Engineer

Bicep

Azures eigene Sprache zur Deklaration von Infrastruktur

Was es ist

Bicep ist Microsofts Sprache, um Azure-Infrastruktur als Text zu beschreiben: vorhandene Ressourcen, ihre Namen, Einstellungen und Beziehungen. Eine Bicep-Datei nennt keine Schritte. Sie beschreibt den Endzustand. Azure Resource Manager, die Schicht, durch die Azure-Verwaltungsanfragen laufen, bestimmt, was dafür anzulegen, zu ändern oder beizubehalten ist. Wer „Was ist Bicep?“ fragt, möchte meist wissen, ob es Klicks im Azure-Portal ersetzt. Das tut es, und darin liegt der größte Teil seines Nutzens.

Bicep ist dabei eine lesbarere Schreibweise für etwas, das Azure bereits verstand. Resource Manager akzeptiert seit Langem JSON-Bereitstellungsvorlagen. Sie sind vollständig, aber schwer zu lesen und noch schwerer zu prüfen. Bei einer Bereitstellung wandelt das Bicep-Kommandozeilenwerkzeug `.bicep` in eine solche JSON-Vorlage um. Resource Manager führt sie wie eine handgeschriebene aus. Laut Microsoft unterstützt Bicep jeden Ressourcentyp und jede API-Version sofort bei Einführung durch einen Ressourcenanbieter, einschließlich Vorschauen. Ein neuer Dienst benötigt also kein Warten auf ein Plugin-Update. Bestehende JSON-Vorlagen lassen sich nach Bicep dekompilieren. Die Migration eines älteren Bestands wird dadurch mechanische Arbeit statt einer Neufassung.

Eine kleine Datei zeigt die Form. Zuerst stehen Parameter, danach Ressourcen mit symbolischen Namen, auf die der Rest verweisen kann:

```
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'
}
```

Verweist eine andere Ressource auf `store.id`, erkennt Bicep die Abhängigkeit vom Speicherkonto. Niemand schreibt die Reihenfolge von Hand. Resource Manager stellt voneinander unabhängige Ressourcen parallel bereit. Eine Datei kann eine andere als Modul aufrufen. So wird „eine Datenbank samt Netzwerkregeln“ einmal beschrieben und überall wiederverwendet. Umgebungsspezifische Werte liegen in einer getrennten `.bicepparam`-Datei. Microsoft erklärt ausdrücklich, dass Bicep kostenlos, quelloffen und keine allgemeine Programmiersprache ist. Schleifen, Bedingungen und Funktionen dienen allein der Ressourcenbeschreibung.

Wie steht Bicep zu den Alternativen? Gegenüber ARM JSON nutzt es dieselbe Engine mit erheblich kürzerem Quelltext. Gegenüber Terraform unterscheiden sich vor allem Reichweite und Zustand: Terraform deckt viele Clouds ab und hält eine eigene Zustandsdatei. Biceps Azure-Resource-Manager-Workflow führt keine eigene Zustandsdatei, weil Azure den Bereitstellungszustand speichert. Bicep-Erweiterungen unterstützen auch Ressourcen außerhalb von Resource Manager, darunter Kubernetes und Microsoft Graph. Gegenüber Pulumi mit allgemeinen Sprachen wie TypeScript oder Go tauscht Bicep Flexibilität gegen eine kleinere, im Review schnell lesbare Sprache.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Bicep?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die gesamte Einrichtung wird in Dateien festgehalten, damit sie sich exakt wiederherstellen lässt und nicht nur im Kopf einer einzelnen Person existiert.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Von Hand gebaute Infrastruktur hat keine Geschichte. Eine Portaländerung hinterlässt keinen Grund, erhält keine vorherige Prüfung, und der Wiederaufbau nach einem schlechten Tag hängt von der Erinnerung an jeden Klick ab. Als

Dateien im Repository erhält Infrastruktur, was Anwendungscode bereits hat: Pull Request, Reviewer, Verlauf und Rückweg.

Der zweite Gewinn ist Gleichheit. Aus denselben Dateien mit verschiedenen Parametern gebaute Entwicklungs-, Staging- und Produktionsumgebungen unterscheiden sich nur an den angegebenen Stellen. Bicep-Bereitstellungen sind idempotent: Dieselbe Datei liefert laut Microsoft wiederholt „dieselben Ressourcentypen im selben Zustand". Die Wiederholung ist deshalb sicher. Eine frische Umgebung entsteht mit demselben Befehl gegen eine neue Ressourcengruppe.

Drittens werden Änderungen vorab sichtbar. What-if vergleicht Datei und laufenden Bestand und listet vor jeder Änderung die anzulegenden, zu verändernden oder zu löschenden Ressourcen. Zusammen mit dem Pull Request wird „Ich glaube, das ändert nur die Firewallregel" zu etwas, das ein Reviewer überprüfen kann.

Die Grenzen und Abwägungen verdienen eine klare Benennung. Welche Ressourcen unterstützt werden, hängt vom Bereitstellungsworkflow und seinen Erweiterungen ab. DNS bei einem anderen Anbieter oder Ressourcen in einer zweiten Cloud können daher weiterhin ein separates Werkzeug erfordern. Im Resource-Manager-Workflow beschreibt Bicep Azure-Ressourcen; Pakete und Dateien innerhalb einer virtuellen Maschine gehören zum Konfigurationsmanagement, etwa mit Ansible. Außerdem überraschen die Voreinstellungen von Resource Manager für Löschung und erneute Bereitstellung diejenigen, die von einem Sollzustandswerkzeug das Entfernen nicht mehr genannter Ressourcen erwarten. Abschnitt vier behandelt das.

Wie es in der Praxis aussieht

Mehrere Azure-Projekte dieser Website bauen auf Bicep. Drei zeigen seinen Nutzen für ein Team über die Zeit.

Auf einer Plattform steht der ganze Azure-Bestand in Bicep. Jede Umgebung entsteht aus denselben Vorlagen: api und www als Azure Container Apps, PostgreSQL Flexible Server, Redis für Caching, Front Door mit vorgelagerter WAF-Richtlinie und das Netz darunter mit VNet, NSG und privatem DNS. Log Analytics ist zur Diagnose angeschlossen, Images kommen aus der Azure Container Registry des Projekts. Durchgängige Parametrisierung macht neue

Umgebungen und Änderungen zu Pull Requests. Laut Seite wurden Umgebungen reproduzierbar und prüfbar. Abweichungen verloren ihren rätselhaften Charakter, weil der Code die maßgebliche Quelle wurde.

Dieselben Vorlagen tragen den Veröffentlichungsweg. Releases gehen von Entwicklung über Staging in die Produktion. Bicep hält die Zielinfrastruktur identisch, damit ein Build nicht an einen unmerklich anderen Ort befördert wird. Abweichende Umgebungskonfiguration bleibt von Geheimnissen getrennt. Ein gebautes Artefakt durchläuft deshalb ohne Neubau jede Stufe.

Bicep enthält auch einen Teil des Wiederherstellungsplans. Die Aufbewahrungsfrist der Sicherung steht neben der geschützten Datenbank im Bicep-Code. Point-in-Time Restore wird damit zur Vorlageneigenschaft statt zum einmaligen Portalklick. Die umgebende Infrastruktur ist ebenfalls Code. Das zählt, weil Daten in eine nur aus dem Gedächtnis neu aufbaubare Umgebung zu bringen kaum Wiederherstellung ist. Die Seite benennt danach die offenen Punkte: Georedundanz ist ausgeschaltet, das Wiederherstellungszeitziel vorgeschlagen statt gemessen. Eine Vorlage zeigt die Konfiguration. Nur eine Übung zeigt die Dauer der Wiederherstellung.

Wo es schiefgeht

Meist wird erwartet, Bicep besitze alles in einer Ressourcengruppe. Standardmäßig besitzt es nur das ausdrücklich Genannte und nur in der beschriebenen Weise.

Eine Portaländerung verschwindet bei der nächsten Bereitstellung. Eine Ressource erneut bereitzustellen wendet alle Eigenschaften an und setzt ausgelassene auf Standardwerte. Der manuelle Hotfix eines Incidents wird beim nächsten normalen Release rückgängig. Behandeln Sie die Datei als vollständigen Endzustand, schreiben Sie alle vom Standard abweichenden Werte auf und führen Sie auch dringliche Änderungen per Pull Request aus. Sperreinstellungen eines Deployment Stacks können Bearbeitung außerhalb der Bereitstellung vollständig verhindern.

Eine entfernte Ressource läuft und kostet weiter. Der Standardmodus ist inkrementell. Vorhandene, nicht mehr genannte Ressourcen bleiben unangetastet. Eine aus dem Code entfernte Datenbank bleibt in Azure. Microsoft empfiehlt Deployment Stacks. Ein Stack erfasst seine Ressourcen.

`actionOnUnmanage` bestimmt, ob aus der Datei verschwundene Ressourcen abgetrennt oder gelöscht werden. Auch der vollständige Modus löscht Fehlendes, wird aber von Microsoft nicht empfohlen und schrittweise eingestellt.

What-if listet nie vorgenommene Änderungen. Eine ausgelassene, bei der Bereitstellung von Azure ergänzte Standardeigenschaft kann als gelöscht erscheinen. Microsoft nennt das Rauschen. Werte aus `reference` erscheinen jedes Mal als Änderung. `utcNow()` oder sichere Parameter existieren erst während der Bereitstellung und werden daher unausgewertet gezeigt. Führen Sie What-if für jeden Pull Request aus, lesen Sie es als Entwurf und setzen Sie störende Eigenschaften ausdrücklich, damit echte Änderungen auffallen:

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

Ein Passwort erscheint im Bereitstellungsverlauf. Ein gewöhnlicher Stringparameter wird mitprotokolliert und ist für jeden mit Leserechten auf den Verlauf sichtbar. Kennzeichnen Sie geheime Parameter mit `@secure()`, das ihren Wert aus Verlauf und Protokollen fernhält. Das Geheimnis selbst gehört in Key Vault statt in eine Parameterdatei im Repository.

Weiterführende Lektüre

- Was ist Bicep? — Microsofts Überblick über Sprache, Vorteile und bewusst nicht verfolgte Ziele.
- Bicep-Module — Aufteilung einer Bereitstellung in einander aufrufende Dateien.
- Parameter in Bicep-Dateien — Typen, Dekoratoren, Standardwerte und die Wirkung von `@secure()` auf den Verlauf.
- Bereitstellungsmodi — inkrementell und vollständig und der jeweilige Umgang mit nicht genannten Ressourcen.
- What-if für Bicep-Bereitstellungen — Vorschau, Änderungstypen und bekannte Ursachen von Rauschen.
- Deployment Stacks — gemeinsame Ressourcenverwaltung mit Lösch- und Sperreinstellungen.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Bicep?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/bicep/