Engineer

Bicep

el lenguaje propio de Azure para declarar infraestructura

Qué es

Bicep es un lenguaje de Microsoft para describir la infraestructura Azure como texto: qué recursos existen, cómo se llaman, cómo se configura cada uno y cómo se conectan entre sí. Un archivo Bicep no enumera pasos. Declara el resultado final, y Azure Resource Manager, la capa por la que pasan las solicitudes de gestión de Azure, calcula qué crear, cambiar o conservar para alcanzarlo. Quien pregunta qué es Bicep suele querer saber si sustituye a recorrer el portal Azure haciendo clic. Lo hace, y ahí reside la mayor parte de su valor.

Por debajo, Bicep es una forma más cómoda de escribir algo que Azure ya entendía. Resource Manager acepta desde hace tiempo plantillas de despliegue escritas en JSON, que son completas pero difíciles de leer y aún más de revisar. Durante un despliegue, la herramienta de línea de comandos de Bicep convierte un archivo `.bicep` en una de esas plantillas JSON, y Resource Manager la despliega exactamente como desplegaría una plantilla JSON escrita a mano. Microsoft afirma que Bicep admite todos los tipos de recursos y versiones API en cuanto los introduce un proveedor de recursos, incluidas las versiones preliminares, por lo que un servicio nuevo se puede utilizar sin esperar a que alguien actualice un plugin. Las plantillas JSON existentes pueden descompilarse a Bicep, lo que convierte trasladar un entorno antiguo en un trabajo mecánico en vez de una reescritura.

Un archivo pequeño muestra la estructura. Primero aparecen los parámetros y después los recursos, cada uno con un nombre simbólico al que puede referirse el resto del archivo:

```
param env string
param location string = resourceGroup().location

resource store 'Microsoft.Storage/storageAccounts@2023-05-01' = {
```

```
  name: 'st${env}${uniqueString(resourceGroup().id)}'
  location: location
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'
}
```

Cuando otro recurso menciona `store.id`, Bicep deduce que debe esperar a la cuenta de almacenamiento, de modo que nadie escribe el orden a mano, y Resource Manager despliega en paralelo todo lo que no depende de otra cosa. Un archivo puede llamar a otro como módulo, lo que permite describir una sola vez una base de datos con sus reglas de red y reutilizarla donde haga falta. Los valores que difieren según el entorno residen en un archivo `.bicepparam` separado para cada entorno. Microsoft declara expresamente que Bicep es gratuito, de código abierto y que no pretende ser un lenguaje de programación de propósito general: tiene bucles, condiciones y funciones, pero únicamente al servicio de describir recursos.

¿Cómo se compara Bicep con las alternativas? Frente a ARM JSON, es el mismo motor con un código fuente mucho más corto. Frente a Terraform, las principales diferencias son el alcance y el estado: Terraform cubre muchas nubes y mantiene un archivo de estado propio, mientras que el flujo de Azure Resource Manager de Bicep no mantiene un archivo de estado propio, porque Azure almacena el estado del despliegue. Las extensiones de Bicep también admiten recursos ajenos a Resource Manager, incluidos Kubernetes y Microsoft Graph. Frente a Pulumi, que describe infraestructura en lenguajes de propósito general como TypeScript o Go, Bicep sacrifica flexibilidad a cambio de un lenguaje más pequeño que se lee con rapidez durante la revisión.

Utilice el concepto en una decisión

Aplique «¿Qué es Bicep?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Toda la configuración escrita en archivos para poder recrearla con exactitud y evitar que exista solo en la memoria de una persona.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 1.000 reglas RBAC para servidores Linux — 1,000 reglas de acceso escritas para cerrar una brecha de seguridad

1.000 reglas RBAC para servidores Linux Póngase en contacto →

Por qué importa

La infraestructura construida a mano no tiene historia. Un ajuste cambiado en el portal no deja detrás el motivo, nadie lo revisa antes de hacerlo y reconstruir el entorno después de un mal día depende de que alguien recuerde cada clic. En cuanto la infraestructura se convierte en un conjunto de archivos en el repositorio, obtiene lo que ya tiene el código de aplicación: una pull request, alguien que la revise, un historial y una forma de volver atrás.

La segunda ganancia es la uniformidad. Desarrollo, staging y producción, construidos desde los mismos archivos con parámetros distintos, solo difieren donde los parámetros dicen que deben hacerlo. Los despliegues Bicep son idempotentes, lo que, en palabras de Microsoft, significa que desplegar el mismo archivo muchas veces produce «los mismos tipos de recursos en el mismo estado». Por tanto, un despliegue puede repetirse con seguridad, y un entorno nuevo se obtiene con el mismo comando dirigido a otro grupo de recursos.

La tercera ganancia es la visibilidad antes del cambio. La operación what-if compara un archivo con lo que está en ejecución y enumera cada recurso que crearía, modificaría o eliminaría antes de tocar nada. Combinada con una pull request, esa vista previa convierte la creencia de que solo se cambia una regla del cortafuegos en algo que quien revisa puede comprobar.

Los costes merecen nombrarse. Los recursos admitidos dependen del flujo de despliegue y sus extensiones, por lo que el DNS de otro proveedor o los recursos de una segunda nube pueden seguir necesitando una herramienta aparte. En el flujo de Resource Manager, Bicep describe recursos de Azure; los paquetes y archivos dentro de una máquina virtual corresponden a la gestión de configuración, por ejemplo con Ansible. Y los valores predeterminados de Resource Manager para la eliminación y el redespliegue sorprenden a quienes esperan que una herramienta de estado deseado elimine lo que ya no menciona el archivo, como se explica en el cuarto apartado.

Cómo se aplica en la práctica

Varios de los proyectos Azure de este sitio se apoyan en Bicep, y tres muestran lo que aporta a un equipo a lo largo del tiempo.

En una plataforma, toda la infraestructura Azure está escrita en Bicep. Cada entorno sale de las mismas plantillas: api y www como Azure Container Apps, un PostgreSQL Flexible Server, Redis para caché, Front Door con una política WAF delante y la red debajo —VNet, NSG y DNS privado—, con Log Analytics conectado para diagnóstico e imágenes obtenidas del Azure Container Registry del proyecto. Como todo está parametrizado, poner en marcha un entorno nuevo o cambiar uno existente es una pull request. La página describe el resultado como entornos que se volvieron reproducibles y revisables, y una desviación que dejó de ser un misterio porque el código es la fuente de verdad.

Las mismas plantillas sostienen la ruta de publicación. Las versiones pasan de desarrollo a staging y de ahí a producción, y Bicep mantiene idéntica la infraestructura de destino de un entorno al siguiente, por lo que nunca se promociona una compilación a un lugar sutilmente distinto. La configuración que difiere por entorno se mantiene separada de los secretos, lo que permite que un único artefacto ya construido recorra todas las etapas sin volver a construirlo.

Bicep también contiene parte del plan de recuperación. La retención de copias de seguridad se configura en Bicep junto a la base de datos que protege, de modo que restaurar a un instante determinado es una propiedad de la plantilla en vez de un ajuste que alguien seleccionó una vez con un clic. El entorno que rodea la base de datos también es código, algo importante porque restaurar datos en un entorno que debe reconstruirse de memoria apenas puede llamarse recuperación. La página enumera después lo que sigue pendiente: la georredundancia está desactivada y el objetivo de tiempo de recuperación es una propuesta, no una medición. Una plantilla muestra qué está configurado; solo un simulacro muestra cuánto tarda una restauración.

Dónde surgen los problemas

La mayoría de los problemas con Bicep nace de esperar que se comporte como una herramienta que gobierna todo lo que hay en un grupo de recursos. De forma predeterminada solo gobierna lo que menciona el archivo, y únicamente de la forma que el archivo indica.

Un ajuste cambiado en el portal desaparece en el siguiente despliegue. Redesplegar un recurso reaplica todas sus propiedades, y cualquier propiedad que omita la plantilla se restablece a su valor predeterminado. Una corrección urgente hecha a mano durante un incidente se deshace en la siguiente publicación rutinaria. Trate el archivo como el estado final completo de cada

recurso, escriba en él todos los valores distintos del predeterminado y haga los cambios urgentes mediante una pull request. Los ajustes de denegación de una pila de despliegue pueden bloquear por completo las ediciones ajenas al despliegue.

Un recurso retirado de la plantilla sigue funcionando y costando dinero. El modo de despliegue predeterminado es incremental: los recursos del grupo que no menciona el archivo se conservan. Eliminar una base de datos del código la deja en Azure. La respuesta de Microsoft son las pilas de despliegue. Una pila registra los recursos que gestiona, y su ajuste `actionOnUnmanage` decide si uno que desaparece del archivo se desvincula o se elimina. El modo completo también elimina lo que el archivo omite, pero Microsoft lo desaconseja y lo está retirando progresivamente.

La vista previa what-if enumera cambios que nadie ha hecho. Una propiedad que el archivo omite, pero que Azure rellena con un valor predeterminado durante el despliegue, puede aparecer como eliminada, y Microsoft llama a esto ruido. Cualquier propiedad cuyo valor proceda de la función `reference` aparece como modificada cada vez, y valores como `utcNow()` o un parámetro seguro solo existen durante el despliegue, por lo que what-if los muestra sin evaluar. Ejecútelo en cada pull request, léalo como un borrador y establezca explícitamente las propiedades ruidosas para que destaquen los cambios reales:

```
az deployment group what-if \
  --resource-group rg-staging \
  --template-file main.bicep \
  --parameters staging.bicepparam
```

Una contraseña aparece en el historial de despliegues. Un parámetro de cadena ordinario se registra con el despliegue, donde puede verlo cualquiera con acceso de lectura al historial de despliegues. Marque los parámetros secretos con el decorador `@secure()`, que excluye su valor del historial y de los registros, y conserve el secreto en Key Vault en vez de un archivo de parámetros del repositorio.

Lecturas adicionales

- Qué es Bicep — la descripción general de Microsoft: el lenguaje, sus ventajas y lo que no pretende ser.
- Módulos Bicep — dividir un despliegue en archivos que se llaman entre sí.

- Parámetros en archivos Bicep — tipos, decoradores, valores predeterminados y lo que `@secure()` excluye del historial.
- Modos de despliegue — modo incremental frente al completo y qué hace cada uno con los recursos que no menciona el archivo.
- Operación what-if de despliegue Bicep — vista previa de un despliegue, tipos de cambio y fuentes de ruido conocidas.
- Pilas de despliegue — gestión de recursos como una unidad, con ajustes de eliminación y denegación.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Bicep?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/bicep/