ARM Infrastructure as code

Damian Flynn

What are ARM Templates?

- They are Azure's *Infrastructure as Code*
- They are *declarative*
- They are *idempotent*
- They allow deployment of complex resources
- They can be nested (call other templates)
- You can do clever stuff with sequencing
- They are part of a healthy balanced DevOps diet
- They are authored in ***J**ava **S**cript **O**bject **N**otation*

Quickly deploying through Azure Portal

Demo

Parameters

```
"parameters": {
    "NamePrefix": {
      "type": "string",
      "minLength": 1,
      "defaultValue": "Env1"
      "metadata": {
        "description": "Used to construct all resource names as prefix+service.
Should be all lowercase to avoid naming issues"
      }
    }
  }
```

Variables

```
"variables": {
    "WebSiteName": "[concat(parameters('NamePrefix'),'WebSite')]",
    "AppPlanName":
"[concat(parameters('NamePrefix'),'AppPlan',uniqueString(resourceGroup().id))]",
    "RedisCache": {
      "Name":
"[concat(parameters('NamePrefix'),'Redis',uniqueString(resourceGroup().id))]",
      "SKUName": "Basic",
      "SKUFamily": "C",
      "SKUCapacity": 0
    }
  }
```

Functions

```
"functions": [
  {
    "namespace": "sampleFunction",
    "members": {
      "getNicName": {
        "parameters": [
          { "name": "prefix",
            "type": "string"
          },
          { "name": "id",
            "type": "string"
          }
        ],
        "output": {
          "type": "string",
          "value": "[concat(toLower(parameters('prefix')),'-nic-',padLeft(parameters('id'),2,'0'))]"
        }
      }
    }
  }
]
```

Resources

```
"resources": [
    {
      "name": "[variables('AppPlanName')]",
      "type": "Microsoft.Web/serverfarms",
      "location": "[resourceGroup().location]",
      "apiVersion": "2014-06-01",
      "dependsOn": [],
      "tags": {
        "displayName": "AppPlan"
      },
      "properties": {
        "name": "[variables('AppPlanName')]",
        "sku": "[parameters('AppPlanSKU')]",
        "workerSize": "[parameters('AppPlanWorkerSize')]",
        "numberOfWorkers": 1
      }
    }
]
```

Outputs

```
 "outputs": {
    "WebsiteHostName": {
      "value":
"[reference(resourceId('Microsoft.Web/sites',variables('WebSiteName'))).hostNames[
0]]",
      "type": "string"
    }
  }
```

Template Tips

- Small numbers of parameters
- Make use of objects in variables
- If your template is huge, consider splitting
- Use clear naming, especially if using references and outputs
- Avoid Parameters.json
- Accept the azuredeploy.json naming
- Know and love Resource Explorer

Automation Script and Resource Explorer

Demo

Getting Clever: Copy

- Copy loops allow you to create multiple copies of a given resource with unique properties
  e.g. The same web site in multiple regions

```
"apiVersion": "2015-08-01",
"name": "[concat(variables('WebSiteName'), copyIndex(),'/staging')]",
"copy": {
        "count": "[length(parameters('WebLocations'))]",
        "name": "WebSlotLoop"
},
"type": "Microsoft.Web/sites/slots",
"location": "[parameters('WebLocations')[copyIndex()]]",
"dependsOn": ["WebAppLoop"],
"properties": {}
```

Getting Clever: Sequencing

01

A resource can only be defined once in a deployment

02

Nested deploys allow you to define it more than once

03

Dependencies and references allow you to *redefine* resource properties
e.g. change the DNS value of vNet based on the IP of a new VM

Getting Clever: Condition, If

- Pass different JSON into the final template based on logic

```
"apiVersion": "2017-06-01",
      "condition": "[greater(length(parameters('WebLocations')),1)]",
      "type": "Microsoft.Network/trafficManagerProfiles",
      "name": "[variables('TrafficMgr').Name]“

"TrafficMgrFQDN": {
      "type": "string",
      "value":
"if(greater(length(parameters('WebLocations')),1),[reference(resourceId('Microsoft.
Network/trafficManagerProfiles', variables('TrafficMgr').Name)).dnsConfig.fqdn,
json('null'))]"
    }
```

Getting Clever: Undocumented Stuff

Automation Script doesn’t give you everything

Not everything is fully documented (yet)

Resource Explorer is your friend

Parameter Usage

👌 `parameters` accept inputs

👍 `parameters` pass data to `variables`

👍 `variables` transform/isolate `parameters`

👎 ~~`parameters` used directly with `resources`~~

👍 `variables` used directly with `resources`

```
{
  "$schema": "...",
  "contentVersion": "...",
  "parameters": {},
  "variables": {},
  "functions": [],
  "resources": [],
  "outputs": {}
}
```

ARM Functions

- Simplifies complex expressions
- Reusable within the template

```
"functions": [
  {
    "namespace": "sampleFunction",
    "members": {
      "getName": {
        "parameters": [
          { "name": "prefix",
            "type": "string"
          },
          { "name": "id",
            "type": "string"
          }
        ],
        "output": {
          "type": "string",
          "value": "[concat(toLower(parameters('prefix')),'-nic-',padLeft(parameters('id'),2,'0'))]"
        }
      }
    }
  }
]
```

ARM Functions Usage

- Simplifies complex expressions
- Reusable within the template

```
"variables": {
 "fullname": "[concat(parameters('prefix'), '-{0}-', parameters('location'))]",
  "nickname": "[sampleFunction.getName(variables('fullname'), 'nic')]",
  "nsgName": "[sampleFunction.getName(variables('fullname'), 'nsg')]",
}
```

ARM Outputs

- Visualize the results of the deployments
- Exposes Data to be consumed by next stage efforts

```
"outputs": {
    "websiteHostName": {
      "value": "[reference(resourceId('Microsoft.Web/sites',variables('WebSiteName'))).hostNames[0]]",
      "type": "string"
    },
    "storageAccountKey": {
      "value": "[listKeys('resourceId', 'apiVersion').keys[0].value]",
      "type": "string"
    },
    "appInsightsKey": {
      "value": "[reference('resourceName').instrumentationKey]",
      "type": "string"
    }
}
```

Complex Templates

Demo

Authoring Considerations

Consider Yet Another Markup Language

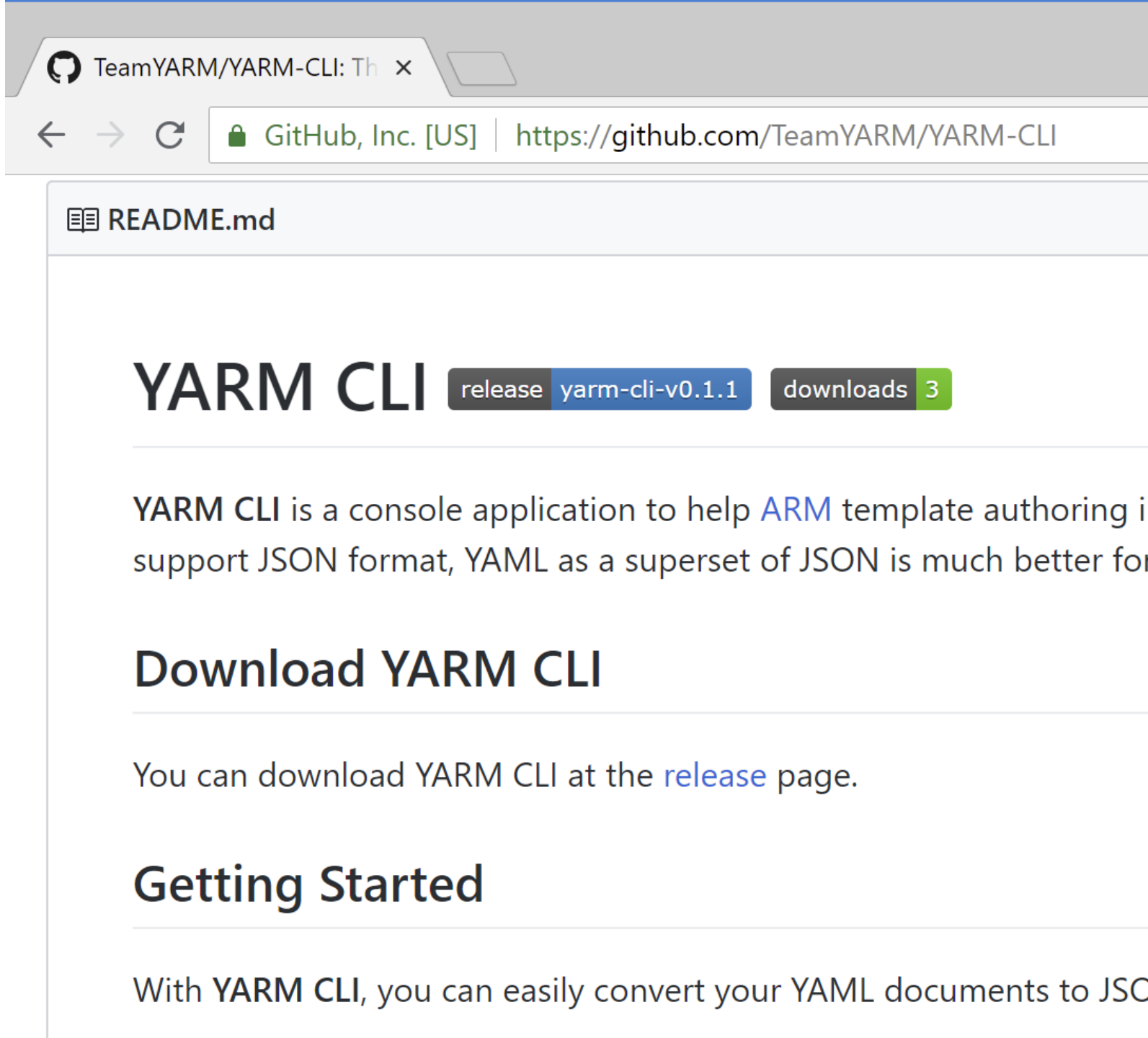
- Author in YAML
- Superset of JSON
- Python Pretty Code Compliant
  - Delimited using Spaces
  - No complex nest of braces {[(..]})
  - Supports Comments
  - Much easier read
  - Easily convertible to and from JSON/YAML

Use in Deployment Automation

- Avoid parameters.json in code repo
- PowerShell supports inferred parameters
- Store your parameter values in your release pipeline
- Use the same template for dev, test, release

YAML for ARM

- CLI Interface
- Cross Platform
- Pipeline Compliant



Useful Links

Reference
https://github.com/rjmax

Azure QuickStart Templates
https://github.com/Azure/azure-quickstart-templates
http://azure.microsoft.com/en-us/documentation/templates/

ARM Template Documentation
https://docs.microsoft.com/en-us/azure/templates/

Azure Resource Explorer
https://resources.azure.com/

Thank You!

Gold

itnetX

cireson

Hewlett Packard Enterprise

ACP

StarWind

Silver

HYCU

INOVIT

SquaredUp

MPexperts

NiCE EVIEW TECHNOLOGY

Advanced Enterprise Management Packs for Today's Leading IT Platforms

DERDACK SIGNL4

adaptiva

Callisto from Apajove

Conference Partner

Microsoft