AZ-400 · CHEAT SHEET · CERTIFYDECK

Microsoft Certified: DevOps Engineer Expert

Dominios, pesos oficiales y habilidades clave del temario de AZ-400 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

10–15%

Design and implement traceability and flow of work

- GitHub flow es un flujo ligero basado en ramas: creas una rama descriptiva, confirmas cambios aislados, abres una PR (en borrador si quieres opinión temprana), atiendes la revisión, combinas y eliminas la rama. En Azure Repos, Microsoft recomienda ramas de característica, PR con validación de compilación y revisores automáticos en main, y GitHub, las plantillas y formularios de incidencias (.github/ISSUE_TEMPLATE) estructuran los informes, config.yml puede ocultar la incidencia en blanco, y las subincidencias crean jerarquías. Las notificaciones llegan a la bandeja web, móvil o correo según si participas o vigilas el repositorio. Azure DevOps entrega avisos por correo o
- Enlaces de servicio con sus tablas, tableros personales, de equipo, de proyecto, incidencias y PR, las plantillas de
- Azure Boards se conecta a GitHub con aplicación y vínculos, confirmaciones, PR e incidencias mediante AB#ID, Commit, Pull Request o Integrated in build. En Azure Pipelines puedes vincular resultados de pruebas a requisitos, crear errores desde pruebas fallidas y localizar el cambio de código que inició un fallo. El widget Requirements quality resume la calidad por requisito, y AB# o Fixes #n conectan el código de GitHub con el trabajo.

Design and implement appropriate metrics and queries for DevOps

- El tiempo de entrega va de la creación del elemento a su estado Completed y el de ciclo, desde que entra en In Progress hasta Completed; los widgets Lead Time y Cycle Time muestran la media móvil y la desviación estándar, y el CFD revela cuellos de botella y exceso de WIP. Para la entrega, Cloud Adoption Framework propone Lead Time para cambio, Azure Boards ofrece Velocidad (trabajo completado por sprint, medido al sprint en que se
- Para Planificar, Azure Boards ofrece Velocidad (trabajo completado por sprint, medido al sprint en que se termina), la herramienta de previsión, que usa esa velocidad para repartir el trabajo pendiente en sprints futuros, y widgets Burndown y Burnup que pueden abarcar varios equipos y mostrar el aumento de alcance. Las consultas
- Las métricas de @Current en la consulta de elementos de trabajo y los gráficos históricos forman la base
- Las métricas de pruebas de Azure DevOps salen de los resultados que publica la canalización (p. ej., PublishTestResults@2). Test Analytics y el informe Test failures muestran las pruebas que más fallan y la tendencia de aprobación; la administración de pruebas inestables detecta pruebas flaky y puede excluirlas del resumen. Los
- Las métricas de seguridad (Advanced) de las herramientas de análisis en GitHub, la vista general de seguridad ofrece vistas Overview, Risk, Coverage y por herramienta, y las alertas de código y de Dependabot se filtran con expresiones como is:open o scope:development. GitHub Advanced Security for Azure DevOps ofrece Risk, Coverage y Alerts para Azure Repos, y Defender for Cloud unifica Azure DevOps, GitHub y GitLab.

Configure collaboration and communication

- La wiki de Azure DevOps admite Markdown y diagramas Mermaid con el contenedor ::: mermaid o un bloque delimitado mermaid; los diagramas de flujo se escriben con graph, no con flowchart. Etiquetas como [[_TOC_]] y [[_TOSP_]] generan índices, y Publicar código como wiki asigna una rama y carpeta del repositorio, ordenadas con archivos .order. GitHub interpreta Mermaid en bloques delimitados y en las
- En GitHub, las solicitudes de incorporación de cambios combinadas, los colaboradores y un vínculo al registro de cambios; .github/release.yml las organiza por etiquetas y excluye etiquetas o autores. La documentación de API en ASP.NET Core se genera como documento OpenAPI con Swashbuckle.AspNetCore.Swagger y se muestra con Swagger UI.
- Con líneas cortas para git blame. Separa asunto y cuerpo del mensaje de confirmación. Usa git log --oneline --graph --decorate --all para ver el historial: git log con rangos como v2.3.0..HEAD, --no-merges, --grep y --pretty=format crea registros de cambios; git shortlog -sn resume confirmaciones por autor; git describe genera versiones a partir de la etiqueta anotada más cercana. La API de GitHub genera notas de la versión a partir de etiquetas con el punto de conexión HTTPS. En GitHub conviene usar un secreto y validar la firma X-Hub-Signature-256, responder con 2XX enseguida y usar X-GitHub-Delivery contra reenvíos. Azure DevOps envía eventos con suscripciones Web Hooks de enlaces de servicio, y Azure Pipelines admite recursos webhooks entrantes. Incoming Webhook predefinido.
- Los Web Hooks se envían a un punto de conexión HTTPS. Entre GitHub y Azure Boards se conecta a repositorios de GitHub, preferentemente con la aplicación Azure Boards. Las menciones AB# en mensajes de confirmación y descripciones de solicitudes vinculan elementos de trabajo, y palabras como Fixes cambian su estado solo al combinar en la rama predeterminada. También hay distintivos de estado, ramas creadas desde elementos de trabajo y menciones ! de solicitudes.

2. Design and implement a source control strategy

10–15%

Design and implement branching strategies for the source code

- Microsoft recomienda una estrategia sencilla: ramas de características cortas desde main, combinación mediante solicitudes de incorporación de cambios y una rama main siempre estable. Las ramas de versión se crean desde main, no se vuelven a combinar y reciben correcciones con cherry-pick. El desarrollo basado en tronco usa marcas de características en lugar de ramas largas.
- El flujo de solicitudes de incorporación de cambios se controla con directivas de rama en Azure Repos (revisores mínimos, comentarios resueltos, comprobaciones de estado y resolución del combinador antes de aceptar). Estas directivas se aplican en el servidor y bloquean el envío directo a la rama protegida.
- Las restricciones de combinación definen cómo llegan los cambios a una rama. En Azure Repos, la directiva Limit merge types permite solo squash, rebase&fast-forward o combinación básica, y no se pueden invalidar de forma individual. En GitHub, la protección de rama equivalente es Require linear history, que obliga a usar squash o rebase.
- Las ramificaciones de características aíslan el trabajo en curso y permiten publicar versiones con marcas de características. Las ramificaciones de versión fijan el código de una versión publicada para poder aplicar correcciones urgentes sin arrastrar el trabajo posterior.

Configure and manage repositories

- Azure Repos ofrece repositorios Git y TFVC; Git es el predeterminado y el recomendado. Un monorepo conviene cuando el código está muy acoplado y se versiona como un todo; varios repositorios, cuando cada componente tiene su propio ciclo de vida. Scalar optimiza Git para repositorios muy grandes.
- En Azure Repos los permisos se asignan en API repositorios o por repositorio, que hereda de nivel de proyecto salvo que se desactive la herencia. Deny suele prevalecer sobre Allow. Hay dos permisos para omitir directivas: al completar solicitudes y bypass en el envío. Evítalos: rompen la trazabilidad y deberían ser excepcionales.
- Para trabajar en varios repos a la vez, git clone --recurse-submodules o git submodule update --init --recursive. Para actualizar el puntero a la última versión: git submodule update --remote --merge. Olvidar --recursive deja carpetas vacías y errores de compilación difíciles de diagnosticar.
- Los submódulos fijan una versión concreta del repositorio externo, lo que garantiza builds reproducibles pero exige actualizar el puntero a mano. Subtree copia el código dentro del repositorio, sin punteros externos, más fácil para los consumidores pero con historia más difícil de separar.
- Para limpiar el repositorio local, git fsck --lost-found lista confirmaciones inalcanzables y git checkout <sha> -- archivo recupera un archivo eliminado. git revert deshace cambios compartidos sin reescribir el historial. En Azure Repos las ramas eliminadas se restauran desde el portal, sin sus directivas.

3. Design and implement build and release pipelines

50–55%

Design and implement a package management strategy

- GitHub Packages aloja paquetes npm, RubyGems, Maven, Gradle, NuGet e imágenes de contenedor junto al código, con permisos integrados y GITHUB_TOKEN en los flujos de trabajo. Azure Artifacts ofrece fuentes con NuGet, npm, Python, Maven, Cargo y Universal Packages, vistas y orígenes ascendentes. Los artefactos de canalización sirven para pasar resultados de compilación, no para gestionar paquetes.
- Una fuente de Azure Artifacts tiene ámbito de proyecto u organización y las vistas @Local, @Prerelease y @Release controlan la promoción. Los orígenes ascendentes unen fuentes externas (nuget.org, npmjs.com) y protegen contra ataques de sustitución de dependencias.
- Semver: Major cambios incompatibles, Minor funcionalidad nueva compatible, Patch correcciones. Usa Major.Minor.Revision, el número principal indica cambios importantes, el secundario funciones nuevas y la revisión correcciones; las etiquetas como -beta plantean versiones preliminares. Las versiones de paquetes y las versiones de compilación son conceptos distintos.
- En Azure Pipelines el nombre de la ejecución se define con name y tokens como $(Date:yyyyMMdd) y $(Rev:r), que se reinicia al cambiar el resto del número, contiene un contador por prefijo para versiones semánticas. Los enteros autoincrementados son más simples cuando no hace falta significado en el número.
- Para dependencias transitivas, package-lock.json, yarn.lock y packages.lock.json fijan versiones exactas. Dependabot y Renovate abren solicitudes de incorporación de cambios para actualizarlos. Los rangos flotantes encajan con bibliotecas sin versiones publicadas.

Design and implement a testing strategy for pipelines

- Una taxonomía DevOps clasifica las pruebas por dependencias: L0/L1 unitarias, L2 funcionales con SQL o archivos, L3 de integración con servicios reales. Las unitarias son rápidas y herméticas; las de integración, lentas y frágiles.
- VSTest y la tarea .NET Core publican resultados en la pestaña Tests por sí mismas para JUnit, NUnit, xUnit, CTest o Visual Test. Publish Test Results@2 publica informes JUnit, NUnit, VSTest y xUnit en Azure Pipelines.
- Publish Code Coverage Results@2 publica informes Cobertura, JaCoCo y otros formatos en la pestaña Code Coverage, en Linux, .NET usa Coverlet con XPlat Code Coverage. En Azure Repos, azure-pipelines-coverage.yml define autoStop en el YAML de configuración.

Design and implement pipelines

- Azure Pipelines combina definición de GitHub, Azure Repos y otros; ofrece editor clásico, fases en YAML, entornos y agentes. Los agentes hospedados por Microsoft y los ejecutores hospedados por GitHub dan una máquina limpia por trabajo con herramientas preinstaladas; los privados conservan estado entre ejecuciones.
- Azure Pipelines accede a GitHub con la aplicación de GitHub (Identidad de Azure Pipelines y GitHub Checks), OAuth o PAT. Para publicación en repositorios públicos. Agentes y ejecutores inician la conexión saliente por HTTPS. Managed DevOps Pools y los ejecutores efímeros facilitan el escalado automático.
- En Azure Pipelines, trigger define la CI por ramas, rutas y etiquetas; trigger: none la desactiva y batch agrupa ejecuciones. pr define la validación de solicitudes de incorporación de cambios, con filtros de ramas y rutas. Comentario para omitir CI y autoCancel controlan la concurrencia.
- En Azure Pipelines, las expresiones ${{ }} se evalúan al analizar plantillas y pueden usar parámetros, mientras que $[ ] se evalúa en tiempo de ejecución sin acceso a parámetros. Las plantillas de inclusión insertan contenido y las compilaciones de bifurcaciones.

Design and implement deployments

- La exposición progresiva limita el radio de impacto: canary envía la versión a un grupo pequeño y lo amplía, rolling actualiza instancias por lotes, y blue-green cambia el tráfico de golpe entre dos entornos idénticos y cambia el tráfico tras probar. Las marcas de características separan implementación y exposición, y las variantes permiten pruebas A/B por percentiles.
- En App Service se implementa en una ranura de ensayo y se intercambia con producción: el inter-cambio prepara las instancias y se revierte con otro intercambio. La preparación se ajusta con WEBSITE_SWAP_WARMUP_PING_PATH y SITEUSES_STATUSES, y el intercambio automático no existe en Linux ni en contenedores. Los conjuntos de escalado en modo gradual actualizan por lotes con sondeo de estado, y rolling con maxParallel mantiene VM en servicio.
- En el flujo de ramas de versión, la revisión se hace primero en main mediante solicitud y luego se lleva con cherry-pick a la rama de versión, para que no reaparezca. Azure Repos crea con Cherry-pick una nueva solicitud con directivas, y git cherry-pick -x anota el commit de origen. La guía de implementación segura permite que un cambio pase de desarrollo a producción sin sorpresas.
- Para la configuración del sistema operativo como código en Azure y servidores con Arc se recomienda Azure Machine Configuration, sustituye a Automanage Machine Configuration. Las tareas de Azure Pipelines para App Configuration importan pares clave-valor desde archivos (con Dry Run, etiqueta, prefijo y profundidad) y bloquean cambios con candados. Una estrategia de IaC guarda las plantillas en Git, revisa cada cambio con solicitudes de incorporación de cambios y despliega a través de entornos con aprobaciones y puertas.
- Azure Machine Configuration aplica configuración del sistema operativo como código con los modos Audit, AuditAndSet y ApplyAndMonitor, y corrige la deriva cada 15 minutos. Para confirmar cambios sin aplicarlos, Test-AzConfiguration. Las propiedades omitidas vuelven a su valor predeterminado.
- Azure Deployment Environments permite que los desarrolladores creen entornos a petición a partir de definiciones de entorno (plantilla IaC y environment.yaml) publicadas en catálogos de GitHub o Azure DevOps. Los tipos de implementación limitan qué suscripciones y regiones pueden usar cada entorno.

Maintain pipelines

- En Azure Pipelines, los informes de tasa de éxito, duración y errores de pruebas muestran las tareas que más fallan, necesitan el rol Deployment Environments User. Las pruebas inestables las detecta por repetición o mediante API y puede excluirlas del resumen. En GitHub, las métricas de rendimiento de Actions muestran tiempos de ejecución, de cola y tasas de error.
- Para optimizar canalizaciones: la caché (Cache@2 en Azure Pipelines; actions/cache en GitHub) reutiliza dependencias con claves y rutas de restauración en cascada. Los artefactos pasan datos entre trabajos, no para caché. Fetch depth limita la extracción de Git, timeoutInMinutes cancela trabajos bloqueados; y los agentes en espera de Managed DevOps Pools evitan esperas de aprovisionamiento.
- En Azure Pipelines la retención de ejecuciones se configura solo a nivel de proyecto; una ejecución concreta se conserva indefinidamente en la organización. maxParallel limita las matrices, y el bloqueo exclusivo con lockBehavior serializa las implementaciones.
- Solo las canalizaciones creadas con el diseñador de compilación clásico se exportan a YAML desde la vista de definición; las de versión clásicas se migran a tarea. Tras pegar el YAML exportado hay que redefinir las variables del editor clásico y revisar los cron, que en YAML van en UTC. Los grupos de implementación pasan a entornos, las aprobaciones y puertas a comprobaciones, y los grupos de tareas a plantillas.

4. Develop a security and compliance plan

10–15%

Design and implement authentication and authorization methods

- Las identidades administradas evitan gestionar credenciales. La asignada por el sistema vive y muere con su recurso y solo ese recurso la usa; la asignada por el usuario es un recurso independiente, se comparte y admite asignaciones de roles previas al despliegue. Fuera de Azure, como en GitHub Actions, se usa una aplicación de Entra o una identidad asignada por el usuario con credencial federada.
- El GITHUB_TOKEN es un token de instalación creado para cada trabajo, limitado al repositorio y ajustable con permissions; lo que hace no desencadena nuevas ejecuciones salvo excepciones. Las GitHub Apps son la opción para integraciones duraderas: permisos específicos, tokens de corta duración y sin depender de un usuario. Los tokens personales sirven para scripts breves; se prefieren los específicos.
- Las conexiones de servicio tienen los roles Reader, User, Creator y Administrator, y cada canalización debe autorizarse por separado en lugar de abrirlas a todas. Para Azure se recomienda la federación de identidades de carga de trabajo. Los PAT de Azure DevOps son un último recurso: ámbito de organización, ámbitos mínimos y vida corta; nunca superan los permisos de su propietario y regenerarlos invalida el valor anterior.
- Los roles de repositorio de GitHub van de menos a más acceso: Read, Triage, Write, Maintain y Admin. Triage gestiona incidencias sin escribir, Maintain administra el repositorio sin acciones sensibles y Admin controla protección de ramas, webhooks y borrado. En la organización hay roles predefinidos como security manager, billing manager, moderador o GitHub App manager, y los propietarios tienen acceso total.
- En Azure DevOps los permisos se asignan a grupos de seguridad en ámbito de organización, proyecto u objeto, con estados Allow, Deny y Not set. En el mismo ámbito un Deny suele prevalecer sobre un Allow, y una asignación explícita en un nodo hijo sustituye a la heredada. Project Collection Administrators gestiona procesos, directivas y extensiones; Project Administrators, equipos, rutas y repositorios.

Design and implement a strategy for managing sensitive information in automation

- En Azure Pipelines, la tarea AzureKeyVault@2 descarga secretos como variables; con RunAsPreJob quedan disponibles para todo el trabajo, y la identidad necesita Get y List sobre secretos. Un grupo de variables vinculado a Key Vault solo asigna nombres de secretos (no claves ni certificados) y obtiene el valor más reciente en cada ejecución. En GitHub Actions se inicia sesión con OIDC y se enmascara el valor con ::add-mask::.
- La federación de identidades de carga de trabajo evita guardar secretos: en Azure Pipelines se recomienda para las conexiones de servicio de Azure Resource Manager, y las creadas automáticamente y usadas por un solo proyecto se pueden convertir. En GitHub Actions, el trabajo necesita id-token: write y azure/login intercambia el token OIDC por un token de Azure de corta duración, válido solo durante el trabajo.
- Los archivos seguros de Azure Pipelines (certificados, almacenes de claves, perfiles de aprovisionamiento, claves SSH) se guardan cifrados, son un recurso protegido con permisos de canalización, aprobaciones y comprobaciones, y no se pueden editar tras subirlos. DownloadSecureFile@1 expone la ruta en $(nombre.secureFilePath) y borra el archivo al terminar el trabajo. En GitHub Actions se usan secretos en Base64 o archivos cifrados con gpg.
- Para evitar fugas, nunca imprimas secretos ni los pases por la línea de comandos: asígnalos a variables de entorno y no uses datos estructurados como secreto, porque la ocultación depende de coincidencias exactas. Los secretos no llegan a compilaciones de bifurcaciones ni, en GitHub, a flujos reutilizables de forma automática. Ancla las acciones a un SHA completo y usa variables de entorno intermedias frente a la inyección.

Automate security and compliance scanning

- Una estrategia de análisis combina análisis de secretos (todo el historial más protección de inserción), análisis de código con CodeQL, análisis de dependencias y control de licencias. En GitHub, las alertas de Dependabot detectan dependencias vulnerables en la rama predeterminada y la acción de revisión de dependencias bloquea solicitudes con paquetes vulnerables o licencias no permitidas (allow-licenses o deny-licenses).
- La seguridad de DevOps de Defender for Cloud conecta Azure DevOps, GitHub y GitLab mediante conectores y centraliza resultados de código, secretos, dependencias e IaC. La extensión y la acción Microsoft Security DevOps ejecutan analizadores como Bandit, Checkov, Template Analyzer o Trivy; en Azure DevOps los SARIF se publican en el artefacto CodeAnalysisLogs. Las anotaciones en solicitudes requieren validación de compilación.
- En GitHub, Advanced Security ofrece análisis de secretos con protección de inserción, análisis de código con CodeQL y alertas de Dependabot. En Azure Repos, GitHub Advanced Security for Azure DevOps usa tareas como AdvancedSecurity-Codeql-Init@1, AdvancedSecurity-Codeql-Analyze@1 y AdvancedSecurity-Dependency-Scanning@1, y las comprobaciones de estado NewHighAndCritical o AllHighAndCritical bloquean solicitudes.
- La integración de GitHub Advanced Security con Defender for Cloud usa el conector de GitHub para asignar repositorios a cargas de trabajo en ejecución y añadir factores de riesgo (exposición a Internet, datos sensibles, recursos críticos, movimiento lateral) a las alertas de GHAS. Las campañas se crean en la organización de GitHub con el filtro Runtime Risks, y las incidencias generadas desde Defender se sincronizan en ambos sentidos.
- Para analizar imágenes en la canalización se usan Microsoft Security DevOps con Trivy (break: true hace fallar la compilación) o la CLI de Defender for Cloud (defender scan image); Defender for Containers evalúa además las imágenes del registro y las que están en ejecución. CodeQL debe ejecutarse en el mismo contenedor en que se compila el código (container: image: en el trabajo) y no admite imágenes Alpine con musl.

5. Implement an instrumentation strategy

5–10%

Configure monitoring for a DevOps environment

- Azure Pipelines se integra con Azure Monitor mediante la comprobación o puerta Query Azure Monitor alerts, que solo deja avanzar si no hay reglas de alertas activadas. Application Insights crea elementos de trabajo en Azure DevOps o incidencias en GitHub con plantillas que son libros de Azure Monitor. La auditoría de Azure DevOps puede enviarse a Azure Monitor Logs para consultarla con KQL en la tabla AzureDevOpsAuditing.
- Para las máquinas virtuales se instala el agente de Azure Monitor y se asocian reglas de recopilación de datos; las métricas del host se recogen sin agente. En AKS se combinan Prometheus administrado, la recopilación de registros de contenedores y una configuración de diagnóstico para el plano de control. Application Insights usa una cadena de conexión y OpenTelemetry, y Storage insights y Network insights funcionan sin configuración.
- Las estadísticas de GitHub Projects permiten crear gráficos actuales y gráficos históricos (eje X en Time), filtrarlos con la misma sintaxis que las vistas y sumar campos numéricos en el eje Y; no siguen elementos archivados. Los repositorios ofrecen Pulse y el gráfico de tráfico, y la organización ofrece métricas de uso (minutos) y de rendimiento (colas, errores) de Actions.
- En GitHub, la configuración de notificaciones de Actions permite recibir correos o avisos web y limitarlos a los flujos con error; para automatizar alertas, workflow_run con types: [completed] y una condición sobre github.event.workflow_run.conclusion reacciona a otro flujo. En Azure DevOps, las suscripciones de equipo usan plantillas como Build fails o Run stage waiting for approval y se entregan por correo o enlace de servicio.

Analyze metrics from instrumentation

- Las métricas del host de una máquina virtual, como Percentage CPU, Available Memory Bytes, OS Disk Queue Depth o Network In Total, se recopilan sin agente y se analizan en el explorador de métricas con agregaciones, filtros y divisiones. La pestaña Performance de VM insights usa datos del invitado y muestra gráficos Top N de CPU, memoria, disco y red, incluido el tráfico entre redes virtuales.
- Application Insights ofrece herramientas de uso: Users, Sessions y Events, Funnels para la conversión en una secuencia fija, User Flows para explorar rutas, Cohorts, retención e Impact Analysis. Los usuarios se cuentan por identificadores anónimos salvo que se establezca un identificador autenticado. Las vistas Performance y Failures enumeran operaciones con recuento, duración media y usuarios afectados.
- Application Insights correlaciona la telemetría distribuida con operation_Id, compartido por todos los elementos de una operación, y operation_ParentId, que apunta al padre inmediato; con W3C Trace Context, traceparent lleva esos identificadores. El mapa de aplicación identifica componentes por el nombre del rol en la nube, y los detalles de la transacción muestran un diagrama de Gantt de solicitudes y dependencias.
- Una consulta KQL empieza por una tabla y encadena operadores con |: where filtra (== distingue mayúsculas y =~ no), project elige columnas, extend añade columnas, summarize agrega con funciones como count() o avg() y bin() agrupa por intervalos de tiempo. take devuelve registros arbitrarios, top ordena en el servidor y render debe ir al final.