

# Microsoft Certified: Agentic DevOps Engineer Expert

Dominios, pesos oficiales y habilidades clave del temario de AZ-401, para repasar antes del examen.

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

# 1. Design and implement processes and communications

15-20%

## Design traceability and flow of work

- Diseña el flujo de trabajo enlazando fases: planifica en Azure Boards o GitHub Projects (columnas, columnas divididas, límites WIP y definición de terminado), compila con vinculación automática de elementos a compilaciones, despliega con entornos que registran historial y elementos desplegados, y opera creando elementos de trabajo desde Application Insights con su contexto.
- Los ciclos de comentarios en Azure DevOps se basan en suscripciones: personales, de equipo, de proyecto o globales, entregadas por correo o mediante enlaces de servicio a servicios externos como Teams, Service Bus o Datadog. La entrega por rol limita los avisos a los implicados y Skip initiator evita avisar a quien hizo el cambio. En GitHub Actions, las notificaciones de flujos programados siguen a su creador o a quien los rehabilita.
- Elige el proceso según el equipo: Basic (épicas, incidencias, tareas), Agile (historias de usuario), Scrum (elementos del backlog con estados Approved y Committed) o CMMI (control de cambios, riesgos y revisiones). Organiza el trabajo con rutas de área por equipo o componente, que también restringen el acceso, y rutas de iteración por sprint, y usa planes de entrega para ver varios equipos y sus dependencias en un calendario.
- La trazabilidad une elementos de trabajo con ramas, commits, solicitudes de incorporación de cambios, compilaciones, versiones y pruebas. En Azure Repos, la directiva Check for linked work items exige vínculos; con GitHub, AB# en commits o descripciones vincula y, con fixes, cambia el estado al combinar en la rama predeterminada. Los conjuntos de pruebas basados en requisitos y los informes de trazabilidad miden la calidad de cada requisito.

## Design and implement success metrics for DevOps

- Convierte cada KPI en una métrica medible: tiempo de entrega (de la creación a Completed) para compromisos con clientes, tiempo de ciclo y diagrama de flujo acumulado para eficiencia y cuellos de botella, informes de tasa de éxito, duración y fallos de pruebas para la calidad de la entrega, y la información general de seguridad de GitHub (Coverage, Remediation) para la seguridad, con el registro de auditoría para datos de cumplimiento.
- Con Azure DevOps MCP Server en modo agente, un asistente traduce preguntas en lenguaje natural a consultas WIQL y llamadas a herramientas sobre elementos, compilaciones y pruebas. Analytics expone OData (LeadTimeDays y CycleTimeDays solo para elementos completados) para Power BI. Azure Copilot genera KQL de ejemplo para registros de Azure Monitor, y las métricas de uso de Copilot miden la adopción de la IA.

## Configure collaboration and communication

- Documenta el proyecto con wikis de Azure DevOps (wiki de proyecto o código publicado como wiki, sincronizado con la rama), README en .github, raíz o docs, notas de versión generadas y configuradas en .github/release.yml, un archivo LICENSE y CODEOWNERS, donde gana el último patrón coincidente. Mermaid dibuja diagramas de proceso, y Copilot redacta resúmenes de solicitudes de incorporación de cambios partiendo de una descripción vacía.
- Define requisitos para agentes con instrucciones en .github/copilot-instructions.md, archivos .instructions.md con applyTo y AGENTS.md; habilidades en .github/skills con SKILL.md; y agentes personalizados cuyo campo tools limita herramientas. Copilot cloud agent solo inserta en su rama copilot/ y no aprueba sus solicitudes. Los flujos agénticos programados se ejecutan con permisos de lectura y escriben solo mediante safe-outputs.
- Azure DevOps MCP Server puede ser remoto (<https://mcp.dev.azure.com/{organización}>), HTTP y Entra ID), filtrado con X-MCP-Toolsets, X-MCP-Tools o X-MCP-Readonly, o local (stdio, PAT o Entra ID). El servidor MCP de GitHub remoto usa OAuth o PAT. Para Copilot cloud agent, el JSON del repositorio solo admite herramientas, usa secretos COPILOT\_MCP\_ y el servidor de GitHub integrado lee solo el repositorio actual.
- Los webhooks de GitHub y los enlaces de servicio de Azure DevOps envían eventos (push, revisiones, elementos de trabajo) a sistemas externos sin sondeo. Los servidores MCP dan herramientas a Copilot: en Copilot cloud agent se configuran en el repositorio y leen secretos de Agents con prefijo COPILOT\_MCP\_. Las skills son carpetas con SKILL.md (name y description) en .github/skills.
- El servidor MCP de Azure DevOps da a los asistentes de IA en modo agente acceso a elementos de trabajo, solicitudes de cambios y canalizaciones. La versión remota (<https://mcp.dev.azure.com/{org}>) requiere Microsoft Entra; la local usa npx y stdio. Los encabezados X-MCP-Toolsets y X-MCP-Readonly limitan sus herramientas, y desde Azure Boards se puede iniciar Copilot en repositorios de GitHub.

## 2. Design and implement a source control strategy

10–15%

## Design and implement branching strategies for Azure DevOps and GitHub

- Microsoft implementa ramas de características breves combinadas en main mediante solicitudes de cambios y una rama main siempre estable (enfoque basado en tronco). Las ramas de versión se crean desde main, no se combinan de vuelta y reciben correcciones con cherry-pick. GitHub flow elimina la rama tras combinada, y git workflow permite que Azure Repos imita los tipos de combinación (básica, squash, rebase con avance rápido, rebase con commit de trabajo), en varias ramas a la vez sobre un mismo repositorio. Los cambios aprobados, revisión de propietarios de código, Copilot cloud agent solo inserta en su rama copilot, respeta las protecciones de rama y las comprobaciones de código de aprobación en consolas y resolución de conflictos en Azure DevOps.
  - Configuración de ramas obligatorias, si un conjunto de reglas lo bloquea, se le añade como actor con omisión. No puede aprobar ni combinar configuraciones y ramas obligatorias en GitHub.
  - En Azure Repos, los permisos se asignan por Al Repositories, en cada repositorio a una persona, una acción o un grupo de usuarios que aprueba. Bypass policies y when completing pull requests y Bypass policies when pushing separan omitir directivas al completar o al insertar. En GitHub, los roles Read, Triage, Write, Maintain y Admin escalan el acceso, solo Admin gestiona reglas de rama.
  - La trazabilidad que codifica y trabaja en GitHub, las menciones @# vinculan commits y solicitudes con Azure y a la vez a cadena, solo Admin gestiona reglas de configuración trigger, tags y en GitHub Actions on push, tags filtra por nombre de etiqueta, los conjuntos de reglas también protegen etiquetas.
  - Los hooks de GitHub son scripts ejecutables en el repositorio en la ruta de core hooks path que se disparan en puntos de entrada de etiqueta, los conjuntos de reglas también protegen etiquetas.
  - GitHub Enterprise impone mediante la extensión gh add-on y tokens en GHA PAT READ y PAT, migra desde Azure DevOps Cloud (no desde Server) el código Git, las solicitudes de incorporación y pre-push, por lo que las directivas obligatorias deben aplicarse también en el servidor.
- pero no los objetos LFS. Conviene hacer una migración de prueba, detener el trabajo durante la real y, después, revisar el registro, ajustar la visibilidad y reclamar los maniqués.

### 3. Design and implement build and release pipelines

35–40%

## Design and implement a package management strategy

- [illegible]

## 4. Design and implement a security and compliance plan

15–20%

### Design and implement authentication and authorization methods

- Para cargas hospedadas en Azure se prefiere una identidad administrada, cuyas credenciales gestiona Azure; la asignada por el sistema vive y muere con su recurso y la asignada por el usuario se comparte entre recursos. Fuera de Azure se usa una entidad de servicio con federación o certificado mejor que con secreto. En Azure DevOps la identidad se agrega explícitamente a la organización y se autoriza con niveles de acceso y grupos.
- GITHUB\_TOKEN es un token de instalación limitado al repositorio del flujo de trabajo; con la clave permissions se aplica el mínimo privilegio y lo no especificado queda en none. Para otros repositorios o integraciones duraderas se usa una GitHub App, que no depende de un usuario, con actions/create-github-app-token. Los tokens de acceso personal sirven para scripts breves, y se prefieren los fine-grained a los clásicos.
- La autenticación sin secretos usa OIDC y federación de identidades de carga de trabajo: en GitHub Actions, el trabajo necesita id-token: write y azure/login intercambia el token por uno de Azure de corta duración, con la credencial federada en una aplicación de Entra o una identidad administrada asignada por el usuario. En Azure Pipelines, las conexiones de servicio con federación eliminan los secretos almacenados.
- Las conexiones de servicio autentican Azure Pipelines frente a servicios externos; para Azure Resource Manager se recomienda la federación de identidades de carga de trabajo, que elimina los secretos, y cada canalización debe autorizarse por separado. Los PAT solo se usan cuando no hay autenticación de Microsoft Entra: con ámbito de organización, ámbitos mínimos, vigencia corta y guardados en un almacén de secretos.
- En Azure DevOps, los niveles de acceso deciden qué características del portal se ven y los grupos de seguridad conceden Allow o Deny; un Deny de otro grupo en el mismo ámbito suele prevalecer y una asignación explícita en un nodo hijo sustituye la heredada. En GitHub se eligen roles de repositorio de Read a Admin, y el bloque permissions limita el GITHUB\_TOKEN.

### Design and implement a strategy for managing sensitive information in automation

- En Azure Pipelines, la tarea AzureKeyVault@2 descarga secretos como variables mediante una conexión de servicio con permisos Get y List, y un grupo de variables puede vincularse a un almacén para leer los valores en tiempo de ejecución. En GitHub Actions se inicia sesión con azure/login y OIDC, se lee el secreto con la CLI y se enmascara con ::add-mask::.
- Key Vault guarda secretos, claves y certificados; con Azure RBAC se asigna el rol mínimo, como Key Vault Secrets User para leer secretos. La eliminación temporal permite recuperar objetos y la protección de purga impide borrarlos antes de la retención. Las directivas de rotación crean nuevas versiones de clave, que los servicios toman con el URI sin versión, y los certificados de CA integradas se renuevan solos.
- Los archivos seguros de Azure Pipelines guardan certificados, perfiles y claves cifrados; son recursos protegidos con aprobaciones y permisos de canalización, y DownloadSecureFile@1 los descarga al inicio de la fase y los borra al terminar el trabajo. En GitHub Actions, los secretos pueden ser de repositorio, de organización o de entorno, y estos últimos solo se liberan tras sus reglas de protección.
- El enmascaramiento de secretos es un esfuerzo razonable, no una garantía: no se ocultan subcadenas ni partes de datos estructurados, así que hay que usar un secreto por valor, asignarlos a variables de entorno en vez de pasarlos por la línea de comandos y registrar con ::add-mask:: los valores derivados. Las compilaciones de bifurcaciones no deben recibir secretos ni ejecutar código no confiable con privilegios.

### Automate security and compliance scanning

- Una estrategia de análisis combina herramientas por momento y tipo de contenido: revisión de dependencias en cada solicitud para bloquear paquetes vulnerables, CodeQL para el código, la extensión Microsoft Security DevOps para plantillas IaC, contenedores y binarios, y Defender for Cloud para ver en una consola los hallazgos de Azure DevOps, GitHub y GitLab.
- En Azure Repos, GitHub Advanced Security for Azure DevOps activa la protección de inserción y el análisis de secretos del historial; el análisis de dependencias y CodeQL son tareas de canalización, y las comprobaciones de estado AdvancedSecurity bloquean solicitudes vulnerables. En GitHub, la configuración predeterminada de CodeQL requiere Actions y la omisión delegada limita quién salta la protección de inserción.
- Defender for Cloud conecta Azure DevOps, GitHub y GitLab desde Configuración del entorno con conectores nativos. GitHub exige instalar la aplicación de seguridad de DevOps como propietario de la organización; Azure DevOps exige Project Collection Administrator, nivel Basic y OAuth de terceros activado. Las operaciones se ejecutan con la identidad que autoriza, así que conviene una cuenta de servicio dedicada.

## 5. Implement an instrumentation strategy

15–20%

### Configure monitoring for Azure DevOps and GitHub platforms

- En Azure DevOps, el widget Lead Time mide desde la creación del elemento hasta su estado Completado y Cycle Time desde que entra en curso; ambos muestran media móvil y desviación estándar por equipo. El Cumulative Flow Diagram enseña el flujo de trabajo por columna y revela abultamientos. Para la recuperación se sigue el tiempo medio de restauración del servicio tras un incidente.
- Azure DevOps puede transmitir su registro de auditoría a Azure Monitor Logs, donde se consulta la tabla AzureDevOpsAuditing y se crean alertas; también admite Splunk y Event Grid. GitHub Enterprise Cloud transmite eventos de auditoría y de Git a Azure Event Hubs o Blob Storage desde que se habilita. Application Insights recibe anotaciones de versión de Azure Pipelines y crea elementos de trabajo en GitHub o Azure DevOps.
- GitHub ofrece información a varios niveles: Pulse y el tráfico por repositorio, métricas de uso y de rendimiento de GitHub Actions por organización, y gráficos en la información de Projects. Los gráficos actuales agrupan elementos por campo y los históricos ponen Time en el eje X; el eje Y puede sumar o promediar un campo numérico. Un rol personalizado permite ver las métricas de Actions sin ser propietario.
- En Azure Pipelines, las alertas se configuran con suscripciones de notificaciones de equipo o proyecto, con la aplicación Azure Pipelines para Slack o Teams, que permite filtrar por compilaciones fallidas y aprobar fases, y con la comprobación Query Azure Monitor Alerts en entornos. En GitHub Actions, cada usuario recibe avisos de las ejecuciones que desencadena, y workflow\_run con una condición sobre conclusion crea alertas propias.

### Monitor Azure workloads by using AI agents

- Azure Copilot Observability Agent permite chatear con registros y métricas e investigar incidentes sin aprovisionar nada. Para las operaciones autónomas, que correlacionan alertas y crean incidencias de Azure Monitor, se crea un recurso Observability Agent con identidad administrada, ámbito en Application Insights e instrucciones personalizadas. El agente nunca reinicia recursos ni cambia la configuración: deciden las personas.
- Para crear Azure SRE Agent hace falta Propietario, o Colaborador más User Access Administrator, y el creador recibe SRE Agent Administrator. Después se conectan repositorios de código, orígenes de registros como Application Insights y, opcionalmente, grupos de recursos con nivel Reader o Privileged para su identidad administrada. Las directivas de acceso a herramientas solo permiten denegar en el ámbito global.
- Azure SRE Agent investiga con hipótesis que valida o descarta, propone mitigaciones y, en modo Review, espera la aprobación de un SRE Agent Administrator para operaciones de infraestructura de Azure. Nunca ejecuta eliminaciones, respeta los bloqueos ReadOnly y registra cada comando az como evento AgentAzCliExecution en Application Insights. También puede crear elementos de trabajo, notificar o lanzar flujos de trabajo.

### Analyze metrics from instrumentation by using AI agents

- Las métricas de host de una máquina virtual (CPU, red y uso de disco) están disponibles sin agente; la memoria del invitado y el espacio libre en disco requieren el agente de Azure Monitor y una DCR, por ejemplo con VM insights, que escribe en InsightsMetrics. Azure SRE Agent consulta métricas y registros con su identidad administrada sin conectores y analiza series temporales; Observability Agent correlaciona mejor si se supervisan las VM.
- Application Insights ofrece las vistas Performance, Failures y Search y la vista End-to-end transaction details; Live metrics muestra datos con un segundo de latencia sin conservarlos. Azure Copilot Observability Agent ejecuta investigaciones profundas desde una alerta que correlaciona señales de aplicación, infraestructura y plataforma; sus resultados son temporales y se conservan guardándolos como issue de Azure Monitor.
- Application Insights agrupa la telemetría de una transacción distribuida con operation\_Id (operación raíz) y enlaza cada elemento con su padre inmediato mediante operation\_ParentId. Application map identifica los componentes por el cloud role name, y End-to-end transaction details muestra un diagrama de Gantt con solicitudes, dependencias y excepciones. El muestreo en la distribución evita trazas rotas; el de ingesta las hace más probables.
- La actividad de los agentes de Copilot en GitHub se revisa en el registro de auditoría de la empresa con actor:Copilot; cada evento incluye user y agent\_session\_id. Para un historial largo, se transmite a un destino como Azure Event Hubs. Los datos de clientes locales se recogen con OpenTelemetry, y la clase identity atribuye la actividad sin capturar prompts ni respuestas.
- Desde la página Logs, Observability Agent responde en lenguaje natural sobre registros y métricas, y Azure Copilot genera KQL de ejemplo. En KQL, que distingue mayúsculas y minúsculas, where filtra (== exacto, =~ sin distinguir mayúsculas), summarize con bin agrupa por intervalos de tiempo, project y extend controlan columnas y top ordena y limita; search es más lento que filtrar.