Consistent cloud environments with Infrastructure as Code

Matthew Sheehan

Planet DDS

https://github.com/MattSheehanDev/codemash2022-bicep

What is Infrastructure as Code?

- Infrastructure as Code is the process of automating the provisioning of your infrastructure.
- Enables managing your infrastructure in a descriptive way, by using scripts and configuration files instead of graphical user interfaces.
  - Azure Resource Manager (ARM)
  - AWS CloudFormation
  - Terraform

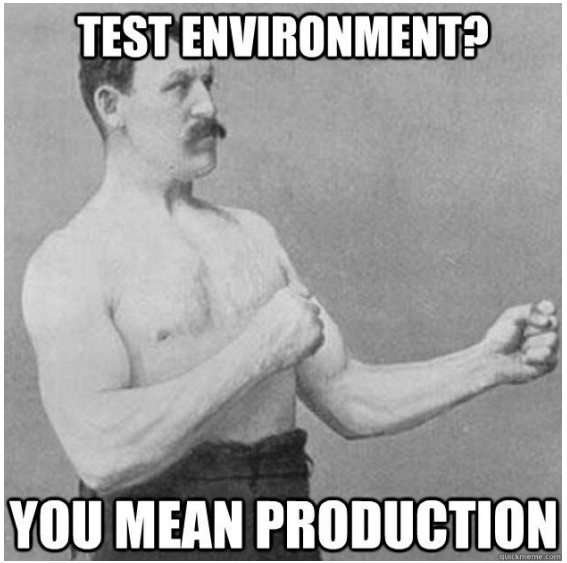
Why use Infrastructure as Code?

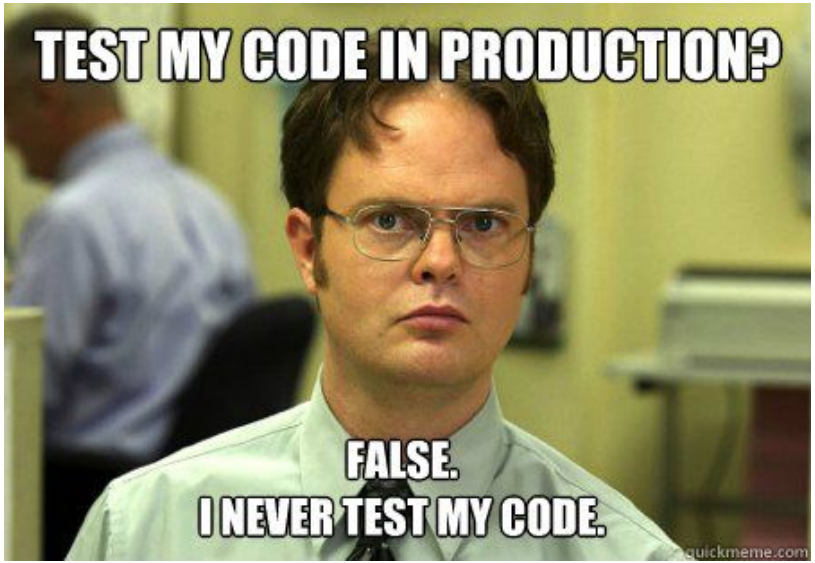
- Better understand your infrastructure and how the pieces connect.
- Allows your team to version control their infrastructure.
- Follow standard best practices such as peer reviewing to detect problems in configurations.
- Consistent deployments and avoid *configuration drift*.
- Shift access away from people to an automated process and tooling.

So this is a story all about how… my team started using bicep

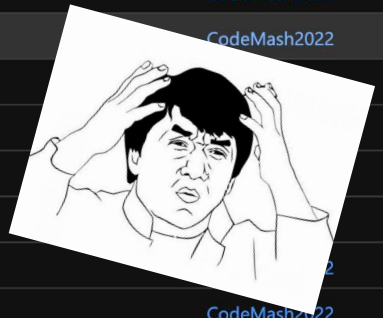
Challenges facing the team

- Resources allocated over more than half a decade.
- 404: No naming conventions found.
- Have production as our one environment for testing.

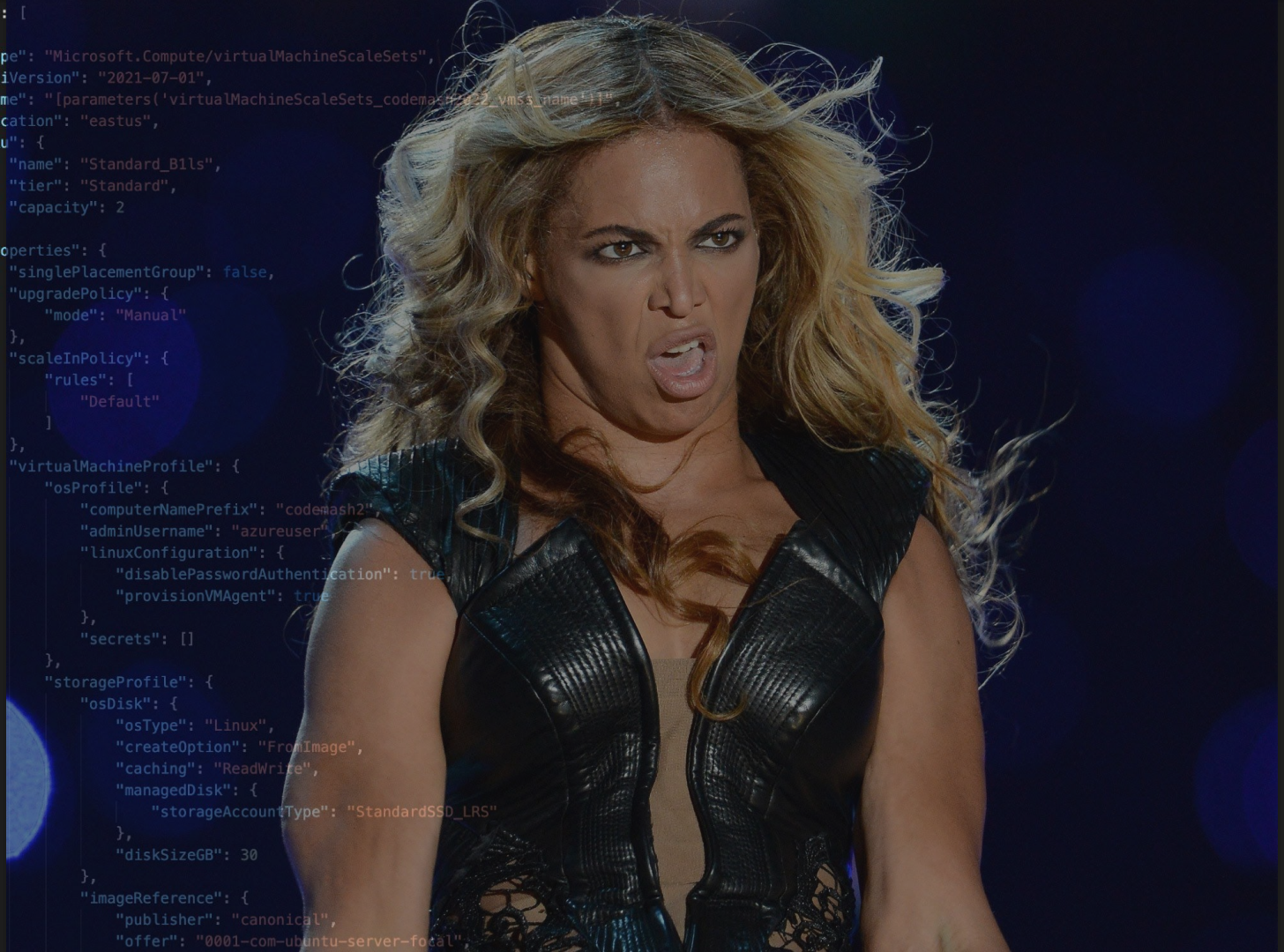




ASP-apps-96a2	App Service plan	apps	Central US	CodeMash2022
ASP-infrastructure-92e1	App Service plan	infrastructure	Central US	CodeMash2022
basicNsgvmss-mjs-test-vnet-nic01	Network security group	vmss-mjs-test	East US	
customerlist (product1/customerlist)	SQL database	infrastructure	East US	
DefaultWorkspace-1415b3bb-311b-4857-851d-42ce9c18fbfc-CUS	Log Analytics workspace	DefaultResourceGroup-CUS	Central US	
kfc1-prod	Virtual machine scale set	vmss-mjs-test	East US	
logsproduct2	Storage account	infrastructure	East US	
NetworkWatcher_eastus	Network Watcher	NetworkWatcherRG	East US	CodeMash2022
p_key	SSH key	vmss-mjs-test	East US	CodeMash2022
pool1 (product1/pool1)	SQL elastic pool	infrastructure	East US	CodeMash2022
product1	SQL server	infrastructure	East US	CodeMash2022
product2east	App Service	infrastructure	Central US	CodeMash2022
product2east	Application Insights	infrastructure	Central US	CodeMash2022
productEXT	Function App	apps	Central US	CodeMash2022
productEXT	Application Insights	apps	Central US	CodeMash2022
vmss-mjs-test-vnet	Virtual network	vmss-mjs-test	East US	CodeMash2022
ASP-apps-96a2	App Service plan	apps	Central US	CodeMash2022
ASP-infrastructure-92e1	App Service plan	infrastructure	Central US	CodeMash2022
basicNsgvmss-mjs-test-vnet-nic01	Network security group	vmss-mjs-test	East US	CodeMash2022
customerlist (product1/customerlist)	SQL database	infrastructure	East US	CodeMash2022
DefaultWorkspace-1415b3bb-311b-4857-851d-42ce9c18fbfc-CUS	Log Analytics workspace	DefaultResourceGroup-CUS	Central US	CodeMash2022
kfc1-prod	Virtual machine scale set	vmss-mjs-test	East US	CodeMash2022
logsproduct2	Storage account	infrastructure	East US	CodeMash2022
NetworkWatcher_eastus	Network Watcher	NetworkWatcherRG	East US	CodeMash2022




ARMS™



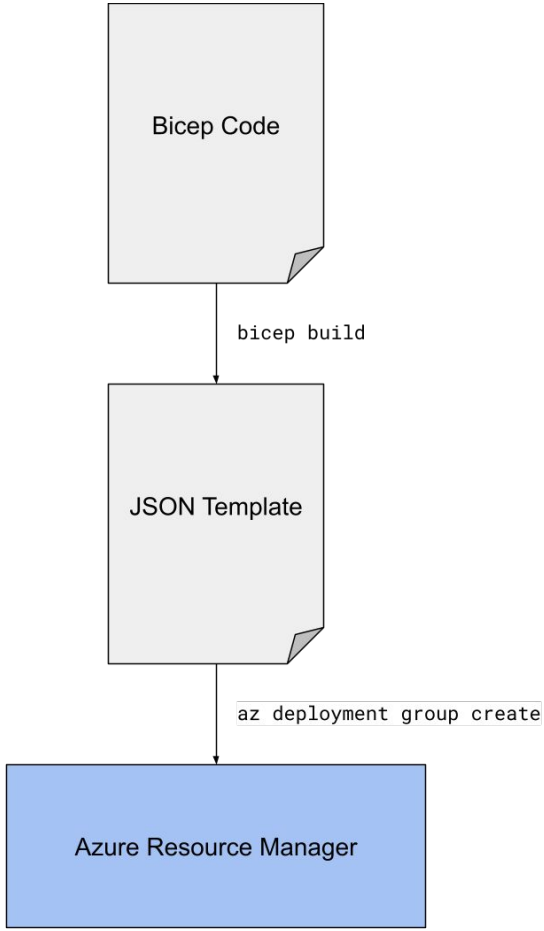
```
27      "resources": [
28          {
29              "type": "Microsoft.Compute/virtualMachineScaleSets",
30              "apiVersion": "2021-07-01",
31              "name": "[parameters('virtualMachineScaleSets_codemash-02_vmss_name')]",
32              "location": "eastus",
33              "sku": {
34                  "name": "Standard_B1ls",
35                  "tier": "Standard",
36                  "capacity": 2
37              },
38              "properties": {
39                  "singlePlacementGroup": false,
40                  "upgradePolicy": {
41                      "mode": "Manual"
42                  },
43                  "scaleInPolicy": {
44                      "rules": [
45                          "Default"
46                      ]
47                  },
48                  "virtualMachineProfile": {
49                      "osProfile": {
50                          "computerNamePrefix": "codemash2",
51                          "adminUsername": "azureuser",
52                          "linuxConfiguration": {
53                              "disablePasswordAuthentication": true,
54                              "provisionVMAgent": true
55                          },
56                          "secrets": []
57                      },
58                      "storageProfile": {
59                          "osDisk": {
60                              "osType": "Linux",
61                              "createOption": "FromImage",
62                              "caching": "ReadWrite",
63                              "managedDisk": {
64                                  "storageAccountType": "StandardSSD_LRS"
65                              },
66                              "diskSizeGB": 30
67                          },
68                          "imageReference": {
69                              "publisher": "canonical",
70                              "offer": "0001-com-ubuntu-server-focal",
```

Introducing 💪

- Bicep is a DSL abstraction over ARM templates.
- Bicep code is transpiled to standard ARM template files before being deployed.

```
az bicep build --file main.bicep
```

- Retains the core functionality and the same runtime as ARM templates.


Bicep Code
bicep build
JSON Template
az deployment group create
Azure Resource Manager

Why Bicep?

- First party support from Microsoft.
- Immediate support for new resources in preview and API versions.
- Simpler syntax and more module than JSON templates.
- Built in state management.
- Existing ARM templates can be easily converted to Bicep code.

```
az bicep decompile --file main.json
```

- Tooling already included in the Azure CLI.
- Intellisense and VS Code support.

Everyone @CodeMash · Jan 13, 2022

Weird flex but ok

Me @BiceplaC · Jan 13, 2022

- First party support from Microsoft.
- Immediate support for new resources in preview and API versions.
- Simpler than JSON.
- Existing ARM templates can be easily converted to Bicep code.

```
az bicep decompile --file main.json
```

- Tooling already included in the Azure CLI.
- Top notch VS Code support.

Is Bicep a good fit for your team?

- You are a single public cloud team that deploys to Azure. ✅
- You have an existing stash of ARM templates (or no IaC). ✅
- You really like VS Code. ✅
- You need a solution that will deploy to multiple cloud providers. ❎
- You have an existing IaC toolset 🤷

Bicep Syntax

Arm Template

```
"resources": [
  {
    "type": "Microsoft.Web/sites"
    "apiVersion": "2021-02-01",
    "name": "[variables('appname')]",
    "location": "[parameters('applocation')]",
    "kind": "app",
    ...
  }
]
```

Bicep Code

```
resource app Microsoft.Web/sites@2021-02-01' =
{
  name: appname
  location: applocation
  kind: 'app'
}
```

```
Microsoft.Compute
Microsoft.Storage
Microsoft.Sql
Microsoft.Network
Microsoft.EventHub
Microsoft.KeyVault
Microsoft.Authorization
...
```

Parameters and Variables

```
// arm.json

"parameters": {
  "applocation": {
    "defaultValue": "Central US",
    "type": "String"
  },
},
"variables": {
  "appname": "codemashapp"
}
```

```
// main.bicep

param applocation string = 'Central US'

var appname = 'codemashapp'
```

array
bool
int
string
object

References and Dependencies

```
resource servicePlan 'Microsoft.Web/serverfarms@2021-02-01' existing = {
 name: '${appname}-sp'
 ...
}

resource app 'Microsoft.Web/sites@2021-02-01' = {
 name: appname
 properties: {
   serverFarmId: servicePlan.id
 }
}
```

```
"resources": [
  {
    "type": "Microsoft.Web/sites",
    "apiVersion": "2021-02-01",
    "name": "[parameters('appname')]",
    "dependsOn": ["[resourceId('Microsoft.Web/serverfarms/', concat(parameters('appname'), '-sp'), '2021-02-01')]"],
    "properties": {
      "serverFarmId": "[resourceId('Microsoft.Web/serverfarms/', concat(parameters('appname'), '-sp'), '2021-02-01')]"
      …
    }
  }
]
```

Parent and Child Resources

```
// Parent and child resource
resource server 'Microsoft.Sql/servers@2021-05-01' = {
  name: '${env}-cm-sqlserver'
  properties: {
    ...
}

resource db 'Microsoft.Sql/servers/databases@2021-05-01' = {
  parent: server
  name: 'codemash-db'
  ...
}
```

```
// Parent and nested child resource
resource server 'Microsoft.Sql/servers@2021-05-01' = {
  name: '${env}-cm-sqlserver'
  properties: {
    ...
  }

  resource db 'databases' = {
    name: 'codemash-db'
    ...
  }
}


// Can be accessed with :: notation
sqlServer::sqlDb
```

Bicep Modules

```
// sql.bicep

resource sqlServer 'Microsoft.Sql/servers@2014-04-01' ={
  name: sqlServerName
  location: resourceGroup().location
}
...
```

```
// main.bicep

module sql './sql.bicep' = {
  name: 'sqlDeploy'
  params: {
    ...
  }
}
```

Module Outputs

```
// module1.bicep

resource appService 'Microsoft.Web/sites@2021-02-01' = {
  name: 'my-app-name'
  location: resourceGroup().location
  properties: {
    ...
  }
}

output appName string = appService.name
```

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

```
// module2.bicep

module module1 'module1.bicep' = {
  name: 'moduleDeploy'
  params: {
    ...
  }
}

resource app 'Microsoft.Web/sites@2021-02-01' existing = {
  name: module1.outputs.appName
}
```

Module Outputs

- Answer the question, “How do I share information from a module?”.
  - It’s a good idea to output information a parent template might need to use.
- Outputs are logged as part of the resource group deployment history.
  - It’s a bad idea to output values that should remain a secret (keys, connection strings).
- Consider alternatives for secret values,
  - Output the resource name and the parent template can lookup the resource and properties using the `existing` resource definition.
  - Write the secret to an Azure Key Vault secret and look up the secret from the vault when needed.

Parameter decorators

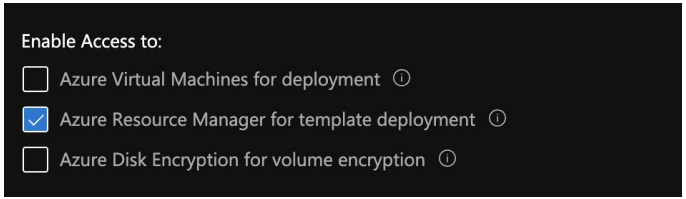
```
@description('Must be at least S1 tier to support VNet integration')
param appSku string = 'S1'


@allowed([
 'dev'
 'test'
 'prod'
])
param env string


@secure()
param password string
```

```
allowed
secure
minLength / maxLength
minValue / maxValue
description
metadata
```

Working with `@secure` secrets

```
// main.bicep

resource kv 'Microsoft.KeyVault/vaults@2019-09-01' existing = {
 scope: resourceGroup('kv-bicep-rg')
 name: 'kv-bicep-deployment'
}

module sql './sql.bicep' = {
 name: 'sqlDeploy'
 params: {
   username: 'codemash'
   password: kv.getSecret('sql-password')
 }
}
```

```
// sql.bicep

param username string

@secure()
param password string


resource sqlServer 'Microsoft.Sql/servers@2014-04-01' = {
 name: sqlServerName
 location: resourceGroup().location
 properties: {
   administratorLogin: username
   administratorLoginPassword: password
 }
}
```



Deployment scope

```
"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#"
```

```
"$schema": "https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#"
```

```
"$schema": "https://schema.management.azure.com/schemas/2019-08-01/managementGroupDeploymentTemplate.json#"
```

```
"$schema": "https://schema.management.azure.com/schemas/2019-08-01/tenantDeploymentTemplate.json#"
```

```
targetScope = 'resourceGroup'
targetScope = 'subscription'
targetScope = 'managementGroup'
targetScope = 'tenant'
```

Deployments

```
az deployment group create -f <path-to-bicep> -g <resource-group-name> --parameters ./parameters.json
```

```
az deployment sub create -f <path-to-bicep> --location <location> --parameters ./parameters.json
```

Putting it together


QUESTION.
DOES ANYONE HAVE ANY QUESTIONS?


THANK
YOU