GH-300 · CHEAT SHEET · CERTIFYDECK

GitHub Copilot

Objetivos, tareas y habilidades clave del temario de GH-300, para repasar antes del examen.

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

1. Use GitHub Copilot responsibly

15–20%

Understand responsible AI principles

- Copilot puede alucinar: generar código o texto plausible pero falso. Su calidad depende del volumen de datos de entrenamiento de cada lenguaje, puede reproducir sesgos, coincidir con código público y no ve problemas de arquitectura fuera del contexto inmediato. Los resúmenes de pull request no se actualizan solos y en un Space solo se procesa una parte del contenido.
- GitHub sigue los seis principios de IA responsable de Microsoft: equidad, fiabilidad y seguridad, privacidad y seguridad, inclusión, transparencia y responsabilidad. Las application cards aportan transparencia sobre usos y limitaciones. En decisiones trascendentes, como vivienda o empleo, las personas afectadas deben poder entender, recurrir y corregir los datos de la decisión.
- Daños posibles: contenido ofensivo, código inseguro, coincidencias con código público, exfiltración de datos e instrucciones ocultas, y la dependencia excesiva de la salida. Mitigaciones: filtrado de contenido, bloqueo de código público y análisis de propiedad intelectual; en cloud agent, firewall por defecto, push solo a ramas copilot/, secretos sólo del entorno copilot y respuesta sólo a usuarios con escritura.

Validate and operate AI tools

- El usuario asume los riesgos del código generado: errores, vulnerabilidades e infracciones de propiedad intelectual. El código puede parecer válido y ser incorrecto o inseguro, y los tests generados pueden no cubrir todos los casos. Conviene entender cada sugerencia, revisarla, apoyarse en linting, code scanning e IP scanning y usar Copilot code review como complemento de la revisión humana.
- Usar Copilot con responsabilidad implica lanzar Copilot CLI solo desde directorios de confianza, revisar cada comando antes de aprobarlo y conceder permisos mínimos: --deny-tool prevalece sobre --allow-all-tools. Aprobar una herramienta para toda la sesión permite cualquier uso de ella. Las tareas para cloud agent deben estar bien acotadas y, con BYOK, los datos quedan sujetos a las políticas del proveedor.

2. Use GitHub Copilot features

25–30%

Use GitHub Copilot in the IDE

- En VS Code basta con iniciar sesión con la cuenta de GitHub: las extensiones se instalan solas y, sin plan, la cuenta pasa a Copilot Free. En JetBrains se instala el plugin y se autoriza con un código de dispositivo. Las sugerencias se activan por lenguaje con github.copilot.enable en settings.json; en JetBrains, en Settings | Tools | GitHub Copilot.
- En el IDE Copilot ofrece completado inline (ghost text, aceptar con Tab), chat en la vista de Chat con modos Ask, Agent y Plan, e inline chat para ediciones puntuales en el editor. El modo agente resuelve tareas de varios pasos y usa herramientas MCP. Copilot CLI puede conectarse a VS Code con /ide para usar la selección del editor y revisar los diffs.
- Controla qué archivos ve Copilot con .copilotignore (admite negación con !) o la opción de contenido excluido de GitHub.com. Las reglas de repositorio se aplican a todos los usuarios con Copilot en ese repo; las de organización y de empresa configuran administradores de repositorio, propietarios de organización y de empresa con rutas tipo fnmatch; las reglas de empresa afectan a todos los usuarios con Copilot en la organización.

Use GitHub Copilot CLI

- Lleva Copilot al agente de IA al terminal: responde preguntas, escribe y depura código y trabaja con GitHub.com, por ejemplo creando issues o pull requests en tu nombre. Tiene una interfaz interactiva (con modo plan) y otra programática para scripts (-p, --allow-all-tools, --output-format json). Se instala con npm install -g @github/copilot, Homebrew (brew install copilot-cli), WinGet (winget install GitHub.Copilot) o el script de instalación para macOS y Linux. Requiere una suscripción activa a Copilot y Node.js 22 o superior como entorno de ejecución.
- En modo interactivo, @ añade archivos como contexto, / lanza comandos slash y Shift+Tab alterna entre los modos edit, plan y autopilot. Las sesiones se guardan: copilot --continue reanuda la más reciente, --resume abre un selector y /new empieza de cero. /compact resume el historial para liberar contexto y /context muestra el uso de tokens.
- Copilot CLI trabaja en un entorno aislado: un worktree por sesión en ~/.copilot/session-state/, checkpoints que se pueden restaurar con /rewind y un modelo de permisos por herramienta, ruta y URL. Los flags --allow-all-tools, --allow-tool, --deny-tool (que tiene prioridad) y --allow-all ajustan esos permisos. Con @ruta se añade un archivo como contexto, /add-dir concede acceso a otro directorio y /cwd muestra o cambia el directorio de trabajo.
- Agent mode itera hasta terminar, ejecuta comandos e itera hasta terminar; edit mode (Copilot Edits) limita los cambios al working set y permite aceptar o descartar cada edición. Los servidores MCP añaden herramientas para sistemas externos. Los subagentes trabajan en su propio contexto y devuelven solo el resultado, evitando ensuciar el principal. Confirma la confianza de la carpeta antes de usar agentes o MCP.
- Copilot Chat ofrece slash commands como /tests, /fix o /explain, chat participants como @terminal, @github y variables como #file. Sus respuestas pueden ser incorrectas o inseguras, así que hay que revisarlas y probarlas; los pulgares valoran cada respuesta. Los prompt files (.prompt.md) guardan prompts reutilizables que se lanzan con / y se definen a nivel de repo o de usuario.

Manage organization-wide settings and policies

- Las políticas de Copilot se gestionan en la empresa; si esta elige un valor concreto, la organización no puede cambiarlo, y si delega, decide cada organización. Con licencias de varias organizaciones suele ganar la opción menos restrictiva. Copilot code review se activa por política y se automatiza con ruleset, agent mode, MCP, CLI y hooks.
- El registro de auditoría recoge cambios en el plan de Copilot (directivas, configuración, asignaciones) y la actividad de agentes; se filtra con action:copilot o, para agentes, con actor:Copilot y campos como agent_session_id. Se exporta en JSON o CSV y se consulta por API REST (/orgs/{org}/audit-log).
- La API REST de gestión de usuarios de Copilot permite asignar y retirar puestos con last_activity_at y asignar o retirar puestos por usuario o por equipo. Al retirarlos, quedan pendientes de cancelación hasta el final del ciclo. Requiere la gestión de puestos en miembros seleccionados y una directiva de código público configurada.

3. Understand GitHub Copilot data and architecture

10–15%

Describe data handling and flow

- GitHub no entrena modelos con datos de Copilot Business ni Enterprise; en los planes individuales el usuario puede desactivar el uso de sus datos para entrenamiento. Con BYOK, los prompts van al proveedor elegido y quedan sujetos a sus políticas. Las sesiones locales de la CLI se guardan en el equipo y no se comparten por defecto; las del cloud agent son visibles para quien tenga acceso al repositorio.
- Antes de llamar al modelo, Copilot construye un prompt: para las sugerencias en línea combina el código antes y después del cursor (Fill-in-the-Middle) con fragmentos de los archivos abiertos. En VS Code, cada petición del chat o del agente reúne instrucciones del sistema, personalizaciones, el mensaje, el historial, contexto implícito, referencias explícitas y resultados de herramientas. Los archivos excluidos no alimentan el prompt.
- El prompt pasa por un proxy alojado en un tenant de Azure propiedad de GitHub que bloquea intentos de manipulación, y por filtros de contenido dañino. Tras generar la respuesta, se eliminan contenidos dañinos y el proxy revisa calidad, seguridad y, si se ha activado, coincidencias con código público; lo que no supera los filtros se trunca o descarta. Estos filtros se aplican sea cual sea el modelo elegido.

Understand lifecycle and limitations

- Una sugerencia de Copilot sigue un ciclo: se recopila el contexto, el prompt pasa por el proxy y los filtros, el modelo genera, el posprocesamiento revisa la respuesta y solo entonces se entrega. En el editor aparece diferenciada y solo se aplica si se acepta, total o parcialmente; las next edit suggestions predicen dónde y qué cambiar. Las referencias de código se comprueban al aceptar una sugerencia sin modificar.
- Copilot genera código de forma probabilística a partir de sus datos de entrenamiento y del contexto disponible, así que puede producir código que parece válido pero es incorrecto o inseguro, sesgos, coincidencias con código público o respuestas inventadas (alucinaciones). La calidad baja en lenguajes poco representados y en prompts que no están en inglés. Revisa, prueba y analiza siempre lo que acepte.

4. Apply prompt engineering and context crafting

10–15%

Craft effective prompts

- Un buen prompt empieza por el objetivo general y luego enumera los requisitos concretos. En Copilot Chat del IDE se combina con palabras clave: participantes con @ (como @terminal), slash commands con / (como /tests) y variables de chat con # (como #selection) para aportar contexto. Con las sugerencias en línea, el prompt es un comentario que describe el código esperado.
- Para las sugerencias en línea, Copilot combina el código antes y después del cursor (Fill-in-the-Middle) con fragmentos de las pestañas abiertas. En el chat de VS Code, el prompt reúne instrucciones del sistema, personalizaciones, el mensaje, el historial de ese chat, el contexto implícito (selección, archivo activo), las referencias con # y los resultados de herramientas; el agente busca más contexto con búsqueda semántica, texto o grep.
- En zero-shot, Copilot genera código sin ejemplos y se apoya solo en su entrenamiento: va bien para patrones comunes. En one-shot se da un ejemplo para que siga tus convenciones, y en few-shot varios, lo que ayuda con formatos propios, varios escenarios y casos límite. Sirven como ejemplos los datos de entrada, las salidas esperadas, implementaciones y pruebas unitarias.
- Divide las tareas complejas en pasos pequeños, evita términos ambiguos como "esto" y nombra la función o la respuesta a la que te refieres. Indica la biblioteca que quieres usar, sigue buenas prácticas de código (nombres descriptivos, estilo coherente), incluye pruebas o criterios de aceptación para verificar el resultado y refina con mensajes de seguimiento en lugar de empezar de cero.

Engineer prompts for performance

- La ingeniería de prompts consiste en redactar instrucciones claras para que Copilot genere código adecuado al proyecto. Se resume en las 4 S: Single (una sola tarea bien definida), Specific (instrucciones explícitas y detalladas), Short (conciso sin perder claridad) y Surround (nombres de archivo descriptivos y archivos relacionados abiertos). El role prompting pide a Copilot que actúe como un experto concreto.
- El prompt viaja por HTTPS con el contexto reunido, pasa por un proxy que bloquea intentos de manipularlo y por filtros de toxicidad, y el LLM genera la respuesta; después se revisan calidad, seguridad y, si está activado, código público, y lo que no pasa se trunca o descarta. Chat usa el historial de la conversación como contexto: abre un hilo nuevo para otra tarea, borra lo irrelevante y compacta las conversaciones largas.

5. Improve developer productivity with GitHub Copilot

10–15%

Enhance productivity and code quality

- Las inline suggestions completan código, código repetitivo y código a partir de comentarios mientras escribes; Copilot Chat genera bloques grandes que luego se refinan, refactoriza (nombres, condicionales, métodos largos, patrones de diseño) y documenta con comandos como /doc. Para documentación de varios archivos enlazados conviene el modo agente; para predecir la siguiente edición tras un cambio, next edit suggestions.
- Copilot ayuda a entender código desconocido sin salir del flujo: en GitHub.com explica repositorios, archivos y commits; Copilot CLI actúa como experto disponible si se le pide basarse en el código, citar archivos y no adivinar; los Copilot Spaces reúnen código y documentación para preguntas repetidas, y el inline chat del terminal sugiere comandos de shell. Verifica siempre las respuestas.
- Copilot genera datos ficticios a partir de un esquema o de unos registros de ejemplo, indicando cuántas filas y con qué patrones; con @mssql necesita conexión a la base de datos para conocer el esquema. Para modernizar código heredado conviene escribir antes pruebas del comportamiento actual, explicar archivos y flujo de datos, convertir por partes pequeñas, trabajar en ramas y revisar siempre las sugerencias.

Support testing and security

- Copilot genera pruebas unitarias con /tests o con prompts que acotan alcance y escenarios, y pruebas de integración que usan mocks para simular dependencias externas y verificar las llamadas. Conviene reutilizar el framework del proyecto, dar como contexto pruebas existentes (pestañas abiertas o #file), planificar con Plan los trabajos de varios archivos y ampliar casos con ghost text.
- Para que Copilot cubra algo más que el camino feliz hay que pedir de forma explícita casos límite, excepciones y validación de datos, y probar en cada límite y justo fuera de él. Los valores esperados salen de los requisitos, no de la implementación ni de la propia función probada. Ante un fallo, se busca la causa y nunca se debilitan ni se omiten aserciones solo para que pasen.
- Copilot Chat puede detectar vulnerabilidades comunes (XSS, inyección SQL) y proponer correcciones, y sugerir optimizaciones como mejores estructuras de datos, transacciones cortas o un orden de acceso coherente. Copilot Autofix propone correcciones para alertas de code scanning. Copilot es una capa adicional: no sustituye a code scanning ni a las revisiones, y sus sugerencias se revisan y prueban siempre.

6. Configure privacy, content exclusions, and safeguards

10–15%

Manage privacy settings and exclusions

- Content exclusion impide que Copilot use determinados archivos: sin inline suggestions en ellos, sin influir en respuestas ni en code review. La configuran administradores de repositorio, propietarios de organización (solo para sus licencias) y de empresa (para todos), con rutas fnmatch; los cambios quedan en el audit log. En VS Code, github.copilot.enable activa o desactiva Copilot por lenguaje.
- GitHub no reclama la propiedad de las entradas ni de las salidas de Copilot, y si una sugerencia puede tener propietario depende de la legislación aplicable. La salida se ofrece tal cual: puede ser incorrecta, insegura o parecerse a código público, por lo que el usuario debe revisarla, probarla y comprobar si necesita una licencia de terceros, igual que con cualquier código de origen desconocido.

Apply safeguards and troubleshoot

- La directiva de sugerencias que coinciden con código público admite Allow o Block. Con Block, Copilot compara cada sugerencia y su contexto con el código público de GitHub y oculta las coincidencias; con Allow, muestra referencias con la URL y la licencia. Los usuarios de Pro la fijan en su configuración; con licencia de organización se hereda y, si hay conflicto, gana la más restrictiva.
- Si faltan sugerencias en algunos archivos, el icono tachado de la barra de estado indica una content exclusion. Los cambios tardan en llegar al IDE: en VS Code se recarga la ventana. Las reglas de organización solo afectan a quien recibe la licencia de esa organización, no cubren enlaces simbólicos ni sistemas de archivos remotos, y los cambios se auditan con copilot.content_exclusion_changed.