preservado nos resumos de compactação	**/session checkpoints**
Registro navegável da conversa para abrir no navegador	**/share html**
Mesmo pedido toda semana mudando só um nome	**Prompt file com variáveis de entrada**
Responder sempre em português, curto e em tópicos	**Instruções pessoais no GitHub.com**
Agente que lê e pesquisa mas nunca edita	**tools só com leitura e busca**
Agente só no cloud agent e não no VS Code	**Propriedade target**
Priorizar custo nas tarefas rápidas usando Auto	**Camada Efficiency**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Sem tools no perfil, o agente fica com todas as ferramentas; só description é obrigatória.
- Agente personalizado só aparece no seletor depois do merge no branch padrão.
- Rewind volta ao estado anterior ao prompt escolhido, é irreversível e não funciona em sessão remota.
- Regra direta vence meta-instrução; requisitos importantes vão no início do prompt.
- /compact resume, /clear zera, /restart-session mantém o histórico.

Essenciais para a prova (checklist)

☐ Sei comparar app, CLI e Chat no GitHub.com e o que cada um faz.

☐ Sei usar ícone de nova tentativa, subthreads e os comandos /clear, /delete, /new e /rename no Chat do GitHub.

☐ Sei retomar sessões com --continue, --resume e /resume e renomear com /rename.

☐ Sei os seis subcomandos do /chronicle e quando usar cada um.

☐ Sei a diferença entre /compact, /clear e /restart-session e como ver checkpoints.

☐ Sei como o rewind funciona e quais são suas limitações.

☐ Sei aplicar as práticas de prompt do GitHub e os 4 Ss.

☐ Sei criar um .agent.md com as propriedades certas e restringir ferramentas.

☐ Sei onde ficam agentes de organização e de enterprise e como testá-los antes de liberar.

☐ Sei escolher entre prompt file, instrução, skill e agente personalizado.

Questões de revisão do capítulo

R1.1 · Aumentar a produtividade usando o GitHub Copilot

Uma analista de negócios da Seguradora Horizonte achou fraca uma resposta do Copilot Chat em github.com e quer ver como outro modelo responderia à mesma pergunta, sem reescrever o prompt. O que ela deve fazer?

A. Pedir na mesma conversa que o Copilot simule a resposta de um modelo diferente
B. Clicar no ícone de nova tentativa abaixo da resposta e escolher outro modelo
C. Editar a pergunta para abrir um subthread, que troca o modelo por conta própria
D. Usar o comando /new seguido do nome do modelo para repetir a mesma pergunta

R1.2 · Aumentar a produtividade usando o GitHub Copilot

No Copilot Chat em github.com, a analista de negócios da Vinícola Alto Vale quer testar uma versão alternativa de uma pergunta anterior sem perder a resposta original. O que ela usa?

A. O comando /clear para limpar a conversa e recomeçar o assunto
B. O comando /delete para apagar a resposta e perguntar outra vez
C. Uma automação com gatilho manual para repetir a conversa inteira
D. Um subthread, editando a pergunta anterior e enviando de novo

R1.3 · Aumentar a produtividade usando o GitHub Copilot

O analista de mercado da Startup Rumo criou um agente personalizado com a ferramenta web para pesquisar concorrentes e quer usá-lo no Copilot cloud agent. O que a referência de configuração alerta?

A. O alias web, de busca e fetch, não se aplica ao cloud agent hoje
B. O alias web fica liberado depois de compilar um .lock.yml
C. O alias web só funciona se o agente usar o modelo mais caro
D. O alias web exige um servidor MCP da organização registrado

R1.4 · Aumentar a produtividade usando o GitHub Copilot

A analista da Floricultura Jardim quer tirar da barra lateral do GitHub Copilot app uma conversa antiga sem perder o histórico, podendo trazê-la de volta depois. O que ela faz?

A. Exclui o chat e recupera depois pelo /chronicle reindex
B. Renomeia o chat com /rename para escondê-lo da lista
C. Arquiva o chat e, se precisar, restaura em Manage sessions
D. Exporta o chat com /export-gist e apaga a conversa original

Respostas comentadas

R1.1. B) Clicar no ícone de nova tentativa abaixo da resposta e escolher outro modelo

Por quê: No Copilot Chat do GitHub, o seletor de modelo fica na parte de baixo do chat, e o ícone de nova tentativa sob a resposta permite regenerá-la com outro modelo e comparar os resultados.

✗ A) Incorreta. Pedir uma simulação continua usando o mesmo modelo; não é uma comparação real entre modelos.
✗ C) Incorreta. Subthreads servem para explorar variações da pergunta editada; eles não trocam o modelo sozinhos.
✗ D) Incorreta. O /new inicia uma conversa nova; ele não recebe nome de modelo nem repete a pergunta anterior.

> **Dica de prova:** Comparar modelos na mesma pergunta: ícone de nova tentativa sob a resposta.

R1.2. D) Um subthread, editando a pergunta anterior e enviando de novo

Por quê: Subthreads são ramos de uma conversa. Passe o mouse sobre uma pergunta anterior, clique em Edit message, ajuste e envie; navegue entre as respostas pelo contador.

✗ A) Incorreta. /clear limpa a conversa, perdendo o contexto anterior.
✗ B) Incorreta. /delete exclui a conversa e perde a resposta original.
✗ C) Incorreta. Automações não repetem conversas do Chat.

> **Dica de prova:** Variação de pergunta sem perder a original: subthread.

R1.3. A) O alias web, de busca e fetch, não se aplica ao cloud agent hoje

Por quê: Os aliases de ferramentas de agentes personalizados variam por ambiente. O alias web (busca e leitura de URLs) está marcado como não aplicável ao cloud agent; em outros ambientes, como o CLI, a pesquisa na web pode ser usada.

✗ B) Incorreta. .lock.yml é o arquivo compilado de agentic workflows, sem relação com agentes personalizados.
✗ C) Incorreta. A disponibilidade da ferramenta não depende do modelo escolhido.
✗ D) Incorreta. A ferramenta web é um alias nativo; a restrição é do ambiente do cloud agent.

> **Dica de prova:** Ferramenta web em agente personalizado: não vale no cloud agent.

R1.4. C) Arquiva o chat e, se precisar, restaura em Manage sessions

Por quê: No app, Archive chat tira o chat da barra sem perder o histórico. Em Settings, Sessions, Manage sessions, você busca, arquiva, exclui e restaura chats arquivados.

✗ A) Incorreta. Excluir apaga o chat; o reindex reconstrói índice, não recupera itens excluídos.
✗ B) Incorreta. Renomear não esconde o chat da lista.
✗ D) Incorreta. Exportar e apagar perde o chat no app.

> **Dica de prova:** Tirar sem perder: Archive chat.

02

CAPÍTULO 2 · 25-30% DA PROVA

Projetar e implementar fluxos de trabalho e automações agênticos

É o domínio de maior peso, então preciso diferenciar com segurança três coisas que a prova embaralha: GitHub Agentic Workflows (arquivo Markdown versionado no repositório e compilado em .lock.yml), automações do Copilot (configuradas na aba Agents ou no app, fora do conteúdo do repositório) e dynamic workflows (processo definido em código, em extensões do Copilot). As questões pedem campo do frontmatter certo (on, permissions, safe-outputs, engine, tools, max-ai-credits, source), comando gh aw certo e requisitos, escopo, cobrança e visibilidade das automações. Também caem skills (estrutura da pasta, SKILL.md, allowed-tools, gh skill com --pin) e quando um servidor MCP faz sentido. Erro onde troco safe-outputs por permissions, Run now por gh aw run e o que a automação herda do cloud agent. Por fim, preciso reconhecer padrões de desenho: chat x agente x automação, decomposição, pesquisar-planejar-iterar e autocrítica com rubber duck.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Chat, agente e automação

Chat explora e orienta por conversa; agente cumpre um objetivo raciocinando e usando ferramentas; automação inicia trabalho repetível a partir de evento, agenda ou outro gatilho e pode rodar passos fixos ou chamar um agente. Nem toda automação é agêntica. Para escolher a experiência: guiar o trabalho de perto: GitHub Copilot app, CLI ou sessão colaborativa no IDE. Delegar uma tarefa bem delimitada e revisar depois: Copilot cloud agent. Repetir o mesmo trabalho a partir de um gatilho: fluxo automatizado. Em todos, a pessoa define objetivo e limites e revisa o resultado.

Decomposição e padrão pesquisar, planejar e iterar

Tarefas grandes devem ser divididas em etapas menores pedidas uma de cada vez. Quando o escopo é incerto, o fluxo do cloud agent é pesquisar o repositório, pedir um plano e refiná-lo, implementar revisando o diff e só então criar o pull request.

Casos de uso citados

Agentic workflows: triagem e rotulagem de issues, investigação de falhas de CI, relatórios diários ou semanais de status, manutenção da documentação e melhoria da cobertura de testes. Automações: triar issues novas, corrigir testes com falha toda noite abrindo PR em rascunho e preparar notas de versão semanais. Aprovar e mesclar PR sem revisão humana ou rotacionar credenciais não são casos citados.

Arquivo de um GitHub Agentic Workflow

É um arquivo Markdown em .github/workflows com duas partes: frontmatter YAML (quando roda, permissões, escritas permitidas, motor de IA) e corpo Markdown com as instruções em linguagem natural. É compilado com gh aw compile num .lock.yml de mesmo nome, e os dois arquivos são commitados juntos no branch padrão; ele roda como workflow do GitHub Actions.

Campos do frontmatter

on: gatilho na mesma sintaxe do Actions (por exemplo, semanal na segunda). permissions: padrão read-all. safe-outputs: operações de escrita permitidas (create-issue, add-comment, create-pull-request, add-labels, set-issue-type). engine: motor de IA, copilot por padrão. tools: ferramentas, como github com toolsets [issues]. max-ai-credits: teto de inferência por execução (padrão 1.000 AIC). source: origem para gh aw update.

Guardrails dos agentic workflows

O agente roda em contêineres com firewall e tokens somente leitura por padrão; escritas só passam pelos safe outputs, que antes passam por detecção de ameaças; secrets ficam fora do runtime do agente, em jobs isolados; há controle de quem pode disparar ou alterar. As saídas são revisáveis e aprovações e merges continuam com pessoas.

Motor e autenticação

Aceita motores Copilot, Anthropic, OpenAI e Google, declarados em engine, cada um com seu secret. Em organização com Copilot, o recomendado é GITHUB_TOKEN: a organização habilita Allow use of Copilot CLI billed to the organization e o workflow declara copilot-requests: write. Em repositório pessoal, usa-se o secret COPILOT_GITHUB_TOKEN com token pessoal fine-grained (ou chave de API nos motores de terceiros).

Comandos gh aw e pré-requisitos

init prepara o repositório para autoria com agente de código (skills, instruções e agente personalizado); add e add-wizard importam (o wizard escolhe motor, configura secret, gera e compila os arquivos e abre PR); compile gera o .lock.yml; run dispara uma execução; logs lista execuções com duração, tokens e AIC estimado; audit detalha uma execução; update traz melhorias da origem indicada em source. Pré-requisitos: conta num motor, repositório com escrita, Actions habilitado e GitHub CLI autenticado com a extensão github/gh-aw.

Custo de um agentic workflow

É a soma dos minutos do GitHub Actions com o custo de inferência do motor. A inferência é acompanhada em AI Credits (1 AIC = US$ 0,01); no motor Copilot vira AI credits na cobrança do Copilot, e motores de terceiros cobram pelo provedor. Os AIC exibidos são estimativas.

Definição e gatilhos das automações do Copilot

Uma automação tem nome, prompt, um ou mais gatilhos, modelo e ferramentas. Gatilhos: agenda (por hora, dia ou semana), issue criada, pull request aberto e pull request sincronizado; issue criada aceita filtro de busca, e os de PR aceitam filtro de busca e de arquivos alterados. Com vários gatilhos, a automação roda quando qualquer um ocorre.

Requisitos, escopo e herança das automações

Exigem acesso de escrita, repositório privado ou interno, cloud agent habilitado e automações permitidas pela organização; existem nos planos Pro, Pro+, Max, Business e Enterprise e não valem em repositórios de managed user accounts. Têm escopo de um único repositório e herdam a configuração do cloud agent dele: instruções personalizadas, agent skills, regras de firewall e secrets e variáveis.

Visibilidade, cobrança e versionamento das automações

A automação é privada de quem a criou, mas as sessões, prompts, logs e PRs que ela gera são visíveis a quem acessa o repositório. Cada execução consome minutos do Actions e AI credits, cobrados de quem criou, e PRs e commits são atribuídos a essa pessoa. Automações ficam fora do conteúdo do repositório e não passam por PR; para lógica versionada e revisada, use agentic workflow.

Painel Automations e ferramentas

No painel do repositório você vê suas automações e as sessões iniciadas, edita nome, prompt, gatilhos, ferramentas e modelo, desabilita, reabilita, exclui e testa com Run now; a visão Automations do usuário reúne todas as suas automações. Selecione só as ferramentas necessárias, à mão ou com Suggest tools, inclusive para limitar o impacto de prompt injection.

Automações no GitHub Copilot app

Há automações locais e na nuvem: Run in the cloud roda num ambiente remoto mesmo com o computador desligado, e nas da nuvem as ferramentas ficam em Tools. Gatilhos: Manual, Hourly, Daily, Weekly, CRON (automação local, com validação e prévia), Issue e Pull request. No prompt, / chama uma skill; abaixo dele escolhem-se modelo, esforço de raciocínio e agente. Create salva, Create and run salva e roda, o play roda uma salva.

Agent skills

Skill é uma pasta com SKILL.md (frontmatter com name e description) e scripts e modelos citados nele; o nome da pasta vai em minúsculas com hífens. Projeto: .github/skills, .claude/skills ou .agents/skills; pessoal: ~/.copilot/skills ou ~/.agents/skills (o agente na nuvem pode não ter acesso). allowed-tools pré-aprova ferramentas e pré-aprovar shell exige confiança total. Seguem padrão aberto; no CLI, cite /nome-da-skill e use /skills list, /skills reload e /skills info.

gh skill

O GitHub CLI pesquisa, instala, atualiza e publica skills. gh skill install com --pin trava versão ou commit; gh skill update atualiza uma, todas com --all ou de forma interativa, mas pula as fixadas. Para mudar uma skill fixada, reinstale com outro --pin.

Servidores MCP

Use MCP quando o agente precisa de dados de um sistema externo ou de agir em outro serviço; evite se as ferramentas nativas já resolvem, e uma skill pode descrever como usar as ferramentas MCP. MCP configurado nas configurações do repositório vale para o cloud agent e o code review (o CLI usa arquivos próprios). Padrões do cloud agent: servidor do GitHub (token somente leitura do repositório atual) e Playwright (só localhost); o cloud agent chama ferramentas MCP sem pedir aprovação e só usa ferramentas, não recursos nem prompts. No app, descubra servidores pela aba Customize ou /af (Agent Finder).

Autocrítica, paralelismo e dynamic workflows

O rubber duck critica planos, código e testes usando um modelo diferente do da sessão, só com leitura, classificando achados em bloqueantes, não bloqueantes e sugestões, sem comentar estilo; por padrão é chamado sob pedido (/rubber-duck) e no CLI há invocação proativa pelo seletor /subagents. Dynamic workflow é um processo definido em código pelo autor, combinando passos automáticos e agentes, com paralelismo, passagem de resultados entre etapas, verificação mútua, pedido de input e checkpoints, distribuído em extensões do Copilot e disponível no app, CLI e SDK. /fleet coloca vários agentes numa tarefa; /spawn cria sessão filha; /fork bifurca; /orchestrate coordena.

Macetes para memorizar

💡 Frontmatter do agentic workflow

O-P-S-E: On (quando), Permissions (lê), Safe-outputs (escreve), Engine (quem pensa).

💡 Ciclo de vida do agentic workflow

Escreve o .md, compila o .lock, commita os dois, roda no Actions.

💡 Comandos gh aw

init prepara, add traz, compile gera, run dispara, logs lista, audit investiga, update atualiza.

💡 Automação do Copilot

Privada de quem cria, paga por quem cria, um repositório só, nunca em repositório público.

💡 Herança da automação

Herda o kit do cloud agent: instruções, skills, firewall e secrets.

💡 Skill

Pasta minúscula-com-hífen + SKILL.md com name e description; --pin congela, update pula o congelado.

MCP no cloud agent

Vem com GitHub e Playwright, usa ferramentas sem pedir licença: libere o mínimo.

Tabela de memorização: não confunda

Quando é...	...e quando é
GitHub Agentic Workflow Lógica versionada no repositório, revisada por PR, rodando no Actions com frontmatter e .lock.yml.	**Automação do Copilot** Tarefa definida uma vez na aba Agents ou no app, com prompt, gatilhos, modelo e ferramentas, guardada fora do repositório.
safe-outputs Declarar as operações de escrita que o agente pode fazer (comentar, abrir PR, rotular).	**permissions** Definir o acesso ao repositório, com padrão read-all; dar escrita ampla contraria o modelo dos safe outputs.
gh aw logs Acompanhar várias execuções recentes com duração, tokens e AIC estimado.	**gh aw audit RUN-ID** Examinar em detalhe uma única execução, como a que falhou ontem.
Run now (painel Automations) Testar uma automação do Copilot na hora.	**gh aw run ou aba Actions** Rodar uma vez um agentic workflow já compilado no branch padrão.
GITHUB_TOKEN com copilot-requests: write Repositório de organização com Copilot, com cobrança na organização.	**COPILOT_GITHUB_TOKEN** Repositório pessoal, com token pessoal fine-grained.
Dynamic workflow Processo repetível em que o autor fixa passos, condições e passagens em código.	**Autopilot ou /fleet** Autopilot foca em autonomia na execução; /fleet incentiva delegar o trabalho a subagentes em paralelo.
Skill Procedimento longo usado às vezes, carregado só quando relevante.	**Instruções do repositório** Orientações amplas e enxutas que valem sempre no escopo.

Números que caem

Item	Valor
Padrão de max-ai-credits por execução de agentic workflow	**1.000 AIC**
Valor de 1 AIC	**US$ 0,01**
Arquivos de um agentic workflow em .github/workflows	**Dois: o .md e o .lock.yml de mesmo nome**
Servidores MCP padrão do cloud agent	**Dois: GitHub e Playwright**

Peso do domínio no exame	**25-30%**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Editar o .lock.yml à mão ou achar que o .md só é lido na primeira execução.
Como não cair: A fonte é o Markdown; edite o .md, rode gh aw compile e commite os dois arquivos.

⚠ **Alerta de prova:** Achar que gh aw run recompila e publica o workflow.
Como não cair: gh aw run só dispara uma execução; quem gera o .lock.yml é gh aw compile.

⚠ **Alerta de prova:** Colocar o agendamento numa frase do corpo Markdown ou no campo engine.
Como não cair: Quando roda vai em on, na sintaxe do Actions; o corpo traz só as instruções da tarefa.

⚠ **Alerta de prova:** Supor que o padrão de permissions é nenhuma permissão, escrita total ou a de quem compilou.
Como não cair: O padrão é read-all; escritas só pelos safe-outputs.

⚠ **Alerta de prova:** Limitar ferramentas do servidor MCP do GitHub com uma frase no corpo ou em safe-outputs.
Como não cair: Use tools no frontmatter, com github e o toolset desejado, como [issues].

⚠ **Alerta de prova:** Criar COPILOT_GITHUB_TOKEN com token de administrador para cobrar a organização.
Como não cair: Para organização, o recomendado é GITHUB_TOKEN com copilot-requests: write e a cobrança habilitada pela organização.

⚠ **Alerta de prova:** Achar que automações rodam em repositórios públicos se o plano for Business.
Como não cair: Automações só existem em repositórios privados e internos, em qualquer plano elegível.

⚠ **Alerta de prova:** Supor que a automação alcança outros repositórios da organização ou os citados no prompt.
Como não cair: Escopo de um único repositório, sem exceção.

⚠ **Alerta de prova:** Achar que administradores veem as automações de colegas ou que a conta vai para a organização.
Como não cair: Automação é privada de quem criou e cobrada dessa pessoa; só as sessões e saídas são visíveis.

⚠ **Alerta de prova:** Contar instruções pessoais de quem criou ou views do project entre o que a automação herda.
Como não cair: Herda do cloud agent do repositório: instruções, skills, firewall e secrets e variáveis.

⚠ **Alerta de prova:** Usar gatilho CRON local para rodar com o notebook desligado.
Como não cair: Automação local depende da máquina ligada; ative Run in the cloud.

⚠ **Alerta de prova:** Achar que o cloud agent pede aprovação antes de cada chamada de ferramenta MCP.
Como não cair: Ele usa as ferramentas MCP sozinho; libere só o necessário.

⚠ **Alerta de prova:** Rodar gh skill update --all para atualizar uma skill instalada com --pin.
Como não cair: Skills fixadas são puladas; reinstale com novo valor em --pin.

⚠ **Alerta de prova:** Pré-aprovar shell em allowed-tools de toda skill para evitar pedidos.
Como não cair: Pré-aprovar shell só depois de revisar a skill e confiar na fonte; sem isso, abre espaço para prompt injection.

Como cai na prova

📝 **Como cai:** Cenário pede onde configurar algo num agentic workflow e as alternativas misturam frontmatter, corpo, permissions e safe-outputs.
Como resolver: Classifique o pedido: quando roda (on), o que lê (permissions), o que escreve (safe-outputs), quem pensa (engine), quais ferramentas (tools), quanto gasta (max-ai-credits); instruções em linguagem natural vão no corpo.

📝 **Como cai:** Gestor quer algo recorrente e as alternativas são chat, agente personalizado, prompt file e automação.
Como resolver: Se o trabalho precisa rodar sem alguém pedir, a resposta é automação por gatilho ou agenda; chat, prompt file e agente escolhido à mão dependem de alguém iniciar.

📝 **Como cai:** Pergunta sobre requisito, escopo, visibilidade ou cobrança de automação.
Como resolver: Aplique o pacote: escrita, privado ou interno, cloud agent habilitado, permitido pela organização; um repositório; privada de quem criou; minutos do Actions e AI credits cobrados de quem criou.

📝 **Como cai:** Qual comando gh aw ou gh skill atende a uma necessidade.
Como resolver: Associe o verbo: preparar (init), importar (add, add-wizard), gerar lock (compile), disparar (run), acompanhar (logs), detalhar (audit), atualizar da origem (update); para skills, install com --pin trava e update pula fixadas.

📝 **Como cai:** Escolha entre Autopilot, /fleet e dynamic workflow, ou entre skill, MCP e instruções.
Como resolver: Processo fixado pelo time em código: dynamic workflow; acesso a sistema externo: MCP; procedimento ocasional: skill; orientação ampla e curta: instruções.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Onde ficam as instruções do agentic workflow	**Corpo Markdown, em linguagem natural**
Quais escritas o agente pode fazer	**safe-outputs no frontmatter**
Evitar que uma execução gaste créditos demais	**max-ai-credits (padrão 1.000 AIC)**
Trazer melhorias de um workflow importado	**source no frontmatter + gh aw update**
Preparar o repositório para criar workflows com agente de código	**gh aw init**
Motor de IA de terceiros	**engine + secret de autenticação do motor**
Automação só quando o PR mexer numa pasta	**Filtro de arquivos alterados no gatilho de PR**
Lógica de triagem versionada e revisada em PR	**GitHub Agentic Workflow em vez de automação**
Conferir agora se a automação funciona	**Run now no painel Automations**
Automação do app às 7h com notebook desligado	**Run in the cloud**
Horário preciso de segunda a sexta numa automação local	**Gatilho CRON**
Dados ou ações num sistema externo	**Servidor MCP (e skill para ensinar o passo a passo)**
Processo repetido com etapas e condições definidas pelo time	**Dynamic workflow**
Segunda opinião independente sobre o plano	**Rubber duck, com modelo diferente**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Agentic workflow é versionado no repositório e compilado em .lock.yml; automação do Copilot fica fora do repositório e não passa por PR.
- permissions padrão read-all; escrita só por safe-outputs, depois de detecção de ameaças.
- Automação: repositório privado ou interno, escopo de um repositório, privada e cobrada de quem criou.
- O cloud agent usa ferramentas MCP sem pedir aprovação.
- Skill fixada com --pin não é atualizada por gh skill update.

Essenciais para a prova (checklist)

- ☐ Sei montar a estrutura de um agentic workflow e explicar o papel do .md e do .lock.yml.
- ☐ Sei para que servem on, permissions, safe-outputs, engine, tools, max-ai-credits e source.
- ☐ Sei os guardrails de segurança dos agentic workflows.
- ☐ Sei escolher entre GITHUB_TOKEN com copilot-requests: write e COPILOT_GITHUB_TOKEN.
- ☐ Sei o que fazem init, add-wizard, compile, run, logs, audit e update do gh aw.
- ☐ Sei os gatilhos, filtros, requisitos, escopo e cobrança das automações do Copilot.
- ☐ Sei diferenciar automações locais e na nuvem no app e os gatilhos disponíveis.
- ☐ Sei criar uma skill, chamá-la e controlar versões com gh skill --pin.
- ☐ Sei quando usar MCP e quais servidores o cloud agent traz por padrão.
- ☐ Sei o que o rubber duck faz e quando usar dynamic workflow, /fleet ou Autopilot.

Questões de revisão do capítulo

R2.1 · Projetar e implementar fluxos de trabalho e automações agênticos

Ao criar uma automação na aba Agents do repositório da Papelaria Central, o analista não sabe quais ferramentas liberar para o prompt de rotular issues. Que recurso pode ajudar?

A. O botão Suggest tools, que sugere ferramentas a partir do prompt
B. O modo Autopilot, que libera as ferramentas conforme a necessidade
C. O comando /context, que lista as ferramentas usadas na última sessão
D. A política MCP servers in Copilot, que escolhe as ferramentas certas

R2.2 · Projetar e implementar fluxos de trabalho e automações agênticos

O gerente de vendas da Moda Prisma criou uma automação no repositório da loja. Um colega, administrador do repositório, diz que não a enxerga na lista. O que a documentação explica?

A. Automações ficam ocultas até alguém aprovar o primeiro pull request delas
B. Automações são privadas de quem as criou, mas as sessões delas são visíveis
C. Automações só aparecem para administradores depois da primeira execução
D. Automações aparecem na wiki do repositório, e não na aba Agents

R2.3 · Projetar e implementar fluxos de trabalho e automações agênticos

A administradora do repositório da Rede Atacado Bom configurou um servidor MCP de estoque nas configurações do repositório no GitHub. Onde essa configuração passa a valer?

A. No Copilot CLI de quem clonar o repositório, sem outra configuração
B. No Copilot cloud agent e no Copilot code review do repositório
C. No Copilot Chat de todos os membros da organização no navegador
D. Nas automações de outros repositórios da mesma organização

R2.4 · Projetar e implementar fluxos de trabalho e automações agênticos

A analista da Rede Escolar Saber Mais criou uma automação no GitHub Copilot app e quer que o prompt use a skill de boletins do repositório. Como faz no campo do prompt?

A. Digita # e o nome do arquivo SKILL.md
B. Digita @ e o nome do agente personalizado
C. Digita o caminho completo da pasta da skill no repositório
D. Digita / para escolher uma skill disponível

Respostas comentadas

R2.1. A) O botão Suggest tools, que sugere ferramentas a partir do prompt

Por quê: As ferramentas escolhidas definem o que a automação pode fazer. Selecione só as necessárias, manualmente ou com o botão Suggest tools.

✗ B) Incorreta. Autopilot é modo de sessão do app e do CLI; não define ferramentas de automações.
✗ C) Incorreta. O /context mostra o uso de contexto da sessão atual; não configura automações.
✗ D) Incorreta. Essa política controla o uso de servidores MCP; não seleciona ferramentas para uma automação.

> **Dica de prova:** Dúvida sobre ferramentas da automação: Suggest tools.

R2.2. B) Automações são privadas de quem as criou, mas as sessões delas são visíveis

Por quê: Visibilidade: a automação é privada de quem a criou, mas as sessões, os prompts, os logs e os PRs que ela gera ficam visíveis a quem tem acesso ao repositório.

✗ A) Incorreta. Aprovar PRs não muda a visibilidade da automação.
✗ C) Incorreta. Administradores não veem automações de outras pessoas.
✗ D) Incorreta. Automações não são publicadas na wiki.

> **Dica de prova:** Automação privada; sessões visíveis.

R2.3. B) No Copilot cloud agent e no Copilot code review do repositório

Por quê: Administradores de repositório configuram MCP nas configurações do repositório no GitHub; essa configuração vale para o cloud agent e o code review. Ela é diferente dos arquivos de configuração de MCP do CLI.

✗ A) Incorreta. O CLI usa arquivos de configuração próprios.
✗ C) Incorreta. O Chat do GitHub tem ferramentas pré-configuradas próprias, separadas dessa configuração.
✗ D) Incorreta. A configuração vale para o repositório onde foi feita.

> **Dica de prova:** MCP do repositório: cloud agent + code review.

R2.4. D) Digita / para escolher uma skill disponível

Por quê: Ao criar uma automação no app, descreva a tarefa no prompt (digite / para usar uma skill) e use os controles abaixo do prompt para modelo, esforço de raciocínio e agente.

✗ A) Incorreta. # referencia issues no app.
✗ B) Incorreta. @ adiciona arquivos; agentes são escolhidos nos controles abaixo do prompt.
✗ C) Incorreta. Não é preciso digitar o caminho da pasta.

> **Dica de prova:** Skill no prompt da automação do app: digite /.

03

CAPÍTULO 3 · 20-25% DA PROVA

Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

Aqui eu preciso saber onde a pessoa entra no ciclo: escolher o modo de trabalho (Interactive, Plan, Autopilot), definir permissões de ferramentas, aprovar ou recusar pedidos, redirecionar um agente em execução, parar e desfazer, e revisar a saída antes do merge. A prova testa muito que Autopilot e permissões são controles separados, o que cada tecla faz no CLI (Enter, Ctrl+Enter, Esc, Ctrl+C) e o que acontece com o trabalho quando paro uma sessão. Também cai o pacote de mitigações do cloud agent e das automações: quem pode aprovar o PR, por que os workflows não rodam sozinhos, aprovação extra e rastreabilidade dos commits. Nas automações de triagem, preciso decorar os quatro níveis de automação e que Cautious é o padrão. Erro quando acho que parar ou cancelar desfaz o trabalho ou que aprovações são uma fronteira de segurança.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

Responsabilidade humana em todos os níveis

Agentes trabalham em sessão colaborativa, assíncrona ou em fluxo automatizado, e em todos a pessoa define objetivo e limites, concede acesso e revisa o resultado. Delegar não transfere a responsabilidade ao GitHub nem ao agente.

Modos Interactive, Plan e Autopilot

Interactive: você e o agente trabalham juntos, para explorar o problema ou quando espera decisões frequentes. Plan: o agente faz perguntas, define escopo e monta um plano que você aprova antes de executar, pegando mal-entendidos antes de qualquer mudança. Autopilot: execução autônoma para tarefa com resultado claro e detalhe suficiente. No CLI, Shift+Tab alterna os modos.

Fluxo Plan e depois Autopilot

Descreva objetivo, restrições e como a conclusão será validada; revise o plano e peça ajustes até atender aos requisitos; escolha Accept plan and continue in Autopilot execution mode; acompanhe e, no fim, revise mudanças e testes como em qualquer outra mudança. O prompt deve trazer o resultado esperado e as checagens, com condição de parada clara.

Autopilot não é permissão

Autopilot controla a continuidade, não concede acesso irrestrito; se uma ação necessária não estiver permitida, o agente pode não terminar. Ao entrar no Autopilot sem todas as permissões, o CLI oferece habilitar todas, Use Manual Approval for this session (pedidos que exigiriam aprovação são negados) ou cancelar. Três controles separados: Autopilot (continuidade), --allow-all ou --yolo (aprovações) e --no-ask-user (sem perguntas de esclarecimento).

Permissões de ferramentas no app

Três níveis: Ask every time (mais supervisão), Assisted (um modelo aprova o seguro e pergunta o duvidoso) e Approve all (sem aprovações individuais). Escolhe-se por sessão, abaixo do prompt, ou como padrão para novas sessões locais em Settings, Sessions, Tool permissions.

Permissões assistidas

Um modelo avalia pedidos elegíveis ainda não resolvidos: se confiar que é seguro, aprova; se não, pede sua aprovação. No Autopilot e em execuções não interativas, o que exigiria aprovação é negado. Regras existentes, aprovações humanas e regras de negação da enterprise vêm antes do modelo. Nas configurações gerenciadas, disableBypassPermissionsMode e disableAssistedPermissionsMode são independentes.

Aprovação manual e regras de ferramenta no CLI

Na aprovação manual, comandos que não são só leitura, edição de arquivo e acesso a URL pedem aprovação, que pode valer uma vez ou para o resto da sessão. Aprovações salvas ficam em arquivos de configuração; --allow-tool e --deny-tool valem só para a sessão atual, e negação tem precedência. --available-tools e --excluded-tools definem quais ferramentas o modelo conhece (por exemplo, excluir web_fetch e web_search).

Hooks como caminho de escalonamento

Hooks executam comandos em pontos do fluxo do agente. O preToolUse pode aprovar ou negar ferramentas, bloquear comandos fora de uma lista permitida e registrar uso antes da execução; sessionEnd roda no fim e é tarde demais para bloquear.

Teclas e direcionamento no CLI

Um prompt enviado com Enter enquanto o Copilot trabalha vira steering da tarefa atual; Ctrl+Enter ou Ctrl+Q enfileira para depois. Esc com diálogo de permissão aberto nega o pedido; com o Copilot trabalhando, o primeiro Esc mostra lembrete e o segundo cancela; ocioso, dois Esc abrem o rewind. Ctrl+C cancela sem a segunda confirmação e limpa a fila. Ao recusar uma permissão, dá para dar feedback inline dizendo o que fazer no lugar.

Cancelar, parar e desfazer

Cancelar (Esc ou Ctrl+C) para o trabalho em andamento: uma escrita de arquivo em curso termina sem corromper e o resto é abandonado, mas nada concluído é desfeito; para isso existe o rewind. No Autopilot, parar o agente também não desfaz as mudanças. Stop session no cloud agent encerra a execução do GitHub Actions e preserva os commits já enviados.

Steering e acompanhamento de sessões

No cloud agent, o follow-up escrito abaixo do log é aplicado depois da chamada de ferramenta atual, sem reiniciar; cada mensagem consome AI credits, e o recurso não existe para agentes de código de terceiros. No CLI, /ask faz pergunta paralela que não usa ferramentas nem entra no histórico (indisponível em sessões remotas). Remote control (/remote, sujeito a política) permite ver progresso e responder perguntas e permissões pelo GitHub.com ou GitHub Mobile. Limite de AI credits por sessão é suave: /limits set ajusta e a sessão continua; /limits unset remove.

Sessões: visibilidade, arquivamento e exclusão

Sessões do cloud agent são compartilhadas por padrão com quem acessa o repositório e só podem ser arquivadas, não excluídas. Sessões locais (CLI, VS Code, JetBrains, app) não são compartilhadas por padrão, podem ser compartilhadas só para visualização pela aba Agents e podem ser excluídas. Consultar o histórico em linguagem natural cobre apenas as sessões que você iniciou.

Mitigações do cloud agent

Só quem tem escrita aciona o agente, e comentários de quem não tem escrita não chegam a ele; caracteres ocultos (como comentários HTML) são filtrados; o acesso à internet passa por firewall; ele envia a um único branch (como copilot/) com credenciais de envio simples e segue rulesets e proteções. Abre PR em rascunho e não pode marcá-lo como pronto, aprová-lo nem fazer merge. Antes de concluir, valida o próprio trabalho com CodeQL, secret scanning, Advisory Database e code review.

Revisão do PR feito pelo agente

Revise com o mesmo cuidado de qualquer contribuição. Workflows do Actions não rodam quando o Copilot envia mudanças: alguém com escrita revisa e clica em Approve and run workflows, olhando com atenção mudanças em .github/workflows. Quem pediu o PR não pode aprová-lo para cumprir Required approvals; se o PR não for atribuído a uma pessoa, uma aprovação extra é exigida (em rulesets vem habilitada por padrão). Peça mudanças com @copilot num comentário ou commits no branch, e dê feedback com os botões de polegar.

Rastreabilidade e o PR que não aparece

Commits do cloud agent têm o Copilot como autor e quem pediu como coautor, são assinados (Verified) e trazem link para o log da sessão; administradores têm logs e eventos de auditoria. No fluxo pesquisar, planejar e iterar, a sessão não cria PR sozinha: revise em Diff e clique em Create pull request (ou peça o PR no prompt). O Copilot Chat responde sobre o PR do agente usando os logs da sessão, sem substituir sua revisão.

Justificativa, confiança e aprovações em issues

Vale para automações do cloud agent, agentic workflows e APIs, e cobre labels, campos, tipo, fechamento e responsáveis de issues. O nível do repositório (Settings, Planning, Agent suggestions for issues) define o limiar: Full control (tudo para revisão), Cautious (padrão; aplica só alta confiança), Balanced (aplica o rotineiro) e Full automation (aplica tudo, salvo o incerto). Para segurar uma mudança específica, peça no prompt que o agente sugira; o ícone de informação mostra a justificativa e o painel tem Accept all e Decline all.

Aprovações não são fronteira de segurança

Aprovações seguram mudanças de baixa confiança para revisão, mas quem limita o que a automação pode fazer são as permissões do repositório e do agente; um agente com permissão pode mudar issues direto, inclusive pelas APIs. Recusar uma sugestão não impede execuções futuras. Mitigações das automações: trabalho atribuído a quem criou (que não aprova os próprios PRs), eventos de quem não tem escrita ignorados por padrão e workflows que exigem aprovação humana para rodar em PRs.

Monitoramento

Na barra lateral do CLI, círculos indicam ocupada, ociosa ou concluída, ! indica pedido de permissão, ? indica pergunta e cores indicam os modos plan e autopilot. No app, cada cartão de Automations mostra nome, agenda, repositório e status da última execução. Agentic workflows: gh aw logs e gh aw audit. Na enterprise, AI controls traz Agent sessions (recentes e últimas 24 horas) e Audit logs (com streaming), além de OpenTelemetry e API de métricas de uso.

Macetes para memorizar

Quando usar cada modo

Explorar = Interactive; revisar a rota = Plan; rota aprovada = Autopilot.

Três controles independentes

Autopilot anda, --allow-all libera, --no-ask-user cala.

Teclas do CLI

Enter corrige, Ctrl+Enter enfileira, Esc nega ou freia, Ctrl+C mata a fila, Shift+Tab troca o modo.

Parar x desfazer

Parar segura o que vem, não apaga o que veio; para apagar, rewind.

Níveis de automação

Do mais humano ao mais automático: Full control, Cautious (padrão), Balanced, Full automation.

Quem aprova o PR do agente

Agente não aprova, quem pediu não aprova, sem dono humano pede uma aprovação a mais.

Comandos de PR no app

/pr-open abre, /pr-fix-checks conserta checks, /pr-resolve-comments responde revisão, /pr-merge mescla.

Tabela de memorização: não confunda

Quando é...	...e quando é
Interactive Requisitos ainda instáveis, exploração ou decisões frequentes.	**Autopilot** Plano já detalhado e resultado claro, validado por testes.
Modo Plan Modo de sessão: o agente monta o plano e só executa depois da sua aprovação.	**Permissão Assisted** Opção de permissão de ferramentas: um modelo aprova o seguro e pergunta o duvidoso.
Enter durante o trabalho Redirecionar a tarefa atual (steering).	**Ctrl+Enter (ou Ctrl+Q)** Enfileirar uma instrução para depois, sem interferir.
Esc Controle gradual: nega diálogo de permissão; com o Copilot ativo, o primeiro mostra lembrete e o segundo cancela.	**Ctrl+C** Cancelar de uma vez, sem confirmação extra, limpando a fila.
Cancelar ou parar Interromper o trabalho em andamento, mantendo o que já foi concluído.	**Rewind** Desfazer trabalho concluído, voltando conversa (e arquivos, se escolhido).
Sessão do cloud agent Compartilhada por padrão; pode ser arquivada, não excluída.	**Sessão local** Não compartilhada por padrão; compartilhamento só para visualização; pode ser excluída.

Full control Nenhuma mudança de triagem entra sem revisão humana.	**Cautious** Padrão: só mudanças de alta confiança são aplicadas sozinhas.

Números que caem

Item	Valor
Níveis de automação para triagem de issues	**Quatro: Full control, Cautious (padrão), Balanced e Full automation**
Agent sessions em AI controls da enterprise	**Sessões recentes e das últimas 24 horas**
Peso do domínio no exame	**20-25%**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Achar que o Autopilot libera as ferramentas de que o agente precisar.
Como não cair: Autopilot é continuidade; permissões são outro controle, e sem elas o agente pode não concluir.

⚠ **Alerta de prova:** Supor que, com Use Manual Approval no Autopilot, o pedido fica esperando a resposta.
Como não cair: O pedido é negado, o que pode impedir o agente de terminar.

⚠ **Alerta de prova:** Acreditar que o modelo das permissões assistidas pode aprovar um comando que a enterprise nega.
Como não cair: Regras de negação da enterprise vêm antes; o comando é negado.

⚠ **Alerta de prova:** Pensar que --deny-tool fica salvo para as próximas sessões.
Como não cair: --allow-tool e --deny-tool valem só para a sessão atual.

⚠ **Alerta de prova:** Usar --mode autopilot ou --allow-all para evitar perguntas de esclarecimento.
Como não cair: Sem perguntas e sem dar autonomia nem ferramentas: --no-ask-user (--yolo é alias de --allow-all).

⚠ **Alerta de prova:** Achar que Ctrl+C ou parar no Autopilot reverte o que o agente já fez.
Como não cair: Cancelar e parar não desfazem; o concluído só volta com rewind.

⚠ **Alerta de prova:** Supor que Stop session apaga commits ou exclui a sessão do cloud agent.
Como não cair: Encerra a execução no Actions, mantém os commits enviados, e a sessão só pode ser arquivada.

⚠ **Alerta de prova:** Achar que colega com quem compartilhei a sessão local pode direcioná-la ou aprovar permissões.
Como não cair: Compartilhamento é só de visualização.

⚠ **Alerta de prova:** Esperar o PR do cloud agent criado automaticamente ao fim da sessão.
Como não cair: Revise em Diff e clique em Create pull request, ou peça o PR no prompt.

⚠ **Alerta de prova:** Achar que quem pediu o PR ao agente, ou quem criou a automação, pode aprovar o PR.
Como não cair: A aprovação dele não conta; outra pessoa precisa aprovar.

⚠ **Alerta de prova:** Contar o parecer do Copilot code review como aprovação obrigatória.
Como não cair: Por padrão não conta; só se a opção de aprovações do Copilot for habilitada.

⚠ **Alerta de prova:** Marcar Ready for review ou comentar @copilot para os checks rodarem.
Como não cair: É preciso Approve and run workflows, depois de revisar as mudanças.

⚠ **Alerta de prova:** Tratar aprovações das automações como controle de segurança ou achar que cobrem pushes e PRs.
Como não cair: São conveniência de fluxo e cobrem só mudanças em issues; o limite real são as permissões.

⚠ **Alerta de prova:** Dizer que Balanced ou Full automation é o nível padrão.
Como não cair: O padrão do repositório é Cautious.

Como cai na prova

📝 **Como cai:** Cenário descreve o grau de clareza da tarefa e pede o modo de sessão.
Como resolver: Requisitos instáveis ou exploração levam a Interactive; abordagem a revisar antes leva a Plan; plano pronto com validação leva a Autopilot; descarte Assisted e Approve all, que são permissões.

📝 **Como cai:** Agente parou ou não concluiu no Autopilot e a pergunta é por quê.
Como resolver: Verifique se faltou permissão: Autopilot não concede acesso, e com aprovação manual os pedidos são negados.

📝 **Como cai:** Pessoa aperta uma tecla ou para uma sessão e a pergunta é o que acontece com o trabalho.
Como resolver: Cancelar e parar interrompem, mas preservam o concluído e os commits; só o rewind desfaz; Stop session encerra a execução do Actions.

📝 **Como cai:** Regra de aprovação obrigatória com PR do cloud agent ou de automação.
Como resolver: Agente não aprova; quem pediu ou quem criou a automação não conta; PR sem pessoa atribuída exige aprovação extra; parecer do Copilot code review não conta por padrão; workflows precisam de Approve and run workflows.

📝 **Como cai:** Triagem automática de issues com pedido de mais ou menos revisão humana.
Como resolver: Escolha o nível certo (Full control, Cautious padrão, Balanced, Full automation) em Settings, Planning, Agent suggestions for issues; para uma mudança específica, peça sugestão no prompt; lembre que permissões, e não aprovações, limitam a automação.

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Ainda explorando, decisões frequentes	**Interactive**
Montar plano e só executar depois de aprovado	**Modo Plan**
Plano revisado, implementar sem esperar a cada passo	**Accept plan and continue in Autopilot**
Modelo aprova o seguro e pergunta o arriscado	**Assisted (permissões assistidas)**
Script verifica cada comando contra lista permitida antes de executar	**Hook preToolUse**
Mandar instrução para depois da tarefa atual	**Ctrl+Enter (ou Ctrl+Q)**
Negar rápido um pedido de permissão	**Esc uma vez**
Recusar e dizer o que fazer no lugar	**Feedback inline na recusa da permissão**
Acompanhar e responder permissões pelo celular	**Remote control (GitHub.com ou GitHub Mobile)**
Sessão parou no limite de AI credits e quer continuar	**/limits set**
Símbolo ! na barra lateral de sessões	**Pedido de permissão aguardando resposta**

Checks não rodaram no PR do agente	**Approve and run workflows**
Automação não reagiu a issue de voluntário sem escrita	**Eventos de quem não tem escrita são ignorados por padrão**
Commits do agente não alterados depois de feitos	**Assinados e marcados como Verified**

🎯 **Se você lembrar de só uma coisa deste capítulo...**

- Autopilot e permissões são controles separados; Autopilot não libera ferramentas.
- Cancelar, parar e Stop session não desfazem trabalho concluído; commits enviados ficam.
- O agente não aprova nem mescla PR, e quem pediu o trabalho não conta como aprovação.
- Nível padrão de automação das issues é Cautious; aprovações não são fronteira de segurança.
- Workflows não rodam no PR do agente até alguém com escrita clicar em Approve and run workflows.

Essenciais para a prova (checklist)

- ☐ Sei escolher entre Interactive, Plan e Autopilot pelo cenário.
- ☐ Sei descrever o fluxo Plan e depois Autopilot e o que o prompt deve conter.
- ☐ Sei diferenciar Ask every time, Assisted e Approve all e onde configurar o padrão.
- ☐ Sei como permissões assistidas interagem com regras de negação da enterprise.
- ☐ Sei o que fazem Enter, Ctrl+Enter, Esc, Ctrl+C e Shift+Tab no CLI.
- ☐ Sei o que acontece ao cancelar, parar e usar Stop session.
- ☐ Sei redirecionar uma sessão do cloud agent e usar /ask, remote control e /limits set.
- ☐ Sei as mitigações do cloud agent e as regras de aprovação do PR que ele cria.
- ☐ Sei os quatro níveis de automação, o padrão e onde configurá-los.
- ☐ Sei onde monitorar sessões e execuções no CLI, no app, no gh aw e na enterprise.

Questões de revisão do capítulo

R3.1 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

A automação de triagem da Clínica Bem Viver deixou três mudanças pendentes numa issue: um label, um tipo e um responsável. A coordenadora concorda com todas. Qual é a forma mais rápida de agir?

A. Reexecutar a automação depois de mudar o nível para Full automation
B. Clicar em Accept all no painel de sugestões da issue
C. Aplicar cada mudança editando a issue à mão
D. Comentar @copilot approve na issue para aplicar

R3.2 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

A coordenadora da Escola Novo Saber vê, numa issue, um label aplicado pela automação de triagem com um ícone de informação ao lado. O que esse ícone mostra?

A. O custo em AI credits que a automação gastou nessa mudança
B. A justificativa registrada pela automação para essa mudança
C. O nome do modelo que a automação usou para essa mudança
D. A lista de pessoas que aprovaram essa mudança na issue

R3.3 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

Uma automação da Rede Lojas Prisma abriu um pull request, e o PR disparou outra automação, que abriu outro PR. O gerente teme que workflows do Actions rodem em cadeia sem controle. O que a documentação garante?

A. Workflows não rodam nesses PRs até alguém com escrita aprovar
B. Workflows rodam e o custo de cada execução vai para o owner da organização
C. Workflows rodam e são revertidos se o merge for recusado
D. Workflows rodam em cadeia, mas param depois de três PRs

R3.4 · Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

A diretoria jurídica da Contábil Exata quer que nenhuma mudança das automações de triagem nas issues entre sem revisão humana. Qual nível de automação atende?

A. Cautious
B. Balanced
C. Full automation
D. Full control

Respostas comentadas

R3.1. B) Clicar em Accept all no painel de sugestões da issue

Por quê: Sugestões de automações aparecem num painel na issue, onde você pode aceitar ou recusar cada uma, ou usar Accept all e Decline all para agir em várias sugestões.

✗ A) Incorreta. Mudar o nível e reexecutar é desnecessário; as sugestões já estão prontas para aprovação.
✗ C) Incorreta. Editar à mão funciona, mas ignora o painel feito para aprovar as sugestões de uma vez.
✗ D) Incorreta. Não há comando de comentário para aprovar sugestões; a aprovação é feita no painel.

> **Dica de prova:** Várias sugestões corretas: Accept all.

R3.2. B) A justificativa registrada pela automação para essa mudança

Por quê: Cada ação suportada registra o motivo, aplicada automaticamente ou sugerida. Isso cria uma trilha de auditoria; o ícone de informação revela a justificativa.

✗ A) Incorreta. O ícone não mostra custo.
✗ C) Incorreta. O ícone mostra a justificativa, não o modelo.
✗ D) Incorreta. O ícone não lista aprovadores.

> **Dica de prova:** Ícone de informação na mudança: justificativa da automação.

R3.3. A) Workflows não rodam nesses PRs até alguém com escrita aprovar

Por quê: Mitigações das automações: trabalho atribuído a quem criou, ferramentas escolhidas, eventos de quem não tem escrita ignorados por padrão e workflows que exigem aprovação humana para rodar em PRs.

✗ B) Incorreta. A garantia é a aprovação humana antes de rodar.
✗ C) Incorreta. Não há reversão automática.
✗ D) Incorreta. Não existe limite de três PRs.

> **Dica de prova:** Cadeia de automações: workflows só rodam com aprovação.

R3.4. D) Full control

Por quê: Níveis de automação: Full control (tudo para revisão), Cautious (padrão; só alta confiança aplica), Balanced (rotineiro aplica) e Full automation (tudo aplica, salvo incerto).

✗ A) Incorreta. Cautious aplica mudanças de alta confiança.
✗ B) Incorreta. Balanced aplica mudanças rotineiras e claras.
✗ C) Incorreta. Full automation aplica tudo, salvo o que for incerto.

> **Dica de prova:** Nada sem revisão: Full control.

04

CAPÍTULO 4 · 20-25% DA PROVA

Usar os recursos da plataforma GitHub

Aqui eu preciso dominar duas frentes que a prova mistura: governança do Copilot (AI credits, orçamentos, políticas, secrets, modelos, IA responsável, instruções, skills e MCP) e o trabalho do dia a dia no GitHub (Projects, issues, pull requests, rulesets e wikis). As questões vêm como cenário de gestor ou de PMO e pedem o recurso certo ou o comportamento padrão documentado. O que mais derruba é a cobrança por uso: qual orçamento limita o quê, quando ele começa a valer (no pool ou só depois dele) e qual vence quando há vários. Outro ponto fraco clássico é confundir o que orienta o modelo (instruções) com o que de fato controla (políticas, exclusão de conteúdo, hooks, CODEOWNERS). Os limites numéricos de issues e projects (3, 10, 100, 8, 25, 6) caem como alternativas trocadas entre si.

Neste capítulo

18 tópicos-chave · 14 pegadinhas · 7 macetes · 14 gatilhos · 4 questões comentadas

Tópicos-chave

AI credits e custo de uma interação

Na cobrança por uso de organizações e enterprises, 1 AI credit = US$ 0,01; orçamentos são definidos em dólares e o consumo aparece em créditos (US$ 10 cobrem 1.000 créditos; US$ 25 cobrem 2.500). O custo depende do modelo e dos tokens de entrada, de saída e de cache. Não há tarifa por horário nem influência do número de pessoas com acesso ao repositório.

Pool de créditos incluídos

Cada licença Business ou Enterprise traz créditos mensais que ficam em pool na entidade de cobrança: o uso leve de uns compensa o uso intenso de outros. O pool reinicia com o valor cheio às 00:00:00 UTC do dia 1º de cada mês, e o saldo não usado é perdido (não acumula, não vira desconto, não fica guardado).

Uso adicional e política AI credits paid usage

Quando o pool acaba, se o uso adicional for permitido o consumo continua pelas tarifas publicadas; se não for, o uso é bloqueado até o próximo ciclo. O padrão é permitido. Para garantir zero gasto extra, o admin desabilita a política AI credits paid usage.

Orçamento por usuário (user-level budget)

Limita quantos AI credits uma pessoa consome no ciclo, tanto do pool quanto do uso adicional, e é sempre parada rígida: não existe opção de só alertar, e US$ 0 bloqueia na hora. Esgotado, o acesso é interrompido mesmo com saldo no pool, e não há troca automática para modelo mais barato. Existem três tipos: universal, de cost center e individual, e o mais específico vence (individual > cost center > universal); eles não se somam.

Expiração de orçamento individual

Só o orçamento individual por usuário pode ter expiração: sem expiração (padrão), ao fim do ciclo atual ou numa data específica. Serve para aumentos temporários, como numa sprint ou num incidente, sem depender de alguém desfazer depois.

Orçamento de cost center e de organização

Os dois limitam cobranças de uso adicional (metered) do grupo e só passam a valer depois que o pool compartilhado se esgota. O de organização cobre os usuários licenciados por aquela organização e só pode restringir abaixo do que a enterprise definiu. O orçamento de cost center não estende nem substitui o orçamento por usuário.

Controle de uso incluído do cost center

Limita quanto um cost center consome do pool compartilhado ao valor financiado pelas licenças dele, antes mesmo de o pool acabar; o GitHub define e ajusta esse limite. Ao atingir o limite, você escolhe bloquear ou deixar seguir como uso pago.

O que consome e o que não consome AI credits

Consomem: Copilot Chat, Copilot CLI, cloud agent, Copilot Spaces, Spark e agentes de terceiros. Ficam fora e são ilimitadas nos planos pagos: completions de código e next edit suggestions. Para medir e reduzir tokens: /chronicle cost-tips, visão geral da sessão do agente (uso de tokens e duração), /context no app e, em agentic workflows, gh aw logs e gh aw audit.

Disponibilidade e tipos de modelo

Os modelos que cada pessoa vê dependem do plano, da superfície ou cliente e das políticas dos administradores, além da disponibilidade ao longo do tempo. O Auto respeita as políticas e não usa modelos excluídos. Modelos utilitários rodam em segundo plano (títulos de sessão, mensagens de commit) e não aparecem no seletor. No Business e no Enterprise há o modelo base (usado quando nenhum outro está habilitado) e o modelo LTS, suportado por um ano a partir da designação.

Comparação de modelos por tarefa

Rápidos e econômicos para dúvidas simples e edições repetitivas; raciocínio profundo para trade-offs de arquitetura e depuração com contexto em vários arquivos; multimodais para diagramas e capturas de tela (anexar imagem exige modelo que aceite imagem). A regra é usar a capacidade que a tarefa exige, e não mais.

Seis princípios de IA responsável

Equidade (tratar grupos de forma justa, monitorar por segmento), confiabilidade e segurança (consistência, resistir a manipulação), privacidade e segurança (consentimento, coleta mínima, anonimização, criptografia), inclusão (envolver todos, acessibilidade), transparência (sistema compreensível, documentar funcionamento e limitações) e responsabilização (pessoas respondem pelos resultados).

Instruções personalizadas e personalizações

Instruções existem em três níveis: pessoais, de repositório (.github/copilot-instructions.md e NOME.instructions.md em .github/instructions para caminhos específicos) e de organização (linguagem, segurança, compliance). Elas orientam, mas não garantem cumprimento. Prompt files são acionados sob demanda; skills guiam tarefas; hooks executam código em eventos definidos (registro garantido); MCP conecta sistemas externos.

Exclusão de conteúdo

Recurso do Business e do Enterprise configurado por admins de repositório, owners de organização e owners de enterprise. Nos arquivos excluídos não há sugestões inline, o conteúdo não informa respostas nem sugestões em outros arquivos e não passa pelo Copilot code review. Não é suportada nos modos Edit e Agent do Copilot Chat no VS Code, e os arquivos continuam no repositório.

Secrets para o cloud agent e para agentic workflows

O cloud agent usa secrets e variáveis do tipo Agents (organização ou repositório), expostos como variáveis de ambiente e mascarados nos logs; com o prefixo COPILOT_MCP_ o valor fica só para servidores MCP. O que o agente não deve ver continua em Actions secrets. Nunca coloque segredo no prompt de automação, pois sessões e logs são visíveis. Em agentic workflows, os secrets ficam fora do runtime do agente, em jobs isolados.

Políticas do Copilot que a prova cobra

MCP servers in Copilot vem desabilitada e vale só para licenças Business e Enterprise. A política do GitHub Copilot app é separada da política do Copilot CLI (desligar uma não desliga a outra). Store local sessions in the Cloud, no mínimo como View from cloud, permite consultar sessões locais pelo GitHub.com, sem dar acesso aos administradores. Políticas da enterprise formam a base, e as organizações podem restringir mais.

Rulesets, CODEOWNERS e estratégia de branch

Vários rulesets podem valer no mesmo branch (camadas que se somam), o status de enforcement muda sem excluir o ruleset, há lista de bypass e quem tem leitura vê os rulesets ativos; na proteção de branch só uma regra vale por branch. Para exigir aprovação de mudanças em arquivos de instruções, agentes e MCP, use CODEOWNERS com Require review from Code Owners. O GitHub flow é: branch, commits, PR, revisão, merge e excluir o branch.

GitHub Projects

Projects nascem do zero (Table, Roadmap ou Board) ou de template, e você confere campos, views, workflows e insights antes de criar; owners recomendam até seis templates. Make a copy leva views, campos, workflows (exceto auto-add) e insights, mas não itens nem colaboradores. Workflows padrão: item fechado e PR com merge vão para Done; auto-add só pega itens criados ou atualizados depois. Papel base: No access, Read, Write ou Admin; status update registra status (como At risk), datas e mensagem.

Issues, pull requests e wikis

Até 10 responsáveis por issue ou PR, até três issues fixadas, até 100 sub-issues por pai com oito níveis, até 25 tipos de issue geridos por owners da organização; dependências blocked by e blocking exigem triage. Closes (ou fixes, resolves) seguido do número na descrição do PR fecha a issue no merge ao branch padrão; branch criado na seção Development da issue vincula o PR sozinho. Ready for review pede revisão dos code owners. Wiki é para documentação longa, com histórico, Compare Revisions e reversão por quem pode editar.

Macetes para memorizar

💡 Limites de issues e projects

3-10-100-8-25-6: fixadas 3, responsáveis 10, sub-issues 100 em 8 níveis, tipos de issue 25, templates recomendados 6.

💡 Qual orçamento por usuário vence

Quem é mais específico manda: Individual > Cost center > Universal. Nunca soma.

💡 Quando cada orçamento entra

Pessoa trava no pool e depois dele; grupo (cost center, organização) só trava depois do pool.

💡 Pool de créditos

Compartilha, não acumula e reinicia com o valor cheio no dia 1º às 00:00:00 UTC; o saldo não usado é perdido.

💡 Instrução x controle

Instrução sugere, hook executa, política proíbe, CODEOWNERS aprova.

💡 Secrets do cloud agent

Agente precisa: Agents. Agente não pode ver: Actions. Só o MCP usa: COPILOT_MCP_.

Seis princípios de IA responsável

Grupo tratado desigual = equidade; resiste a manipulação = confiabilidade; coleta mínima = privacidade; acessibilidade = inclusão; explicar como funciona = transparência; quem responde = responsabilização.

Tabela de memorização: não confunda

Quando é...	...e quando é
Orçamento por usuário (ULB) Limitar o consumo de cada pessoa, no pool e no uso adicional, com parada rígida.	**Orçamento do cost center** Limitar o gasto extra total do grupo, que só começa depois que o pool acaba. Os dois podem ser aplicados ao mesmo cost center.
Orçamento do cost center Teto de cobrança adicional do departamento depois do pool.	**Controle de uso incluído do cost center** Impedir que o departamento consuma do pool mais do que as próprias licenças financiam, antes de o pool acabar.
Rulesets Camadas de regras no mesmo branch (organização + repositório), pausar via enforcement sem excluir, bypass list, valem para branches e tags.	**Regras de proteção de branch** Proteção de branch, em que só uma regra se aplica por branch.
Agents secrets Valores que o cloud agent precisa usar (registro interno de pacotes, chaves de MCP com prefixo COPILOT_MCP_).	**Actions secrets** Valores que o agente não deve acessar; para eles, continue usando os secrets e variáveis do GitHub Actions.
Instruções personalizadas Orientar o modelo de forma persistente num escopo, sem garantia de cumprimento.	**Hooks** Executar código num evento definido, como registrar todo uso de ferramenta para auditoria, sem depender do modelo lembrar.
Instruções da organização Padrão comum de linguagem, segurança e compliance para todo uso no contexto da organização.	**Instruções pessoais** Preferências de uma pessoa só; não garantem padrão comum.
Modelo de raciocínio profundo Trade-offs de arquitetura e depuração com contexto em vários arquivos.	**Modelo rápido e econômico** Dúvidas simples de sintaxe e edições pequenas e repetitivas.

Números que caem

Item	Valor

Valor de 1 AI credit	**US$ 0,01**
Reinício do pool de créditos incluídos	**00:00:00 UTC do primeiro dia de cada mês**
Suporte do modelo LTS (Business e Enterprise)	**Um ano a partir da designação**
Issues fixadas por repositório	**Até três**
Responsáveis (assignees) por issue ou PR	**Até 10**
Sub-issues	**Até 100 por issue pai e até oito níveis de aninhamento**
Tipos de issue por organização	**Até 25**
Templates recomendados da organização	**Até seis**
Peso do domínio no exame	**20-25%**

Alertas de prova (pegadinhas)

⚠ **Alerta de prova:** Achar que, quando o orçamento acaba, o Copilot troca sozinho para um modelo mais barato.
Como não cair: Não há fallback automático: orçamento por usuário esgotado interrompe o acesso, mesmo com saldo no pool e em qualquer plano.

⚠ **Alerta de prova:** Supor que orçamento por usuário pode só gerar alerta ao administrador.
Como não cair: Orçamento por usuário é sempre parada rígida; US$ 0 bloqueia na hora.

⚠ **Alerta de prova:** Somar orçamentos universal, de cost center e individual da mesma pessoa.
Como não cair: Eles não se somam: vale o mais específico (individual > cost center > universal).

⚠ **Alerta de prova:** Colocar data de expiração no orçamento universal ou criar cost center temporário para um aumento pontual.
Como não cair: Expiração existe só no orçamento individual por usuário; é a solução para aumentos temporários.

⚠ **Alerta de prova:** Acreditar que o uso adicional vem desabilitado ou que o GitHub bloqueia sozinho quando o pool zera.
Como não cair: O padrão é uso adicional habilitado; para barrar gasto extra, desabilite a política AI credits paid usage.

⚠ **Alerta de prova:** Tratar créditos incluídos como cotas individuais ou como saldo que acumula para o mês seguinte.
Como não cair: É um pool compartilhado na entidade de cobrança que reinicia no dia 1º; saldo não usado é perdido.

⚠ **Alerta de prova:** Pensar que orçamento de cost center ou de organização limita o consumo do pool.
Como não cair: Ambos só valem depois que o pool acaba; quem limita a fatia do pool é o controle de uso incluído, e quem limita cada pessoa é o orçamento por usuário.

⚠ **Alerta de prova:** Achar que instruções da organização bloqueiam respostas, viram política ou geram registro no audit log.
Como não cair: Instrução orienta, não garante; controles efetivos vêm de políticas, exclusão de conteúdo, hooks e revisão humana.

⚠ **Alerta de prova:** Supor que o Auto pode escolher um modelo que o administrador desabilitou.
Como não cair: O Auto respeita políticas e plano; modelos excluídos ficam de fora, e a complexidade da tarefa não muda isso.

⚠ **Alerta de prova:** Dizer que a política MCP servers in Copilot vem habilitada ou que governa Copilot Pro.
Como não cair: Vem desabilitada e só vale para licenças Business e Enterprise; Free, Pro, Pro+ e Max não são governados por ela.

⚠ **Alerta de prova:** Achar que exclusão de conteúdo apaga os arquivos do repositório ou vale no agent mode do VS Code.
Como não cair: A exclusão afeta só o Copilot e os arquivos continuam no repositório; ela não é suportada nos modos Edit e Agent do Copilot Chat no VS Code.

⚠ **Alerta de prova:** Concluir que desabilitar a política do Copilot CLI bloqueia o GitHub Copilot app.
Como não cair: App e CLI têm políticas separadas no Business e no Enterprise.

⚠ **Alerta de prova:** Usar o prefixo COPILOT_MCP_ para esconder do agente um segredo que ninguém deve usar.
Como não cair: O prefixo só restringe o valor aos servidores MCP; o que o agente não pode ver vai em Actions secrets.

⚠ **Alerta de prova:** Trocar os limites: três responsáveis, dez issues fixadas, coluna que bloqueia card, auto-add que puxa issues antigas.
Como não cair: Fixadas: 3; responsáveis: 10; limite de coluna só destaca a contagem; auto-add só pega itens criados ou atualizados depois de ativado.

Como cai na prova

📝 **Como cai:** Cenário de cobrança com vários orçamentos e a pergunta qual limite vale ou o que acontece ao esgotar.
Como resolver: Identifique se o controle é por pessoa ou por grupo; por pessoa vale no pool e no adicional e é parada rígida; por grupo só depois do pool; se houver vários por usuário, aplique o mais específico; descarte qualquer alternativa com troca automática de modelo.

📝 **Como cai:** Gestor quer garantir uma regra (CPF, registro, aprovação) e a alternativa tentadora é escrever isso nas instruções.
Como resolver: Pergunte se o enunciado pede orientar ou garantir; para garantir, escolha política, exclusão de conteúdo, hook, ruleset ou CODEOWNERS.

📝 **Como cai:** Dilema de comitê de IA descrito em palavras de negócio, pedindo o princípio de IA responsável.
Como resolver: Procure a palavra-chave: desigual entre grupos, manipulação, dados pessoais, deficiência, explicar funcionamento, quem responde pelo resultado.

📝 **Como cai:** PMO quer um efeito em Projects ou issues e as alternativas misturam wiki, labels, iteração e fixar issue.
Como resolver: Mapeie o recurso nativo: status update para patrocinadores, roadmap para linha do tempo, Make a copy para estrutura, Convert to issue para draft, blocked by para dependência.

📝 **Como cai:** Questão de número exato com distratores que são outros limites reais.
Como resolver: Lembre a sequência 3-10-100-8-25-6 e desconfie de alternativa que usa o número de outro recurso (por exemplo, dez é de responsáveis, não de sub-issues).

Gatilhos do enunciado

Se o enunciado trouxer...	Pense em
Quantos créditos cobre um valor em dólares	**1 AI credit = US$ 0,01 (multiplique o valor por 100)**

Estagiário com orçamento US$ 0	**Orçamento por usuário é parada rígida: bloqueado na hora**
Aumento temporário durante incidente ou sprint	**Orçamento individual com data de expiração**
Departamento não pode consumir do pool mais que suas licenças	**Controle de uso incluído do cost center**
Ninguém pode gerar cobrança depois que o pool acabar	**Desabilitar a política AI credits paid usage**
Por que pessoas veem modelos diferentes	**Plano + superfície ou cliente + políticas**
Estabilidade de modelo no Business ou Enterprise	**Modelo LTS (um ano a partir da designação)**
Registrar toda vez que o agente usar ferramenta, sem depender do modelo	**Hook**
Mudanças em instruções, agentes e MCP só com aprovação da equipe de plataforma	**CODEOWNERS + Require review from Code Owners**
Regra da organização e regra do repositório no mesmo branch	**Rulesets em camadas**
Desligar um ruleset por uns dias sem perder a configuração	**Mudar o status de enforcement**
Reaproveitar views e campos sem levar itens nem pessoas	**Make a copy do project**
Issue que só anda depois de outra	**Dependência blocked by**
Fechar a issue automaticamente quando a correção entrar no branch padrão	**Closes #número na descrição do PR**

🎯 Se você lembrar de só uma coisa deste capítulo...

- Orçamento por usuário é parada rígida e nunca troca para modelo mais barato; orçamentos de grupo só valem depois do pool.
- Uso adicional vem habilitado por padrão; quem corta é a política AI credits paid usage.
- Instrução orienta e não garante; garantia vem de política, exclusão de conteúdo, hook, ruleset ou CODEOWNERS.
- Completions e next edit suggestions não consomem AI credits nos planos pagos.
- Política MCP servers in Copilot: desabilitada por padrão, só Business e Enterprise.

Essenciais para a prova (checklist)

☐ Sei converter dólares em AI credits e explicar que o custo depende de modelo e tokens.

☐ Sei dizer qual orçamento vence entre universal, de cost center e individual e quando cada um passa a valer.

☐ Sei diferenciar orçamento do cost center de controle de uso incluído do cost center.

- ☐ Sei onde guardar segredos que o agente usa e os que ele não pode ver, e para que serve COPILOT_MCP_.
- ☐ Sei relacionar cada um dos seis princípios de IA responsável a um cenário.
- ☐ Sei escolher entre modelo rápido, de raciocínio e multimodal e explicar base, LTS, Auto e utilitário.
- ☐ Sei explicar por que rulesets permitem camadas e como pausar um sem excluir.
- ☐ Sei os limites 3, 10, 100, 8, 25 e 6 de issues e projects.
- ☐ Sei o que Make a copy leva e não leva, e quais workflows de project vêm ligados por padrão.
- ☐ Sei vincular PR a issue para fechá-la no merge e usar a wiki com histórico e reversão.

Questões de revisão do capítulo

R4.1 · Usar os recursos da plataforma GitHub

A gerente de engenharia da Plataforma Viaje Já quer combinar uma regra da organização que exige pull request com outra regra do repositório que exige checks, no mesmo branch principal. O que torna isso possível com rulesets?

A. Rulesets da organização desligam as regras definidas no repositório
B. Cada branch aceita um só ruleset, que precisa reunir as duas regras
C. Rulesets só valem para tags, e branches usam proteção de branch
D. Vários rulesets podem valer ao mesmo tempo para o mesmo branch

R4.2 · Usar os recursos da plataforma GitHub

A enterprise do Grupo Vale Forte vai liberar o Copilot cloud agent. A diretoria quer uma base comum de restrições, com liberdade para cada organização apertar mais. O que o tutorial de guardrails orienta?

A. Planejar políticas na enterprise como base, que as organizações podem restringir mais
B. Deixar cada organização definir tudo sozinha, sem nenhuma política definida pela enterprise
C. Usar instruções da organização no lugar das políticas da enterprise
D. Definir políticas só nas organizações, que depois sobem para a enterprise

R4.3 · Usar os recursos da plataforma GitHub

A área de risco da Seguradora Laço quer estabilidade nas respostas de um processo crítico e pergunta se o Copilot Business oferece algum modelo com suporte por período estendido. O que a documentação informa?

A. Há o modelo LTS, com suporte por um período definido de um ano
B. Não há opção, porque os modelos mudam sem nenhum aviso prévio
C. Há o modelo Auto, que mantém o mesmo modelo por todo o contrato
D. Há o modelo utilitário, que o usuário escolhe no seletor do Chat

R4.4 · Usar os recursos da plataforma GitHub

A owner da organização Varejo Nordeste, dentro de uma enterprise que já definiu orçamentos, quer criar um orçamento próprio para os usuários com licença pela organização. O que a documentação diz sobre esse orçamento de organização?

A. Ele vale para usuários de outras organizações da mesma enterprise
B. Ele só pode restringir mais, abaixo dos orçamentos da enterprise
C. Ele substitui o orçamento da enterprise para todos da organização
D. Ele limita o consumo do pool antes mesmo de o pool se esgotar

Respostas comentadas

R4.1. D) Vários rulesets podem valer ao mesmo tempo para o mesmo branch

Por quê: Rulesets permitem camadas: vários podem valer para o mesmo branch, e o status de enforcement pode mudar sem excluir o ruleset. Com proteção de branch, só uma regra se aplica por branch.

✗ A) Incorreta. Rulesets de níveis diferentes se somam; um não desliga o outro.
✗ B) Incorreta. Essa é a limitação das regras de proteção de branch (só uma vale por branch), não dos rulesets.
✗ C) Incorreta. Rulesets valem para branches e tags.

> **Dica de prova:** Regras em camadas no mesmo branch: rulesets.

R4.2. A) Planejar políticas na enterprise como base, que as organizações podem restringir mais

Por quê: Antes de liberar o cloud agent, planeje: onde ele será habilitado, que MCP será configurado, quais políticas não se aplicam a ele, rulesets, runners e secrets. Políticas da enterprise formam a base; organizações podem restringir mais.

✗ B) Incorreta. O tutorial recomenda planejar as políticas na enterprise antes de liberar o agente.
✗ C) Incorreta. Instruções orientam o modelo; não substituem políticas de acesso.
✗ D) Incorreta. As políticas não sobem das organizações para a enterprise.

> **Dica de prova:** Enterprise define a base; organização aperta.

R4.3. A) Há o modelo LTS, com suporte por um período definido de um ano

Por quê: Copilot Business e Enterprise têm duas designações: modelo base (usado quando nenhum outro está habilitado) e modelo LTS (suportado por período estendido, hoje um ano a partir da designação).

✗ B) Incorreta. Existem as designações base e LTS justamente para estabilidade.
✗ C) Incorreta. O Auto escolhe modelos conforme a tarefa e a disponibilidade; não fixa um modelo.
✗ D) Incorreta. Modelos utilitários rodam em segundo plano e não aparecem no seletor.

> **Dica de prova:** Estabilidade de modelo no Business: LTS.

R4.4. B) Ele só pode restringir mais, abaixo dos orçamentos da enterprise

Por quê: O orçamento de organização limita cobranças de uso adicional dos usuários licenciados pela organização, depois que o pool acaba. Ele só restringe abaixo do que a enterprise definiu.

✗ A) Incorreta. Ele cobre usuários que recebem a licença por aquela organização.

✗ C) Incorreta. Ele não sobrepõe nem substitui o orçamento definido pela enterprise.

✗ D) Incorreta. Como o de cost center, o orçamento de organização vale depois que o pool compartilhado acaba.

> **Dica de prova:** Orçamento da organização só aperta, não afrouxa.

DIA DA PROVA

Estratégia de prova

A prova GH-700 é voltada a público de negócio (gerente de produto, gerente de programa, gerente técnico de projeto, analista de negócios) e não cobra código: os enunciados são cenários de empresa em que alguém quer um resultado e você escolhe o recurso, a configuração ou o comportamento padrão documentado. Os distratores costumam ser recursos reais usados no lugar errado (wiki, labels, prompt, Autopilot, .lock.yml) ou números de outro recurso. Em questões de múltipla resposta (Escolha DUAS ou TRÊS), é preciso saber cada afirmação por inteiro. O exame valoriza controle humano e governança: quando uma alternativa promete garantia por instrução ou autoaprovação do agente, quase sempre está errada.

Como responder

✓ Administre o tempo: são 100 minutos numa prova supervisionada, em inglês, com possíveis componentes interativos e aprovação com 700; a quantidade de questões não é divulgada, então treine com 50 questões, cerca de 2 minutos cada, e marque as dúvidas para voltar.

✓ Como a prova só é oferecida em inglês, se o inglês não for seu idioma preferido avalie pedir os 30 minutos extras previstos pelo guia.

✓ Antes de ler as alternativas, identifique a superfície (Chat do GitHub.com, app, CLI, cloud agent, agentic workflow, automação) e o verbo pedido; muitos distratores existem, mas em outra superfície.

✓ Em Escolha DUAS ou TRÊS, julgue cada alternativa como verdadeira ou falsa isoladamente; uma palavra absoluta (sempre, todas, qualquer) costuma derrubar a afirmação.

✓ Separe orientar de garantir: instruções, prompts e skills orientam; políticas, rulesets, CODEOWNERS, hooks, permissions e safe-outputs garantem.

✓ Em cobrança, pergunte primeiro se o controle é por pessoa ou por grupo e se age no pool ou depois dele; depois aplique o mais específico.

✓ Desconfie de alternativas em que o agente aprova, mescla ou decide sozinho algo de negócio; a resposta da prova mantém a decisão final com uma pessoa.

✓ Use os pesos a seu favor: o domínio de fluxos e automações agênticos vale 25-30% e os outros três 20-25% cada; reforce primeiro agentic workflows e automações.

Erros que derrubam candidatos

✗ Confundir agentic workflow (arquivo versionado e compilado em .lock.yml) com automação do Copilot (configurada fora do repositório).

✗ Achar que Autopilot libera ferramentas ou que parar e cancelar desfazem o trabalho já feito.

✗ Trocar orçamento por usuário, de cost center e controle de uso incluído, ou supor fallback para modelo mais barato.

✗ Tratar instruções personalizadas como controle garantido e aprovações de automação como fronteira de segurança.

✗ Misturar comandos de barra parecidos (/clear, /delete, /compact, /restart-session, /chronicle standup, tips, search, improve).

✗ Esquecer padrões: uso adicional habilitado, política MCP desabilitada, nível Cautious, permissions read-all, tools omitido libera tudo.

✗ Trocar limites numéricos entre recursos (3 fixadas, 10 responsáveis, 100 sub-issues em 8 níveis, 25 tipos, 6 templates, 30.000 caracteres).

Na semana da prova

- ☐ Refaça as provas simuladas focando nos erros e reescreva cada erro como uma regra curta (recurso certo x recurso tentador).
- ☐ Revise as seções lembrar e numeros de cada capítulo e recite os padrões de fábrica: uso adicional habilitado, MCP desabilitada, Cautious, read-all, engine copilot, tools omitido libera tudo.
- ☐ Monte uma tabela própria de comandos por superfície: Chat do GitHub.com, app, CLI e gh aw, com o que cada comando faz.
- ☐ Treine a leitura em inglês dos termos que aparecem como estão na prova (safe outputs, steering, rewind, Approve and run workflows, Full control).
- ☐ Na véspera, revise só o frontmatter do agentic workflow, as mitigações do cloud agent e a hierarquia de orçamentos, e confira as regras da prova supervisionada.

REVISÃO FINAL

Folha de cola

Leia na véspera e na manhã da prova: os gatilhos e as pegadinhas mais importantes de cada domínio.

1. Aumentar a produtividade usando o GitHub Copilot

‣ Ver como outro modelo responderia sem reescrever o prompt → **Ícone de nova tentativa abaixo da resposta**

‣ Testar versão alternativa da pergunta sem perder a resposta original → **Subthread (Edit message)**

‣ Conversa de dois meses atrás incompleta → **Mensagens mantidas por 28 dias**

‣ Sessão longa e lenta, mas quer manter o trabalho → **/compact**

‣ Voltar exatamente à sessão de ontem sem lista → **copilot --continue**

⚠ Sem tools o agente recebe todas as ferramentas; para restringir, liste só as necessárias.

⚠ Prompt orienta e target escolhe o ambiente; quem limita ferramentas é a propriedade tools.

⚠ Precisa de merge no branch padrão e atualizar a aba de agentes.

2. Projetar e implementar fluxos de trabalho e automações agênticos

‣ Onde ficam as instruções do agentic workflow → **Corpo Markdown, em linguagem natural**

‣ Quais escritas o agente pode fazer → **safe-outputs no frontmatter**

‣ Evitar que uma execução gaste créditos demais → **max-ai-credits (padrão 1.000 AIC)**

‣ Trazer melhorias de um workflow importado → **source no frontmatter + gh aw update**

‣ Preparar o repositório para criar workflows com agente de código → **gh aw init**

⚠ A fonte é o Markdown; edite o .md, rode gh aw compile e commite os dois arquivos.

⚠ gh aw run só dispara uma execução; quem gera o .lock.yml é gh aw compile.

⚠ Quando roda vai em on, na sintaxe do Actions; o corpo traz só as instruções da tarefa.

3. Realizar tarefas com humano no ciclo (human-in-the-loop) em fluxos de trabalho agênticos

‣ Ainda explorando, decisões frequentes → **Interactive**

‣ Montar plano e só executar depois de aprovado → **Modo Plan**

‣ Plano revisado, implementar sem esperar a cada passo → **Accept plan and continue in Autopilot**

‣ Modelo aprova o seguro e pergunta o arriscado → **Assisted (permissões assistidas)**

‣ Script verifica cada comando contra lista permitida antes de executar → **Hook preToolUse**

⚠ Autopilot é continuidade; permissões são outro controle, e sem elas o agente pode não concluir.

⚠ O pedido é negado, o que pode impedir o agente de terminar.

⚠ Regras de negação da enterprise vêm antes; o comando é negado.

4. Usar os recursos da plataforma GitHub

‣ Quantos créditos cobre um valor em dólares → **1 AI credit = US$ 0,01 (multiplique o valor por 100)**

‣ Estagiário com orçamento US$ 0 → **Orçamento por usuário é parada rígida: bloqueado na hora**

‣ Aumento temporário durante incidente ou sprint → **Orçamento individual com data de expiração**

‣ Departamento não pode consumir do pool mais que suas licenças → **Controle de uso incluído do cost center**

‣ Ninguém pode gerar cobrança depois que o pool acabar → **Desabilitar a política AI credits paid usage**

⚠ Não há fallback automático: orçamento por usuário esgotado interrompe o acesso, mesmo com saldo no pool e em qualquer plano.

⚠ Orçamento por usuário é sempre parada rígida; US$ 0 bloqueia na hora.

⚠ Eles não se somam: vale o mais específico (individual > cost center > universal).

GABARITO

Respostas do diagnóstico

D1. A) Um prompt file .prompt.md com variáveis de entrada → revise o capítulo 1

Por quê: Prompt files (.prompt.md, em geral em .github/prompts) são modelos de prompt reutilizáveis, com variáveis de entrada e contexto adicional, acionados para uma interação. Instruções personalizadas valem automaticamente no escopo delas.

✗ B) Incorreta. Automações rodam por gatilho; ele quer acionar quando precisar, no chat.

✗ C) Incorreta. Instruções do repositório entram automaticamente em todas as conversas do escopo, não só quando ele chama.

✗ D) Incorreta. Rulesets protegem branches e tags; não guardam prompts.

> **Dica de prova:** Prompt reutilizável sob demanda: prompt file.

D2. C) Cada sessão roda num workspace isolado, com branch próprio → revise o capítulo 1

Por quê: O app roda várias sessões isoladas ao mesmo tempo, cada uma com worktree e branch dedicados; a sessão pode rodar numa worktree nova, no repositório local ou num sandbox na nuvem.

✗ A) Incorreta. As sessões não compartilham o mesmo branch.

✗ B) Incorreta. O app foi feito justamente para sessões paralelas.

✗ D) Incorreta. Não há fila entre sessões.

> **Dica de prova:** Tarefas paralelas no app: sessões isoladas, um branch cada.

D3. A) Single, Specific, Short e Surround → revise o capítulo 1

Por quê: Os 4 Ss: Single (foco numa tarefa ou pergunta), Specific (instruções explícitas e detalhadas), Short (conciso) e Surround (contexto relevante ao redor).

✗ B) Incorreta. Smart e Shared não fazem parte dos 4 Ss.

✗ C) Incorreta. Secure e Summary não fazem parte dos 4 Ss.

✗ D) Incorreta. Simple, Secure, Scalable e Structured não formam os 4 Ss do módulo.

> **Dica de prova:** 4 Ss: Single, Specific, Short, Surround.

D4. B) Somente leitura (read-all), salvo concessão explícita → revise o capítulo 2

Por quê: No frontmatter, permissions define o que o agente pode fazer no repositório; o padrão é read-all. Operações de escrita passam pelos safe-outputs.

✗ A) Incorreta. O padrão é leitura, não ausência total de permissões.

✗ C) Incorreta. O padrão não é escrita; o prompt não substitui permissões.

✗ D) Incorreta. As permissões vêm do frontmatter, não de quem compilou.

> **Dica de prova:** Agentic workflow nasce somente leitura.

D5. B) Usa CodeQL, secret scanning, a Advisory Database e uma segunda opinião do code review

→ revise o capítulo 2

Por quê: O cloud agent valida o próprio trabalho antes de concluir: CodeQL, Advisory Database (malware e vulnerabilidades altas ou críticas), secret scanning e code review. Os detalhes ficam no log da sessão, e as ferramentas podem ser desativadas.

✗ A) Incorreta. Não há auditoria manual do suporte.

✗ C) Incorreta. Essa validação não exige licença de Secret Protection, Code Security ou Advanced Security.

✗ D) Incorreta. O agente tenta resolver os problemas antes de concluir o PR.

> **Dica de prova:** Autocrítica do cloud agent: CodeQL, Advisory Database, secret scanning e code review.

D6. B) Subagentes, que fazem trabalho delegado em contexto separado → revise o capítulo 2

Por quê: Na comparação de personalizações, subagentes fazem trabalho delegado separado do agente principal, com contexto próprio; o trabalho intermediário não ocupa a janela de contexto principal.

✗ A) Incorreta. Labels classificam issues.

✗ C) Incorreta. Prompt files são modelos reutilizáveis de prompt.

✗ D) Incorreta. Instruções pessoais guardam preferências, não dividem trabalho.

> **Dica de prova:** Dividir sem poluir o contexto: subagentes.

D7. A) gh aw audit com o ID da execução → revise o capítulo 3

Por quê: Para custos e falhas: gh aw logs mostra execuções recentes com duração, tokens e AIC estimado; gh aw audit RUN-ID detalha uma execução.

✗ B) Incorreta. gh aw add importa um workflow de outro repositório.

✗ C) Incorreta. gh aw init prepara o repositório para autoria.

✗ D) Incorreta. gh aw compile gera o .lock.yml a partir do Markdown.

> **Dica de prova:** Investigar uma execução: gh aw audit.

D8. B) Pegar mal-entendidos de escopo antes que alguma mudança seja feita → revise o capítulo 3

Por quê: O plan mode do CLI ajuda a pegar mal-entendidos antes de qualquer código e a manter o controle de tarefas complexas: o Copilot faz perguntas, define escopo e monta o plano. No modo interativo, Shift+Tab alterna os modos.

✗ A) Incorreta. Liberar ferramentas sem aprovação é papel de opções como --allow-all, não do plan mode.

✗ C) Incorreta. Compactar o histórico é função do /compact.

✗ D) Incorreta. Seguir as etapas sem pausa descreve o Autopilot, não o plan mode.

> **Dica de prova:** Plan mode: perguntas e plano antes de mexer em algo.

D9. D) Nome, agenda, repositório associado e status da última execução → revise o capítulo 3

Por quê: Em Automations, no app, você vê as automações salvas com nome, agenda, repositório e status da última execução, e roda qualquer uma sob demanda pelo botão de play.

✗ A) Incorreta. O cartão não mostra custo, autores de commits nem revisores.

✗ B) Incorreta. Essas configurações ficam em outras áreas do app.

✗ C) Incorreta. O cartão não exibe o prompt completo nem todo o histórico.

> **Dica de prova:** Cartão de automação: nome, agenda, repositório e última execução.

D10. A) O card entra, e a contagem acima do limite fica destacada → revise o capítulo 4

Por quê: No layout de board, Set column limit comunica a capacidade desejada da coluna. A contagem e o limite aparecem no topo e ficam destacados ao exceder, mas nada impede adicionar cards.

✗ B) Incorreta. O limite não devolve nem bloqueia cards.

✗ C) Incorreta. Limites de coluna são próprios de cada view.

✗ D) Incorreta. Não existe aprovação de exceção para limite de coluna.

> **Dica de prova:** Limite de coluna sinaliza, não bloqueia.

D11. A) Definir data de expiração no orçamento individual desse usuário → revise o capítulo 4

Por quê: Orçamentos individuais por usuário podem ter expiração: sem expiração (padrão), ao fim do ciclo atual ou numa data específica. É útil para aumentos temporários, como numa sprint ou num incidente.

✗ B) Incorreta. Cost centers não se apagam sozinhos; a expiração resolve o caso sem estrutura nova.

✗ C) Incorreta. A expiração existe só para orçamentos individuais, não para o universal nem para o de cost center.

✗ D) Incorreta. Não é preciso automação; a própria configuração do orçamento tem expiração.

> **Dica de prova:** Aumento temporário: orçamento individual com expiração.

D12. B) Um status update do project, com status At risk e datas → revise o capítulo 4

Por quê: Status updates ficam no painel lateral do project: status, datas de início e alvo e mensagem. Quem tem escrita adiciona; quem tem leitura vê e pode assinar.

✗ A) Incorreta. Iterações agrupam itens por período; não registram status do project.

✗ C) Incorreta. A wiki não integra o status ao project.

✗ D) Incorreta. Labels em issues não comunicam o status do project como um todo.

> **Dica de prova:** Status do project para patrocinadores: status update.

Agora, treine no formato real

Faça as 6 provas completas do curso GH-700: 300 questões inéditas, tempo do exame, explicação de cada alternativa e link da documentação para revisar.

Ver as 6 provas completas e o cupom do mês →

Até o dia da prova, uma revisão por dia no seu feed

Instagram @strecht.simulados

Um post de prova por dia: questão comentada, pegadinhas e o "não confunda" de cada certificação.

Seguir no Instagram →

YouTube Strecht Simulados

Shorts diários com pegadinhas de prova e, aos sábados, um simulado comentado.

Inscrever-se no canal →

Material independente de preparação, sem afiliação ou endosso de GitHub. As questões são inéditas e não reproduzem a prova real.