Engineer

Bicep

Azures eget språk for å deklarere infrastruktur

Hva det er

Bicep er et språk fra Microsoft for å skrive ned Azure-infrastruktur som tekst: hvilke ressurser som finnes, hva de heter, hvordan hver av dem er konfigurert, og hvordan de henger sammen. En Bicep-fil lister ikke opp steg. Den oppgir sluttresultatet, og Azure Resource Manager, laget administrasjonsforespørsler til Azure går gjennom, finner ut hva som må opprettes, endres eller få være i fred for å komme dit. Den som spør «hva er Bicep?», vil som regel vite om det erstatter klikking gjennom Azure-portalen. Det gjør det, og der ligger det meste av verdien.

Under overflaten er Bicep en vennligere måte å skrive noe Azure allerede forsto. Resource Manager har lenge tatt imot maler for utrulling skrevet i JSON, som er komplette, men tunge å lese og enda tyngre å gjennomgå. Under en utrulling gjør kommandolinjeverktøyet for Bicep en `.bicep` -fil om til en slik JSON-mal, og Resource Manager ruller den ut nøyaktig slik den ville rullet ut en håndskrevet mal. Microsoft oppgir at Bicep støtter hver ressurstype og API-versjon så snart en ressursleverandør innfører den, forhåndsversjoner inkludert, så en ny tjeneste kan tas i bruk uten å vente på at noen oppdaterer et tillegg. Eksisterende JSON-maler kan dekompileres til Bicep, og det gjør flyttingen av et eldre oppsett til en mekanisk jobb i stedet for en omskriving.

En liten fil viser formen. Parameterne kommer først, så ressursene, hver med et symbolsk navn resten av filen kan vise 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 annen ressurs nevner `store.id`, slutter Bicep seg til at den må vente på lagringskontoen, så ingen skriver rekkefølgen for hånd, og Resource Manager ruller ut parallelt alt som ikke avhenger av noe annet. En fil kan kalle en annen fil som en modul, og det lar et team beskrive «en database med nettverksreglene sine» én gang og bruke den overalt. Verdier som varierer mellom miljøene, ligger i en egen `.bicepparam`-fil. Microsoft sier tydelig at Bicep er gratis, har åpen kildekode og ikke er ment som et generelt programmeringsspråk: det har løkker, betingelser og funksjoner, men bare for å beskrive ressurser.

Hvordan står Bicep seg mot alternativene? Mot ARM-JSON er det den samme motoren med langt kortere kildekode. Mot Terraform er de viktigste forskjellene rekkevidde og tilstand: Terraform dekker mange skyer og fører sin egen tilstandsfil, mens Biceps arbeidsflyt med Azure Resource Manager ikke fører noen egen tilstandsfil, fordi Azure lagrer utrullingstilstanden. Biceps utvidelser støtter også ressurser utenfor Resource Manager, blant annet Kubernetes og Microsoft Graph. Mot Pulumi, som beskriver infrastruktur i generelle programmeringsspråk som TypeScript eller Go, bytter Bicep bort fleksibilitet mot et mindre språk som er raskt å lese i en gjennomgang.

Bruk begrepet i en beslutning

Bruk «Hva er Bicep?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Hele oppsettet skrevet ned som filer, slik at det kan gjenskapes nøyaktig og aldri bare finnes i hodet på én person.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Skrev 1 000 RBAC-regler for å sikre Linux-servere — 1,000 tilgangsregler skrevet for å tette et sikkerhetshull

Skrev 1 000 RBAC-regler for å sikre Linux-servere Ta kontakt →

Hvorfor det er viktig

Infrastruktur bygd for hånd har ingen historikk. En innstilling som endres i portalen, etterlater ingen begrunnelse, ingen ser over den før den skjer, og å bygge opp miljøet igjen etter en dårlig dag avhenger av at noen husker hvert

klikk. Når infrastrukturen er et sett filer i repoet, får den det applikasjonskoden allerede har: en pull request, en kollega som leser den, en historikk og en vei tilbake.

Den andre gevinsten er likhet. Utvikling, staging og produksjon bygd fra de samme filene med ulike parametere skiller seg bare der parameterne sier at de skal. Utrullinger med Bicep er idempotente, som med Microsofts ord betyr at den samme filen rullet ut mange ganger gir de samme ressurstypene i samme tilstand. En utrulling er dermed trygg å gjenta, og et nytt miljø er den samme kommandoen rettet mot en ny ressursgruppe.

Den tredje er innsyn før endringen. Operasjonen what-if sammenligner en fil med det som kjører, og lister opp hver ressurs den ville opprettet, endret eller slettet, før noe blir rørt. Sammen med en pull request gjør den forhåndsvisningen «jeg tror dette bare endrer brannmurregelen» om til noe den som gjennomgår endringen, kan sjekke.

Kostnadene er verdt å nevne. Ressursdekningen avhenger av arbeidsflyten for utrulling og utvidelsene som brukes, så DNS hos en annen leverandør eller ressurser i en annen sky kan fortsatt kreve et eget verktøy. I arbeidsflyten med Resource Manager beskriver Bicep Azure-ressurser; pakker og filer inne i en virtuell maskin hører hjemme i konfigurasjonsstyring som Ansible. Og det Resource Manager gjør som standard ved sletting og ny utrulling, overrasker folk som venter at et verktøy for ønsket tilstand fjerner det filen ikke lenger nevner. Det tar delen om hvor det går galt, for seg nedenfor.

Slik ser det ut i praksis

Flere av Azure-prosjektene på dette nettstedet hviler på Bicep, og tre av dem viser hva det gjør for et team over tid.

På én plattform er hele Azure-oppsettet skrevet i Bicep. Hvert miljø kommer ut av de samme malene: api og www kjører som Azure Container Apps, med en PostgreSQL Flexible Server, Redis til caching og Front Door med en WAF-policy foran. Nettverket under består av VNet, NSG og privat DNS, Log Analytics samler diagnostikken, og containerbildene kommer fra prosjektets eget Azure Container Registry. Fordi alt er parameterisert, er det å sette opp et nytt miljø eller endre et eksisterende en pull request. Siden oppgir resultatet som miljøer som ble reproduserbare og lot seg gjennomgå, og avvik som sluttet å være et mysterium, fordi koden er sannhetskilden.

De samme malene bærer utgivelsesveien. Utgivelser går fra utvikling til staging til produksjon, og Bicep holder målinfrastrukturen identisk fra ett miljø til det neste, slik at et bygg aldri føres videre til et miljø som er litt annerledes. Den delen av konfigurasjonen som skiller ett miljø fra et annet, ligger atskilt fra hemmelighetene, og det lar én bygd artefakt passere hvert trinn uten å bygges på nytt.

Og Bicep holder en del av gjenopprettingsplanen. Oppbevaringstiden for sikkerhetskopier er satt i Bicep ved siden av databasen den beskytter, så gjenoppretting til et valgt tidspunkt følger av malen, i stedet for å være en innstilling noen satte i portalen en gang. Miljøet rundt databasen er også kode, og det betyr noe, fordi å gjenopprette data inn i et miljø som må bygges opp igjen etter hukommelsen, knapt er gjenoppretting. Siden nevner deretter det som fortsatt står åpent: georedundans er slått av, og målet for gjenopprettingstid er foreslått og ikke målt. En mal viser hva som er konfigurert; bare en øvelse viser hvor lang tid en gjenoppretting tar.

Hvor det går galt

De fleste problemer med Bicep kommer av å vente at det skal oppføre seg som et verktøy som eier alt i en ressursgruppe. Som standard eier det bare det filen nevner, og bare på den måten filen sier.

En innstilling som ble endret i portalen, forsvinner ved neste utrulling. Når en ressurs rulles ut på nytt, brukes alle egenskapene dens igjen, og en egenskap malen utelater, settes tilbake til standardverdien. En hasteretting gjort for hånd under en driftshendelse blir angret av neste rutinemessige utgivelse. Behandle filen som den fullstendige sluttilstanden for hver ressurs, skriv hver verdi som avviker fra standarden, inn i den, og gjør hasteendringer gjennom en pull request. Innstillingene for å nekte endringer (deny settings) i en deployment stack kan blokkere redigering utenfor utrullingen helt.

En ressurs som er fjernet fra malen, fortsetter å kjøre og å koste penger. Standardmodusen for utrulling er inkrementell: ressurser i ressursgruppen som filen ikke nevner, får være i fred. Slett en database fra koden, og den blir værende i Azure. Microsofts svar er deployment stacks. En stack registrerer ressursene den styrer, og innstillingen `actionOnUnmanage` avgjør om en ressurs som faller ut av filen, kobles fra eller slettes. Komplett modus (complete) sletter også det filen utelater, men Microsoft fraråder den og er i ferd med å fase den ut.

Forhåndsvisningen what-if lister opp endringer ingen har gjort. En egenskap filen utelater, men som Azure fyller inn med en standardverdi under utrullingen, kan bli rapportert som slettet, og Microsoft kaller dette støy. En egenskap som settes fra funksjonen `reference`, rapporteres som endret hver gang, og verdier som `utcNow()` eller en sikker parameter finnes bare under utrullingen, så what-if viser dem uevaluert. Kjør den på hver pull request, les den som et utkast, og sett de støyende egenskapene eksplisitt, slik at de virkelige endringene skiller seg ut:

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

Et passord dukker opp i utrullingshistorikken. En vanlig strengparameter blir lagret sammen med utrullingen, der alle med lesetilgang til historikken kan se den. Merk hemmelige parametere med dekoratøren `@secure()`, som holder verdien utenfor utrullingshistorikken og loggene, og oppbevar selve hemmeligheten i Key Vault i stedet for i en parameterfil i repoet.

Les mer

- What is Bicep? — Microsofts egen oversikt: språket, fordelene og hva det ikke prøver å være.
- Bicep modules — å dele en utrulling opp i filer som kaller hverandre.
- Parameters in Bicep files — typer, dekoratører, standardverdier og hva `@secure()` holder utenfor historikken.
- Deployment modes — inkrementell mot komplett, og hva hver av dem gjør med ressurser filen ikke nevner.
- Bicep deployment what-if operation — forhåndsvisning av en utrulling, endringstypene og de kjente kildene til støy.
- Deployment stacks — å styre ressurser som én enhet, med innstillinger for sletting og for å nekte endringer.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er Bicep?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/bicep/