

Open Source North

05.24.2022 | St. Paul, MN

ICF B – 9:00 am

laC Azure - Bakeoff

Mike Benkovich

@mbenko | mike@benko.com | www.benkoTips.com

Mike Benkovich

- **Developer** tools evangelist
- Cloud **Architect** & **Consultant**
- Live in **Chanhassen**
- Follow **@mbenko** on **Twitter**
- Blog www.benkoTIPS.com
- Owner of **Imagine Technologies**, Inc.
- Developing Courses for **LinkedIn** Learning
- Founded **TechMasters** – Toastmasters for Geeks
- Send me **Feedback!** mike@benko.com



Mike Benkovich

Enterprise Cloud Architect,
Consultant, Developer Tools Ev...



Today

Infrastructure as Code and Azure
Hello ARM and Bicep
Hello Terraform
Considerations

An Application is an Idea...

Data

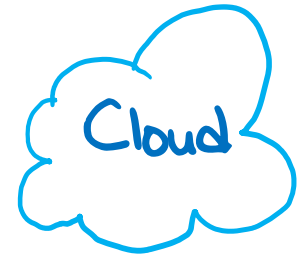
Code

+ Infrastructure

= Application



Cloud Application

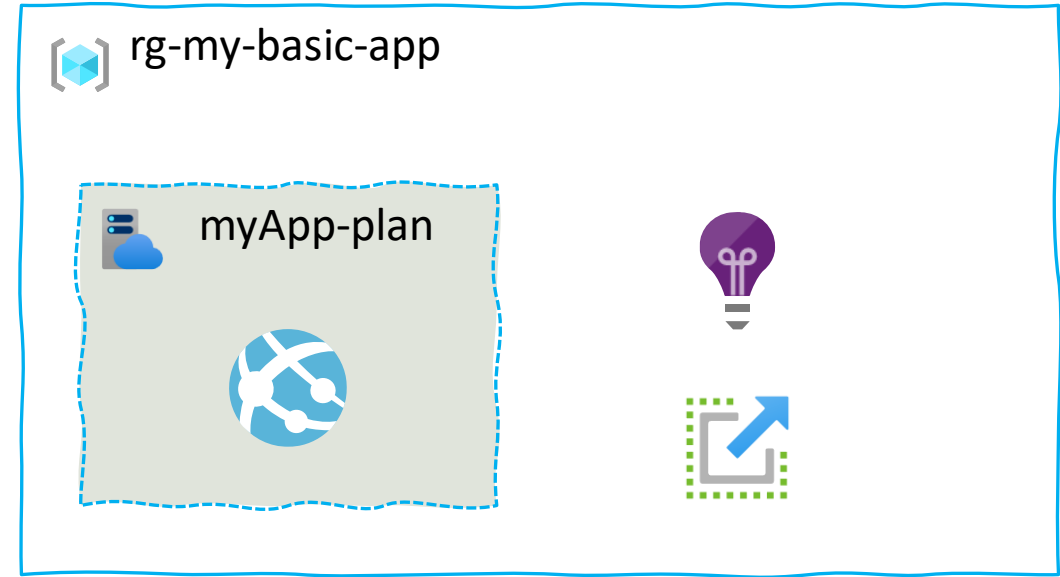


- Runs in a cloud datacenter
- Virtualized hardware
- Monitored
- Configurable
- Scalable

Basic Infrastructure

Simple web app

- Resource Group
- Hosting Plan
- App Svc
- Insights
- Autoscale rule



Questions?

What should I use?

What tools do I need?

Cross Cloud support or on-prem?

Who manages the state?

Learning curve?

Readability of the language?

Immediate support for new features?

Does it support modules?

How does it do DevOps processes?

Azure Deployment Options

Azure Native

- Azure Portal
- Azure PowerShell
- Azure CLI
- ARM Templates
- Bicep

Cross Cloud (3rd Party)

- Terraform
- Pulumi
- Ansible
- Chef
- and more...

Infrastructure as Code (IaC)

Code

Versioned & managed

Repeatable

Descriptive vs Procedural

Environment drift

Idempotent

Current

```
main.bicep M x
deploy > Bicep > main.bicep > ...
1 | @description('Specifies the location for resources.')
2 | param location string = 'centralus'
3 |
4 | param appName string
5 | param envName string
6 | param color string
7 | @secure()
8 | param secretValue string
9 |
10 | targetScope = 'subscription'
11 |
12 | resource rg 'Microsoft.Resources/resourceGroups@2021-04-01' = {
13 |   name: 'bnk-${appName}-${envName}-rg'
14 |   location: location
15 | }
16 |
17 | module site 'mywebsite.bicep' = {
18 |   scope: resourceGroup(rg.name)
19 |   name: deployment().name
20 |   params: {
21 |     appName: appName
22 |     envName: envName
23 |     secretValue: secretValue
24 |     color: color
25 |   }
26 | }
27 |
28 | output rgName string = rg.name
```


ARM



ARM Template

- A declarative way to work with a resource provider
- Includes one or more resources
- Provides configuration information
- Each resource is translated into the REST call

Template Structure

```
{  
  "$schema": "https://schema.management.azure.com/schemas/20...  
  "contentVersion": "1.0.0.0",  
  "parameters": {  },  
  "variables": {  },  
  "resources": [  ],  
  "outputs": {  }  
}
```

Resource section

```
"resources": [  
  {  
    "name": "[parameters('storageName')]",  
    "type": "Microsoft.Storage/storageAccounts",  
    "location": "[resourceGroup().location]",  
    "apiVersion": "2016-01-01",  
    "sku": {...},  
    "dependsOn": [...],  
    "tags": {...},  
    "kind": "Storage"  
  } ... ]
```

Tools for ARM

Visual Studio – Resource Group Project

VS Code – ARM Extension

Azure Portal

GitHub

ARM Summary

Native to Azure

View deployments in Azure Portal

Verbose

Variables enable naming standards

Parameters ease testing across environments

Bicep



Project Bicep

- ARM Transpiler, generates ARM as output
- Simpler syntax reduces complexity of ARM
- Modularity
- Support for all resource types and API versions
- A domain-specific-language for Azure
- No state or state files to manage
- No cost, open source

Bicep File

```
targetScope = '<scope>'

@<decorator>(<argument>)
param <parameter-name> <parameter-data-type> = <default-value>

var <variable-name> = <variable-value>

resource <resource-symbolic-name> '<resource-type>@<api-version>' = {
  <resource-properties>
}

module <module-symbolic-name> '<path-to-file>' = {
  name: '<linked-deployment-name>'
  params: {
    <parameter-names-and-values>
  }
}

output <output-name> <output-data-type> = <output-value>
```

Example Bicep Parameters

```
@minlength(3)
@description('Application Name')
param appName string

@allowed([
    'eus'
    'wus'
    'cus'
])
@description('Location of Data Center')
param loc string
```

Example Bicep Variables

```
var prefix = '${loc}-poc-'  
var hostName_var = '${prefix}${appName}-plan'  
var siteName_var = '${prefix}${appName}-site'
```

Example Bicep Resources

```
resource host 'Microsoft.Web/serverfarms@2021-01-15' = {  
  name: hostName  
  location: resourceGroup().location  
  sku: {  
    name: 'F1'  
  }  
}
```

Get started with Bicep

Decompile existing ARM templates

Code Extension enables snippets to simplify dev

Deployment via same calls as for ARM

Terraform



Created by Hashicorp

HCL Language

Multi-cloud

Tool for versioning infrastructure

Uses state information for execution plan

Install is simple download of the executable and run

Install/setup Terraform

Built in to the Azure Cloud Shell in the portal

Download the executable from Terraform's site, copy exe to path

Use Chocolatey installation

```
> choco install terraform
```


Workspace and Files

Create folder for workspace

Initialize terraform in folder

Associates workspace with backend

Loads necessary modules

Add template files *.tf and *.tfvar files
main.tf, vars.tf, output.tf, etc.

Terraform Providers

```
terraform {  
  required_providers {  
    azurerm = {  
      source  = "hashicorp/azurerm"  
      version = ">= 2.0"  
    }  
    ...  
  }  
  provider "azurerm" { ...  
}
```

Terraform Variables

```
variable "prefix" {  
    type = string  
    default = "dadapp"  
}
```

```
variable "src" {  
    type = list  
    default = ["azARM", "code", "bicep", "terraform"]  
}
```

Terraform Resources

```
resource "azurerm_app_service_plan" "plan" {  
  name          = "tf-${var.prefix}-plan"  
  location      = "${azurerm_resource_group.main.location}"  
  resource_group_name = "${azurerm_resource_group.main.name}"  
  
  sku {  
    tier = "Free"  
    size = "F1"  
  }  
}
```

Terraform Commands

init	Initializes the environment
plan	Compares the template to the saved state and shows what will change if applied
apply	Runs the template
destroy	Removes what was created

Terraform Summary

HCL Language less noisy

Cross cloud support

Environmental testing code takes some thought

State management

Secrets

Consider how you will secure your state

IaC Considerations

Learning Curve & Tools
Variables and Parameters
Preview
History
Modularity
Looping
Security
Current

IaC Comparison notes...

ARM/Bicep

- JSON or Bicep language
- Parameters & variables
- State managed by Azure natively
- can see deployments in Az Portal
- path to modules
- dynamic map data type
- module convention = 1 file
- secrets

Terraform

- HCL
- Locals, variables & TFVars
- State managed by TF*
- No History of deployments*
- need to run tf init to load modules
- map defines structure & elements
- module convention = 3 files
- secrets in plain text in state

State Management

Working with same code in different environments

Securing it is your responsibility

Updating/fixing if something goes wrong

How to import external changes

Comparison

Feature	ARM/Bicep	Terraform	Pulumi
Language	JSON + Bicep	HCL/DSL	Code Native, e.g. JavaScript, Python, C#...
Clouds	Azure only	Agnostic + on-prem	Agnostic + on-prem
State Files	Uses Azure Resource Manager natively	Plain-text	Encrypted
Naming standards	Variables & Parameters	Locals	Language native
Environments	Parameter files	Folder structure	Stacks
Preview Changes	az deployment ... what-if	terraform plan	pulumi preview
Infrastructure Cleanup	No	terraform destroy	pulumi destroy
Deployment History	Yes – View in Portal	SCM, TF Cloud*	SCM, Pulumi Enterprise*
Code Reuse	Hosted JSON URIs	Modules + Registry	Code native packages, NPM

* refers to a premium feature from vendor, i.e. Terraform Cloud or Pulumi Enterprise

Source: <https://julie.io/writing/arm-terraform-pulumi-infra-as-code>



ARM

Overview

Azure native solution for infrastructure as code, provides idempotent declarative way to describe infrastructure shape and the ARM engine in Azure makes it so

PROS

- Native to Azure
- Works with the Portal
- Tooling is ok in VS, better in Code

CONS

- Verbose, hard to read
- Complex
- Cloud specific



Bicep

A domain specific transpiler for creating Azure ARM templates from a language that provides constructs for variables, looping, modules and scoped deployments

PROS

- Day 1 Current
- Easier to read and write thanks to tooling
- Output is ARM
- History is Azure Resource Manager native
-

CONS

- Newer to the field
- Cloud specific
-



A popular cross-cloud tool for managing infrastructure by processing templates written in HCL into calls to management APIs, keeping track of state information describing cloud resources and services, available in Open Source and paid versions

PROS

- Declarative description of infrastructure
- HCL is easier to read, less clutter
- Works in multiple Cloud providers & on prem
- Broad adoption

CONS

- State management separate from cloud
- Changing between environments
- History/visibility of deployments
- Secrets stored in clear text in state



A developer focused collection of packages and libraries that can be run from within a custom application to operate cloud APIs to create and manage infrastructure.

PROS

- Multiple languages and APIs
- Compiled into native runtime
- Developers don't have to another language
- State and secret management
-

CONS

- Vendor managed running of api's
- Need paid plan for CI/CD integration
- State is 3rd party to Azure

Conclusion

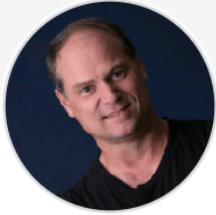
Terraform is a powerful and popular tool for IaC that supports multi-cloud deployments, but you have to be careful with your state and secrets

Bicep is always current, uses Azure Resource Manager for State, secures secrets by default, and is a powerful domain specific language for IaC with Azure with great tooling

Understanding/architecting an effective **cloud infrastructure** still requires thought, vision and execution, but you have to decide on your priorities.

References

- [Bicep and Terraform compared · Thorsten Hans' blog \(thorsten-hans.com\)](https://thorsten-hans.com)
- [Design Principles and Practices for Terraform | by Fernando Villalba | The Startup | Medium](https://medium.com/@fernando.villalba/design-principles-and-practices-for-terraform-by-fernando-villalba-the-startup-1234567890)
- [Getting started with Azure Bicep for ARM template creation \(zimmergren.net\)](https://zimmergren.net)
- [How to Create Terraform Multiple Environments \(getbetterdevops.io\)](https://getbetterdevops.io)
- [ARM Templates vs Terraform: Comparison and Fundamental Differences | Dinarys](https://dinarys.com)
- [ARM Templates vs Terraform vs Pulumi - Infrastructure as Code in 2021 | Julie Ng](https://julieng.com)



Mike Benkovich

Enterprise Cloud Architect,
Consultant, Developer Tools Ev...



Call to Action...

Where can I get more info?

Give me feedback on LinkedIn
(Scan the QR Code to the left)

Visit my blog www.benkotips.com

Azure Office Hour Fridays!
<https://bit.ly/BnkAzHrs>

Try it out with **low hanging fruit**

Takeaways from today

How to go from **Idea** to **Cloud** App

Discover **Visual Studio** tools to build **Connected** Apps

Use **Resource Group** projects to manage cloud **access**!

Enable Continuous **Value** with **DevOps**

Blueprints create **Landing Zones** with **Cloud Governance**

The Dad App

Web Application

- Mobile friendly
- ASP.NET Core Web App
- Calls API for Joke

Configuration

Monitored

Keep it simple!

- Use existing tools
- Click to deploy
- Manage in Portal



Azure DevOps vs. Github

Azure DevOps

- Microsoft Team Services
- Enterprise focus
- Private repos by default
- One-stop-shop

Repos, Boards, Pipelines

Artifacts, Testing

Github

- Open Source favorite
- Community focus
- Public repos by default
- New capabilities
 - Actions
 - Boards



GitHub Actions for CI/CD

Triggered by changes to Code

YAML based syntax

Defines steps to build & deploy

Extendable





Azure DevOps



Boards

Manage the work of a project



Repositories

Stores your code in GIT or TFVC



Pipelines

Automation workflow to build and publish code



Test Plans

Automate the testing process



Artifacts

The output of Pipelines and Code



DevOps Server

An on-premises installation of Azure DevOps (TFS)

Azure DevOps Pipelines

A cloud **automation service** to enable building, testing and delivery of code of any language to project type

Enables **consistent execution** of build and delivery processes

Hosted and custom pipeline agents including Windows, Linux and MacOS support processing of **any** software project type

Supports **triggers** and wide variety of **approvals** and **quality** gates