GITHUB COPILOT ONBOARDING · PLAYBOOK

El onboarding estructurado decide si GitHub Copilot entrega valor o se estanca en 23%.

Las empresas no fracasan con GitHub Copilot porque la herramienta sea débil. Fracasan porque la alfabetización es desigual. Este playbook convierte una licencia en un programa: veinticuatro personas del SDLC en siete clústeres, una jornada de cinco etapas de la preparación a la autonomía, una victoria rápida en las primeras setenta y dos horas y un Centro de Excelencia que asume la cadencia después de que el programa formal termina. Los equipos con onboarding formal alcanzan 80 por ciento de adopción activa en noventa días; los equipos sin él se estacionan cerca de 23 por ciento.

AUTORA

Paula Silva

CARGO

AI-Native Software Engineer

CONTACTO

in/paulanunes

TIEMPO DE LECTURA

~ 45 minutos, de punta a punta

24

PERSONAS DEL SDLC

7

CLÚSTERES DE PERSONAS

5

ETAPAS DE ONBOARDING

80%

META DE ADOPCIÓN ACTIVA

CAPÍTULOS

Chapter 00

Por qué el onboarding estructurado decide el retorno

Por qué GitHub Copilot no entrega valor cuando la alfabetización es desigual, qué dice la evidencia sobre la adopción activa, las siete trampas que estancan la mayoría de los rollouts y los seis principios sobre los que se construye el programa.

Fundamentos →

Chapter 01

Veinticuatro personas del SDLC en siete clústeres

Los veinticuatro roles del SDLC agrupados en siete clústeres, el dolor que carga cada uno hoy y la ganancia que traen los agentes, la matriz de tres planos y un vocabulario por clúster que reporta a un dashboard compartido.

Personas →

Chapter 02

Un programa de cinco etapas de la preparación a la autonomía

Las cinco etapas de una puerta de preparación a la autonomía, cada una con un criterio de salida explícito, apoyadas en una pila de habilitación de cuatro capas de personas, etapas, labs y medición.

Framework →

Chapter 03

Rutas de alfabetización por clúster y herramienta Copilot

Cómo cada clúster mapea a una superficie de GitHub Copilot: agent mode para ingeniería, instrucciones de repositorio y MCP para contexto, el selector de modelos como decisión de costo y rutas para plataforma, datos, producto y seguridad.

Alfabetización →

Chapter 04

Quick wins en setenta y dos horas, medidos cada quincena

Un quick win concreto por clúster dentro de la ventana de setenta y dos horas, tarjetas de patrón y una biblioteca de prompts compartida, seis KPIs del alcance al riesgo y gobernanza de costo con GitHub AI Credits.

Métricas →

Chapter 05

Sostén la adopción con gestión del cambio y un Centro de Excelencia

El mapa de resistencia y una conversión en cuatro pasos, Kotter aplicado a la adopción agéntica, el Agents Center of Excellence y la comunidad de práctica, y el roadmap de seis meses con acciones del lunes por la mañana.

Adopción →

CONSEJOS DE LECTURA

> **ATAJOS DE TECLADO**
>
> Presiona `/` para buscar. Usa `j` y `k` para moverte entre capítulos.
> Presiona `t` para cambiar tema. Presiona `g` para ir al inicio.

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 LA BRECHA DE ALFABETIZACIÓN

Por qué el onboarding estructurado decide el retorno.

Las empresas no fracasan con GitHub Copilot porque la herramienta sea débil. Fracasan porque la alfabetización es desigual. Se compran licencias, se provisionan asientos, y la adopción igual se estanca: los ingenieros senior corren con la herramienta, QA nunca la abre, el gerente de producto no ve la conexión con su trabajo, y el CISO tiene dudas abiertas. La diferencia entre pagar por IA y obtener valor de ella es el programa que envuelve a la herramienta. Este capítulo mide esa brecha, expone la evidencia sobre la adopción activa, nombra las siete trampas que estancan la mayoría de los rollouts y presenta los seis principios sobre los que se construye el programa.

La brecha de alfabetización que nadie presupuesta

Las organizaciones compran licencias de GitHub Copilot y esperan que la adopción venga detrás. No viene. Según el GitHub Octoverse 2025, los equipos con onboarding formal alcanzan 80% de adopción activa en 90 días. Los equipos sin él llegan al 23%. Eso es una brecha de 3,5x. El diferenciador no es el presupuesto, no es la antigüedad, no es el stack tecnológico. Es el programa construido alrededor de la herramienta. La brecha tiene un nombre que ninguna línea de costo captura: alfabetización desigual. Los ingenieros senior corren con la herramienta, QA nunca la abre, el gerente de producto no ve la conexión con su trabajo, y el CISO tiene dudas abiertas. La licencia es uniforme. La alfabetización no.

El rollout por defecto es una licencia y un correo de bienvenida. La adopción activa alcanza su pico alrededor de la segunda semana y luego cae a aproximadamente 12% en el día 60 (Forrester 2025). El modelo de facturación vuelve legible ese desperdicio. GitHub Copilot pasó a facturación basada en uso el 1 de junio de 2026, tarifada en GitHub AI Credits, donde un crédito equivale a un centavo de dólar. Las ejecuciones en agent mode y los premium requests consumen esos créditos. Una

organización que provisiona los asientos pero nunca construye alfabetización paga por los créditos y captura casi nada del retorno. El onboarding se trata como un evento. Tiene que correr como un programa.

Qué dice la evidencia sobre la adopción activa

La evidencia es consistente entre las fuentes. Los programas formales alcanzan 80% de adopción activa en 90 días; los rollouts informales se estacionan cerca del 23% (GitHub Octoverse 2025). Cada dólar invertido en formación estructurada retorna USD 4,20; la formación autodidacta retorna USD 1,80. MIT Sloan 2024 constató que enmarcar el programa como aumentación, no automatización, produce 2,8x más compromiso. El número que paga el programa no es el conteo de asientos. Es la tasa de usuarios activos, el porcentaje de usuarios con licencia activos al menos cinco días por semana.

La profundidad importa tanto como el alcance. La adopción activa se define como al menos una sugerencia de GitHub Copilot aceptada por día laboral en las dos semanas anteriores. Esa definición separa a quien tiene asiento de quien practica. Un desarrollador que corre una sesión de agent mode de múltiples pasos por semana usa la plataforma de forma distinta a quien acepta un autocompletado y sigue. El programa enseña el segundo comportamiento: elegir el modelo correcto en el Copilot picker para la tarea, Claude Fable 5 u Opus 4.8 para razonamiento difícil, Sonnet 5 o Haiku 4.5 para ediciones rápidas, Gemini 3.1 Pro o un modelo GPT-5.x cuando la tarea lo pide, y escribir el prompt de agent mode que convierte un ítem de backlog en un pull request. Alcance sin profundidad es una métrica de vanidad.

Siete trampas que estancan la mayoría de los rollouts

La evidencia de campo del McKinsey Global Survey on AI 2024 y del Forrester AI Adoption Pitfalls 2025 apunta a una lista corta de modos de fallo recurrentes. Formación puntual sin seguimiento: la adopción cae al 12% en 60 días. Programas exclusivos para ingenieros: silos de adopción, y cerca de la mitad del retorno potencial queda sin capturar porque QA, producto y seguridad nunca empiezan. Sin casos de uso reales: los tutoriales genéricos matan la motivación en la primera

semana. Sin política clara de uso de IA: el CISO bloquea el rollout y los equipos se paralizan.

Las tres restantes son más silenciosas e igual de fatales. Sin champions internos: la práctica nunca se difunde socialmente, así que queda atrapada en el bolsillo de los primeros adoptantes. Sin métricas, o con las equivocadas: el presupuesto se recorta en la primera revisión trimestral porque nadie puede mostrar el retorno. Un currículo único para todos: alta deserción y baja relevancia, porque un gerente de producto y un ingeniero backend no aprenden la misma herramienta de la misma manera. Cada una de las siete remite a un solo error, tratar un evento de despliegue como un programa terminado.

Seis principios sobre los que se construye el programa

El programa se apoya en seis principios que la investigación valida como determinantes del éxito. Persona-first: cada rol tiene sus propias motivaciones, miedos, vocabulario y contexto de trabajo, así que una guía genérica de GitHub Copilot no funciona para un gerente de producto, y la alfabetización tiene que hablar el idioma de cada persona. Quick wins primero: los hábitos cambian con refuerzo positivo, así que el programa diseña una victoria visible dentro de las primeras 72 horas antes de introducir conceptos avanzados. Aprendizaje social: Rogers, en Diffusion of Innovations, muestra que la adopción se acelera cuando la gente ve a sus pares tener éxito, no solo a expertos, y por eso las comunidades internas de práctica superan a la formación top-down.

Los otros tres gobiernan la confianza y la durabilidad. Seguridad psicológica: las personas no experimentan con una herramienta nueva cuando temen parecer incompetentes, así que el ambiente tiene que ser explícitamente seguro para el ensayo y error. Gobernanza visible: la adopción sube cuando las reglas están claras, qué puede y qué no puede hacer el agente, quién es responsable de qué, cómo se protegen los datos, y esa claridad importa más para las personas no técnicas y los líderes. Medición continua: sin métricas el programa se vuelve una caja negra, así que la adopción, la calidad y la satisfacción se miden cada quincena y el programa se ajusta. En la etapa de autonomía estos principios se endurecen en artefactos: un

.github/copilot-instructions.md compartido que codifica las convenciones una vez, y servidores MCP que dan al agent mode acceso gobernado al contexto interno.

> DEFINICIÓN
>
> Seis principios, una regla: el onboarding no es un evento, es una plataforma. Persona-first sobre genérico. Quick wins sobre teoría. Aprendizaje social sobre top-down. Seguridad psicológica sobre miedo. Gobernanza visible sobre ambigüedad. Medición continua sobre fe.

Referencias

Fuentes primarias para la brecha de adopción, la evidencia empírica, las siete trampas y los principios sobre los que se construye el programa.

- GitHub Octoverse 2025, la brecha de adopción: 80% de adopción activa con onboarding formal frente a 23% sin él.
- Forrester, AI Adoption Pitfalls, la adopción cae a cerca del 12% en 60 días sin un programa sostenido.
- MIT Sloan Management Review, AI in the Workplace, el enfoque de aumentación produce 2,8x más compromiso que el de automatización.
- McKinsey Global Survey on AI, evidencia de campo para los modos de fallo recurrentes de adopción.
- Everett M. Rogers, Diffusion of Innovations, 5th Edition, la base de aprendizaje social para las comunidades de práctica por encima de la formación top-down.
- GitHub Copilot Documentation, la plataforma de referencia para agent mode, el model picker y el .github/copilot-instructions.md.
- Usage-based billing and GitHub AI Credits, el modelo de créditos que vuelve visible el costo de una adopción estancada.

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 ROLES Y CLÚSTERES

Veinticuatro personas del SDLC en siete clústeres.

Las organizaciones que obtienen valor de la IA agéntica hacen onboarding de toda la organización de entrega, no solo de sus desarrolladores senior. Este capítulo mapea esa organización: veinticuatro roles del SDLC agrupados en siete clústeres, cada uno con su dolor de hoy, su ganancia con agentes, sus tres planos de herramientas y su vocabulario. Los clústeres difieren en casi todo, excepto en los dashboards a los que reportan. Ese es el diseño. Habla el idioma de cada clúster, mídelos a todos en un mismo panel.

Siete clústeres, veinticuatro roles

El SDLC no es un único rol, y cualquier programa que lo trate como tal falla en el momento en que llega a alguien que no es desarrollador senior. Este capítulo mapea toda la organización de entrega: veinticuatro roles, agrupados en siete clústeres. El agrupamiento se apoya en tres referencias que las organizaciones de entrega ya usan para describirse, el DORA 2024 State of DevOps report, el SAFe 6.0 y el IEEE SWEBOK v4. Los siete clústeres son Ingeniería con seis roles, Calidad y Confiabilidad con tres, Plataforma e Infraestructura con cuatro, Producto y Diseño con tres, Datos e IA con tres, Seguridad y Cumplimiento con dos, y Liderazgo y Estrategia con tres.

El clúster, y no el individuo, es la unidad de onboarding. Dos ingenieros de backend y un ingeniero mobile comparten una herramienta principal, un flujo de revisión y un vocabulario, así que se forman juntos y se miden juntos. Un ingeniero de QA y un gerente de producto no lo comparten, así que ponerlos en la misma sala produce una demo que no conecta con ninguno de los dos. El agrupamiento mantiene cada sesión anclada en el trabajo real de un clúster: los artefactos en pantalla vienen del sprint actual de ese clúster, y las convenciones que el clúster ya sigue van a un archivo compartido .github/copilot-instructions.md que toda sesión de agente de ese clúster lee antes de escribir una línea.

El dolor de cada persona hoy y la ganancia con agentes

Cada clúster carga un dolor específico que es anterior a cualquier agente. Ingeniería pierde horas con boilerplate y refactorización repetitiva, y los ingenieros full stack pierden aún más con el cambio de contexto entre capas. Calidad y Confiabilidad escribe el esqueleto de las pruebas a mano y diagnostica incidentes a partir de logs dispersos. Plataforma e Infraestructura repite configuración y persigue drift entre muchos sistemas. Producto y Diseño sintetiza feedback y requisitos manualmente. Datos e IA depura pipelines y espera por largos ciclos de experimentación. Seguridad y Cumplimiento hace triaje de alertas y falsos positivos y mapea controles a políticas a mano. Liderazgo y Estrategia no tiene una lectura limpia de productividad ni de ROI de herramientas.

De autor a revisor

La ganancia tiene la misma forma en todos los clústeres: el trabajo mecánico pasa al agente, y el juicio se queda con la persona. Ingeniería usa GitHub Copilot en agent mode para llevar un ítem de baja complejidad del backlog desde la rama hasta el pull request sin salir del editor. Calidad genera una suite completa de pruebas unitarias para una clase sin cobertura en unos quince minutos frente a tres a cuatro horas a mano, y los equipos que generan pruebas a partir de requisitos reportan de 60 a 70 por ciento menos tiempo para escribirlas. Plataforma describe un módulo Bicep o Terraform y revisa el código generado. Producto construye un agente en Copilot Studio que responde preguntas del backlog desde una hoja de SharePoint, y los PMs reportan recuperar de cuatro a seis horas por semana. Seguridad hace el triaje de los principales findings con GitHub Advanced Security y Copilot Autofix. Liderazgo lee métricas DORA en lugar de anécdotas.

El hilo común es un cambio de rol. El ingeniero de plataforma deja de escribir Terraform desde cero y pasa a revisar el código generado. El ingeniero de QA deja de escribir el esqueleto y pasa a decidir cuáles de un centenar de casos generados realmente importan. Los desarrolladores formados formalmente producen código con 55% más de calidad que sus pares autodidactas con la misma herramienta, que es la distancia medible entre aceptar sugerencias y dirigirlas.

La matriz de herramientas con planos primario, secundario y de gobernanza

Cada clúster recibe tres planos de herramientas, no uno. El plano primario es donde el clúster hace su trabajo diario. El plano secundario lo extiende. El plano de gobernanza mantiene el trabajo auditable. Ingeniería usa GitHub Copilot en agent mode como primario, GitHub Actions como secundario y GitHub Advanced Security como gobernanza. Calidad usa GitHub Copilot para pruebas, Azure Monitor con Application Insights como secundario y GitHub Actions en CI como gobernanza. Plataforma usa GitHub Copilot para infraestructura como código, el Azure Developer CLI como secundario y Defender for Cloud como gobernanza. Producto usa Copilot Studio, Microsoft 365 Copilot como secundario y Purview sobre datos de investigación como gobernanza. Datos e IA usa Azure AI Foundry, Microsoft Fabric como secundario y Purview como catálogo. Seguridad usa GitHub Advanced Security, Defender for Cloud como secundario y Purview con Compliance Manager como gobernanza. Liderazgo usa Microsoft 365 Copilot, GitHub Insights con métricas DORA como secundario y Azure AI Foundry para gobernanza.

Los tres planos comparten una superficie de facturación y una superficie de modelo. El trabajo en agent mode se factura como GitHub AI Credits, donde un crédito equivale a un centavo de dólar, bajo la facturación por uso que entró en vigor el 1 de junio de 2026, así que la métrica de profundidad y la factura leen la misma señal. El modelo detrás de cada sesión viene del selector de Copilot: Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro y la familia GPT-5.x, así que un clúster combina un modelo con una tarea sin salir de la herramienta. En la etapa de autonomía, los clústeres extienden el plano primario con Copilot Extensions construidas sobre el Model Context Protocol, que es como los propios sistemas de un clúster se convierten en contexto de primera clase para sus agentes.

> DEFINICIÓN
>
> Tres planos por clúster: el primario es donde ocurre el trabajo diario, el secundario lo extiende, la gobernanza lo mantiene auditable. Los tres se facturan como GitHub AI Credits y toman su modelo del selector de Copilot, así que profundidad y costo leen una misma señal.

Un vocabulario por clúster, un conjunto de dashboards

Clústeres distintos hablan idiomas distintos, y el programa lo respeta. Una demo para ingenieros habla de branches, pull requests y refactorización. Una demo para gerentes de producto habla de backlog, feedback y roadmap. Una demo para seguridad habla de findings, autofix y trazas de auditoría. La misma herramienta mostrada en el vocabulario equivocado suena como la herramienta de otra persona, que es la razón más común para que un clúster nunca la abra. El encuadre también importa aquí. El MIT Sloan constata que el mensaje de aumentación produce 2,8 veces el compromiso del mensaje de automatización, y la aumentación es más fácil de escuchar en el propio vocabulario.

El vocabulario diverge, los dashboards convergen. Cada clúster reporta al mismo panel, así que el liderazgo lee una única señal en toda la organización. Alcance es la tasa de usuarios activos. Profundidad son acciones agénticas por semana, el conteo de sesiones de agent mode con múltiples pasos que separa el autocompletado de la orquestación. Throughput es el cuarteto DORA de lead time, frecuencia de despliegue, tasa de fallos de cambio y tiempo de recuperación. Riesgo son los findings de GitHub Advanced Security corregidos y fusionados. Un vocabulario por clúster mantiene la adopción real; un conjunto de dashboards la mantiene financiable.

> DEFINICIÓN
>
> Un clúster, una herramienta principal, un vocabulario, una fila en el dashboard compartido. Los veinticuatro roles nunca se forman como veinticuatro currículos separados y nunca reportan como veinticuatro métricas separadas. Siete clústeres es la unidad lo bastante pequeña para hablar cada idioma y lo bastante grande para medir como un programa.

Referencias

Fuentes primarias para el mapa de siete clústeres, la matriz de herramientas por clúster y el dashboard compartido.

- DORA, 2024 State of DevOps Report, referencia para mapear los roles de entrega en los siete clústeres.
- Scaled Agile Framework, SAFe 6.0, la taxonomía de roles detrás del mapa de clústeres.
- IEEE SWEBOK v4, Software Engineering Body of Knowledge, la referencia para los perfiles de rol de cada clúster.
- GitHub Copilot Documentation, el plano primario de los clústeres de Ingeniería, Calidad y Plataforma.
- GitHub Advanced Security, el plano de gobernanza de Ingeniería y el plano primario de Seguridad y Cumplimiento.
- Microsoft Copilot Studio Documentation, el plano primario del clúster de Producto y Diseño.
- Azure AI Foundry Documentation, el plano primario del clúster de Datos e IA.
- DORA, the four key metrics, la señal de throughput en el dashboard compartido.

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 ETAPAS

Un programa de cinco etapas, de la preparación a la autonomía.

Una licencia de GitHub Copilot no crea fluidez. Un programa sí. Este capítulo presenta las cinco etapas que el programa recorre a lo largo de veinticuatro semanas: una puerta de preparación que debe cerrarse antes de que cualquier persona empiece, el enfoque de aumentación y las victorias rápidas que construyen el primer hábito, y la fluidez operacional y la autonomía que permiten que el programa formal se retire. Cada etapa lleva un criterio de salida explícito, así que el progreso se mide por evidencia, no por el calendario. Toda la secuencia se apoya en una pila de habilitación de cuatro capas, y saltarse cualquier capa rompe la de arriba.

Etapa 0: preparación y la puerta de prerrequisitos

La Etapa 0 va de la semana menos dos a la semana cero, y es una puerta, no un calentamiento. Nueve prerrequisitos deben estar en su lugar antes de que una sola persona vea una demo, sin excepciones. Publica la política de uso de IA por escrito. Provisiona GitHub Copilot Enterprise con agent mode habilitado, y define un archivo .github/copilot-instructions.md por repositorio para que cada sesión de agente lea las mismas convenciones. Provisiona los espacios de trabajo adyacentes que los clústeres van a necesitar: Azure AI Foundry, Copilot Studio, GitHub Advanced Security, Defender for Cloud, Microsoft Fabric y Purview. Identifica dos champions por equipo y somételos a dos semanas de formación avanzada antes del kickoff general. Define los KPIs del programa por escrito para que el primer informe tenga una línea base contra la cual medir. El criterio de salida es concreto: política publicada, champions formados, sandboxes listos.

La puerta importa porque las tres formas más comunes en que un rollout se estanca se remontan todas a un prerrequisito que se saltó. Sin una política escrita, el CISO bloquea el rollout y los equipos se paralizan. Sin champions formados, no hay quién responda a la primera pregunta difícil, y la adopción nunca se difunde más allá de los

primeros voluntarios. Sin sandboxes provisionados, la primera victoria rápida pierde su ventana y el hábito nunca se forma. El presupuesto también pertenece a la puerta. Desde el 1 de junio de 2026, GitHub Copilot cobra el uso premium en GitHub AI Credits, donde un crédito es un centavo de dólar, así que el responsable del programa dimensiona la asignación de créditos antes del piloto en lugar de reaccionar a la primera factura. La puerta de prerrequisitos no es burocracia. Es el seguro más barato que compra el programa.

Etapas 1 y 2: enfoque de aumentación y victorias rápidas

La Etapa 1 es la concienciación, semanas uno y dos, y todo su trabajo es el enfoque. Presenta el programa como aumentación, no automatización. La distinción no es cosmética: MIT Sloan reporta que un enfoque de aumentación produce cerca de 2,8 veces el compromiso de un enfoque de automatización, porque las personas se acercan a una herramienta que las hace mejores y se alejan de la que leen como reemplazo. Ejecuta demos en vivo de sesenta a noventa minutos, una por clúster, y construye cada demo sobre un artefacto real tomado del sprint actual. Los tutoriales genéricos fracasan aquí, todas las veces, porque un QA que observa una refactorización de backend no ve nada que mapee al trabajo frente a él. El criterio de salida es participación con relevancia: cada persona asistió a una demo que usó su propio backlog, no un repositorio de ejemplo.

La Etapa 2 son las victorias rápidas, semanas tres a seis, y está cronometrada para la ventana de hábito. El primer éxito medible debe ocurrir dentro de 72 horas, el umbral por debajo del cual un nuevo comportamiento tiende a fijarse y por encima del cual tiende a perderse. Cada clúster obtiene una victoria concreta. Ingeniería lleva un ítem de baja complejidad del backlog desde la rama hasta el pull request en agent mode sin salir del editor. QA genera una suite completa de pruebas unitarias para una clase sin cobertura, lo que normalmente toma cerca de 15 minutos frente a 3 a 4 horas a mano. Plataforma genera un pipeline de GitHub Actions para un nuevo servicio, usando la configuración del proyecto como contexto. Producto construye un agente en Copilot Studio que responde preguntas del backlog desde una hoja de SharePoint. Seguridad realiza el triaje de los tres principales CVEs con GHAS Security Autofix. Datos genera un notebook exploratorio en cerca de 15 minutos. El selector de modelos es parte del oficio: dirige las tareas de alto volumen y bajo

riesgo a Haiku 4.5 para conservar créditos, y reserva Opus 4.8 o Claude Fable 5 para el trabajo pesado de razonamiento. Cada victoria se captura como una tarjeta de patrón de una página, la semilla de la biblioteca de prompts y recetas. El criterio de salida: cada persona registró al menos un artefacto de victoria rápida.

Etapas 3 y 4: fluidez operacional y autonomía

La Etapa 3 es la fluidez operacional, semanas siete a catorce, cuando el uso ocasional se vuelve práctica diaria. La mecánica es hands-on: sesiones de pair programming, dojos de prompt donde los equipos refinan y comparan prompts sobre una tarea compartida, y un showcase mensual donde las personas presentan lo que entregaron. Es aquí donde el archivo .github/copilot-instructions.md se paga. Una convención que un equipo discute en un dojo, cómo maneja excepciones, cómo nombra pruebas, cómo estructura un servicio, se vuelve contexto permanente que cada sesión de agente futura lee sin necesidad de una nueva explicación. La biblioteca de prompts deja de ser una carpeta de fragmentos y se convierte en un activo curado. El criterio de salida tiene tres partes: 60% de uso semanal activo, una biblioteca de prompts curada y un showcase corriendo en cadencia fija.

La Etapa 4 es la autonomía, semanas quince a veinticuatro, y es donde el programa planea su propia salida. Los equipos construyen agentes personalizados para sus flujos recurrentes y extienden el agent mode con servidores MCP que exponen sistemas internos, un rastreador de issues, una API de despliegue, un repositorio de documentación, como contexto de primera clase que el agente puede consultar directamente. La selección de modelo se vuelve deliberada, no por defecto: los equipos eligen en el selector de Copilot por tarea, emparejando Sonnet 5 o Gemini 3.1 Pro con la forma del trabajo y el presupuesto de créditos. Los OKRs empiezan a llevar métricas de IA como una línea normal, no una novedad. El criterio de salida es la señal más clara de todo el programa: los champions retiran la estructura formal y una comunidad de práctica asume la cadencia. La etapa de autonomía tiene éxito cuando el programa logra retirarse.

Criterios de salida y la pila de habilitación de cuatro capas

Los cinco criterios de salida son la columna vertebral del programa, y son acumulativos. Un equipo no avanza porque pasaron seis semanas; avanza porque cumplió el criterio, una demo asistida, un artefacto registrado, un umbral de uso semanal. Puertas, no calendarios, deciden el progreso, lo que impide que una etapa se declare terminada mientras la mitad de las personas siguen atascadas. Cuando una etapa se estanca, el criterio dice exactamente qué capa reforzar antes de seguir.

La pila de habilitación de cuatro capas

Bajo las etapas hay una pila de habilitación de cuatro capas, y cada capa depende de la de abajo. La capa uno son las personas: los 24 roles del SDLC agrupados en siete clústeres, la base a la que todo lo demás se ajusta. La capa dos es la jornada de cinco etapas que lleva esas personas de la conciencia a la autonomía. La capa tres son los labs: el pair programming, los dojos de prompt y los showcases que transforman el enfoque en práctica. La capa cuatro es la medición: los KPIs, la señal de DevEx y el Centro de Excelencia que sostiene la cadencia después de que los champions se retiran. Sáltate la base y la adopción vuelve al umbral de 23% que reportan los programas sin estructura.

> DEFINICIÓN
>
> Cinco etapas, un criterio de salida cada una: la preparación cierra una puerta de nueve ítems, la concienciación enmarca aumentación sobre automatización, las victorias rápidas ocurren dentro de 72 horas, la fluidez alcanza 60% de uso semanal y la autonomía retira el programa hacia una comunidad de práctica. La pila de abajo, personas, etapas, labs, medición, solo se sostiene si cada capa descansa sobre la de abajo.

Referencias

Fuentes primarias para el modelo de etapas, la base de gestión del cambio, la evidencia del enfoque de aumentación y la mecánica de la plataforma GitHub Copilot

referenciada en este capítulo.

- GitHub Copilot Documentation, agent mode, .github/copilot-instructions.md y MCP, la mecánica de plataforma sobre la que se construye el programa.
- Customize AI in Visual Studio Code, cómo los archivos de instrucciones del repositorio y el contexto personalizado moldean el comportamiento del agente.
- Usage-based billing for individuals: GitHub AI Credits, el modelo de facturación contra el que presupuesta la puerta de preparación, un crédito es un centavo de dólar.
- Prosci, ADKAR change management model, la base de gestión del cambio para secuenciar la concienciación antes de la adopción.
- The four stages of competence, la progresión de competencia que las cinco etapas reflejan.
- BJ Fogg, Tiny Habits, la base de ventana de hábito para la meta de victoria rápida en 72 horas.
- MIT Sloan Management Review, AI in the Workplace, la evidencia del enfoque de aumentación frente a automatización.

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 RUTAS DE APRENDIZAJE

Rutas de alfabetización por clúster y herramienta Copilot: del autocomplete a la orquestación.

La alfabetización no es un único currículo. Los 24 roles del SDLC se dividen en siete clústeres, y cada clúster trabaja con una superficie diferente de GitHub Copilot. Un ingeniero vive en agent mode dentro del editor. Un ingeniero de plataforma genera pipelines. Un product manager construye un agente en Copilot Studio. Este capítulo mapea cada clúster a su herramienta primaria de Copilot, a su primera victoria rápida, y a las decisiones de modelo y contexto que hacen que la herramienta se pague sola.

Ingeniería: del autocomplete a la orquestación en Agent Mode

Ingeniería es el clúster más grande: seis roles, backend, frontend, full stack, junior, mobile y arquitecto. La superficie primaria es GitHub Copilot Agent Mode y Chat. La ruta de alfabetización pasa por tres niveles. El nivel uno es el autocomplete: aceptar o rechazar sugerencias inline. El nivel dos es el Chat: preguntar sobre el archivo abierto, pedir un refactor, generar una prueba. El nivel tres es el agent mode: entregar una tarea a Copilot y dejarlo planificar, editar varios archivos, ejecutar la terminal y abrir un pull request. La mayoría de los desarrolladores se estanca en el nivel uno porque nadie les mostró los dos siguientes.

La primera victoria rápida es deliberada y pequeña. Toma un ítem de baja complejidad del backlog y llévalo de la rama al pull request sin salir del editor. Ese es el criterio de salida de la Etapa 2 para el clúster, diseñado para caber en la ventana de hábito de 72 horas. La métrica de profundidad que importa aquí es acciones agénticas por semana: el conteo de sesiones de agent mode con múltiples pasos por desarrollador. Es el número que separa a un equipo que autocompleta de un equipo que orquesta.

DEFINICIÓN

La ruta de ingeniería en una línea: el nivel uno acepta sugerencias, el nivel dos pregunta al Chat, el nivel tres entrega una tarea al agent mode y revisa el pull request. La adopción se estanca cuando un equipo nunca sale del nivel uno.

Contexto de repositorio con .github/copilot-instructions.md y MCP

El agent mode es tan bueno como el contexto que puede ver. Dos mecanismos llevan ese contexto en GitHub Copilot. El primero es un archivo de instrucciones del repositorio, .github/copilot-instructions.md, versionado en el repo. Le dice a Copilot las convenciones que la base de código ya codifica: cómo se manejan las excepciones, cómo se organizan las pruebas, qué librerías están aprobadas, cuál es el comando de build. Cada desarrollador del repo hereda las mismas instrucciones, así que el agente deja de adivinar hechos que el equipo ya conoce.

El segundo mecanismo es el Model Context Protocol (MCP). El MCP permite que Copilot alcance herramientas y datos fuera del editor: un issue tracker, un schema de base de datos, un design system, una API interna. En vez de pegar contexto en el chat, el agente consulta la fuente. En el programa de onboarding, las extensiones MCP pertenecen a la Etapa 4, autonomía, porque suponen un equipo que ya ejecuta agent mode con fluidez y sabe qué contexto vale la pena conectar. Los dos mecanismos responden preguntas diferentes: el archivo de instrucciones dice cómo funciona este repositorio, y el MCP dice qué existe más allá de él. El archivo de instrucciones es de la semana uno; el MCP es del mes cuatro.

Elegir modelos en el selector de GitHub Copilot

GitHub Copilot no es un único modelo. El selector de modelos expone un menú que incluye Claude Fable 5, Opus 4.8, Sonnet 5 y Haiku 4.5, junto a Gemini 3.1 Pro y la familia GPT-5.x. La habilidad de alfabetización es saber cuál elegir para la tarea que tienes enfrente. Una completación inline rápida no necesita el modelo de razonamiento más capaz. Un refactor en varios archivos de un servicio desconocido

sí. Elegir Haiku 4.5 para un rename y Opus 4.8 para un cambio de arquitectura es la misma disciplina que elegir la herramienta correcta de la caja. El selector es un clic, así que el costo de una elección equivocada es pequeño, y el hábito de elegir es lo que la ruta construye.

La elección de modelo ahora también es una elección de presupuesto. Desde el 1 de junio de 2026, GitHub Copilot cobra por uso mediante GitHub AI Credits, donde un crédito equivale a un centavo de dólar. Modelos diferentes en el selector consumen cantidades diferentes de créditos por solicitud, así que un equipo que recurre al modelo más grande en cada edición trivial gasta más que un equipo que combina modelo y tarea. Por eso la métrica de profundidad y el selector de modelos se enseñan juntos: orquestación sin disciplina de modelo es orquestación cara.

> DEFINICIÓN
>
> Alfabetización de modelo en una línea: combina el modelo con la tarea, no la tarea con el modelo. Haiku 4.5 para ediciones baratas, Opus 4.8 o Claude Fable 5 para razonamiento difícil, y observa el medidor de AI Credits mientras aprendes.

Rutas para plataforma, datos, producto y seguridad

Los demás clústeres reciben cada uno una ruta construida sobre sus propios artefactos. Plataforma, es decir DevOps, cloud, DBA e infraestructura, usa Copilot para generar infraestructura como código en Bicep, Terraform y Bash, y su victoria rápida es un pipeline de GitHub Actions para un nuevo servicio, generado con la configuración del proyecto como contexto. Calidad y SRE usan agent mode con GitHub Actions; la victoria rápida de QA es una suite completa de pruebas unitarias para una clase sin cobertura, que el artículo cronometra en cerca de 15 minutos frente a 3 a 4 horas a mano.

Datos e IA, es decir ingeniero de datos, científico de datos y analista de BI, trabajan en Azure AI Foundry y Microsoft Fabric; la victoria rápida es un notebook de análisis exploratorio producido en cerca de 15 minutos. Producto y Diseño, es decir PM, UX

designer y analista de negocio, construyen agentes en Copilot Studio; la victoria rápida del PM es un agente que responde preguntas sobre el backlog desde una hoja de SharePoint. Seguridad y GRC combinan GitHub Advanced Security con Purview; la victoria rápida es el triaje de los tres principales CVEs con el GHAS Security Autofix. Liderazgo usa Microsoft 365 Copilot para resúmenes ejecutivos. Mismo programa, siete vocabularios, un único conjunto de dashboards. El vocabulario cambia por clúster, pero la métrica de profundidad y la disciplina de modelo de la ruta de ingeniería atraviesan los siete.

Referencias

Fuentes primarias para las superficies de Copilot, los archivos de personalización, el selector de modelos y el modelo de facturación detrás de estas rutas.

- GitHub Copilot Documentation, la referencia para agent mode, Chat y las superficies que aprende cada clúster.
- Customize AI in Visual Studio Code, cómo escribir el .github/copilot-instructions.md y personalizar Copilot en el editor.
- Usage-based billing for individuals: GitHub AI Credits, cómo funcionan los GitHub AI Credits y por qué la elección de modelo es una decisión de costo.
- GitHub Copilot is moving to usage-based billing, el cambio del 1 de junio de 2026 a facturación por uso detrás del medidor de AI Credits.
- DORA 2025 State of AI-assisted Software Development, evidencia de campo sobre prácticas de desarrollo de software asistido 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, VICTORIAS Y MÉTRICAS

Quick wins en setenta y dos horas, medidos cada quincena.

La confianza se gana o se pierde en las primeras setenta y dos horas. El presupuesto se mantiene o se recorta en la revisión trimestral. Este capítulo conecta ambos. Entrega un quick win concreto por clúster de personas, convierte cada victoria en una tarjeta de patrón reutilizable, define seis KPIs que van del alcance al riesgo, y gobierna la cuenta con GitHub AI Credits. La cadencia de medición es quincenal, porque un programa sin números pierde financiación.

Un quick win concreto por clúster

Los hábitos se forman rápido o no se forman. La ventana de hábito de BJ Fogg se cierra alrededor de las setenta y dos horas, así que el programa diseña un quick win concreto por clúster para caber dentro de ella. Ingeniería lleva un ítem de baja complejidad del backlog desde la rama hasta el pull request en agent mode sin salir del editor. Calidad genera una suite completa de pruebas unitarias para una clase sin cobertura, típicamente quince minutos de trabajo frente a tres a cuatro horas a mano. Plataforma genera un pipeline de GitHub Actions para un nuevo servicio, usando la configuración del proyecto como contexto. Producto construye un agente en Copilot Studio que responde preguntas sobre el backlog desde una hoja de SharePoint. Datos genera un notebook de análisis exploratorio en quince minutos. Seguridad realiza el triaje de los tres principales CVEs con el autofix de GitHub Advanced Security. Liderazgo produce un resumen ejecutivo de retrospectiva desde Teams o OneNote con Microsoft 365 Copilot.

Una regla sostiene todas las victorias: usar el backlog de la propia persona, nunca un tutorial genérico. Cada quick win entrega un artefacto real, un pull request, una suite de pruebas, un despliegue o un agente en funcionamiento. Ese es el sentido de la ventana de setenta y dos horas. Pasado ese punto, la curva de alfabetización se rompe y la persona deserta en silencio. Una victoria que la persona puede mostrar a un colega vale más que una demo que miró.

Tarjetas de patrón y una biblioteca de prompts compartida

Cada quick win se captura como una tarjeta de patrón de una página. La tarjeta es la semilla de la biblioteca de prompts y recetas del equipo. Registra seis cosas: la tarea, el modelo elegido en el selector de Copilot, el prompt que funcionó, los archivos de contexto que leyó, el artefacto que produjo y el tiempo que ahorró. Una tarjeta es lo bastante pequeña para escribirla en cinco minutos y lo bastante específica para que la siguiente persona repita el resultado.

La biblioteca vive en el repositorio, no en un wiki que nadie abre. Las convenciones compartidas van en .github/copilot-instructions.md, para que cada sesión de Copilot las herede sin copiar y pegar. Las recetas reutilizables se vuelven prompt files bajo control de versiones. Los sistemas internos llegan al agent mode mediante servidores Model Context Protocol, para que el agente consulte el contexto real en vez de adivinarlo.

De una tarjeta a una convención

Las tarjetas se acumulan, y las confiables merecen promoción. Cuando un patrón se prueba entre varias personas, muévelo al copilot-instructions.md o a un prompt file. El agent mode entonces lo lee en cada tarea, y el quick win se convierte en la forma predeterminada de trabajar del equipo.

Seis KPIs del alcance al riesgo

Un programa sin medición pierde financiación, así que seis KPIs cubren la señal completa de adopción, leídos cada quincena. Alcance es la tasa de usuarios activos, el porcentaje de usuarios con licencia activos al menos cinco días por semana, con un objetivo de ochenta por ciento en el día noventa. Profundidad es acciones agénticas por semana, un conteo de sesiones de agent mode con múltiples pasos por desarrollador, que separa el autocompletado de la orquestación real. Calidad es la tasa de aceptación y reversión, la tasa de aceptación de sugerencias junto con la tasa de rollback dentro de siete días, que expone la dependencia excesiva antes de que se vuelva un pasivo.

Throughput es el cuarteto DORA: lead time, frecuencia de despliegue, tasa de fallos de cambio y tiempo medio de restauración. Son señales rezagadas, pero son las que justifican el presupuesto. DevEx es una encuesta trimestral alineada con SPACE que incluye una pregunta sobre seguridad psicológica para escépticos e ingenieros junior. Riesgo es el conteo de hallazgos de GitHub Advanced Security corregidos automáticamente y fusionados por semana, más las recomendaciones de Defender for Cloud ejecutadas.

Señales principales primero, señales rezagadas para el CFO

Lee alcance y profundidad cada semana para atrapar un estancamiento mientras aún es barato corregirlo. El cuarteto DORA se mueve despacio, pero es lo que sobrevive una revisión de presupuesto. El otro lado está documentado: sin onboarding estructurado, la adopción activa cae al doce por ciento en sesenta días.

Gobernanza de costo con GitHub AI Credits

Desde el 1 de junio de 2026, GitHub Copilot cobra por uso mediante los GitHub AI Credits. Un crédito es un centavo de dólar, así que mil créditos son diez dólares. Cada solicitud consume créditos según el modelo y el modo. El selector de Copilot expone la elección directamente: Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro y la familia GPT-5.x. El enrutamiento ahora es una decisión de costo, no solo de calidad.

Gobernanza significa presupuestos y visibilidad, no un congelamiento. Define presupuestos de créditos por licencia y por equipo, y lee el panel de uso junto a los seis KPIs. Enruta los quick wins mecánicos a un modelo rápido y barato, y reserva los modelos premium para arquitectura y depuración difícil. Un quick win que quema créditos premium en una edición trivial es una lección para la tarjeta de patrón, no un resultado a celebrar. Como cada tarjeta ya registra el modelo y los créditos, la biblioteca funciona también como una guía de enrutamiento que se afina cada quincena.

DEFINICIÓN

Gobernanza de costo en una regla: empareja el modelo con la tarea. Un GitHub AI Credit es un centavo de dólar. Un arreglo de prueba de una línea no necesita Opus 4.8, y un refactor entre servicios no debería correr en Haiku 4.5. La tarjeta de patrón registra el modelo, los créditos gastados y el tiempo ahorrado, para que el enrutamiento sea más barato y mejor cada quincena.

Referencias

Fuentes primarias para el modelo de facturación, la superficie de personalización y las métricas de throughput.

- GitHub Copilot Documentation, la plataforma de referencia para el agent mode, el selector de modelos y los patrones de onboarding de este capítulo.
- GitHub Copilot is moving to usage-based billing, el cambio a facturación por consumo que entró en vigor el 1 de junio de 2026.
- Usage-based billing for individuals: GitHub AI Credits, cómo se miden los créditos por solicitud y por modelo, la base de la gobernanza de costo.
- Customize AI in Visual Studio Code, copilot-instructions.md y prompt files, la superficie de la biblioteca de prompts compartida.
- DORA 2025 State of AI-assisted Software Development, el cuarteto de throughput y la evidencia más amplia de adopción detrás de los 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 SOSTÉN LA ADOPCIÓN

Sostén la adopción con gestión del cambio y un Centro de Excelencia.

Un rollout que entrega licencias y un correo de bienvenida alcanza su pico alrededor de la segunda semana y cae a cerca de 12% de uso activo en el día sesenta. La adopción sobrevive cuando el programa trata la resistencia como una señal a redirigir, no como un obstáculo a remover, y cuando una estructura durable asume la cadencia después de que el programa formal termina. Este capítulo mapea la resistencia, la convierte en cuatro pasos, aplica Kotter a la secuencia y levanta un Centro de Excelencia.

El mapa de resistencia y la conversión en cuatro pasos

La resistencia a una herramienta nueva no es irracional. Es la respuesta de profesionales que aprendieron a ser competentes de una forma y ahora enfrentan la incertidumbre de aprender de otra. William Bridges, en Managing Transitions, describe la curva de cambio por la que las personas pasan a velocidades distintas: negación, resistencia, exploración y compromiso. El programa tiene que estar listo para encontrar a cada persona en cualquier punto de esa curva. Cada persona lleva una resistencia característica, un miedo de raíz y un disparador de conversión específico. El dev senior dice que la salida del agente bajará la calidad del código; el miedo es la identidad como experto técnico; el disparador es mostrar que el AI review es más estricto, no más laxo. El gerente de producto dice que esto es una herramienta de desarrollador; el disparador es una demo de Copilot Studio que resuelve un problema real de producto. El CISO dice que la empresa no controla lo que sale de ella; el disparador es una sesión de arquitectura de seguridad que muestra dónde quedan los datos y que GitHub Copilot Enterprise no entrena modelos con el código de la empresa.

La conversión sigue un patrón de cuatro pasos derivado de Cialdini sobre influencia y del framework Prosci ADKAR. Primero, reconoce la objeción sin descartarla: nómbrala como exactamente la falla que el programa fue construido para evitar. Segundo, ofrece evidencia específica, no una afirmación genérica: los desarrolladores formalmente formados producen una calidad de código mensurablemente mayor que los autodidactas, así que cita el número para la preocupación propia de esa persona. Tercero, dale control a la persona: deja que defina el criterio de éxito y la fecha de revisión, porque la autonomía reduce la resistencia más que la persuasión. Cuarto, crea la primera experiencia positiva con la persona resistente como protagonista, resolviendo en vivo un problema real suyo, para que la victoria le pertenezca a ella y no a la herramienta.

> DEFINICIÓN
>
> Cuatro pasos para convertir la resistencia: reconoce la objeción sin descartarla, ofrece evidencia específica en lugar de una afirmación genérica, dale a la persona control sobre el criterio de éxito y arma la primera experiencia positiva con la persona resistente como protagonista. La victoria tiene que pertenecer a la persona, no a la herramienta.

Kotter aplicado a la adopción agéntica

El modelo de ocho etapas de Kotter para liderar el cambio es el framework de cambio más documentado en contextos corporativos, y se ajusta con claridad a un programa de adopción agéntica. Crea urgencia presentando evidencia de mercado de que los equipos que no adoptan quedan atrás; el patrocinador ejecutivo es dueño de esta etapa. Forma la coalición conductora nombrando un Champion para cada cluster y un patrocinador C-level antes de que el programa comience; el coordinador del CoE es dueño de ella. Crea la visión definiendo, de forma concreta y por cluster, cómo es el trabajo en seis meses con agentes. Comunica la visión mediante showcases, un canal en Teams, demos en vivo y testimonios de quienes ya usan la herramienta.

La segunda mitad del modelo es donde los programas se estancan. Remueve obstáculos resolviendo bloqueos de licencia, acceso y seguridad antes de que se

vuelvan excusa. Genera victorias de corto plazo con los quick wins en las semanas tres a seis, difundidos ampliamente. Consolida los logros usando esas victorias para justificar expansión, nuevos casos de uso y más recursos. Ancla el cambio en la cultura poniendo KPIs de adopción de IA en los OKRs de los equipos y en los criterios de desempeño. La falla más común es saltar de la etapa uno directo a la etapa cinco, removiendo obstáculos técnicos antes de que existan urgencia, coalición y visión. El resultado es un programa técnicamente impecable que nadie usa.

El Agents Center of Excellence y la comunidad de práctica

El Centro de Excelencia es la estructura que sostiene el programa después de los primeros seis meses. No es un departamento nuevo. Es una red de roles distribuidos en equipos existentes, basada en el framework de Center of Excellence de Gartner. El coordinador del CoE es dueño de la agenda, las métricas y la comunicación, cerca de medio FTE para una organización de 100 a 500 desarrolladores. Un Champion por cluster es un especialista técnico que recibe formación avanzada, responde las dudas del equipo y documenta casos de uso, aproximadamente un Champion por cada veinte personas. Un patrocinador ejecutivo firma las decisiones de política y autoriza el budget, sin el cual el CoE se vuelve un comité sin poder. Un guardián de gobernanza, típicamente de seguridad o cumplimiento, confirma que los nuevos casos de uso de agentes siguen la política aprobada antes de llegar a producción.

La comunidad de práctica complementa al CoE formal. Siguiendo a Wenger, McDermott y Snyder, una comunidad efectiva necesita un dominio definido, un cuerpo de miembros y práctica compartida. En la operación, eso es un canal dedicado en Microsoft Teams, un show-and-tell quincenal de 45 minutos con dos casos y discusión abierta, y un repositorio interno de templates de prompt, guías de troubleshooting y casos documentados. La capa durable de convención vive en .github/copilot-instructions.md, donde cada equipo codifica cómo funciona realmente su base de código, para que el agent mode y las herramientas MCP la hereden. La gobernanza del conocimiento mantiene la biblioteca actual: cada caso de uso en producción se documenta con el problema, la persona, la herramienta, el prompt o la configuración, el modelo elegido en el picker de GitHub Copilot y el resultado medido, revisado cada trimestre.

El roadmap de seis meses y las acciones del lunes por la mañana

El programa corre un prelanzamiento más cuatro etapas a lo largo de veinticuatro semanas, y cada transición lleva un hito de go/no-go. Champions formados llega a 100% de los clusters antes del lanzamiento, o el lanzamiento espera. Licencias provisionadas llega a 100% de los participantes en el día cero, o escala a TI y al patrocinador. Los quick wins superan el 70% de los participantes hacia la semana seis, o el CoE corre sesiones extra de soporte por cluster. El uso activo semanal supera el 60% hacia el mes tres, o el equipo revisa la relevancia de los casos de uso de cada cluster. El ROI se documenta con al menos tres casos mensurables hacia el mes seis, o las métricas se reestructuran hacia casos más visibles. El consumo vía agent mode se factura como GitHub AI Credits, donde un crédito equivale a un centavo de dólar, así que el argumento de ROI reconcilia la productividad medida con una línea real de créditos.

Tres acciones empiezan el programa el lunes. Primero, publica la política de uso de IA por escrito; nada avanza mientras no exista, y su ausencia es el bloqueo que el CISO tiene a mano. Segundo, nombra dos Champions por equipo y reserva su formación avanzada de dos semanas. Tercero, programa la primera demo específica por cluster usando un artefacto real del backlog actual, porque los tutoriales genéricos fracasan en la primera semana. El onboarding no es un evento. Ejecútalo como una plataforma, con un mapa de resistencia, un patrón de conversión, una secuencia de cambio y un Centro de Excelencia que asume la cadencia mucho después de que el programa formal se retire.

> DEFINICIÓN
>
> El lunes por la mañana, tres acciones: publica la política de uso de IA por escrito, nombra dos Champions por equipo y reserva su formación avanzada de dos semanas, y programa la primera demo por cluster usando un artefacto real del backlog actual. El onboarding no es un evento. Ejecútalo como una plataforma.

Referencias

Fuentes primarias para los modelos de gestión del cambio, la estructura de la comunidad de práctica y la plataforma GitHub Copilot sobre la que se apoya este capítulo.

- Kotter, Leading Change, el modelo de ocho etapas aplicado a la secuencia de adopción.
- Cialdini, Influence: The Psychology of Persuasion, la base del patrón de conversión en cuatro pasos.
- Prosci ADKAR, el framework de gestión del cambio detrás de la conversión de personas.
- William Bridges, Managing Transitions, la curva de cambio que sigue el mapa de resistencia.
- Wenger, McDermott y Snyder, Cultivating Communities of Practice, el modelo para la comunidad de práctica.
- Gartner, Infrastructure and Operations, el framework de Center of Excellence.
- GitHub Copilot Documentation, la plataforma de referencia para agent mode, copilot-instructions.md y 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