GH-700 · CHEAT SHEET · CERTIFYDECK

GitHub Certified: Agentic AI Business Professional

Dominios, pesos oficiales y habilidades clave del temario de GH-700, para repasar antes del examen.

Nivel: Intermediate (Associate) Duración: 100 min Nota de corte: 700/1000 Fuente: guía de estudio oficial de Microsoft Learn

1. Use GitHub platform features

20–25%

Apply GitHub management best practices and policies

- Los presupuestos por usuario limitan el consumo de cada persona y siempre aplican un tope estricto; los de centro de costes, organización y empresa limitan los cargos medidos y solo bloquean si se activa la opción de detener el uso. Se evalúan juntos: bloquea el que tenga menos margen, y uno superior nunca amplía un presupuesto por usuario.
- Cada interacción consume tokens de entrada, de salida y en caché, con un precio según el modelo. Para reducirlos: elige el modelo según la tarea, da prompts claros con condición de parada, mantén el contexto reducido con conversaciones nuevas o /compact, activa solo las herramientas necesarias y no cambies modelo ni razonamiento a mitad de sesión.
- Las instrucciones personalizadas de la organización comunican orientación de idioma, seguridad o cumplimiento a todos los miembros, pero no garantizan el cumplimiento. Para imponer estándares, usa conjuntos de reglas y CODEOWNERS, a los que Copilot también está sujeto, y perfiles de agente personalizados de organización para una configuración coherente.
- El examen de secretos detecta credenciales expuestas y, ante una alerta, lo prioritario es rotarlas; la protección de inserciones las bloquea antes de que lleguen al repositorio. Copilot cloud agent solo recibe secretos de Agents, no los de Actions, y los que llevan el prefijo COPILOT_MCP_ quedan reservados a los servidores MCP.
- Las directivas de Copilot controlan funciones y modelos; si un usuario tiene licencias de varias organizaciones suele aplicarse la menos restrictiva. La exclusión de contenido hace que Copilot ignore archivos, y Copilot cloud agent ofrece trazabilidad: coautoría, registros de sesión y revisión humana obligatoria antes de combinar.

Plan and manage work

- Un proyecto puede partir de cero, de una plantilla integrada de GitHub o de una plantilla de la organización, y las recomendadas se sugieren primero. Una plantilla o una copia traslada las vistas, los campos personalizados, los borradores, los flujos configurados, salvo los de adición automática, y la información de insights, pero una copia no conserva los elementos, los colaboradores ni los vínculos con equipos y repositorios.
- Cada vista de un proyecto puede usar diseño de tabla, tablero o hoja de ruta, con sus filtros, agrupaciones y ordenación. En un tablero, las columnas proceden de un campo de selección única o de iteración, y sus límites solo avisan. Los flujos integrados actualizan el estado, añaden o archivan elementos, y los campos personalizados de fecha, número, texto, selección única o iteración enriquecen los datos.
- En un proyecto de organización, el rol base fija el acceso general y puede completarse con roles de lectura, escritura o administración para equipos y personas. El rol base no afecta a los propietarios ni a quien tiene acceso individual. La visibilidad pública o privada solo se aplica al proyecto: para ver un elemento de un repositorio privado hace falta acceso a ese repositorio.
- Las incidencias registran tareas, errores e ideas, y se pueden crear desde un repositorio, un comentario o una línea de código. Las subincidencias forman jerarquías, las dependencias marcan bloqueos y las etiquetas clasifican el trabajo dentro de cada repositorio: el rol de triaje las aplica y asigna, pero crearlas o editarlas exige acceso de escritura.
- GitHub ofrece varias vías de colaboración: las incidencias para tareas y errores con un responsable claro, las solicitudes de incorporación de cambios para proponer y comentar cambios, y GitHub Discussions para conversaciones abiertas, anuncios y encuestas. Las menciones avisan a personas y equipos, y Copilot ayuda a resumir hilos largos de incidencias y debates.

Track and capture work

- Una solicitud de incorporación de cambios propone fusionar una rama de comparación en una rama base y reúne conversación, comprobaciones y archivos modificados. Un borrador comparte trabajo en curso sin poder fusionarse ni pedir revisión a los propietarios de código. Sin permiso de escritura se parte de una bifurcación, y las plantillas y los resúmenes de Copilot ayudan a dar contexto.
- Las personas asignadas indican quién trabaja en una solicitud de incorporación de cambios, algo distinto de los revisores, a quienes se pide su opinión. Se puede asignar a quien tenga permisos de escritura, a quien haya comentado y a miembros de la organización con lectura, de una en una o en bloque. Los equipos pueden repartir revisiones automáticamente con round robin o equilibrio de carga.
- Vincular una solicitud de incorporación de cambios a una incidencia muestra que hay trabajo en curso y cierra la incidencia al fusionar en la rama predeterminada. Se vincula con palabras clave en la descripción, que se ignoran si la rama de destino es otra, o manualmente con permisos de escritura desde la barra lateral; solo la de la incidencia admite otro repositorio.
- Un proyecto se alimenta de incidencias y solicitudes de incorporación de cambios añadidas una a una, en bloque o con el flujo de trabajo de adición automática, que no incluye los elementos ya existentes. Los borradores solo existen en el proyecto hasta convertirlos en incidencias. Vincular un proyecto a un repositorio lo lista allí, siempre que ambos tengan el mismo propietario.
- La wiki de un repositorio aloja documentación extensa, mientras que el README resume el proyecto. Se edita en GitHub o en local, porque es un repositorio Git: cada cambio es una confirmación que se puede comparar y revertir. Hereda la visibilidad del repositorio y por defecto solo editan quienes tienen acceso de escritura. Si necesitas que los buscadores indexen el contenido, usa GitHub Pages.

2. Enhance productivity by using GitHub Copilot

20–25%

Chat with GitHub Copilot

- Cada modelo de Copilot tiene puntos fuertes distintos: los rápidos sirven para preguntas sencillas y repetitivas, los de razonamiento profundo para sopesar compromisos y analizar registros, y los multimodales para diagramas y capturas. Elige por la tarea, no por el nombre. Copilot Fusion o Auto delegan esa decisión, y la organización puede limitar qué modelos están disponibles.
- En GitHub Copilot CLI, /new inicia una sesión, /resume cambia a otra con su historial completo, /rename le pone nombre y /exit la cierra sin borrarla. Archivar saca una sesión de la lista activa sin eliminarla; las sesiones de Copilot cloud agent en GitHub Actions se archivan, pero no se eliminan. En VS Code puedes anclar una sesión en lo alto de la lista.
- Elige el agente según el trabajo: en el IDE, el desplegable de agentes cambia entre Ask, Plan y Agent; los agentes integrados cubren tareas habituales, como la revisión de código o una segunda opinión; y un agente personalizado define un rol reutilizable con sus instrucciones y herramientas. En GitHub se selecciona al asignar una incidencia o en el panel de agentes, y en GitHub Copilot CLI con /agent.
- En Copilot Chat en GitHub, al escribir @ se abre una lista de elementos adjuntables, como archivos, incidencias, solicitudes de incorporación de cambios y repositorios. En GitHub Copilot CLI, @ añade el contenido de un archivo y # incluye una incidencia o solicitud. En Visual Studio Code, @ invoca participantes de chat. Para consultar servicios externos durante una tarea se usan conectores o servidores MCP.
- Los comandos de barra diagonal evitan escribir peticiones largas, y los disponibles varían según el entorno. En el IDE, /explain explica el código, /fix propone correcciones, /tests genera pruebas y /doc añade documentación. En GitHub, /clear y /delete gestionan conversaciones. /chronicle aprovecha el historial de sesiones para informes de seguimiento, consejos, búsquedas y mejoras de las instrucciones personalizadas.

Write instructions and prompts that produce reliable outputs

- Pide de forma expresa lo que la respuesta debe contener: la clasificación por sentimiento de los comentarios, un resumen cohesionado con tendencias y conclusiones, el público destinatario o el tono. Pon primero los requisitos más importantes y exprésalos directamente. Las preferencias estables van en instrucciones personalizadas, aunque Copilot no siempre las aplica igual, de modo que revisa siempre el resultado.
- Indica con precisión sobre qué datos debe actuar Copilot: abre la incidencia para que se cargue como contexto, incluye su URL o pega el contenido en el chat. Evita referencias ambiguas como «esto». Los archivos de prompts son plantillas reutilizables que pueden definir variables de entrada, de modo que la misma tarea se ejecuta cada vez con datos distintos. Revisa bien los resúmenes de hilos muy largos.
- Para controlar la forma de la respuesta, indica el formato de manera directa, coloca primero los requisitos más importantes y añade un ejemplo de la salida esperada. Las preferencias duraderas pueden ir en instrucciones personalizadas, pero orientan al modelo sin garantizar el cumplimiento, y exigir siempre cierto nivel de detalle puede dar problemas en repositorios grandes.
- En GitHub Copilot CLI, /undo o /rewind abre un selector para volver al estado anterior a una petición, solo en la conversación o también en los archivos que cambió Copilot, sin sobrescribir tus ediciones posteriores. La acción no se puede deshacer. Los puntos de control de Visual Studio Code restauran archivos e historial, pero no revierten comandos ni cambios en servicios externos.

Create and manage custom agents

- Un perfil de agente personalizado concede acceso a la web con el alias web en su lista tools y define cómo usar esa información en el contenido Markdown bajo el frontmatter. La referencia indica que este alias no se aplica a Copilot in the cloud. Para un informe puntual con citas no hace falta crear un agente: basta el comando /research de GitHub Copilot CLI.
- Un perfil de agente se guarda como archivo: en .github/agents para un repositorio, en ~/.copilot/agents para uso personal en la CLI o en /agents del repositorio .github o .github-private para toda la organización. En GitHub aparece tras combinarse en la rama predeterminada. Desde un proyecto, se puede asignar una incidencia a Copilot y elegir ese agente.
- Las medidas de protección de un agente combinan una lista tools restringida, propiedades como disable-model-invocation y hooks como preToolUse, que ejecutan código y pueden denegar una herramienta, mientras que las instrucciones solo orientan al modelo. En Copilot cloud agent se suman el cortafuegos, una única rama de envío y la revisión humana antes de combinar.
- La descripción de la solicitud de incorporación de cambios indica qué agente personalizado trabajó, y cada commit enlaza con registros de sesión que muestran el razonamiento y las herramientas usadas. Para supervisar el uso en la organización, filtra el registro de auditoría por actor:Copilot; las peticiones locales no aparecen ahí. Con /chronicle revisas tus propias sesiones de la CLI.

3. Design and implement agentic workflows and automations

25–30%

Design agentic workflows and automations

- Los flujos de trabajo agénticos encajan con tareas repetitivas que exigen interpretar o investigar, como clasificar incidencias, revisar errores de CI o generar informes, y entregan resultados revisables. Para pasos reproducibles, como compilar o implementar, sigue siendo mejor GitHub Actions. Las automatizaciones de Copilot se configuran con desarrolladores sin confirmarse en Git.
- Divide el caso de uso según la naturaleza de cada tarea: pasos deterministas para recopilar datos y el agente para interpretar. Usa subincidencias si cada parte necesita debate y responsable, y un orquestador con trabajadores si cambian los permisos o hay varios repositorios. Cada fase debe dejar un artefacto concreto y un punto de control humano.
- Las decisiones de criterio van en las instrucciones, con definiciones, ejemplos y lógica condicional; las condiciones objetivas van en el frontmatter (if:, names:, skip-if-match:, on.roles:, stop-after:) y se evalúan antes de ejecutar el agente. Conviene aplicar automáticamente lo de bajo riesgo y dejar como sugerencia prioridades y fechas.
- Elige el patrón según qué inicia el trabajo y cómo fluye: IssueOps para incidencias nuevas, LabelOps para etiquetas, ChatOps para comandos, DispatchOps para ejecuciones manuales, MonitorOps para vigilar otros flujos, BatchOps para grandes volúmenes independientes, WorkQueueOps para trabajo pendiente y ResearchPlanAssignOps cuando el alcance no está claro.

Implement agentic workflows

- Un flujo de trabajo agéntico es un archivo Markdown con frontmatter YAML para la configuración y un cuerpo con instrucciones. Puede crearse con Copilot en la web, con un agente de codificación o a mano; gh aw compile genera el archivo .lock.yml. Para reutilizar uno, adapta con un agente o usa gh aw add y fija la versión si debes controlar los cambios.
- El frontmatter define cuándo se ejecuta el flujo (programaciones difusas, cron o eventos), qué puede leer el agente (permisos de solo lectura por defecto) y qué puede pedir que se escriba (salidas seguras con límites como allowed y max). Las salidas las ejecutan trabajos independientes, y staged: true permite previsualizarlas sin escribir nada.
- El código fuente .md y su archivo .lock.yml compilado viven juntos en .github/workflows y se confirman ambos; usa nombres en kebab-case. Los componentes compartidos suelen ir en .github/workflows/shared, los archivos de agente en .github/agents y las skills en .github/skills. Los archivos temporales del agente deben ir en /tmp/gh-aw/agent/ para conservarse.
- Escribe las instrucciones como si explicaras la tarea a alguien nuevo: concretas, con contexto, criterios, plantillas de salida y gestión de fallos, y con una llamada a noop cuando no haya nada que hacer. Evita la complejidad excesiva y el conocimiento supuesto. El cuerpo Markdown se carga en tiempo de ejecución, así que editarlo no requiere recompilar.
- La autónoma se apoya en varias piezas documentadas: el agente rubber duck revisa planes con otro modelo y sin editar nada; los evals plantean preguntas de sí o no sobre la salida; y la evaluación de resultados consulta el estado real del repositorio. Los bucles fiables limitan los reintentos, aceptan solo con pruebas y guardan un estado duradero.

Implement automations

- Una automatización se ejecuta según una programación, al crearse una incidencia, al abrirse o sincronizarse una solicitud de incorporación de cambios, o mediante la API con contexto adicional. Los filtros acotan cuándo se activa y, de forma predeterminada, se ignoran los eventos de usuarios sin acceso de escritura para reducir la inyección de prompts.
- El prompt describe la tarea que Copilot realiza en cada ejecución y es visible en las sesiones para quien tenga acceso de lectura, así que los secretos van en secretos del repositorio. El modelo y el nivel de razonamiento son opcionales: la selección automática adapta el modelo a la tarea y un razonamiento mayor ayuda en lo complejo, con más tiempo y consumo.
- Las herramientas determinan lo que Copilot puede hacer en una automatización y son la principal forma de limitar su alcance: concede solo las necesarias. Las herramientas de GitHub actúan en un único repositorio; las de servidores MCP, que configura un administrador, llegan a lo que permitan sus credenciales. Revisa ese acceso antes de seleccionarlas.

Integrate skills and MCP servers

- Los skills externos se instalan con gh skill install o moviendo el directorio descargado a una ubicación de proyecto o personal. GitHub no los verifica, así que conviene inspeccionarlos con gh skill preview y fijar versiones con --pin. En GitHub Agentic Workflows se declaran en el campo skills del frontmatter, fijados, en lugar de instalarlos sobre la marcha.
- Un skill personalizado es un subdirectorio propio con un archivo SKILL.md, cuyo frontmatter exige name y description, y un cuerpo Markdown con instrucciones. Puede incluir plantillas y scripts. Conviene usarlo para flujos repetibles que solo a veces hacen falta, y dejar shell fuera de allowed-tools salvo que se haya revisado todo su contenido.
- Copilot decide cuándo usar un skill a partir del prompt y de su descripción, e inyecta SKILL.md en el contexto del agente. En Copilot CLI se pide un skill concreto con una barra y su nombre, y /skills reload recoge los añadidos durante la sesión. La GitHub Copilot app ofrece además skills integrados, como /orchestrate, algunos de ejecución automática.
- Un servidor MCP da a un agente acceso a herramientas y datos externos, y complementa las instrucciones en lugar de sustituirlas. En GitHub Agentic Workflows se conecta en la sección mcp-servers, con allowed para limitar las herramientas; las escrituras pasan por salidas seguras. En Copilot cloud agent, la configuración de repositorio es compartida y conviene permitir solo herramientas de lectura, porque el agente las usa sin pedir aprobación.

4. Perform human-in-the-loop tasks for agentic workflows

20–25%

Design human review and validation practices

- Aunque un agente trabaje con autonomía, ciertas decisiones siguen siendo humanas: revisar cambios y pruebas antes de confirmar, aprobar y combinar la solicitud, y autorizar la ejecución de flujos de trabajo. Las tareas sensibles, críticas, ambiguas o muy complejas conviene que las lleve una persona. Quien pidió el trabajo no puede aportar la aprobación exigida, y manual-approval detiene un flujo de trabajo agéntico antes de ejecutarse.
- El modo interactivo conviene al explorar un problema o tomar decisiones frecuentes; el modo Plan, para revisar el enfoque antes de implementar; y el modo Autopilot, para tareas bien definidas que el agente puede llevar a cabo por pasos sin esperar. Autopilot controla la continuidad, no los permisos: si una acción no está permitida, el agente puede no terminar. Sigues siendo responsable de revisar el resultado.
- La forma en que un agente escala depende de la herramienta y del modo de permisos: aprobación manual, permisos asistidos o incondicional. Las reglas de denegación siempre prevalecen, incluso con allow-all, y un enlace preToolUse puede aprobar o denegar cada uso. En flujos de trabajo agénticos, la directiva protected-files decide si los cambios sensibles van a una revisión o a una incidencia para una persona.

Interact with agentic workflows

- Ante una solicitud de permiso en Copilot CLI puedes aprobar una vez, para la sesión o de forma persistente, o rechazar y orientar al agente con comentarios. En una solicitud de Copilot cloud agent, las revisiones se piden mencionando a @copilot, y conviene agruparlas en una revisión para que trabaje en todas a la vez. Solo responde a quien tiene acceso de escritura y únicamente en solicitudes abiertas.
- Mientras un agente trabaja puedes dirigirlo sin detenerlo: en Copilot cloud agent, con un mensaje de seguimiento bajo el registro de la sesión, y en Copilot CLI, con un mensaje normal o poniéndolo en cola para el siguiente turno. Las sesiones en un espacio aislado en la nube se pueden seguir desde otros clientes. Detener una sesión termina la ejecución y conserva las confirmaciones ya enviadas.
- Ante un error, empieza por las pruebas existentes: los registros de la sesión en Copilot cloud agent o la URL de la ejecución, que un agente puede analizar, en GitHub Agentic Workflows. Una sesión bloqueada se reintenta reasignando la incidencia. Agent Merge vigila una solicitud y corrige flujos de trabajo con errores o conflictos, y las herramientas MCP ausentes suelen deberse a los conjuntos de herramientas o a la lista allowed.
- Detener una sesión de Copilot cloud agent termina la ejecución de GitHub Actions y conserva las confirmaciones ya enviadas; detener no deshace lo hecho. En un flujo de trabajo agéntico, cancelar la ejecución no impide nuevos desencadenamientos, mientras que gh aw disable deshabilita y además cancela lo que esté en curso. En Copilot CLI, Ctrl+C cancela y vacía la cola de prompts.

Supervise and monitor agentic workflows

- Un flujo de trabajo agéntico gana autonomía por etapas: solo informes, modo staged (vista previa de lo que haría), evaluación en la sombra (prueba la escritura real fuera de producción) y escritura directa. Los requisitos se imponen en el frontmatter con safe-outputs, listas allowed, manual-approval y límites, no en las instrucciones. En cloud agent, los criterios de aceptación de la incidencia guían el resultado.
- Las solicitudes de Copilot merecen la misma revisión que cualquier otra: tu aprobación no cuenta si el repositorio exige aprobaciones, y conviene inspeccionar los cambios antes de permitir que se ejecuten los flujos. Cada confirmación enlaza a los registros de sesión para auditar el motivo. En los flujos agénticos, la detección de amenazas anota las salidas revisables, y los resultados se miden por el estado real del repositorio.
- Las sesiones de cloud agent se siguen desde el panel de agentes, con registro, progreso, uso de tokens y duración. Para los flujos agénticos, gh aw health da tasas de éxito, gh aw logs analiza varias ejecuciones y sus resultados con --outcomes, y gh aw audit desglosa o compara ejecuciones concretas. El patrón MonitorOps programa un flujo que resume la actividad y escala los fallos repetidos a incidencias.