Azure Bicep for Terraformers

A Featurewise Comparison

Lucas Albuquerque | 09-03-2021

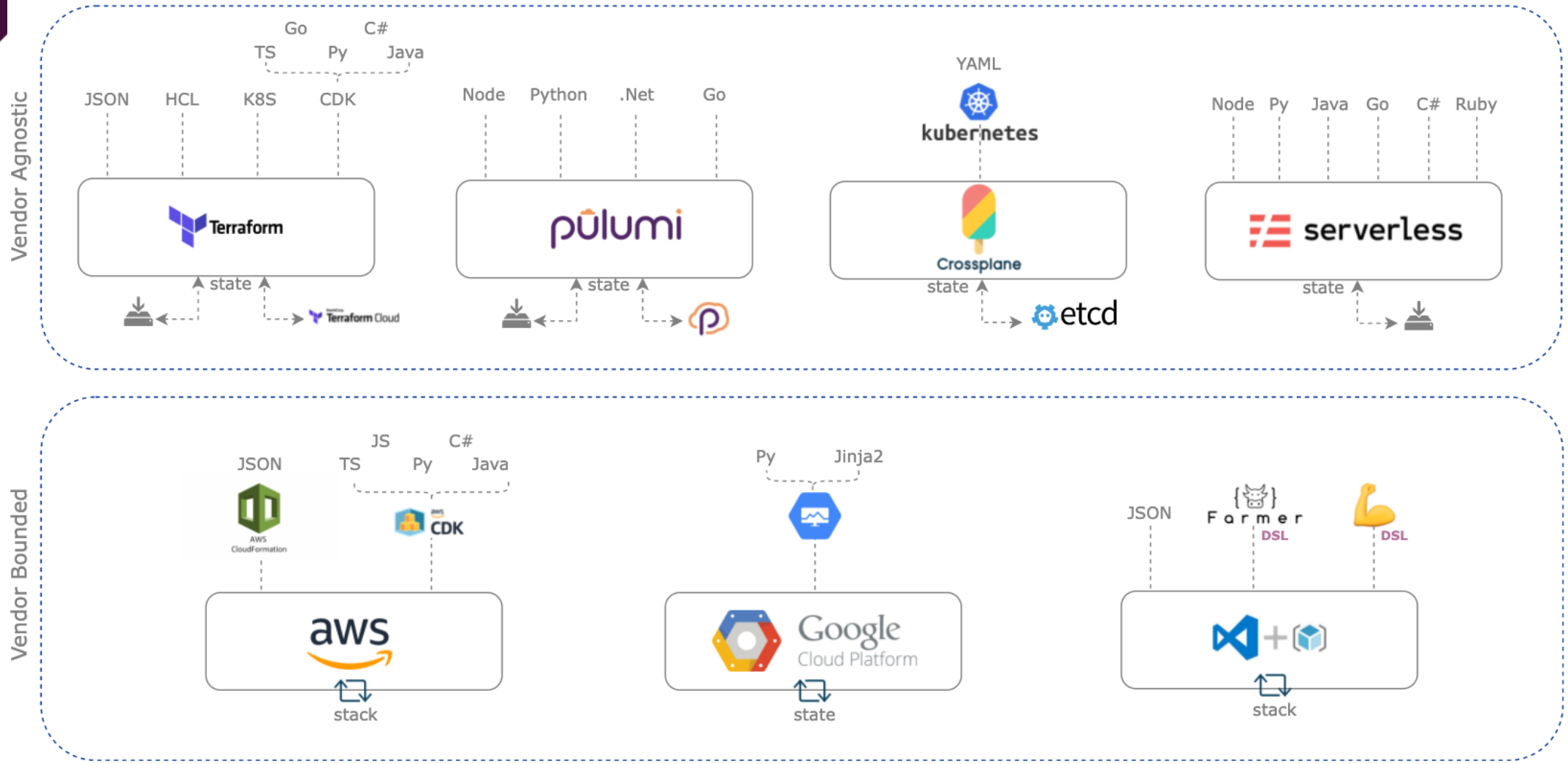
Agenda

- Motivations
- IaC Landscape
- Facts about Terraform
- Facts about Bicep
- Relevant Aspects (Lifecycle, Language, Automation, ...)
- Discussion

Motivation

- A lot of discussions about **which tool to adopt**
- **We need to set the passions aside**
- **Be open minded is paramount for the job we do**
- **Not advocating** in favor of any of them
- Present Bicep resources in **comparison** with TF in a honest way
- **Things change everyday**, hence this presentation can be outdated in one week

IaC Landscape



Facts about Terraform

- Being around **since 2014**
- **Most popular** IaC tool with a **large community**
- **Open-Source** and **Multi-Cloud/Vendor-Independent**
- Huge **Providers** Database (Cloud, On-Prem,...)
- Definitions declared using **HCL or JSON**
- **CDK** for General Purpose Programming Language Support
- **Full Lifecycle** (Create, Update and Destroy)
- **Single-binary** with **State** (Local or Remote)
- **Terraform Registry** for 3rd party **Modules**
- **Terraform Cloud** for **Governance**, **Colab** and **GitOps**

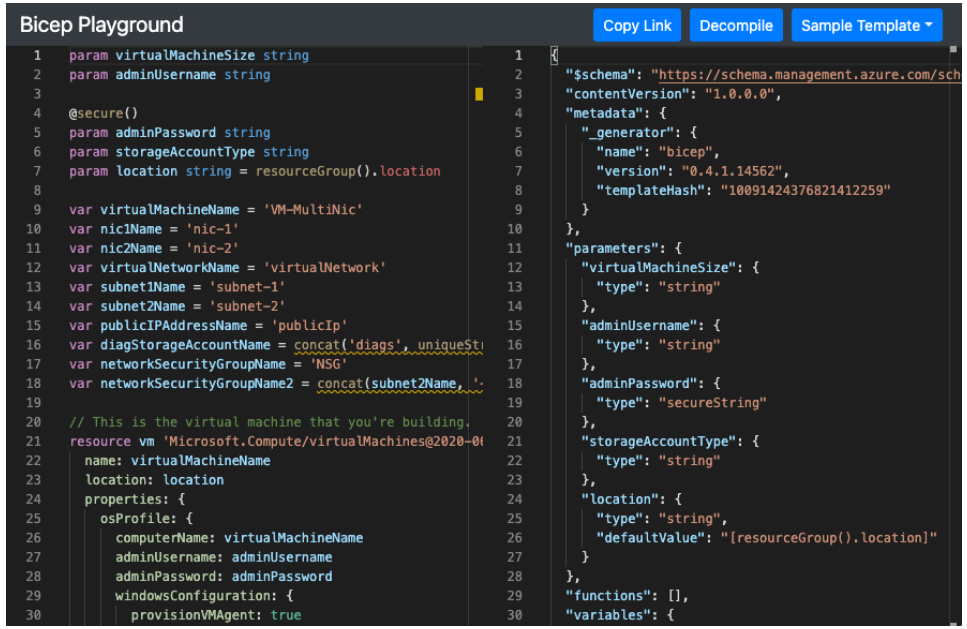
Facts about Bicep

- **New Player** in the IaC Market (but backed by Microsoft)
- **DSL wrapper** for ARM templates
- Supported by Microsoft with **100% parity with ARM**
- **Vendor-Bounded** (Only Azure)
- **Fully-integrated** with Azure Services (Az Policy, Template specs and Blueprints)
- **No integration with 3rd parties resources**
- **No state** or files to manage

Adoption Aspects

- Bicep inherits the Azure ARM community and Support
- Easy to transitioning deployed resources into code (decompile)
  - TF Painfull import (Terraformer, Terraforming, Terracognita,…)
- Bicep Playground (https://aka.ms/bicepdemo)

```
az group export --name "your_resource_group_name" > main.json
az bicep decompile --file main.json
```



Workflow and Lifecycle

- No state on Biceps
- Two Different Deployment modes: **Incremental** and **Complete**

```
az deployment group create \
--mode Complete \
--name ExampleDeployment \
--resource-group ExampleResourceGroup \
--template-file storage.json
```

- **Rollback support** (*--rollback-on-error*)
- **What-if** command
- **Doesn't** have a **destroy** clean up command

Workflow and Lifecycle

Resource Group contains:

- Resource A
- Resource B
- Resource C

Template contains:

- Resource A
- Resource B
- Resource D

When deployed in **incremental** mode,

- Resource A
- Resource B
- Resource C
- Resource D

When deployed in **complete** mode, Resource C is deleted.

- Resource A
- Resource B
- Resource D

Workflow and Lifecycle

- Best practices:
  - Define the **Deployment Scope** (RG, Subs, MG, or Tenant)… the smaller the better!

    ```
    New-AzResourceGroupDeployment -ResourceGroupName <resource-group-name> -TemplateFile <path-to-bicep>
    ```

  - Define the **Deployment Name**

    ```
    New-AzResourceGroupDeployment -Name $deploymentName -ResourceGroupName YourResourceGroup -TemplateFile
    .\appServicePlan.bicep
    ```

  - Use the **What-IF** operation to validate your Bicep template

    ```
    New-AzResourceGroupDeployment -Name $deploymentName -ResourceGroupName flowmon -TemplateFile main.bicep -c
    ```

  - **Avoid hardcoded parameters**. Use in-line or parameters file instead with @secure decorators for sensitive data

    ```
    New-AzResourceGroupDeployment `
    -Name myBicepTemplateDeployment `
    -ResourceGroupName rg-contoso `
    -TemplateFile ./main.bicep `
    -TemplateParameterFile ./main.parameters.json
    ```

Language

- Also leverage on DSL (Domain-Specific Language)

```
@minLength(3)
@maxLength(11)
param storagePrefix string

@allowed([
  'Standard_LRS'
  'Standard_GRS'
  'Standard_RAGRS'
  'Standard_ZRS'
  'Premium_LRS'
  'Premium_ZRS'
  'Standard_GZRS'
  'Standard_RAGZRS'
])
param storageSKU string = 'Standard_LRS'

param location string = resourceGroup().location

var uniqueStorageName = '${storagePrefix}${uniqueString(resourceGroup().id)}'

resource stg 'Microsoft.Storage/storageAccounts@2021-04-01' = {
  name: uniqueStorageName
  location: location
  sku: {
    name: storageSKU
  }
  kind: 'StorageV2'
  properties: {
    supportsHttpsTrafficOnly: true
  }
}

output storageEndpoint object = stg.properties.primaryEndpoints
```

Language

- Supporting **conditionals**, **extensions**, Child Resources,
- **Pull parameters** from other resources (e.g. Az KeyVault)

  az bicep publish storage.bicep --target br:exampleregistry.azurecr.io/bicep/modules/storage:v1

- Supports **Modules (from Files, Urls)** and **Private Registries** (AzContainerRegistry)

- Scopes: RG, Subs, MG
- Templates, Loops,…

```
param staNames array = [
  '4besarraystorage'
  '4besanotherstorage'
]

resource arraySta 'Microsoft.Storage/storageAccounts@2021-04-01' = [for staName in staNames: {
  name: staName
  location: resourceGroup().location
  sku: {
    name: 'Standard_LRS'
  }
  kind: 'StorageV2'

}]
```

- Linter

Automation

- General pipeline using **AzureCLI** (Azure DevOps):

```yaml
trigger:
- master

name: Deploy Bicep files

variables:
  vmImageName: 'ubuntu-latest'

  azureServiceConnection: '<your-connection-name>'
  resourceGroupName: '<your-resource-group-name>'
  location: '<your-resource-group-location>'
  templateFile: './azuredeploy.bicep'
pool:
  vmImage: $(vmImageName)

steps:
- task: AzureCLI@2
  inputs:
    azureSubscription: $(azureServiceConnection)
    scriptType: bash
    scriptLocation: inlineScript
    inlineScript: |
      az --version
      az group create --name $(resourceGroupName) --location $(location)
      az deployment group create --resource-group $(resourceGroupName) --template-file $(templateFil
```

Automation

- GitHub Actions

```yaml
on: [push]
name: Azure ARM
jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:

      # Checkout code
    - uses: actions/checkout@main

      # Log into Azure
    - uses: azure/login@v1
      with:
        creds: ${{ secrets.AZURE_CREDENTIALS }}

      # Deploy Bicep file
    - name: deploy
      uses: azure/arm-deploy@v1
      with:
        subscriptionId: ${{ secrets.AZURE_SUBSCRIPTION }}
        resourceGroupName: ${{ secrets.AZURE_RG }}
        template: ./azuredeploy.bicep
        parameters: storagePrefix=mystore
        failOnStdErr: false
```

Closing words…

“If you're happy using Terraform, there's no reason to switch. Microsoft is committed to making sure Terraform on Azure is the best it can be.”

https://docs.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview

Xebia

Thanks for Joining

Questions?

Lucas Albuquerque | 11-12-2020

Xebia