GITHUB COPILOT ONBOARDING · PLAYBOOK

Onboarding estruturado decide se o GitHub Copilot entrega valor ou empaca em 23%.

Empresas não falham com o GitHub Copilot porque a ferramenta é fraca. Falham porque o letramento é desigual. Este playbook transforma uma licença em um programa: vinte e quatro personas do SDLC em sete clusters, uma jornada de cinco estágios da preparação à autonomia, uma vitória rápida nas primeiras setenta e duas horas e um Centro de Excelência que assume a cadência depois que o programa formal termina. Equipes com onboarding formal atingem 80 por cento de adoção ativa em noventa dias; equipes sem ele estacionam perto de 23 por cento.

AUTORA

Paula Silva

CARGO

AI-Native Software Engineer

CONTATO

in/paulanunes

TEMPO DE LEITURA

~ 45 minutos, fim a fim

24	7	5
PERSONAS DO SDLC	CLUSTERS DE PERSONAS	ESTÁGIOS DE ONBOARDING

80%

META DE ADOÇÃO ATIVA

CAPÍTULOS

Chapter 00

Por que o onboarding estruturado decide o retorno

Por que o GitHub Copilot não entrega valor quando o letramento é desigual, o que a evidência diz sobre adoção ativa, as sete armadilhas que travam a maioria dos rollouts e os seis princípios sobre os quais o programa é construído.

Fundamentos →

Chapter 01

Vinte e quatro personas do SDLC em sete clusters

Os vinte e quatro papéis do SDLC agrupados em sete clusters, a dor que cada um carrega hoje e o ganho que os agentes trazem, a matriz de três planos e um vocabulário por cluster reportando para um dashboard compartilhado.

Personas →

Chapter 02

Um programa de cinco estágios da preparação à autonomia

Os cinco estágios de um portão de preparação à autonomia, cada um com um critério de saída explícito, apoiados em uma pilha de habilitação de quatro camadas de personas, estágios, labs e medição.

Framework →

Chapter 03

Trilhas de letramento por cluster e ferramenta Copilot

Como cada cluster mapeia para uma superfície do GitHub Copilot: agent mode para engenharia, instruções de repositório e MCP para contexto, o seletor de modelos como decisão de custo e trilhas para plataforma, dados, produto e segurança.

Letramento →

Chapter 04

Quick wins em setenta e duas horas, medidos a cada quinzena

Um quick win concreto por cluster dentro da janela de setenta e duas horas, cartões de padrão e uma biblioteca de prompts compartilhada, seis KPIs do alcance ao risco e governança de custo com GitHub AI Credits.

Métricas →

Chapter 05

Sustente a adoção com gestão de mudança e um Centro de Excelência

O mapa de resistência e uma conversão em quatro passos, Kotter aplicado à adoção agêntica, o Agents Center of Excellence e a comunidade de prática, e o roteiro de seis meses com ações de segunda-feira de manhã.

Adoção →

DICAS DE LEITURA

> **ATALHOS DE TECLADO**
>
> Aperte `/` para buscar. Use `j` e `k` para mover entre capítulos. Aperte `t` para alternar tema. Aperte `g` para ir ao topo.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

00 O GAP DE LETRAMENTO

Por que o onboarding estruturado decide o retorno.

Empresas não falham com o GitHub Copilot porque a ferramenta é fraca. Falham porque o letramento é desigual. Licenças são compradas, assentos são provisionados, e a adoção ainda assim empaca: engenheiros sêniores correm com a ferramenta, QA nunca a abre, o gerente de produto não vê conexão com seu trabalho, e o CISO tem dúvidas em aberto. A diferença entre pagar por IA e extrair valor dela é o programa em torno da ferramenta. Este capítulo mede esse gap, apresenta a evidência sobre adoção ativa, nomeia as sete armadilhas que travam a maioria dos rollouts e expõe os seis princípios sobre os quais o programa é construído.

O gap de letramento que ninguém orça

Organizações compram licenças do GitHub Copilot e esperam que a adoção venha atrás. Não vem. Segundo o GitHub Octoverse 2025, equipes com onboarding formal atingem 80% de adoção ativa em 90 dias. Equipes sem ele chegam a 23%. Isso é um gap de 3,5x. O diferencial não é orçamento, não é senioridade, não é o stack tecnológico. É o programa construído em torno da ferramenta. O gap tem um nome que nenhuma linha de custo captura: letramento desigual. Engenheiros sêniores correm com a ferramenta, QA nunca a abre, o gerente de produto não vê conexão com seu trabalho, e o CISO tem dúvidas em aberto. A licença é uniforme. O letramento não é.

O rollout padrão é uma licença e um e-mail de boas-vindas. A adoção ativa atinge o pico por volta da segunda semana e então desaba para cerca de 12% no dia 60 (Forrester 2025). O modelo de cobrança torna esse desperdício legível. O GitHub Copilot passou para cobrança baseada em uso em 1 de junho de 2026, precificada em GitHub AI Credits, onde um crédito equivale a um centavo de dólar. Execuções em agent mode e premium requests consomem esses créditos. Uma organização que provisiona os assentos mas nunca constrói letramento paga pelos créditos e

captura quase nada do retorno. O onboarding é tratado como um evento. Ele precisa rodar como um programa.

O que a evidência diz sobre adoção ativa

A evidência é consistente entre as fontes. Programas formais atingem 80% de adoção ativa em 90 dias; rollouts informais estacionam perto de 23% (GitHub Octoverse 2025). Cada dólar investido em treinamento estruturado retorna USD 4,20; o treinamento autodidata retorna USD 1,80. O MIT Sloan 2024 constatou que enquadrar o programa como augmentação, não automação, produz 2,8x mais engajamento. O número que paga o programa não é a contagem de assentos. É a taxa de usuários ativos, o percentual de usuários licenciados ativos ao menos cinco dias por semana.

Profundidade importa tanto quanto alcance. Adoção ativa é definida como ao menos uma sugestão do GitHub Copilot aceita por dia útil nas duas semanas anteriores. Essa definição separa quem tem assento de quem pratica. Um desenvolvedor que roda uma sessão de agent mode com múltiplas etapas por semana usa a plataforma de forma diferente de quem aceita um autocomplete e segue em frente. O programa ensina o segundo comportamento: escolher o modelo certo no Copilot picker para a tarefa, Claude Fable 5 ou Opus 4.8 para raciocínio difícil, Sonnet 5 ou Haiku 4.5 para edições rápidas, Gemini 3.1 Pro ou um modelo GPT-5.x quando a tarefa pede, e escrever o prompt de agent mode que transforma um item de backlog em um pull request. Alcance sem profundidade é métrica de vaidade.

Sete armadilhas que travam a maioria dos rollouts

Evidências de campo do McKinsey Global Survey on AI 2024 e do Forrester AI Adoption Pitfalls 2025 apontam para uma lista curta de modos de falha recorrentes. Treinamento pontual sem acompanhamento: a adoção cai para 12% em 60 dias. Programas exclusivos para engenheiros: silos de adoção, e cerca de metade do retorno potencial fica sem captura porque QA, produto e segurança nunca começam. Sem casos de uso reais: tutoriais genéricos matam a motivação na

primeira semana. Sem política clara de uso de IA: o CISO bloqueia o rollout e as equipes travam.

As três restantes são mais silenciosas e igualmente fatais. Sem champions internos: a prática nunca se difunde socialmente, então fica presa no bolso dos primeiros adotantes. Sem métricas, ou com as erradas: o orçamento é cortado na primeira revisão trimestral porque ninguém consegue mostrar o retorno. Um currículo único para todos: alta evasão e baixa relevância, porque um gerente de produto e um engenheiro backend não aprendem a mesma ferramenta do mesmo jeito. Cada uma das sete remete a um único erro, tratar um evento de implantação como um programa concluído.

Seis princípios sobre os quais o programa é construído

O programa se apoia em seis princípios que a pesquisa valida como determinantes de sucesso. Persona-first: cada papel tem suas próprias motivações, medos, vocabulário e contexto de trabalho, então um guia genérico do GitHub Copilot não funciona para um gerente de produto, e o letramento precisa falar a língua de cada persona. Quick wins primeiro: hábitos mudam com reforço positivo, então o programa engenheira uma vitória visível nas primeiras 72 horas antes de introduzir conceitos avançados. Aprendizado social: Rogers, em Diffusion of Innovations, mostra que a adoção acelera quando as pessoas veem pares terem sucesso, não só especialistas, e por isso comunidades internas de prática superam o treinamento top-down.

Os outros três governam confiança e durabilidade. Segurança psicológica: pessoas não experimentam uma ferramenta nova quando temem parecer incompetentes, então o ambiente precisa ser explicitamente seguro para tentativa e erro. Governança visível: a adoção sobe quando as regras estão claras, o que o agente pode e não pode fazer, quem é responsável por quê, como os dados são protegidos, e essa clareza importa mais para personas não técnicas e líderes. Medição contínua: sem métricas o programa vira uma caixa-preta, então adoção, qualidade e satisfação são medidas a cada quinzena e o programa se ajusta. No estágio de autonomia esses princípios endurecem em artefatos: um .github/copilot-

instructions.md compartilhado que codifica convenções uma vez, e servidores MCP que dão ao agent mode acesso governado ao contexto interno.

> DEFINIÇÃO
>
> Seis princípios, uma regra: onboarding não é um evento, é uma plataforma. Persona-first sobre genérico. Quick wins sobre teoria. Aprendizado social sobre top-down. Segurança psicológica sobre medo. Governança visível sobre ambiguidade. Medição contínua sobre fé.

Referências

Fontes primárias para o gap de adoção, a evidência empírica, as sete armadilhas e os princípios sobre os quais o programa é construído.

- GitHub Octoverse 2025, o gap de adoção: 80% de adoção ativa com onboarding formal contra 23% sem ele.
- Forrester, AI Adoption Pitfalls, a adoção desaba para cerca de 12% em 60 dias sem um programa sustentado.
- MIT Sloan Management Review, AI in the Workplace, o enquadramento de augmentação produz 2,8x mais engajamento que o de automação.
- McKinsey Global Survey on AI, evidência de campo para os modos de falha recorrentes de adoção.
- Everett M. Rogers, Diffusion of Innovations, 5th Edition, a base de aprendizado social para comunidades de prática acima do treinamento top-down.
- GitHub Copilot Documentation, a plataforma de referência para agent mode, o model picker e o .github/copilot-instructions.md.
- Usage-based billing and GitHub AI Credits, o modelo de créditos que torna a adoção travada um custo visível.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

01 PAPÉIS E CLUSTERS

Vinte e quatro personas do SDLC em sete clusters.

As organizações que extraem valor da IA agêntica fazem onboarding de toda a organização de entrega, não só de seus desenvolvedores sêniores. Este capítulo mapeia essa organização: vinte e quatro papéis do SDLC agrupados em sete clusters, cada um com sua dor de hoje, seu ganho com agentes, seus três planos de ferramentas e seu vocabulário. Os clusters diferem em quase tudo, exceto nos dashboards para os quais reportam. Esse é o desenho. Fale a língua de cada cluster, meça todos em um mesmo painel.

Sete clusters, vinte e quatro papéis

O SDLC não é um único papel, e qualquer programa que o trate como tal falha no momento em que chega a alguém que não é desenvolvedor sênior. Este capítulo mapeia toda a organização de entrega: vinte e quatro papéis, agrupados em sete clusters. O agrupamento se apoia em três referências que as organizações de entrega já usam para se descrever, o DORA 2024 State of DevOps report, o SAFe 6.0 e o IEEE SWEBOK v4. Os sete clusters são Engenharia com seis papéis, Qualidade e Confiabilidade com três, Plataforma e Infraestrutura com quatro, Produto e Design com três, Dados e IA com três, Segurança e Conformidade com dois, e Liderança e Estratégia com três.

O cluster, e não o indivíduo, é a unidade de onboarding. Dois engenheiros de backend e um engenheiro mobile compartilham uma ferramenta principal, um fluxo de revisão e um vocabulário, então treinam juntos e são medidos juntos. Um engenheiro de QA e um gerente de produto não compartilham, então colocá-los na mesma sala produz uma demo que não conecta com nenhum dos dois. O agrupamento mantém cada sessão ancorada no trabalho real de um cluster: os artefatos na tela vêm do sprint atual daquele cluster, e as convenções que o cluster já segue vão para um arquivo compartilhado .github/copilot-instructions.md que toda sessão de agente daquele cluster lê antes de escrever uma linha.

A dor de cada persona hoje e o ganho com agentes

Cada cluster carrega uma dor específica que é anterior a qualquer agente. Engenharia perde horas com boilerplate e refatoração repetitiva, e engenheiros full stack perdem ainda mais com a troca de contexto entre camadas. Qualidade e Confiabilidade escreve o esqueleto dos testes à mão e diagnostica incidentes a partir de logs espalhados. Plataforma e Infraestrutura repete configuração e persegue drift entre muitos sistemas. Produto e Design sintetiza feedback e requisitos manualmente. Dados e IA depura pipelines e espera por longos ciclos de experimentação. Segurança e Conformidade faz triagem de alertas e falsos positivos e mapeia controles para políticas à mão. Liderança e Estratégia não tem uma leitura limpa de produtividade ou de ROI de ferramentas.

De autor a revisor

O ganho tem a mesma forma em todos os clusters: o trabalho mecânico vai para o agente, e o julgamento fica com a pessoa. Engenharia usa o GitHub Copilot em agent mode para levar um item de baixa complexidade do backlog do branch ao pull request sem sair do editor. Qualidade gera uma suíte completa de testes unitários para uma classe sem cobertura em cerca de quinze minutos contra três a quatro horas à mão, e times que geram testes a partir de requisitos relatam de 60 a 70 por cento menos tempo para escrevê-los. Plataforma descreve um módulo Bicep ou Terraform e revisa o código gerado. Produto constrói um agente no Copilot Studio que responde perguntas do backlog a partir de uma planilha no SharePoint, e PMs relatam recuperar de quatro a seis horas por semana. Segurança faz a triagem dos principais findings com o GitHub Advanced Security e o Copilot Autofix. Liderança lê métricas DORA em vez de anedotas.

O fio comum é uma mudança de papel. O engenheiro de plataforma para de escrever Terraform do zero e passa a revisar o código gerado. O engenheiro de QA para de escrever o esqueleto e passa a decidir quais de uma centena de casos gerados realmente importam. Desenvolvedores formalmente treinados produzem código com qualidade 55% maior do que colegas autodidatas com a mesma ferramenta, que é a distância mensurável entre aceitar sugestões e dirigi-las.

A matriz de ferramentas com planos primário, secundário e de governança

Cada cluster recebe três planos de ferramentas, não um. O plano primário é onde o cluster faz seu trabalho diário. O plano secundário o estende. O plano de governança mantém o trabalho auditável. Engenharia usa o GitHub Copilot em agent mode como primário, o GitHub Actions como secundário e o GitHub Advanced Security como governança. Qualidade usa o GitHub Copilot para testes, o Azure Monitor com Application Insights como secundário e o GitHub Actions em CI como governança. Plataforma usa o GitHub Copilot para infraestrutura como código, o Azure Developer CLI como secundário e o Defender for Cloud como governança. Produto usa o Copilot Studio, o Microsoft 365 Copilot como secundário e o Purview sobre dados de pesquisa como governança. Dados e IA usa o Azure AI Foundry, o Microsoft Fabric como secundário e o Purview como catálogo. Segurança usa o GitHub Advanced Security, o Defender for Cloud como secundário e o Purview com o Compliance Manager como governança. Liderança usa o Microsoft 365 Copilot, o GitHub Insights com métricas DORA como secundário e o Azure AI Foundry para governança.

Os três planos compartilham uma superfície de cobrança e uma superfície de modelo. O trabalho em agent mode é cobrado como GitHub AI Credits, em que um crédito equivale a um centavo de dólar, sob a cobrança por uso que entrou em vigor em 1 de junho de 2026, então a métrica de profundidade e a fatura leem o mesmo sinal. O modelo por trás de cada sessão vem do seletor do Copilot: Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro e a família GPT-5.x, então um cluster combina um modelo a uma tarefa sem sair da ferramenta. No estágio de autonomia, os clusters estendem o plano primário com Copilot Extensions construídas sobre o Model Context Protocol, que é como os próprios sistemas de um cluster se tornam contexto de primeira classe para seus agentes.

> DEFINIÇÃO
>
> Três planos por cluster: o primário é onde o trabalho diário acontece, o secundário o estende, a governança o mantém auditável. Os três são cobrados como GitHub AI Credits e tiram seu modelo do seletor do Copilot, então profundidade e custo leem um mesmo sinal.

Um vocabulário por cluster, um conjunto de dashboards

Clusters diferentes falam línguas diferentes, e o programa respeita isso. Uma demo para engenheiros fala de branches, pull requests e refatoração. Uma demo para gerentes de produto fala de backlog, feedback e roadmap. Uma demo para segurança fala de findings, autofix e trilhas de auditoria. A mesma ferramenta mostrada no vocabulário errado soa como a ferramenta de outra pessoa, que é a razão mais comum para um cluster nunca abri-la. O enquadramento também importa aqui. O MIT Sloan constata que a mensagem de augmentação produz 2,8 vezes o engajamento da mensagem de automação, e augmentação é mais fácil de ouvir no seu próprio vocabulário.

O vocabulário diverge, os dashboards convergem. Cada cluster reporta para o mesmo painel, então a liderança lê um único sinal em toda a organização. Alcance é a taxa de usuários ativos. Profundidade é ações agênticas por semana, a contagem de sessões de agent mode com múltiplas etapas que separa autocomplete de orquestração. Throughput é o quarteto DORA de lead time, frequência de deploy, taxa de falha de mudança e tempo de recuperação. Risco são os findings do GitHub Advanced Security corrigidos e mesclados. Um vocabulário por cluster mantém a adoção real; um conjunto de dashboards a mantém financiável.

> DEFINIÇÃO
>
> Um cluster, uma ferramenta principal, um vocabulário, uma linha no dashboard compartilhado. Os vinte e quatro papéis nunca treinam como vinte e quatro currículos separados e nunca reportam como vinte e quatro métricas separadas. Sete clusters é a unidade pequena o bastante para falar cada língua e grande o bastante para medir como um programa.

Referências

Fontes primárias para o mapa de sete clusters, a matriz de ferramentas por cluster e o dashboard compartilhado.

- DORA, 2024 State of DevOps Report, referência para mapear os papéis de entrega nos sete clusters.
- Scaled Agile Framework, SAFe 6.0, a taxonomia de papéis por trás do mapa de clusters.
- IEEE SWEBOK v4, Software Engineering Body of Knowledge, a referência para os perfis de papel de cada cluster.
- GitHub Copilot Documentation, o plano primário dos clusters de Engenharia, Qualidade e Plataforma.
- GitHub Advanced Security, o plano de governança de Engenharia e o plano primário de Segurança e Conformidade.
- Microsoft Copilot Studio Documentation, o plano primário do cluster de Produto e Design.
- Azure AI Foundry Documentation, o plano primário do cluster de Dados e IA.
- DORA, the four key metrics, o sinal de throughput no dashboard compartilhado.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

02 CINCO ESTÁGIOS

Um programa de cinco estágios, da preparação à autonomia.

Uma licença do GitHub Copilot não cria fluência. Um programa cria. Este capítulo apresenta os cinco estágios que o programa percorre ao longo de vinte e quatro semanas: um portão de preparação que precisa fechar antes de qualquer pessoa começar, o enquadramento de augmentação e os quick wins que constroem o primeiro hábito, e a fluência operacional e a autonomia que permitem ao programa formal se encerrar. Cada estágio carrega um critério de saída explícito, então o progresso é medido por evidência, não pelo calendário. Toda a sequência se apoia em uma pilha de habilitação de quatro camadas, e pular qualquer camada quebra a de cima.

Estágio 0: preparação e o portão de pré-requisitos

O Estágio 0 vai da semana menos dois à semana zero, e é um portão, não um aquecimento. Nove pré-requisitos precisam estar no lugar antes de uma única persona ver uma demo, sem exceções. Publique a política de uso de IA por escrito. Provisione o GitHub Copilot Enterprise com o agent mode habilitado, e defina um arquivo .github/copilot-instructions.md por repositório para que toda sessão de agente leia as mesmas convenções. Provisione os workspaces adjacentes que os clusters vão precisar: Azure AI Foundry, Copilot Studio, GitHub Advanced Security, Defender for Cloud, Microsoft Fabric e Purview. Identifique dois champions por equipe e submeta-os a duas semanas de treinamento avançado antes do kickoff geral. Defina os KPIs do programa por escrito para que o primeiro relatório tenha uma linha de base a medir. O critério de saída é concreto: política publicada, champions treinados, sandboxes prontos.

O portão importa porque as três formas mais comuns de um rollout empacar remontam todas a um pré-requisito que foi pulado. Sem uma política escrita, o CISO bloqueia o rollout e as equipes travam. Sem champions treinados, não há quem

responda à primeira pergunta difícil, e a adoção nunca se difunde além dos primeiros voluntários. Sem sandboxes provisionados, o primeiro quick win perde sua janela e o hábito nunca se forma. O orçamento também pertence ao portão. Desde 1º de junho de 2026, o GitHub Copilot cobra o uso premium em GitHub AI Credits, em que um crédito é um centavo de dólar, então o responsável pelo programa dimensiona a alocação de créditos antes do piloto, em vez de reagir à primeira fatura. O portão de pré-requisitos não é burocracia. É o seguro mais barato que o programa compra.

Estágios 1 e 2: enquadramento de augmentação e quick wins

O Estágio 1 é a conscientização, semanas um e dois, e todo o seu trabalho é o enquadramento. Apresente o programa como augmentação, não automação. A distinção não é cosmética: o MIT Sloan relata que um enquadramento de augmentação produz cerca de 2,8 vezes o engajamento de um enquadramento de automação, porque as pessoas se aproximam de uma ferramenta que as torna melhores e se afastam daquela que leem como substituição. Execute demos ao vivo de sessenta a noventa minutos, uma por cluster, e construa cada demo sobre um artefato real tirado do sprint atual. Tutoriais genéricos falham aqui, todas as vezes, porque um QA assistindo a uma refatoração de backend não vê nada que mapeie para o trabalho à sua frente. O critério de saída é participação com relevância: cada persona participou de uma demo que usou seu próprio backlog, não um repositório de exemplo.

O Estágio 2 são os quick wins, semanas três a seis, e é cronometrado para a janela de hábito. O primeiro sucesso mensurável precisa acontecer dentro de 72 horas, o limite abaixo do qual um novo comportamento tende a fixar e acima do qual tende a se perder. Cada cluster ganha uma vitória concreta. Engenharia leva um item de baixa complexidade do backlog do branch ao pull request em agent mode sem sair do editor. QA gera uma suíte completa de testes unitários para uma classe não coberta, o que normalmente leva cerca de 15 minutos contra 3 a 4 horas manualmente. Plataforma gera um pipeline do GitHub Actions para um novo serviço, usando a configuração do projeto como contexto. Produto cria um agente no Copilot Studio que responde perguntas do backlog a partir de uma planilha do SharePoint. Segurança faz a triagem dos três principais CVEs com o GHAS Security Autofix.

Dados gera um notebook exploratório em cerca de 15 minutos. O seletor de modelos faz parte do ofício: direcione tarefas de alto volume e baixo risco ao Haiku 4.5 para conservar créditos, e reserve o Opus 4.8 ou o Claude Fable 5 para o trabalho pesado de raciocínio. Cada vitória é capturada como um cartão de padrão de uma página, a semente da biblioteca de prompts e receitas. O critério de saída: cada persona registrou pelo menos um artefato de quick win.

Estágios 3 e 4: fluência operacional e autonomia

O Estágio 3 é a fluência operacional, semanas sete a quatorze, quando o uso ocasional vira prática diária. A mecânica é hands-on: sessões de pair programming, dojos de prompt em que as equipes refinam e comparam prompts sobre uma tarefa compartilhada, e um showcase mensal em que as personas apresentam o que entregaram. É aqui que o arquivo .github/copilot-instructions.md se paga. Uma convenção que uma equipe discute em um dojo, como trata exceções, como nomeia testes, como estrutura um serviço, vira contexto permanente que toda sessão de agente futura lê sem precisar de nova explicação. A biblioteca de prompts deixa de ser uma pasta de trechos e se torna um ativo curado. O critério de saída tem três partes: 60% de uso semanal ativo, uma biblioteca de prompts curada e um showcase rodando em cadência fixa.

O Estágio 4 é a autonomia, semanas quinze a vinte e quatro, e é onde o programa planeja a própria saída. As equipes constroem agentes personalizados para seus fluxos recorrentes e estendem o agent mode com servidores MCP que expõem sistemas internos, um rastreador de issues, uma API de deploy, um repositório de documentação, como contexto de primeira classe que o agente pode consultar diretamente. A seleção de modelo torna-se deliberada, não padrão: as equipes escolhem no seletor do Copilot por tarefa, casando o Sonnet 5 ou o Gemini 3.1 Pro com o formato do trabalho e o orçamento de créditos. Os OKRs passam a carregar métricas de IA como uma linha normal, não uma novidade. O critério de saída é o sinal mais claro de todo o programa: os champions encerram a estrutura formal e uma comunidade de prática assume a cadência. O estágio de autonomia tem sucesso quando o programa consegue se retirar.

Critérios de saída e a pilha de habilitação de quatro camadas

Os cinco critérios de saída são a espinha do programa, e são cumulativos. Uma equipe não avança porque seis semanas se passaram; avança porque cumpriu o critério, uma demo assistida, um artefato registrado, um patamar de uso semanal. Portões, não calendários, decidem o progresso, o que impede que um estágio seja declarado concluído enquanto metade das personas ainda está travada. Quando um estágio empaca, o critério diz exatamente qual camada reforçar antes de seguir.

A pilha de habilitação de quatro camadas

Sob os estágios está uma pilha de habilitação de quatro camadas, e cada camada depende da de baixo. A camada um são as personas: os 24 papéis do SDLC agrupados em sete clusters, a base a que tudo o mais é ajustado. A camada dois é a jornada de cinco estágios que leva essas personas da consciência à autonomia. A camada três são os labs: o pair programming, os dojos de prompt e os showcases que transformam enquadramento em prática. A camada quatro é a medição: os KPIs, o sinal de DevEx e o Centro de Excelência que sustenta a cadência depois que os champions recuam. Pule a base e a adoção volta ao patamar de 23% que os programas sem estrutura relatam.

> DEFINIÇÃO
>
> Cinco estágios, um critério de saída cada: a preparação fecha um portão de nove itens, a conscientização enquadra augmentação sobre automação, os quick wins acontecem dentro de 72 horas, a fluência atinge 60% de uso semanal e a autonomia encerra o programa em uma comunidade de prática. A pilha por baixo, personas, estágios, labs, medição, só se sustenta se cada camada repousar sobre a de baixo.

Referências

Fontes primárias para o modelo de estágios, a base de gestão de mudança, a evidência do enquadramento de augmentação e a mecânica da plataforma GitHub

Copilot referenciada neste capítulo.

- GitHub Copilot Documentation, agent mode, .github/copilot-instructions.md e MCP, a mecânica de plataforma sobre a qual o programa se constrói.
- Customize AI in Visual Studio Code, como arquivos de instrução do repositório e contexto personalizado moldam o comportamento do agente.
- Usage-based billing for individuals: GitHub AI Credits, o modelo de cobrança contra o qual o portão de preparação orça, um crédito é um centavo de dólar.
- Prosci, ADKAR change management model, a base de gestão de mudança para sequenciar a conscientização antes da adoção.
- The four stages of competence, a progressão de competência que os cinco estágios espelham.
- BJ Fogg, Tiny Habits, a base de janela de hábito para a meta de quick win em 72 horas.
- MIT Sloan Management Review, AI in the Workplace, a evidência do enquadramento de augmentação versus automação.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

03 TRILHAS DE APRENDIZADO

Trilhas de letramento por cluster e ferramenta Copilot: do autocomplete à orquestração.

Letramento não é um único currículo. Os 24 papéis do SDLC se dividem em sete clusters, e cada cluster trabalha com uma superfície diferente do GitHub Copilot. Um engenheiro vive em agent mode dentro do editor. Um engenheiro de plataforma gera pipelines. Um product manager constrói um agente no Copilot Studio. Este capítulo mapeia cada cluster à sua ferramenta primária do Copilot, à sua primeira vitória rápida, e às escolhas de modelo e contexto que fazem a ferramenta se pagar.

Engenharia: do autocomplete à orquestração em Agent Mode

Engenharia é o maior cluster: seis papéis, backend, frontend, full stack, júnior, mobile e arquiteto. A superfície primária é o GitHub Copilot Agent Mode e o Chat. A trilha de letramento passa por três níveis. O nível um é o autocomplete: aceitar ou rejeitar sugestões inline. O nível dois é o Chat: perguntar sobre o arquivo aberto, pedir um refactor, gerar um teste. O nível três é o agent mode: entregar uma tarefa ao Copilot e deixá-lo planejar, editar vários arquivos, rodar o terminal e abrir um pull request. A maioria dos desenvolvedores empaca no nível um porque ninguém mostrou os dois seguintes.

A primeira vitória rápida é deliberada e pequena. Pegue um item de baixa complexidade do backlog e leve-o do branch ao pull request sem sair do editor. Esse é o critério de saída do Estágio 2 para o cluster, desenhado para caber na janela de hábito de 72 horas. A métrica de profundidade que importa aqui é ações agênticas por semana: a contagem de sessões de agent mode com múltiplas etapas por desenvolvedor. É o número que separa um time que autocompleta de um time que orquestra.

DEFINIÇÃO

A trilha de engenharia em uma linha: o nível um aceita sugestões, o nível dois pergunta ao Chat, o nível três entrega uma tarefa ao agent mode e revisa o pull request. A adoção empaca quando um time nunca sai do nível um.

Contexto de repositório com .github/copilot-instructions.md e MCP

O agent mode é tão bom quanto o contexto que ele consegue enxergar. Dois mecanismos carregam esse contexto no GitHub Copilot. O primeiro é um arquivo de instruções do repositório, .github/copilot-instructions.md, versionado no repo. Ele informa ao Copilot as convenções que a base de código já codifica: como exceções são tratadas, como os testes são organizados, quais bibliotecas são aprovadas, qual é o comando de build. Todo desenvolvedor do repo herda as mesmas instruções, então o agente para de chutar fatos que o time já conhece.

O segundo mecanismo é o Model Context Protocol (MCP). O MCP permite que o Copilot alcance ferramentas e dados fora do editor: um issue tracker, um schema de banco de dados, um design system, uma API interna. Em vez de colar contexto no chat, o agente consulta a fonte. No programa de onboarding, as extensões MCP pertencem ao Estágio 4, autonomia, porque pressupõem um time que já roda agent mode com fluência e sabe qual contexto vale a pena conectar. Os dois mecanismos respondem a perguntas diferentes: o arquivo de instruções diz como este repositório funciona, e o MCP diz o que existe além dele. O arquivo de instruções é da semana um; o MCP é do mês quatro.

Escolher modelos no seletor do GitHub Copilot

O GitHub Copilot não é um único modelo. O seletor de modelos expõe um menu que inclui Claude Fable 5, Opus 4.8, Sonnet 5 e Haiku 4.5, ao lado do Gemini 3.1 Pro e da família GPT-5.x. A habilidade de letramento é saber qual escolher para a tarefa à sua frente. Uma completude inline rápida não precisa do modelo de raciocínio mais capaz. Um refactor em vários arquivos de um serviço desconhecido precisa.

Escolher Haiku 4.5 para um rename e Opus 4.8 para uma mudança de arquitetura é a mesma disciplina de escolher a ferramenta certa da caixa. O seletor é um clique, então o custo de uma escolha errada é pequeno, e o hábito de escolher é o que a trilha constrói.

A escolha de modelo agora também é uma escolha de orçamento. Desde 1 de junho de 2026, o GitHub Copilot cobra por uso via GitHub AI Credits, onde um crédito equivale a um centavo de dólar. Modelos diferentes no seletor consomem quantidades diferentes de créditos por requisição, então um time que recorre ao maior modelo em toda edição trivial gasta mais do que um time que combina modelo e tarefa. É por isso que a métrica de profundidade e o seletor de modelos são ensinados juntos: orquestração sem disciplina de modelo é orquestração cara.

> DEFINIÇÃO
>
> Letramento de modelo em uma linha: combine o modelo à tarefa, não a tarefa ao modelo. Haiku 4.5 para edições baratas, Opus 4.8 ou Claude Fable 5 para raciocínio difícil, e observe o medidor de AI Credits enquanto aprende.

Trilhas para plataforma, dados, produto e segurança

Os demais clusters recebem cada um uma trilha construída sobre seus próprios artefatos. Plataforma, ou seja DevOps, cloud, DBA e infraestrutura, usa o Copilot para gerar infraestrutura como código em Bicep, Terraform e Bash, e sua vitória rápida é um pipeline do GitHub Actions para um novo serviço, gerado com a configuração do projeto como contexto. Qualidade e SRE usam agent mode com GitHub Actions; a vitória rápida de QA é uma suíte completa de testes unitários para uma classe sem cobertura, que o artigo cronometra em cerca de 15 minutos contra 3 a 4 horas na mão.

Dados e IA, ou seja engenheiro de dados, cientista de dados e analista de BI, trabalham no Azure AI Foundry e no Microsoft Fabric; a vitória rápida é um notebook de análise exploratória produzido em cerca de 15 minutos. Produto e Design, ou seja PM, UX designer e analista de negócios, constroem agentes no Copilot Studio; a

vitória rápida do PM é um agente que responde perguntas sobre o backlog a partir de uma planilha no SharePoint. Segurança e GRC combinam GitHub Advanced Security com Purview; a vitória rápida é a triagem dos três principais CVEs com o GHAS Security Autofix. Liderança usa o Microsoft 365 Copilot para resumos executivos. Mesmo programa, sete vocabulários, um único conjunto de dashboards. O vocabulário muda por cluster, mas a métrica de profundidade e a disciplina de modelo da trilha de engenharia atravessam os sete.

Referências

Fontes primárias para as superfícies do Copilot, os arquivos de customização, o seletor de modelos e o modelo de cobrança por trás dessas trilhas.

- GitHub Copilot Documentation, a referência para agent mode, Chat e as superfícies que cada cluster aprende.
- Customize AI in Visual Studio Code, como escrever o .github/copilot-instructions.md e customizar o Copilot no editor.
- Usage-based billing for individuals: GitHub AI Credits, como funcionam os GitHub AI Credits e por que a escolha de modelo é uma decisão de custo.
- GitHub Copilot is moving to usage-based billing, a mudança de 1 de junho de 2026 para cobrança por uso por trás do medidor de AI Credits.
- DORA 2025 State of AI-assisted Software Development, evidência de campo sobre práticas de desenvolvimento de software assistido por IA.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

04 PARTE III, VITÓRIAS E MÉTRICAS

Quick wins em setenta e duas horas, medidos a cada quinzena.

A confiança é ganha ou perdida nas primeiras setenta e duas horas. O orçamento é mantido ou cortado na revisão trimestral. Este capítulo conecta os dois. Ele entrega um quick win concreto por cluster de personas, transforma cada vitória em um cartão de padrão reutilizável, define seis KPIs que vão do alcance ao risco, e governa a conta com GitHub AI Credits. A cadência de medição é quinzenal, porque um programa sem números perde financiamento.

Um quick win concreto por cluster

Hábitos se formam rápido ou não se formam. A janela de hábito de BJ Fogg fecha por volta das setenta e duas horas, então o programa desenha um quick win concreto por cluster para caber dentro dela. Engenharia leva um item de baixa complexidade do backlog do branch ao pull request em agent mode sem sair do editor. Qualidade gera uma suíte completa de testes unitários para uma classe sem cobertura, tipicamente quinze minutos de trabalho contra três a quatro horas na mão. Plataforma gera um pipeline do GitHub Actions para um novo serviço, usando a configuração do projeto como contexto. Produto cria um agente no Copilot Studio que responde perguntas sobre o backlog a partir de uma planilha no SharePoint. Dados gera um notebook de análise exploratória em quinze minutos. Segurança faz a triagem dos três principais CVEs com o autofix do GitHub Advanced Security. Liderança produz um resumo executivo de retrospectiva a partir do Teams ou do OneNote com o Microsoft 365 Copilot.

Uma regra sustenta todas as vitórias: usar o backlog da própria persona, nunca um tutorial genérico. Cada quick win entrega um artefato real, um pull request, uma suíte de testes, um deploy ou um agente em funcionamento. É esse o sentido da janela de setenta e duas horas. Passado esse ponto, a curva de letramento quebra e a persona desiste em silêncio. Uma vitória que a pessoa pode mostrar a um colega vale mais do que uma demo que ela assistiu.

Cartões de padrão e uma biblioteca de prompts compartilhada

Cada quick win é capturado como um cartão de padrão de uma página. O cartão é a semente da biblioteca de prompts e receitas da equipe. Ele registra seis coisas: a tarefa, o modelo escolhido no seletor do Copilot, o prompt que funcionou, os arquivos de contexto que ele leu, o artefato que produziu e o tempo que economizou. Um cartão é pequeno o bastante para escrever em cinco minutos e específico o bastante para a próxima pessoa repetir o resultado.

A biblioteca vive no repositório, não em um wiki que ninguém abre. As convenções compartilhadas ficam em .github/copilot-instructions.md, para que toda sessão do Copilot as herde sem copiar e colar. Receitas reutilizáveis viram prompt files sob controle de versão. Sistemas internos chegam ao agent mode por servidores Model Context Protocol, para que o agente consulte o contexto real em vez de adivinhá-lo.

De um cartão a uma convenção

Os cartões se acumulam, e os confiáveis merecem promoção. Quando um padrão se prova entre várias pessoas, mova-o para o copilot-instructions.md ou para um prompt file. O agent mode então o lê em cada tarefa, e o quick win vira o jeito padrão de a equipe trabalhar.

Seis KPIs do alcance ao risco

Um programa sem medição perde financiamento, então seis KPIs cobrem o sinal completo de adoção, lidos a cada quinzena. Alcance é a taxa de usuários ativos, o percentual de usuários licenciados ativos pelo menos cinco dias por semana, com meta de oitenta por cento no dia noventa. Profundidade é ações agênticas por semana, uma contagem de sessões de agent mode com múltiplas etapas por desenvolvedor, que separa o autocomplete da orquestração real. Qualidade é a taxa de aceitação e reversão, a taxa de aceitação de sugestões somada à taxa de rollback em sete dias, que expõe a dependência excessiva antes que ela vire um passivo.

Throughput é o quarteto DORA: lead time, frequência de deploy, taxa de falha de mudança e tempo médio de restauração. São sinais defasados, mas são os que

justificam o orçamento. DevEx é uma pesquisa trimestral alinhada ao SPACE que inclui uma pergunta sobre segurança psicológica para céticos e engenheiros júniores. Risco é a contagem de findings do GitHub Advanced Security auto-corrigidos e mesclados por semana, mais as recomendações do Defender for Cloud acionadas.

Sinais principais primeiro, sinais defasados para o CFO

Leia alcance e profundidade toda semana para pegar uma estagnação enquanto ainda é barato corrigir. O quarteto DORA se move devagar, mas é o que sobrevive a uma revisão de orçamento. O outro lado está documentado: sem onboarding estruturado, a adoção ativa cai para doze por cento em sessenta dias.

Governança de custo com GitHub AI Credits

Desde 1 de junho de 2026, o GitHub Copilot cobra por uso através dos GitHub AI Credits. Um crédito é um centavo de dólar, então mil créditos são dez dólares. Cada requisição consome créditos conforme o modelo e o modo. O seletor do Copilot expõe a escolha diretamente: Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro e a família GPT-5.x. O roteamento agora é uma decisão de custo, não só de qualidade.

Governança significa orçamentos e visibilidade, não congelamento. Defina orçamentos de créditos por licença e por equipe, e leia o painel de uso ao lado dos seis KPIs. Roteie os quick wins mecânicos para um modelo rápido e barato, e reserve os modelos premium para arquitetura e depuração difícil. Um quick win que queima créditos premium em uma edição trivial é uma lição para o cartão de padrão, não um resultado a comemorar. Como cada cartão já registra o modelo e os créditos, a biblioteca funciona também como um guia de roteamento que afia a cada quinzena.

DEFINIÇÃO

Governança de custo em uma regra: case o modelo com a tarefa. Um GitHub AI Credit é um centavo de dólar. Um conserto de teste de uma linha não precisa do Opus 4.8, e um refactor entre serviços não deveria rodar no Haiku 4.5. O cartão de padrão registra o modelo, os créditos gastos e o tempo economizado, para que o roteamento fique mais barato e melhor a cada quinzena.

Referências

Fontes primárias para o modelo de cobrança, a superfície de customização e as métricas de throughput.

- GitHub Copilot Documentation, a plataforma de referência para o agent mode, o seletor de modelos e os padrões de onboarding deste capítulo.
- GitHub Copilot is moving to usage-based billing, a mudança para cobrança por consumo que entrou em vigor em 1 de junho de 2026.
- Usage-based billing for individuals: GitHub AI Credits, como os créditos são medidos por requisição e por modelo, a base da governança de custo.
- Customize AI in Visual Studio Code, copilot-instructions.md e prompt files, a superfície da biblioteca de prompts compartilhada.
- DORA 2025 State of AI-assisted Software Development, o quarteto de throughput e a evidência mais ampla de adoção por trás dos KPIs.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

05 SUSTENTE A ADOÇÃO

Sustente a adoção com gestão de mudança e um Centro de Excelência.

Um rollout que entrega licenças e um e-mail de boas-vindas atinge o pico por volta da segunda semana e desaba para cerca de 12% de uso ativo no dia sessenta. A adoção sobrevive quando o programa trata a resistência como um sinal a redirecionar, não como um obstáculo a remover, e quando uma estrutura durável assume a cadência depois que o programa formal termina. Este capítulo mapeia a resistência, converte-a em quatro passos, aplica Kotter à sequência e ergue um Centro de Excelência.

O mapa de resistência e a conversão em quatro passos

A resistência a uma ferramenta nova não é irracional. É a resposta de profissionais que aprenderam a ser competentes de um jeito e agora enfrentam a incerteza de aprender de outro. William Bridges, em Managing Transitions, descreve a curva de mudança pela qual as pessoas passam em velocidades diferentes: negação, resistência, exploração e comprometimento. O programa precisa estar pronto para encontrar cada persona em qualquer ponto dessa curva. Cada persona carrega uma resistência característica, um medo de raiz e um gatilho de conversão específico. O dev sênior diz que a saída do agente vai piorar a qualidade do código; o medo é a identidade como especialista técnico; o gatilho é mostrar que o AI review é mais rigoroso, não mais frouxo. O gerente de produto diz que isso é ferramenta de desenvolvedor; o gatilho é uma demo do Copilot Studio que resolve um problema real de produto. O CISO diz que a empresa não controla o que sai dela; o gatilho é uma sessão de arquitetura de segurança que mostra onde os dados ficam e que o GitHub Copilot Enterprise não treina modelos com o código da empresa.

A conversão segue um padrão de quatro passos derivado de Cialdini sobre influência e do framework Prosci ADKAR. Primeiro, reconheça a objeção sem

descartá-la: nomeie-a como exatamente a falha que o programa foi construído para evitar. Segundo, ofereça evidência específica, não uma alegação genérica: desenvolvedores formalmente treinados produzem qualidade de código mensuravelmente maior do que colegas autodidatas, então cite o número para a preocupação daquela persona. Terceiro, dê controle à persona: deixe que ela defina o critério de sucesso e a data de revisão, porque autonomia reduz a resistência mais do que a persuasão. Quarto, crie a primeira experiência positiva com a pessoa resistente como protagonista, resolvendo ao vivo um problema real dela, para que a vitória pertença a ela e não à ferramenta.

> DEFINIÇÃO
>
> Quatro passos para converter resistência: reconheça a objeção sem descartá-la, ofereça evidência específica em vez de uma alegação genérica, dê à persona controle sobre o critério de sucesso e monte a primeira experiência positiva com a pessoa resistente como protagonista. A vitória precisa pertencer à pessoa, não à ferramenta.

Kotter aplicado à adoção agêntica

O modelo de oito etapas de Kotter para liderar a mudança é o framework de mudança mais documentado em contextos corporativos, e ele se encaixa com clareza em um programa de adoção agêntica. Crie urgência apresentando evidência de mercado de que times que não adotam ficam para trás; o patrocinador executivo é dono dessa etapa. Forme a coalizão condutora nomeando um Champion para cada cluster e um patrocinador C-level antes de o programa começar; o coordenador do CoE é dono dela. Crie a visão definindo, de forma concreta e por cluster, como é o trabalho em seis meses com agentes. Comunique a visão por showcases, um canal no Teams, demos ao vivo e depoimentos de quem já usa a ferramenta.

A segunda metade do modelo é onde os programas empacam. Remova obstáculos resolvendo bloqueios de licença, acesso e segurança antes que virem desculpa. Gere vitórias de curto prazo com os quick wins nas semanas três a seis, divulgados amplamente. Consolide os ganhos usando essas vitórias para justificar expansão, novos casos de uso e mais recursos. Ancore a mudança na cultura colocando KPIs

de adoção de IA nos OKRs dos times e nos critérios de desempenho. A falha mais comum é pular da etapa um direto para a etapa cinco, removendo obstáculos técnicos antes de existirem urgência, coalizão e visão. O resultado é um programa tecnicamente impecável que ninguém usa.

O Agents Center of Excellence e a comunidade de prática

O Centro de Excelência é a estrutura que sustenta o programa depois dos primeiros seis meses. Ele não é um departamento novo. É uma rede de papéis distribuídos em times existentes, baseada no framework de Center of Excellence do Gartner. O coordenador do CoE é dono da agenda, das métricas e da comunicação, cerca de meio FTE para uma organização de 100 a 500 desenvolvedores. Um Champion por cluster é um especialista técnico que recebe treinamento avançado, responde às dúvidas do time e documenta casos de uso, aproximadamente um Champion para cada vinte pessoas. Um patrocinador executivo assina as decisões de política e autoriza o budget, sem o qual o CoE vira um comitê sem poder. Um guardião de governança, tipicamente de segurança ou conformidade, confirma que novos casos de uso de agentes seguem a política aprovada antes de chegarem à produção.

A comunidade de prática complementa o CoE formal. Seguindo Wenger, McDermott e Snyder, uma comunidade eficaz precisa de um domínio definido, um corpo de membros e prática compartilhada. Na operação, isso é um canal dedicado no Microsoft Teams, um show-and-tell quinzenal de 45 minutos com dois casos e discussão aberta, e um repositório interno de templates de prompt, guias de troubleshooting e casos documentados. A camada durável de convenção vive em .github/copilot-instructions.md, onde cada time codifica como sua base de código realmente funciona, para que o agent mode e as ferramentas MCP a herdem. A governança do conhecimento mantém a biblioteca atual: cada caso de uso em produção é documentado com o problema, a persona, a ferramenta, o prompt ou a configuração, o modelo escolhido no picker do GitHub Copilot e o resultado medido, revisado a cada trimestre.

O roteiro de seis meses e as ações de segunda-feira de manhã

O programa roda um pré-lançamento mais quatro estágios ao longo de vinte e quatro semanas, e cada transição carrega um marco de go/no-go. Champions treinados chega a 100% dos clusters antes do lançamento, ou o lançamento espera. Licenças provisionadas chega a 100% dos participantes no dia zero, ou escala para a TI e o patrocinador. Quick wins passam de 70% dos participantes até a semana seis, ou o CoE roda sessões extras de suporte por cluster. Uso ativo semanal supera 60% até o mês três, ou o time revisita a relevância dos casos de uso de cada cluster. O ROI é documentado com ao menos três casos mensuráveis até o mês seis, ou as métricas são reestruturadas para casos mais visíveis. O consumo via agent mode é cobrado como GitHub AI Credits, onde um crédito equivale a um centavo de dólar, então o argumento de ROI reconcilia a produtividade medida com uma linha real de créditos.

Três ações começam o programa na segunda-feira. Primeiro, publique a política de uso de IA por escrito; nada avança enquanto ela não existir, e sua ausência é o bloqueio que o CISO tem à mão. Segundo, nomeie dois Champions por time e agende seu treinamento avançado de duas semanas. Terceiro, agende a primeira demo específica por cluster usando um artefato real do backlog atual, porque tutoriais genéricos falham na primeira semana. Onboarding não é um evento. Execute-o como uma plataforma, com um mapa de resistência, um padrão de conversão, uma sequência de mudança e um Centro de Excelência que assume a cadência muito depois de o programa formal se encerrar.

> DEFINIÇÃO
>
> Segunda-feira de manhã, três ações: publique a política de uso de IA por escrito, nomeie dois Champions por time e agende seu treinamento avançado de duas semanas, e agende a primeira demo por cluster usando um artefato real do backlog atual. Onboarding não é um evento. Execute-o como uma plataforma.

Referências

Fontes primárias para os modelos de gestão de mudança, a estrutura da comunidade de prática e a plataforma GitHub Copilot sobre a qual este capítulo se apoia.

- Kotter, Leading Change, o modelo de oito etapas aplicado à sequência de adoção.
- Cialdini, Influence: The Psychology of Persuasion, a base do padrão de conversão em quatro passos.
- Prosci ADKAR, o framework de gestão de mudança por trás da conversão de personas.
- William Bridges, Managing Transitions, a curva de mudança que o mapa de resistência segue.
- Wenger, McDermott e Snyder, Cultivating Communities of Practice, o modelo para a comunidade de prática.
- Gartner, Infrastructure and Operations, o framework de Center of Excellence.
- GitHub Copilot Documentation, a plataforma de referência para agent mode, copilot-instructions.md e MCP.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps