Engineer

Bicep

Azures eget sprog til at erklære infrastruktur

Hvad det er

Bicep er et sprog fra Microsoft til at skrive Azure-infrastruktur ned som tekst: hvilke ressourcer der findes, hvad de hedder, hvordan hver enkelt er sat op, og hvordan de hænger sammen. En Bicep-fil indeholder ingen trin. Den beskriver slutresultatet, og Azure Resource Manager, det lag, som forespørgsler om administration af Azure-ressourcer passerer igennem, regner selv ud, hvad der skal oprettes, ændres eller lades i fred for at nå dertil. Den, der spørger »hvad er Bicep?«, vil som regel vide, om det kan afløse at klikke sig gennem Azure-portalen. Det kan det, og det er det meste af værdien.

Under overfladen er Bicep en venligere måde at skrive noget, Azure allerede forstod. Resource Manager har længe taget imod skabeloner skrevet i JSON, som er fuldstændige, men svære at læse og endnu sværere at reviewe. Under en udrulning oversætter Bicep-kommandolinjeværktøjet en `.bicep`-fil til sådan en JSON-skabelon, og Resource Manager udruller den præcis, som var den skrevet i hånden. Ifølge Microsoft understøtter Bicep alle ressourcetyper og API-versioner, så snart en ressourceudbyder indfører dem, previews medregnet, så en ny tjeneste kan tages i brug uden at vente på, at nogen opdaterer et plugin. Eksisterende JSON-skabeloner kan dekompileres til Bicep, og det gør flytningen af et ældre setup til et mekanisk stykke arbejde frem for en omskrivning.

En lille fil viser formen. Først parametre, så ressourcer, hver med et symbolsk navn, som resten af filen kan henvise til:

```
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'
}
```

Når en anden ressource nævner `store.id`, udleder Bicep, at den skal vente på lagerkontoen, så ingen skriver rækkefølgen i hånden, og Resource Manager udruller alt, der ikke afhænger af noget andet, parallelt. En fil kan kalde en anden fil som modul, så et team kan beskrive »en database med dens netværksregler« én gang og genbruge den overalt. Værdier, der skifter fra miljø til miljø, ligger i en separat `.bicepparam`-fil. Microsoft siger klart, at Bicep er gratis, open source og ikke tænkt som et generelt programmeringssprog: der er løkker, betingelser og funktioner, men kun til at beskrive ressourcer med.

Hvordan står Bicep i forhold til alternativerne? Over for ARM-JSON er det den samme motor med en langt kortere kildetekst. Over for Terraform handler forskellen om rækkevidde og tilstand: Terraform dækker mange skyer og fører sin egen tilstandsfil, mens Biceps arbejdsgang med Azure Resource Manager ikke fører nogen egen tilstandsfil, fordi Azure gemmer udrulningens tilstand. Biceps udvidelser understøtter også ressourcer uden for Resource Manager, herunder Kubernetes og Microsoft Graph. Over for Pulumi, der beskriver infrastruktur i almindelige programmeringssprog som TypeScript eller Go, bytter Bicep fleksibilitet for et mindre sprog, der er hurtigt at læse i et review.

Brug begrebet i en beslutning

Brug »Hvad er Bicep?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Hele opsætningen skrevet ned som filer, så den kan genskabes præcist og aldrig kun findes i én persons hoved.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

Infrastruktur bygget i hånden har ingen historik. En indstilling ændret i portalen efterlader ingen begrundelse, ingen ser den igennem, før den sker, og at genopbygge miljøet efter en dårlig dag afhænger af, at nogen husker hvert

eneste klik. Når infrastrukturen er et sæt filer i repositoriet, får den det, applikationskoden allerede har: en pull request, en reviewer, en historik og en vej tilbage.

Den anden gevinst er ensartethed. Udvikling, staging og produktion bygget fra de samme filer med forskellige parametre adskiller sig kun dér, hvor parametrene siger det. Bicep-filer er idempotente, hvilket ifølge Microsoft betyder, at den samme fil udrullet mange gange giver de samme ressourcetyper i den samme tilstand. En udrulning kan derfor trygt gentages, og et nyt miljø er den samme kommando rettet mod en ny ressourcegruppe.

Den tredje er overblik, før noget ændres. What-if-operationen sammenligner en fil med det, der kører, og lister hver ressource, den ville oprette, ændre eller slette, inden noget bliver rørt. Sammen med en pull request gør den forhåndsvisning »jeg tror kun, det ændrer firewallreglen« til noget, en reviewer kan efterprøve.

Omkostningerne er værd at nævne. Hvilke ressourcer der understøttes, afhænger af udrulningsarbejdsgangen og dens udvidelser, så DNS hos en anden udbyder eller ressourcer i en ekstra cloud kan stadig kræve et særskilt værktøj. I arbejdsgangen med Resource Manager beskriver Bicep Azure-ressourcer; pakker og filer inde i en virtuel maskine hører hjemme i konfigurationsstyring som Ansible. Og Resource Managers standardadfærd ved sletning og genudrulning overrasker dem, der forventer, at et værktøj til ønsket tilstand fjerner det, filen ikke længere nævner. Det vender vi tilbage til nedenfor.

Sådan ser det ud i praksis

Flere af Azure-projekterne på dette site hviler på Bicep, og tre af dem viser, hvad det gør for et team over tid.

På én platform er hele Azure-setuppet skrevet i Bicep. Hvert miljø kommer ud af de samme skabeloner: api'et og www som Azure Container Apps, en PostgreSQL Flexible Server, Redis til caching, Front Door med en WAF-politik foran og netværket nedenunder (VNet, NSG, private DNS), med Log Analytics koblet på til diagnostik og images hentet fra projektets Azure Container Registry. Fordi det hele er parametriseret, er et nyt miljø eller en ændring af et eksisterende en pull request. Siden beskriver resultatet som miljøer, der blev reproducerbare og reviderbare, og afvigelser mellem dem, der holdt op med at være et mysterium, fordi koden er kilden til sandhed.

De samme skabeloner bærer releasevejen. Releases bevæger sig fra udvikling til staging til produktion, og Bicep holder målinfrastrukturen ens fra det ene miljø til det næste, så et build aldrig promoveres ind i et lidt anderledes sted. Ved siden af skabelonerne er konfigurationen for hvert miljø adskilt fra secrets, så ét bygget artefakt kan passere alle trin uden at blive bygget om.

Og Bicep rummer en del af genopretningsplanen. Backup-retention er konfigureret i Bicep ved siden af den database, den beskytter, så gendannelse til et tidspunkt er en egenskab ved skabelonen frem for en indstilling, nogen engang klikkede. Miljøet omkring databasen er også kode, og det tæller, for at gendanne data i et miljø, der skal genopbygges efter hukommelsen, er næppe genopretning. Siden navngiver derefter, hvad der står åbent: geo-redundans er slået fra, og målet for genopretningstid er foreslået frem for målt. En skabelon viser, hvad der er konfigureret; kun en øvelse viser, hvor lang tid en gendannelse tager.

Hvor det går galt

Det meste besvær med Bicep kommer af at forvente, at det opfører sig som et værktøj, der ejer alt i en ressourcegruppe. Som standard ejer det kun det, filen nævner, og kun på den måde, filen siger.

En indstilling ændret i portalen forsvinder ved næste udrulning. Når en ressource udrulles igen, anvendes alle dens egenskaber på ny, og enhver egenskab, skabelonen udelader, nulstilles til standardværdien. Et hotfix lavet i hånden under en hændelse bliver rullet tilbage af den næste rutinemæssige release. Behandl filen som den fulde sluttilstand for hver ressource, skriv alle værdier, der afviger fra standarden, ind i den, og lav hasteændringer gennem en pull request. Deny-indstillingerne i en deployment stack kan helt spærre for ændringer uden om udrulningen.

En ressource, der er fjernet fra skabelonen, kører videre og koster stadig penge. Standardtilstanden er inkrementel: ressourcer i ressourcegruppen, som filen ikke nævner, får lov at blive. Slet en database fra koden, og den bliver i Azure. Microsofts svar er deployment stacks. En stack registrerer de ressourcer, den styrer, og dens indstilling `actionOnUnmanage` afgør, om en ressource, der forsvinder fra filen, bliver koblet fra eller slettet. Complete-tilstanden sletter også det, filen udelader, men Microsoft fraråder den og udfaser den gradvist.

What-if-forhåndsvisningen lister ændringer, ingen har lavet. En egenskab, som filen udelader, men som Azure udfylder med en standardværdi under udrulningen, kan blive vist som slettet, og Microsoft kalder det støj. En egenskab sat ud fra funktionen `reference` vises som ændret hver gang, og værdier som `utcNow()` eller en sikker parameter findes kun under selve udrulningen, så what-if viser dem uevaluerede. Kør den på hver pull request, læs den som et udkast, og sæt de støjende egenskaber eksplicit, så de rigtige ændringer træder frem:

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

En adgangskode dukker op i udrulningshistorikken. En almindelig strengparameter gemmes sammen med udrulningen, hvor alle med læseadgang til historikken kan se den. Markér hemmelige parametre med dekoratoren `@secure()`, som holder værdien ude af historik og logs, og opbevar selve hemmeligheden i Key Vault frem for i en parameterfil i repositoriet.

Læs mere

- What is Bicep? — Microsofts egen oversigt: sproget, fordelene og det, det ikke forsøger at være.
- Bicep modules — sådan deles en udrulning op i filer, der kalder hinanden.
- Parameters in Bicep files — typer, dekoratorer, standardværdier, og hvad `@secure()` holder ude af historikken.
- Deployment modes — inkrementel over for complete, og hvad hver af dem gør med ressourcer, filen ikke nævner.
- Bicep deployment what-if operation — forhåndsvisning af en udrulning, ændringstyperne og de kendte kilder til støj.
- Deployment stacks — ressourcer styret som én enhed, med sletning og deny-indstillinger.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er Bicep?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/bicep/