Hur man snabbt sätter upp resurser på Azure med hjälp av Bicepskript

|  |  |
| --- | --- |
| **Namn** | Mazdak Jafari |
| **Utbildning** | DevOps -21 |
| **Uppdrags-givare** | Getit Nordic |
| **Handledare/ examinator** | Viktor Svensson |
| **Datum** | 2024-06-12 |

# Sammanfattning

Detta arbete fokuserar på att utveckla och implementera en Bicep-skriptad template för att snabbt och effektivt sätta upp resurser i Azure-miljöer, specifikt för webbapplikationer och relaterade tjänster. Genom att använda Bicep-skriptet kunde en resursgrupp skapas med olika miljöer (till exempel utveckling och produktion) för att hantera applikationsinfrastrukturen. Dessutom inkluderade templaten uppsättningen av en container registry och konfigurationen av en SendGrid-resurs.

Arbetet utforskade bakgrunden till övergången från serverhallar till molntjänster och betonade fördelarna med att använda molntjänster, särskilt inom kostnadsbesparingar och geografisk tillgänglighet. Syftet med projektet var att demonstrera hur snabbt och effektiv infrastruktur kan skapas och hanteras med hjälp av Bicep-skript.

Genom att använda Bicep kunde resurser deklareras och distribueras på ett enkelt och deklarativt sätt, vilket möjliggjorde automatisering och konsekventa resultat. Bicep-skriptets struktur, inklusive huvudfilen och moduler, bidrog till en organiserad och lättunderhållen kodbas.

Slutligen visade arbetet att Bicep är ett kraftfullt verktyg för att effektivisera Azure-distributioner, och dess användarvänliga syntax och flexibilitet möjliggör snabb och smidig implementering av molninfrastruktur.

Innehåll

[1. Inledning 1](#_Toc100581005)

[1.1 Bakgrund 2](#_Toc100581006)

[1.2 Syfte 2](#_Toc100581007)

[1.3 Problemformulering 2](#_Toc100581008)

[1.4 Avgränsningar och fokus 3](#_Toc100581009)

[1.5 Metod/Arbetssätt 3](#_Toc100581010)

[2. Teori 4](#_Toc100581011)

[3. Resultat 7](#_Toc100581012)

[4. Diskussion 7](#_Toc100581013)

[5. Slutsatser 8](#_Toc100581014)

[5.1 Rekommendationer 8](#_Toc100581015)

[6. Referenslista 9](#_Toc100581016)

[Bilagor 10](#_Toc100581017)

# Inledning

Jag gjorde mitt examensarbete på min LIA-plats Getit Nordic. Bolaget består av moderbolaget GetCap och koncernen består av ca 47 konsulter och medarbetare som jobbar inom IT. GetIT är ett konsultbolag som dessutom har en produktutveckling ”inhouse”. Konsulterna jobbar med bland annat cloud adoption, data intelligence, development and products, technical advisory, business analytics och networking.  
Yousr jobbar mot företag med IT-support med hjälp av client delivery. DB24 unika affärsidé är att de automatiserar den dagliga driften av SQL-databaser.   
Bolaget har sitt huvudkontor i Göteborg. Men de olika bolagen har även kontor i Karlstad, Stockholm och Köpenhamn.

Som första projekt fick jag i uppgift att skapa ett template fil med hjälp av Bicep-script som skulle deployas i Azure. Projektet användes till DB24 och ska användas som underlag för framtida kunder. Templaten består av att en resursgrupp där två olika miljöer (dev = utveckling och prod = produktion) kan köra en webbapp och serviceplan. Den tredje miljön bestod av en container registry. Samt att en sendgrid sattes upp.

Arkitekturen såg ut på detta sätt:

En bild som visar diagram, skärmbild, linje, design

Automatiskt genererad beskrivning

***Fig 1.***

## 

## **Bakgrund**

Övergången från serverhallar till molntjänster är ett faktum när man ser hur marknaden för dessa tjänster ökar. Bara under 2020 så ökade cloud migreringar med 61 % bland företag.   
48 % av företagen har sin viktiga data på molntjänster. Då datorhallar är ganska kostsamma och kräver mycket underhåll, är cloudtjänster ett smartare och billigare sätt att erhålla dessa resurser. Så länge man har en internetuppkoppling är driften säkrad och man har tillgång till sin lagring och tjänster som de förser en med. Det cloudtjänsterna kan bidra med är hela IT-infrastrukturen på ett företag, vilket innebär installera servrar, programvaror och nätverksresurser som behövs. En annan fördel som det innehar är att företagen inte blir geografiskt begränsade. Fjärranvändare kan få tillgång till tjänsterna oavsett plats, så länge en uppkoppling finnes.

Därför väljer och inser allt fler företag värdet av att föra över sina verksamheter på de befintliga cloudtjänsterna. Ett företag kan spara stora kostnader genom att köpa in dessa tjänster och effektivisera verksamheten.

För att snabbt kunna sätta upp resurser som kan skala upp en verksamhet för en produkt/tjänst, så kan ett template snabbt bistå med allt grundläggande inställningar för att tillgodose de nödvändiga funktionerna. På min LIA så utvecklades denna template in-house för pågående projekt samt framtida projekt där man snabbt behövde sätta upp resurser som behövs. Ska man sätta upp dessa resurser manuellt kräver det tid och resurser.

## Syfte

*Syftet med arbetet/uppsatsen är att visa hur snabbt med hjälp av en template i bicepscript, man kan sätta miljöer i Azure för att hantera applikationer och resurser.*

## Problemformulering

Arbetet ska bestå av en template som står i bicepscript. Detta script ska kunna köras och sätta upp resurser i ett befintligt Azure konto. Men vilken uppsättning och hur skriver man scriptet i bicep så att dessa resurser kan deployas och köras på ett subscription.

Den problematik jag kommer utreda är:

För att bespara tid, pengar och resurser ska jag skriva ett skript i bicep som sätter upp en resurs med två olika miljöer i Azure. Detta skript ska även hantera olika typer av funktioner som är relevanta för kunden.

## **1.4 Avgränsningar och fokus**

Detta arbete kommer endast behandla bicepskript även om det finns många likheter med ARM. Det går även att konvertera delar av templaten till ARM från bicep på Azure plattformen.  
Jag kommer hålla mig till Azure och inte beskriva hur det kan fungera eller skriptas på andra plattformar. Bicep är framtaget och utvecklat för just Azure och kan inte deployas till andra plattformar.   
Sendgriden som sätts upp med hjälp av twilio som är en tjänst på Azure kommer uteslutas i detta arbete, då den inte besvarar frågeställningen utan är en lösning som tillämpades för konsulttjänsten som Getit bistod med.

## **1.5 Metod/Arbetssätt**

Arbetet kommer använda sig av kvantitativa metoder för att det mesta materialet och tillämpningar finns givet. Uppsatsen uppgift är att del av befintlig praxis för att kunna deploya så att resurserna sätts upp. I arbetet kommer bicepskript och instruktioner för hur man ska gå till väga i Azure förklaras och beskrivas.   
En person som vill skapa samma uppsättning ska kunna med hjälp av detta arbete återskapa uppsättningen, dock med andra namnsättningar på de olika delarna. Detta arbete kommer använda de namn som vi använde på Getit.

Testning och validering: Efter att jag skrivit bicepsskriptet är det viktigt att testa och validera det för noggrannhet. Detta innebär att köra skriptet och bekräfta att de utlovade resurserna matchar vad som förväntades.  
Distribution och användning: Efter framgångsrik testning kan skriptet deploya för användning i en produktionsmiljö. Detta kan inkludera att installera och konfigurera skriptet på Azure.

# 2. Teori

Med den här uppsatsen vill jag förklara hur man skapar resursgrupper på Azure med ett Bicep-skript. Bicep är ett effektivt sätt att definiera och distribuera molninfrastruktur som kod. Jag kommer att gå igenom processen att ställa in resursgrupper med hjälp av ett Bicep-skript, samt diskutera fördelarna med att använda resursgrupper för att hantera molninfrastruktur. Men först måste man registrera ett konto för att deploya resurserna på.

**Ställa in resursgrupper med ett Bicep-skript**

Bicep skript är ett domänspecifikt språk som används för att definiera och distribuera resurser på Azure. Det är utvecklat av Microsoft och hjälper till att göra Azure-distributioner enklare och hanterbara. Bicep-skriptet gör det möjligt för utvecklare att definiera all sin infrastruktur som kod (IaC), vilket möjliggör bättre automatisering, färre fel och mer konsekventa resultat. Resursgrupper är logiska behållare i Azure som lagrar relaterade resurser. De gör det möjligt för användaren att gruppera och hantera sina resurser enligt de krav som är ställda, till exempel en webbapplikation som kan innehålla en virtuell maskin, lagringskonto och nätverksgränssnitt. Resursgrupper erbjuder flera fördelar: förenklad resurshantering, förbättrad säkerhet, kostnadsoptimering och en allt-i-ett-vy för spårning och övervakning av användning. När du skapar en resursgrupp väljer du ett lämpligt namn som tydligt identifierar dess innehåll och beaktar dess plats baserat på användarens geografiska region. Dessutom övervakar och optimerar man resursanvändningen för att minimera kostnaderna. Genom att skriva bicep-skript kommer kodningsteam som arbetar med Azure att snabbt kunna distribuera och hantera resurser med färre fel och mer effektivitet, vilket gör att de kan fokusera mer av sin tid på andra uppgifter.

Bicep är ett språk som används för att deklarera Azure-resurser. Det är en deklarativ syntax som låter dig definiera och distribuera din infrastruktur som kod. Med Bicep kan du konfigurera resursgrupper och de resurser de innehåller. För att skapa resursgrupper med hjälp av Bicep måste vi deklarera resursgrupperna och de resurser de innehåller.   
Skriptet ska definiera tre resursgrupper och deras platser, inklusive en SendGrid-resurs i prod-resursgruppen. Den innehåller SendGrid-kontodetaljer som API-nyckel, lösenord och användarnamn. Du kan anpassa dina resursgrupper för att möta dina behov genom att lägga till ytterligare resurser som virtuella maskiner, lagringskonton och nätverksgränssnitt. Dessutom kan du definiera beroenden mellan dina resurser för korrekt distributionsordning. När du skapar en resursgrupp väljer du ett lämpligt namn som tydligt identifierar dess innehåll och beakta dess plats baserat på användarens geografiska region. Dessutom övervakar och optimerar man resursanvändningen för att minimera kostnaderna.

**Strukturen i bicep**

Main-filen i Bicep är den huvudfilen i ett projekt. Main-filen innehåller vanligtvis huvuddelen av infrastrukturkod och fungerar som startpunkten för att förklara och definiera Azure-resurserna man vill implementera.

För att skapa Azure-resurser med Bicep är det vanligt att dela upp kod i flera Bicep-filer. Detta hjälper en att organisera och strukturera sitt projekt. Main-filen är den fil där du anger alla resurser, såväl som eventuella moduler eller länkar till andra Bicep-filer.

Bicep-moduler organiserar och återanvänder kod genom att dela upp den i logiska enheter. Med hjälp av moduler kan du skapa komponenter som kan användas igen och organisera din kod på ett sätt som gör den lättare att underhålla och förstå.

Bicep script är ett språk för att uttrycka resurser och distributioner på Azure-plattformen. Den är designad för att vara mer användarvänlig än traditionella ARM-mallar samtidigt som den ger samma kraftfulla funktioner. Bicep är både objektorienterad och deklarativ, vilket innebär att den använder syntax och termer som är lätta för användarna att förstå. Den har också variabler och resursfunktioner som ska användas i deklarationer för att göra distributioner enklare. Syntaxen för Bicep följer formen:

"< resurs > < typ > < namn > = < egenskaper >"   
  
och resurser kan deklareras inuti varandra. Variabler används i Bicep för att definiera värden som kan användas genom hela skriptet. Variabler måste deklareras med syntaxen  
  
"var < namn > = < värde >"  
  
och kan användas någon annanstans i skriptet. Bicep har också resurser som tillhandahåller värdefulla funktioner under hela driftsättningen. Dessa resurser gör det möjligt för användare att skapa parametrar, utdata, moduler och andra element i skriptet. Slutligen erbjuder Bicep ett enkelt sätt att distribuera resurser till Azure. Syntaxen för Bicep ger ett mer intuitivt sätt för användare att distribuera Azure-resurser utan att behöva lära sig komplexa ARM-mallar. Med sin kraftfulla syntax, användarvänliga funktioner och enkla distribution är Bicep script en idealisk resurs för att distribuera resurser till Azure.

Att skriva bicepsskript innebär några steg. För det första måste man deklarera och initiera variabler, vilket kan göras med den grundläggande syntaxen   
  
"var variabel-namn = variabel-värde."  
  
 Variabelns typ härleds automatiskt från värdet och namnet på variabeln kan inte vara detsamma som en parameter, resurs eller utdataidentifierare. När variabler väl har deklarerats och initierats är nästa steg att definiera resurser och deras beroenden. Detta kan göras med ett distributionsskript i Bicep-filen eller i en extern skriptfil. Fördelarna med detta inkluderar enkel kodning, användning, felsökning och att kunna bädda in dem i Bicep-filer. Dessutom kan man köra distributionen manuellt för att se ARM-mallen JSON som skapas. För att distribuera måste man först skapa en resursgrupp, vilket görs med kommandot az group create. Sedan kan resursen skapas med kommandot az deployment group create, följt av att bekräfta resursskapandet med en klump JSON. Slutligen kan man använda distributionsskript i Bicep för att tilldela certifikat till nyckelvalvet och lägga till användare till katalogen. Exempel på detta inkluderar att skapa ett nyckelvalv med distributionsskriptet, skapa en resursgrupp och sedan använda distributionsskriptet för att tilldela ett certifikat, samt skapa en användartilldelad hanterad identitet och tilldela bidragsgivarens roll. Det bästa sättet att använda moduldistributioner är att läsa dem från portalen, eftersom även om toppnivådistributionen misslyckas, kan utdata från framgångsrika moduler läsas såväl som parametrarna de tog. Sammanfattningsvis är Bicep ett kraftfullt och lättanvänt språk för att deklarera och distribuera Azure-resurser, vilket möjliggör skapandet av förutsägbara och konsekventa distributioner.

Sammanfattningsvis kan bicep-skript hjälpa till att effektivisera distribution och resurshantering på Azure. Genom att följa stegen och bästa praxis som beskrivs ovan kan man snabbt lära sig att skriva effektiva bicepsskript. Bicep tillhandahåller ett användbart skriptspråk för att tillåta användare att definiera Azure-resurser med en kortfattad och deklarativ syntax. Detta gör bicepsskript lättare att läsa, skriva och underhålla. Dessutom möjliggör biceps, förmågan att paketera flera resurser i en enda modul (bättre resursorganisation) och effektiv driftsättning. Slutligen tillåter användningen av variabler och parametrisering inom bicep-skript större flexibilitet och skalbarhet, vilket gör att användare snabbt kan uppdatera sina distributioner för olika miljöer och situationer. Med dessa verktyg till sitt förfogande kan användare snabbt och enkelt skapa bicep-skript som är både kraftfulla och effektiva.

# 3. Resultat

Som lösning till fig 1, har jag enligt teori och praxis skapat en main.bicep fil där modulerna finns inpaketerade. Main filen kompletteras av main parameters filen som är en JSON fil för att transmitta och spara data.

Main.bicep

targetScope = 'resourceGroup'

param resourceGroupname string

param appName string

@allowed([

'dev'

'prod'

])

param environments array

param now string = utcNow()

param location string

@allowed([

'F1'

'B1'

'S1'

'P1V2'

])

param appServicePlanSku string

// @allowed([

// 'Basic'

// 'Classic'

// 'Premium'

// 'Standard'

// ])

// param containerRegistrySku string

// @allowed([

// 'Standard'

// 'Premium'

// ])

param containerRegistryName string

module asp 'modules/asp.bicep' = {

name: '${now}-asp'

scope: resourceGroup(resourceGroupname)

params: {

location: location

aspSku: appServicePlanSku

aspName: '${appName}-asp'

}

}

@batchSize(1)

module apps 'modules/app.bicep' = [for env in environments: {

name: '${now}-${env}-app'

scope: resourceGroup(resourceGroupname)

dependsOn: [

asp

]

params: {

appName: '${appName}-${env}-app'

location: location

serverFarmId: asp.outputs.serverFarmId

}

}]

@batchSize(1)

module api 'modules/app.bicep' = [for env in environments: {

name: '${now}-${env}-api'

scope: resourceGroup(resourceGroupname)

dependsOn: [

asp

]

params: {

appName: '${appName}-${env}-api'

location: location

serverFarmId: asp.outputs.serverFarmId

}

}]

@batchSize(1)

module appSettings 'modules/appsettings.bicep' = [for i in range(0, length(environments)): {

name: '${now}-${environments[i]}-appsettings'

scope: resourceGroup(resourceGroupname)

dependsOn: [

apps

]

params: {

appName: apps[i].[outputs.name](http://outputs.name/)

geteventscr: containerRegistryName

}

}]

Main parameters   
{

"$schema": "[https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#](https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json)",

"contentVersion": "1.0.0.0",

"parameters": {

"resourceGroupname": {

"value": "get-events"

},

"containerRegistryName": {

"value": "geteventscr"

},

"appName": {

"value": "get-events"

},

"environments": {

"value": ["dev", "prod"]

},

"location": {

"value": "westeurope"

},

"appServicePlanSku": {

"value": "B1"

}

}

}

**Moduler**

ASP (App Service Plan)

// @description('ENVIRONMENT to run')

// param environment string

@description('Location that this service lands on')

param location string

@description('Provide a tier of your App Service Plan.')

param aspSku string

@minLength(5)

@maxLength(50)

@description('Provide a globally unique name of your App Service Plan')

param aspName string

resource appServicePlan 'Microsoft.Web/serverfarms@2022-03-01' = {

name: aspName

location: location

sku: {

name: aspSku

}

properties: {

reserved: true

}

kind: 'linux'

}

output serverFarmId string = appServicePlan.id

App

@description('Server farm ID')

param serverFarmId string

@description('Location that this service lands on')

param location string

@description('Provide a globally unique name of your App Service')

param appName string

param linuxFxVersion string = 'node|14-lts'

resource identity 'Microsoft.ManagedIdentity/userAssignedIdentities@2022-01-31-preview' = {

name: '${appName}-identity'

location: location

}

resource appService 'Microsoft.Web/sites@2022-03-01' = {

name: appName

location: location

properties: {

serverFarmId: serverFarmId

siteConfig: {

linuxFxVersion: linuxFxVersion

}

}

}

// output principalId string = appService.identity.principalId

output name string = appService.name

Appsettings

param appName string

param geteventscr string

resource webApp 'Microsoft.Web/sites@2022-03-01' existing = {

name: appName

}

resource cr 'Microsoft.ContainerRegistry/registries@2022-02-01-preview' existing = {

name: geteventscr

}

resource appsettings 'Microsoft.Web/sites/config@2022-03-01' = {

name: 'web'

parent: webApp

properties: {

appSettings: [

{

name: 'DOCKER\_REGISTRY\_SERVER\_URL'

value: cr.properties.loginServer

}

{

name: 'DOCKER\_REGISTRY\_SERVER\_USERNAME'

value: [cr.name](http://cr.name/)

}

{

name: 'DOCKER\_REGISTRY\_SERVER\_PASSWORD'

value: cr.listCredentials().passwords[0].value

}

]

}

}

För att deploya skriptet i terminalen på VSCode ska man skriva in följande för att sätta upp resurserna, under förutsättning att permissions finns för att ändra i denna Azure miljö.

1. az login and copy ID of GET-EVENT subscription to use in next step

2. az account set -subscription "GetIT Sandbox"

3. az group list -otable and copy NAME of resource group to run on

4. az deployment group create –resource-group ”get-events” -f main.bicep –parameters @main.parameters.json

Sendgriden går inte att skriptas. Utan man får öppna upp ett konto på Twilio Sendgrid via Azure där en API nyckel ges som kan nyttjas. Det är en SaaS som kallas Twilio Sendgrid.

# 4. Diskussion

Med hjälp av scriptet som deployades, lyckades vi sätta upp en resurs i Azure, där tre miljöer finns uppsatta. Som tidigare nämnt är dessa miljöer Getevents-dev, Getevents-prod samt container registry. Dessa kunde sättas upp på en gemensam resurs och under ett subscription. Givetvis hade vi permissions från subscribern som i detta fall Getit och är en nödvändighet för att scriptet ska fungera. Scriptet kommer att användas som en mall för framtida konsultuppdrag.

Skriptet blev även en del av arbetet för övriga studenter som hade sin LIA på Getit. Deras uppgift var att göra en hemsida där mailutskick samt mötesbokningssystem var tillgängligt. Infrastrukturen som sattes blev en del av hemsidan. Detta blev en del av den arkitektur som Getit kom att använda för framtida kunder.

Med hjälp av min handledare som formade arkitekturen, fick vi justera antalet resurser för att vara kostnadseffektiva, men även driftsäkra. Av ekonomiska skäl avstod vi för att göra infrastrukturen redundant till fler regioner, och nöjde oss med de zoner en region Azure erbjuder. Beroende på kund och budget kan detta justeras i berörda filer.

# 5. Slutsatser och rekommendationer

Slutsatsen av detta arbete visar på den betydande rollen som Bicep-skript spelar i att effektivisera distribution och hantering av resurser på Azure-plattformen. Genom att använda Bicep-skript kan enkla och deklarativa definitioner av Azure-resurser skapas, vilket gör det möjligt för utvecklare att automatisera processer, minimera fel och uppnå konsekventa resultat.

Genom att skapa en template med Bicep-skript för att sätta upp resurser i Azure för olika miljöer (som utveckling och produktion) har detta arbete visat på hur snabbt och effektivt en infrastruktur kan implementeras och hanteras. Genom att använda Bicep-skript kan utvecklare snabbt skapa och distribuera resurser, vilket sparar tid, pengar och resurser för företaget.

Att använda Bicep möjliggör även enkel resurshantering och optimering, samtidigt som det erbjuder flexibilitet och skalbarhet genom användning av variabler och parametrisering. Dessutom möjliggör Bicep skriptorganisering av resurser i moduler, vilket ytterligare förbättrar överskådligheten och underhållbarheten av koden.

Slutligen, med dess användarvänliga syntax och kraftfulla funktioner, är Bicep ett värdefullt verktyg för att effektivisera Azure-distributioner och möjliggör en smidigare och mer effektiv driftsättning av molninfrastruktur.

# 6. Referenslista

***Internet***

[1] Azure Resource Manager overview [Microsoft Azure Documentation]. (n.d.). Retrieved October 4, 2021, from <https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview>

[2] Bicep overview [Microsoft Azure Documentation]. (n.d.). Retrieved October 4, 2021, from <https://docs.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview>

[3] Resource Groups overview [Microsoft Azure Documentation]. (n.d.). Retrieved October 4, 2021, from <https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/manage-resource-groups-portal>

[4] Resource groups: Simplify resource management in Azure [Microsoft Azure Documentation]. (n.d.). Retrieved October 6, 2021, from <https://azure.microsoft.com/en-us/resources/videos/resource-groups-simplify-resource-management-in-azure/>

[5] Azure Resource Manager templates overview [Microsoft Azure Documentation]. (n.d.). Retrieved October 6, 2021, from <https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/overview>

[6] Best practices for managing Azure resources [Microsoft Azure Documentation]. (n.d.). Retrieved October 6, 2021, from <https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-management-best-practices>

Om main o modules

<https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/file>

APA

<https://www.zippia.com/advice/cloud-adoption-statistics/>

<https://www.direktronik.se/kunskapsbanken/datacenter-och-serverhallar/datamoln/>