

Azure Developer Series

Application Migration to Azure

Peter De Tender

CEO & Lead Technical Trainer at
007FFFLearning.com

@pdtit

@007FFFLearning

April 2019

About Me...

Peter De Tender – MCT, Azure MVP

☁ CEO and Lead Technical Trainer of 007FFFlearning.com,
+20 years IT experience, mainly datacenters and
Microsoft Infrastructure background

☁ Full-time in Azure since 2013 (Readiness & Architect)

☁ Azure Advisor, Azure Certified Architect

☁ Technical Writer, Book author, Courseware Creator

☁ Living in Belgium, but traveling worldwide
90% of my time, helping larger Microsoft Partners,
customers and Microsoft FTEs in learning about and
using Azure, by providing workshops with passion



peter@pdtit.be

@pdtit @007FFFlearning

<http://www.facebook.com/pdtit>

<http://www.linkedin.com/in/pdtit>

Application Migration

Session 2: Infrastructure as Code

Deploying Azure Resources using ARM templates

Peter De Tender

@pdtit

@007FFFlearning

Key Objectives

What you will learn in this section

- What is Infrastructure as Code
- Azure Resource Manager – Resource Providers
- Azure Quickstart Templates on GitHub
- Authoring ARM templates with Visual Studio / Visual Studio Code
- Beyond ARM Templates

Setting the scene



Overview of the workshop

About the workshop content...

About:

In this workshop, you will learn how to build a proof of concept (POC) that will transform an existing ASP.NET-based Web application to a container-based application. This POC will deliver a multi-tiered web app solution from a Virtual Machine architecture into Azure, leveraging Azure WebApps and different Azure container solutions available today. You will also migrate the underlying database from a SQL 2014 Virtual Machine architecture to SQL Azure. **Easter Bonus: Every now and then, we will showcase similar steps using a Node.JS and MongoDB, migrating to Azure Web Apps, Containers and CosmosDB.**

At the end of this workshop, you will have a good understanding of container concepts, Docker architecture and operations, Azure Container Services, Azure Kubernetes Services and SQL Azure PaaS solutioning.

Target Audience:

The workshop is targeted to Cloud Architects, Cloud Solution designers, developers and IT sysadmins, CIO's, CTO's and anybody else who is interested in learning about Azure, containers, application cloud migration and digital transformation.

Focus of the workshop (40%) is getting hands-on experience, complemented with presentations and whiteboard sessions (if in-person delivery).

Time Estimate:

16 hours (+/- 10 hours presentations, 6 hours of optional hands-on labs for attendees)

Workshop Agenda - Presentations

What we will talk about...

- Module 1: Digital App Transformation with Azure
- Module 2: Infrastructure as Code using ARM templates
- Module 3: Azure Database Solutions – SQL Azure (+ Azure Cosmos DB)
- Module 4: Azure App Services – Azure Web Apps (.NET) (+ Node.JS)
- Module 5: Introduction to Docker
- Module 6: Deploying Azure Container Registry / Azure Container Instance
- Module 7: Migrating Apps to Azure Container Services / Kubernetes Services
- Module 8: ACS / AKS Management and Monitoring

Workshop Agenda – Hands-On-Labs

Learn by doing...

- **Module 2: Infrastructure as Code using ARM templates**
 - **Lab 1:** Setup your Azure subscription and deploy the source Virtual Machine environment with Visual Studio 2017
- **Module 3: Azure Database Solutions – SQL Azure**
 - **Lab 2:** Migrating a SQL VM database to SQL Azure using SQL Management Studio
- **Module 4: Azure App Services – Azure Web Apps**
 - **Lab 3:** Migrating your legacy ASP.NET application to Azure Web Apps with Visual Studio 2017
- **Module 5: Introduction to Docker**
 - **Lab 4:** Containerizing your legacy ASP.NET application with Docker CE for Windows

Workshop Agenda – Hands-On-Labs

Learn by doing...

- **Module 6: Deploying Azure Container Registry / Azure Container Instance**
 - **Lab 5:** Using Azure Container Registry, Azure Container Instance
- **Module 7: Migrating Apps to Azure Container Services / Kubernetes Services**
 - **Lab 6:** Deploying Azure Container Services with Kubernetes and running Pods
 - **Lab 7:** Deploying Azure Kubernetes Services
- **Module 8: ACS / AKS Monitoring and Operations**
 - **Lab 8:** Integrating ACS monitoring with Azure Monitor and Deploying Kubernetes Dashboard

Node.JS and Cosmos DB labs are available on request

Technical Requirements

What you need...

<Could vary based on the actual delivery-method>, but overall:

- Client workstation running recent Windows, Linux or Mac OS and latest internet browser
- Access to ports 80 (HTTP), 443 (HTTPS) and 3389 (Remote Desktop)
- Full Azure subscription (MSDN, AzurePass, Paid subscription, AE, CSP,...)
Note: Azure trial subscription doesn't work for all required lab steps)
- Lab consumption estimate: \$15-35

Questions and HOL support

msdevseriesupport@007FFFLearning.com

Subject: Azure Developer Series – Containers

Response Time: within 4-8 hours

Check GitHub for FAQ and Updates:

<http://www.github.com/007FFFLearning/MSDevSeriesSupport>

Infrastructure as Code (IAC)



What is Infrastructure As Code (IAC)

[http://en.Wikipedia
.org/wiki/
Infrastructure_as_
Code](http://en.Wikipedia.org/wiki/Infrastructure_as_Code)

- The process of managing and provisioning computer data centers through machine-readable definition files.
- The deployment can use either scripts or declarative definitions, rather than manual processes.
- While the terminology points to Infrastructure, it should not be confused with Infrastructure as a Service (IaaS), as IAC also allows you to deploy other (cloud) components like Platform as a Service (PaaS)

Infrastructure As Code (IAC) - Values

1. FASTER EXECUTION

- Build once – Deploy many
- Minor changes required to deploy different models
- Deployment of compute resources goes faster than manual provisioning

2. REDUCE RISK

- Remove Errors and Mistakes
- Remove Security Violations
- Overwrite changes without touching existing deployment state

3. REDUCE COST

- Manual labor is expensive
- Easily scale out your environment
- Portable across different environments (dev/test/staging/different organizations)

4. INTEGRATION WITH DEVOPS

- Infrastructure as Code provides an integration with DevOps concepts and processes within an organization
- Allows for end-to-end application landscape provisioning, not just 'infrastructure'

Infrastructure As Code (IAC) - Methods

DECLARATIVE (FUNCTIONAL) "WHAT"

- The final state of the system / environment is defined (declared), in such a way that it defines in "what" state it should be.
- When the process is run, it will configure the system/environment to have the declared state as the final result.

IMPERATIVE (PROCEDURAL) "HOW"

- The automation code used to setup or configure the system / environment is written in such a way, it goes through each configuration step-by-step.
- Automation code built in this way, defines the process of "how" the system is to be configured and what steps need to be taken, in the exact order, to obtain the final result state.



A
U
T
O
M
A
T
I
O
N

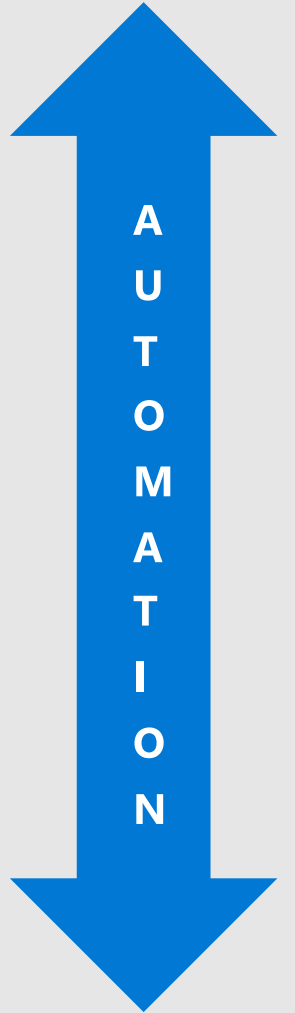
Infrastructure As Code (IAC) – Configuration Options

CONFIGURATION ORCHESTRATION

- Designed and used to automate the deployment of servers and other related (cloud) infrastructure
- Some tools have overlap with configuration management

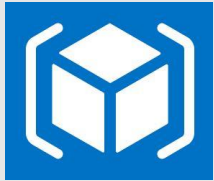
CONFIGURATION MANAGEMENT

- Designed and used to automate the configuration of systems and software on top of the infrastructure which has already been provisioned (out of configuration orchestration)
- Some tools have overlap with configuration Orchestration



Infrastructure as Code – Tools (Azure Popular)

CONFIGURATION ORCHESTRATION



HashiCorp
Terraform



ANSIBLE

CONFIGURATION MANAGEMENT

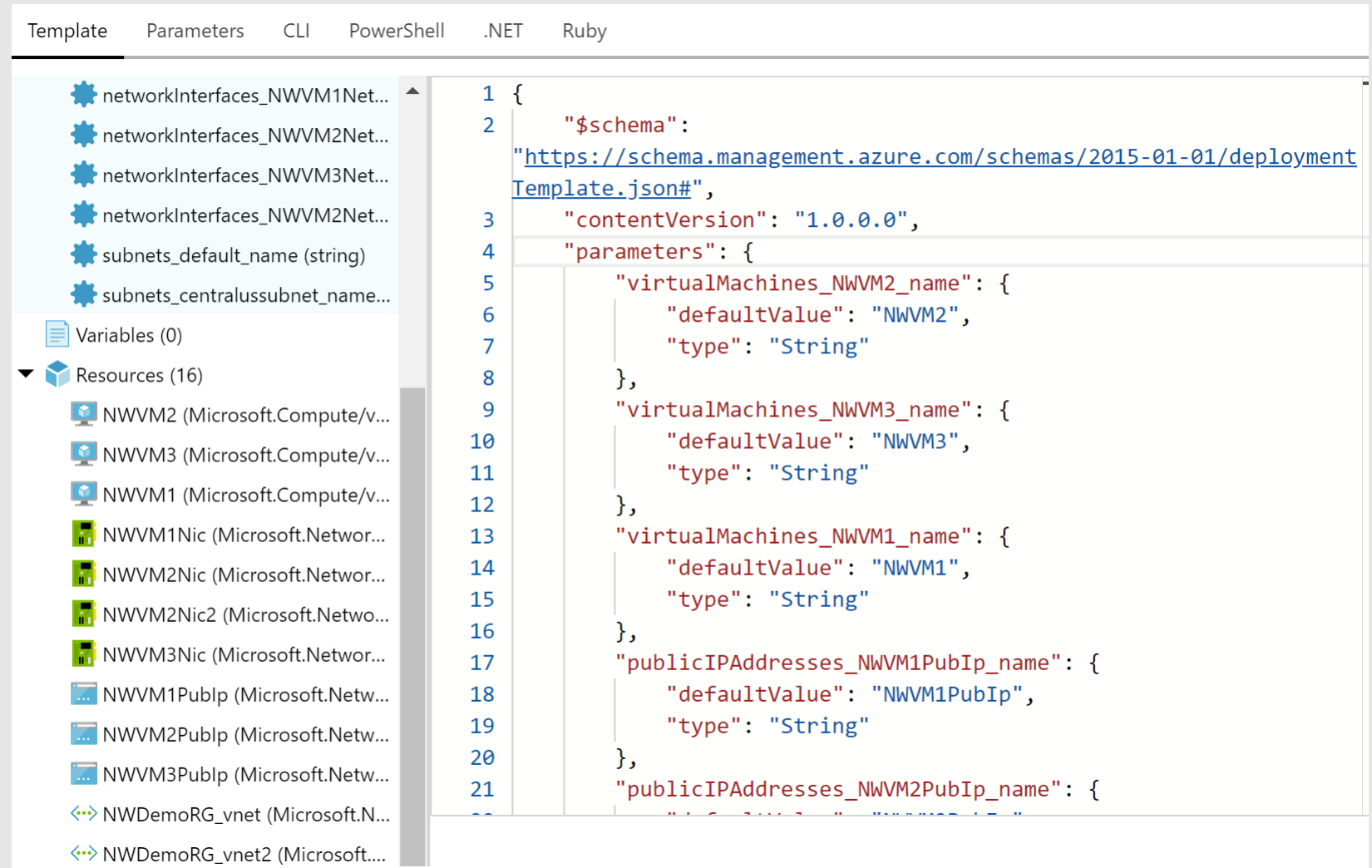


ANSIBLE



Azure Resource Manager Templates

ARM templates are based on JSON syntax



The screenshot displays the Azure Resource Manager Template editor interface. The top navigation bar includes tabs for Template, Parameters, CLI, PowerShell, .NET, and Ruby. The left sidebar shows a tree view of the template structure, including Variables (0) and Resources (16). The main editor area displays the JSON template content, which defines the schema, content version, and parameters for a virtual machine deployment.

```
1 {
2   "$schema":
3     "https://schema.management.azure.com/schemas/2015-01-01/deployment
4     Template.json#",
5   "contentVersion": "1.0.0.0",
6   "parameters": {
7     "virtualMachines_NWVM2_name": {
8       "defaultValue": "NWVM2",
9       "type": "String"
10    },
11    "virtualMachines_NWVM3_name": {
12      "defaultValue": "NWVM3",
13      "type": "String"
14    },
15    "virtualMachines_NWVM1_name": {
16      "defaultValue": "NWVM1",
17      "type": "String"
18    },
19    "publicIPAddresses_NWVM1PubIp_name": {
20      "defaultValue": "NWVM1PubIp",
21      "type": "String"
22    },
23    "publicIPAddresses_NWVM2PubIp_name": {
```

Azure Resource Providers

Resource providers registered for use with your subscription can be [found in the portal \(or via PowerShell, REST API, or CLI\)](#).

Microsoft Azure Sponsorship - Resource providers

Subscription

Search (Ctrl+ /)

Refresh

Microsoft.AzureRM	Registered
Microsoft.KeyVault	Registered
Microsoft.Logic	Registered
Microsoft.MachineLearning	Registered
Microsoft.ManagedIdentity	Registered
Microsoft.Maps	Registered
Microsoft.Network	Registered
Microsoft.NotificationHubs	Registered
Microsoft.Operationallnsights	Registered
Microsoft.OperationsManagement	Registered
Microsoft.PolicyInsights	Registered
Microsoft.Portal	Registered
Microsoft.RecoveryServices	Registered
Microsoft.Relay	Registered
Microsoft.ResourceHealth	Registered

Why use ARM Templates?

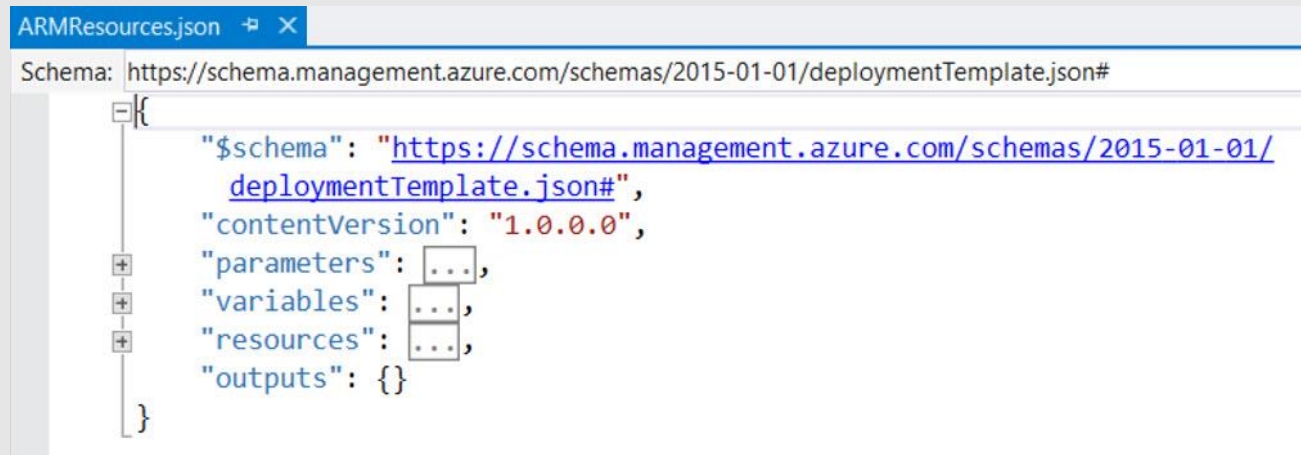
- Include the configuration of Azure resources in source control ("Infrastructure as Code"),
- Repeat the deployment process numerous times,
- Automate deployments,
- Employ continuous integration techniques,
- Utilize DevOps principles and practices,
- Repeatedly utilize testing infrastructure then de-provision it when finished

Ways to create an ARM Template

1. From the automation script available from the Azure Portal (which is imperfect – more on that in a moment), or
2. An ARM template in Visual Studio / Visual Studio Code, or
3. [QuickStart Templates](#) from [GitHub](#) (there's a lot to choose from – the templates that start with 101 are less complex), or
4. Create from the ground up, or
5. Start with a combination of 1 and 2 or 3, customizing the way you like it

Basic Template Structure: \$schema

This refers to the JSON schema. Note that the schema specified is different in the parameters file vs. the main ARM deployment file.

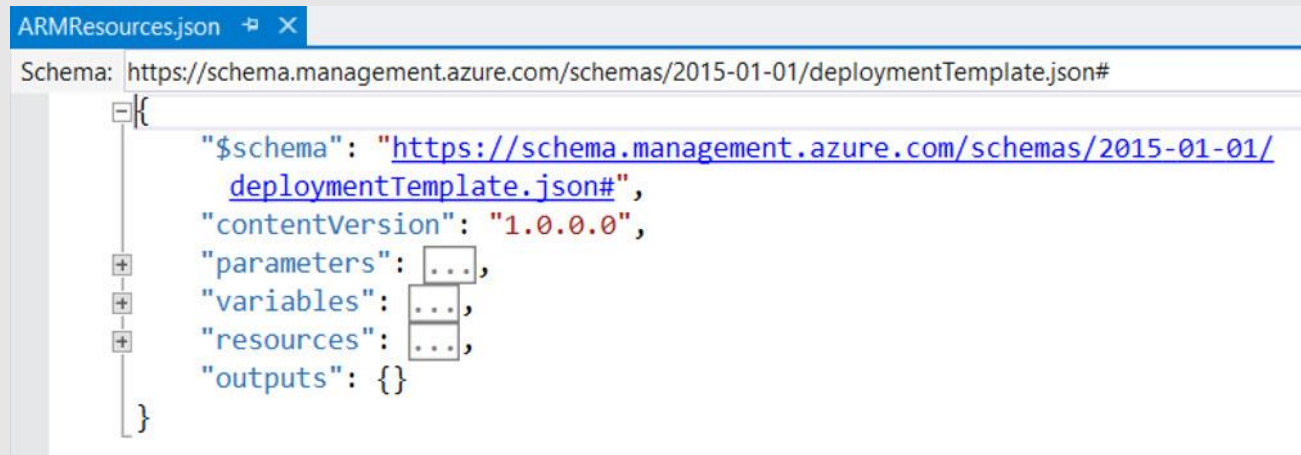


The screenshot shows a code editor window titled 'ARMResources.json'. The 'Schema' field at the top indicates the URL: 'https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#'. The main content is a JSON object representing an ARM template structure. The '\$schema' property is highlighted with a blue link. Other properties shown are 'contentVersion', 'parameters', 'variables', 'resources', and 'outputs'.

```
Schema: https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#  
  
{  
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/  
    deploymentTemplate.json#",  
  "contentVersion": "1.0.0.0",  
  "parameters": {},  
  "variables": {},  
  "resources": {},  
  "outputs": {}  
}
```

Basic Template Structure: Contentversion

You can increment this version if you'd like to manage the changes made over time.
The default is "1.0.0.0."



The screenshot shows a code editor window titled 'ARMResources.json'. The 'Schema' field is set to 'https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#'. The JSON content is as follows:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    ...
  },
  "variables": {
    ...
  },
  "resources": {
    ...
  },
  "outputs": {}
}
```

Basic Template Structure: Parameters

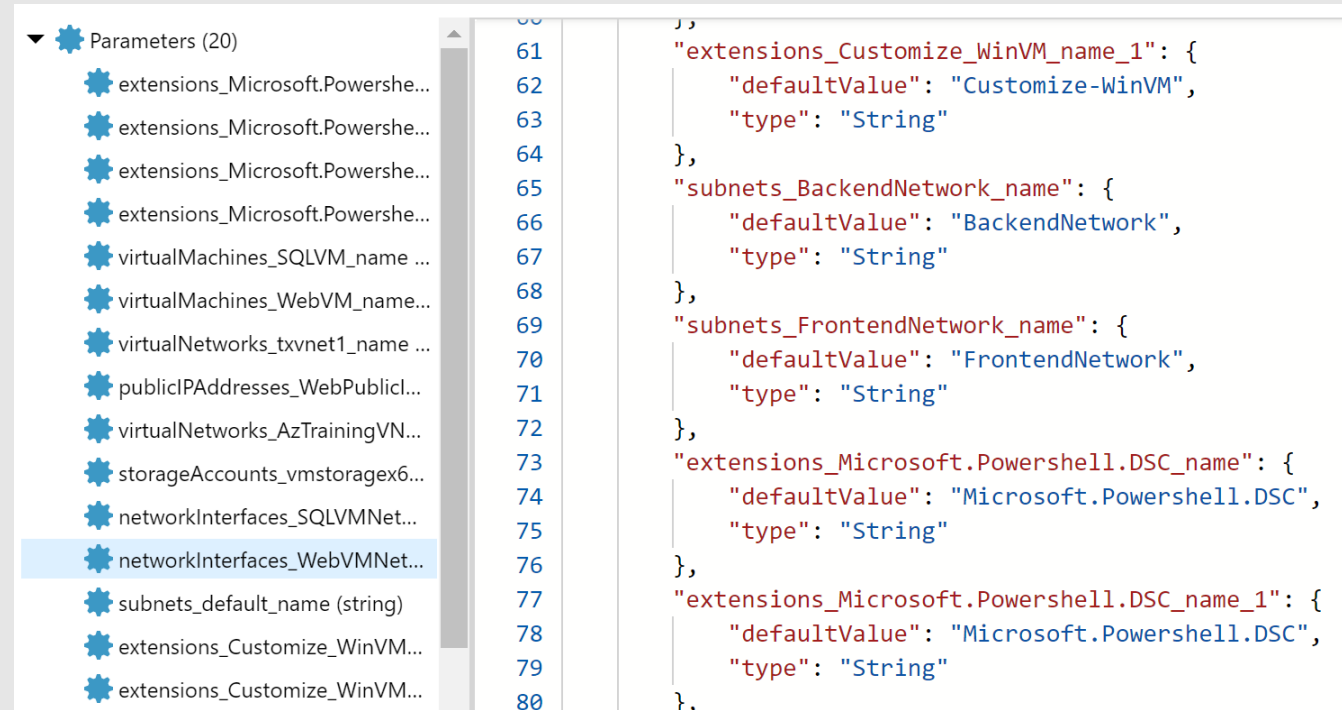
This is where you store the definitions of all parameters used throughout the ARM template

Type

Most common types are string or int;
Password = securestring;

MetaData / Description

Helpful info to remember what it's used for, or any other notes you want to leave for yourself and your team.



Basic Template Structure: Parameters

This is where you store the definitions of all parameters used throughout the ARM template

Allowed Values

Provides a list to choose from
(e.g. restrict list of VM Sizes)

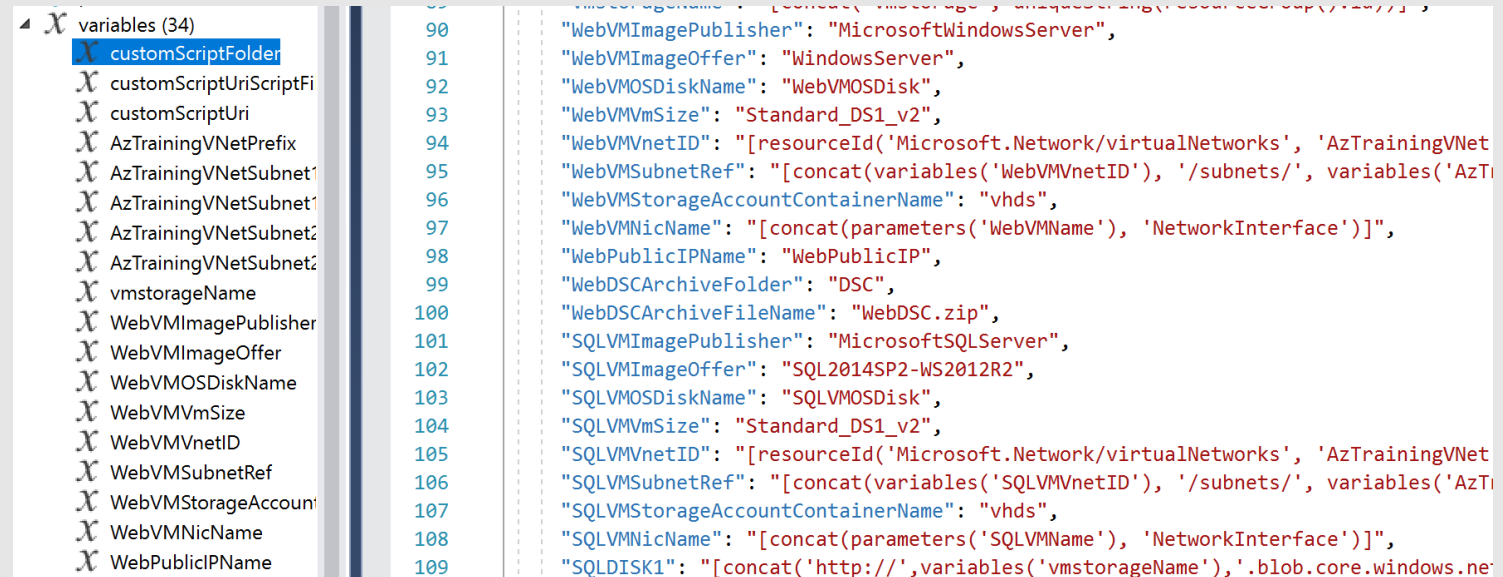
Default Value

Specifies the default value, but leaving choices (e.g. deploy DS2_v2 as VM size, but one can choose another from the list)

```
90     "type": "Microsoft.Compute/virtualMachines",
91     "name": "[parameters('virtualMachines_SQLVM_name')]",
92     "apiVersion": "2017-12-01",
93     "location": "eastus",
94     "tags": {
95       "displayName": "SQLVM"
96     },
97     "scale": null,
98     "properties": {
99       "hardwareProfile": {
100         "vmSize": "Standard_DS1_v2"
101       },
102       "storageProfile": {
103         "imageReference": {
104           "publisher": "MicrosoftSQLServer",
105           "offer": "SQL2014SP2-WS2012R2",
106           "sku": "Standard",
107           "version": "latest"
108         },
109         "osDisk": {
110           "osType": "Windows",
111           "name": "[concat(parameters('virtualMachines_SQLVM_name'), 'osdisk')]"
112         }
113       }
114     }
115   },
116   "resources": [
117     {
118       "type": "Microsoft.Storage/storageAccounts",
119       "name": "[parameters('storageAccount_name')]",
120       "apiVersion": "2016-01-01",
121       "location": "[parameters('location')]",
122       "tags": {
123         "displayName": "Storage Account"
124       },
125       "properties": {
126         "accountType": "Storage",
127         "accessTier": "Hot"
128       }
129     },
130     {
131       "type": "Microsoft.Network/virtualNetworks",
132       "name": "[parameters('virtualNetwork_name')]",
133       "apiVersion": "2016-03-01",
134       "location": "[parameters('location')]",
135       "tags": {
136         "displayName": "Virtual Network"
137       },
138       "properties": {
139         "addressSpace": {
140           "addressPrefixes": [
141             "[parameters('virtualNetwork_addressPrefix')]"
142           ]
143         },
144         "dhcpOptionsId": "[parameters('virtualNetwork_dhcpOptionsId')]",
145         "subnets": [
146           {
147             "name": "[parameters('virtualNetwork_subnet_name')]",
148             "properties": {
149               "addressPrefix": "[parameters('virtualNetwork_subnet_addressPrefix')]"
150             }
151           }
152         ]
153       }
154     },
155     {
156       "type": "Microsoft.Network/publicIPAddresses",
157       "name": "[parameters('publicIP_name')]",
158       "apiVersion": "2016-03-01",
159       "location": "[parameters('location')]",
160       "tags": {
161         "displayName": "Public IP"
162       },
163       "properties": {
164         "publicIPAddressType": "Static",
165         "dnsSettings": {
166           "domainNameLabel": "[parameters('publicIP_domainNameLabel')]"
167         }
168       }
169     },
170     {
171       "type": "Microsoft.Network/routeTables",
172       "name": "[parameters('routeTable_name')]",
173       "apiVersion": "2016-03-01",
174       "location": "[parameters('location')]",
175       "tags": {
176         "displayName": "Route Table"
177       },
178       "properties": {
179         "routes": [
180           {
181             "name": "[parameters('routeTable_route_name')]",
182             "properties": {
183               "addressPrefix": "[parameters('routeTable_route_addressPrefix')]",
184               "nextHopType": "VirtualNetworkGateway",
185               "nextHop": "[parameters('routeTable_route_nextHop')]"
186             }
187           }
188         ]
189       }
190     },
191     {
192       "type": "Microsoft.Network/loadBalancers",
193       "name": "[parameters('loadBalancer_name')]",
194       "apiVersion": "2016-03-01",
195       "location": "[parameters('location')]",
196       "tags": {
197         "displayName": "Load Balancer"
198       },
199       "properties": {
200         "frontendIPConfigurations": [
201           {
202             "name": "[parameters('loadBalancer_frontendIPConfig_name')]",
203             "properties": {
204               "publicIP": "[parameters('publicIP_name')]",
205               "privateIPAllocationMethod": "Static"
206             }
207           }
208         ],
209         "backendIPConfigurations": [
210           {
211             "name": "[parameters('loadBalancer_backendIPConfig_name')]",
212             "properties": {
213               "subnet": "[parameters('virtualNetwork_subnet_name')]",
214               "privateIPAllocationMethod": "Static"
215             }
216           }
217         ],
218         "loadBalancingRules": [
219           {
220             "name": "[parameters('loadBalancer_loadBalancingRule_name')]",
221             "properties": {
222               "frontendIPConfiguration": "[parameters('loadBalancer_frontendIPConfig_name')]",
223               "backendIPConfiguration": "[parameters('loadBalancer_backendIPConfig_name')]",
224               "protocol": "TCP",
225               "port": "[parameters('loadBalancer_loadBalancingRule_port')]",
226               "backendPort": "[parameters('loadBalancer_loadBalancingRule_backendPort')]"
227             }
228           }
229         ],
230         "probes": [
231           {
232             "name": "[parameters('loadBalancer_probe_name')]",
233             "properties": {
234               "protocol": "TCP",
235               "port": "[parameters('loadBalancer_probe_port')]",
236               "request": "[parameters('loadBalancer_probe_request')]"
237             }
238           }
239         ]
240       }
241     },
242     {
243       "type": "Microsoft.Compute/virtualMachines",
244       "name": "[parameters('virtualMachines_SQLVM_name')]",
245       "apiVersion": "2017-12-01",
246       "location": "[parameters('location')]",
247       "tags": {
248         "displayName": "SQLVM"
249       },
250       "properties": {
251         "hardwareProfile": {
252           "vmSize": "Standard_DS1_v2"
253         },
254         "storageProfile": {
255           "imageReference": {
256             "publisher": "MicrosoftSQLServer",
257             "offer": "SQL2014SP2-WS2012R2",
258             "sku": "Standard",
259             "version": "latest"
260           },
261           "osDisk": {
262             "osType": "Windows",
263             "name": "[concat(parameters('virtualMachines_SQLVM_name'), 'osdisk')]"
264           }
265         }
266       }
267     }
268   ],
269   "outputs": {
270     "virtualMachines_SQLVM_name": {
271       "type": "String",
272       "value": "[parameters('virtualMachines_SQLVM_name')]"
273     },
274     "storageAccount_name": {
275       "type": "String",
276       "value": "[parameters('storageAccount_name')]"
277     },
278     "virtualNetwork_name": {
279       "type": "String",
280       "value": "[parameters('virtualNetwork_name')]"
281     },
282     "publicIP_name": {
283       "type": "String",
284       "value": "[parameters('publicIP_name')]"
285     },
286     "routeTable_name": {
287       "type": "String",
288       "value": "[parameters('routeTable_name')]"
289     },
290     "loadBalancer_name": {
291       "type": "String",
292       "value": "[parameters('loadBalancer_name')]"
293     }
294   }
295 }
```

Basic Template Structure: Variables

The variables section contains references to settings, mostly picked up by the resources section later on, making definitions easier



```
variables (34)
  customScriptFolder
  customScriptUriScriptFi
  customScriptUri
  AzTrainingVNetPrefix
  AzTrainingVNetSubnet1
  AzTrainingVNetSubnet1
  AzTrainingVNetSubnet2
  AzTrainingVNetSubnet2
  vmstorageName
  WebVMImagePublisher
  WebVMImageOffer
  WebVMOSDiskName
  WebVMVmSize
  WebVMVnetID
  WebVMSubnetRef
  WebVMStorageAccountName
  WebVMNicName
  WebPublicIPName

90 "WebVMImagePublisher": "MicrosoftWindowsServer",
91 "WebVMImageOffer": "WindowsServer",
92 "WebVMOSDiskName": "WebVMOSDisk",
93 "WebVMVmSize": "Standard_DS1_v2",
94 "WebVMVnetID": "[resourceId('Microsoft.Network/virtualNetworks', 'AzTrainingVNet')]",
95 "WebVMSubnetRef": "[concat(variables('WebVMVnetID'), '/subnets/', variables('AzTrainingVNetSubnet1'))]",
96 "WebVMStorageAccountContainerName": "vhds",
97 "WebVMNicName": "[concat(parameters('WebVMName'), 'NetworkInterface')]",
98 "WebPublicIPName": "WebPublicIP",
99 "WebDSCArchiveFolder": "DSC",
100 "WebDSCArchiveFileName": "WebDSC.zip",
101 "SQLVMImagePublisher": "MicrosoftSQLServer",
102 "SQLVMImageOffer": "SQL2014SP2-WS2012R2",
103 "SQLVMOSDiskName": "SQLVMOSDisk",
104 "SQLVMVmSize": "Standard_DS1_v2",
105 "SQLVMVnetID": "[resourceId('Microsoft.Network/virtualNetworks', 'AzTrainingVNet')]",
106 "SQLVMSubnetRef": "[concat(variables('SQLVMVnetID'), '/subnets/', variables('AzTrainingVNetSubnet1'))]",
107 "SQLVMStorageAccountContainerName": "vhds",
108 "SQLVMNicName": "[concat(parameters('SQLVMName'), 'NetworkInterface')]",
109 "SQLDISK1": "[concat('http://', variables('vmstorageName'), '.blob.core.windows.net/')]"]
```

Basic Template Structure: Resources

The resources section defines each resource to be deployed, with references to parameters and variables as necessary. The elements which are defined vary based on the kind of resource which is being deployed.

```
192 {
193   "name": "[parameters('WebVMName')]",
194   "type": "Microsoft.Compute/virtualMachines",
195   "location": "[resourceGroup().location]",
196   "apiVersion": "2015-06-15",
197   "dependsOn": [
198     "[resourceId('Microsoft.Storage/storageAccounts', variables('vmstorageName'))]",
199     "[resourceId('Microsoft.Network/networkInterfaces', variables('WebVMNicName'))]",
200   ],
201   "tags": {
202     "displayName": "WebVM"
203   },
204   "properties": {
205     "hardwareProfile": {
206       "vmSize": "[variables('WebVMVmSize')]"
207     },
208     "osProfile": {
209       "computerName": "[parameters('WebVMName')]",
210       "adminUsername": "[parameters('WebVMAdminUsername')]",
211       "adminPassword": "[parameters('WebVMAdminPassword')]"
212     },
213     "storageProfile": {
214       "imageReference": {
215         "publisher": "[variables('WebVMImagePublisher')]",
216         "offer": "[variables('WebVMImageOffer')]",
217         "sku": "[parameters('WebVMWindowsOSVersion')]",
218         "version": "latest"
219       }
```

Basic Template Structure: Resources (typical)

- **Location:** You might be tempted to create a parameter for Location. However, a better practice is to inherit the location from the resource group.
- **Type:** The Type element for a resource is a combination of Resource Provider (discussed before) plus the Resource Type (ex: Microsoft.sql/servers).
- **Comments:** Helpful info to clarify what the resource is, or what it's being used for.
- **DependsOn:** This helps Azure understand dependencies, so it can deploy resources in parallel, or sequentially, as appropriate.
- **API Version:** A version is specified for each resource which is associated with a version of the REST API. The version impacts which elements can be specified for the resource, so the versions are updated on occasion.

Azure Automation Script – Resource Group Export

The screenshot displays the Azure portal interface. On the left sidebar, the 'Automation script' option is highlighted under the 'Settings' section. The main area is divided into two panes. The left pane shows a list of resources under the 'Resources (16)' category, including SQLVM, WebVM, SQLVMNic, WebPublicIP, AzTrainingVNet, vmstorage, and various concatenation functions. The right pane displays a JSON schema for a deployment template, with line numbers 1 through 19 visible. The schema defines parameters for extensions and configuration functions, including default values and types like 'SecureString'.

Left Sidebar (Settings):

- Events
- Quickstart
- Resource costs
- Deployments
- Policies
- Properties
- Locks
- Automation script**
- Monitoring
 - Insights (preview)
 - Alerts
 - Metrics
 - Diagnostic settings

Resources (16):

- Parameters (20)
- Variables (0)
- Resources (16)
 - SQLVM (Microsoft.Compute/virtua...
 - WebVM (Microsoft.Compute/virtu...
 - SQLVMNic (Microsoft.Network/ne...
 - [parameters('networkInterfaces_W...
 - WebPublicIP (Microsoft.Network/...
 - AzTrainingVNet (Microsoft.Networ...
 - [parameters('virtualNetworks_txvn...
 - vmstorage (Microsoft.Storage/stor...
 - [concat(parameters('virtualMachin...
 - Customize-WinVM (Microsoft.Co...
 - SQLDSC (Microsoft.Compute/virtu...
 - Customize-WinVM (Microsoft.Co...
 - WebDSC (Microsoft.Compute/virt...
 - [concat(parameters('virtualNetwor...
 - [concat(parameters('virtualNetwor...
 - [concat(parameters('virtualNetwor...

JSON Schema (Right Pane):

```
1 {  
2   "$schema":  
   "https://schema.management.azure.com/schemas/2015-01-01/deployment  
   Template.json#",  
3   "contentVersion": "1.0.0.0",  
4   "parameters": {  
5     "extensions_Microsoft.Powershell.DSC_modulesUrl": {  
6       "defaultValue": null,  
7       "type": "SecureString"  
8     },  
9     "extensions_Microsoft.Powershell.DSC_configurationFunction": {  
10      "defaultValue": null,  
11      "type": "SecureString"  
12    },  
13    "extensions_Microsoft.Powershell.DSC_modulesUrl_1": {  
14      "defaultValue": null,  
15      "type": "SecureString"  
16    },  
17    "extensions_Microsoft.Powershell.DSC_configurationFunction_1": {  
18      "defaultValue": null,  
19      "type": "SecureString"
```

DEMO

Azure Automation Script

Questions Landing Spot

“...If you want good answers,
ask better questions...”

© Randy Glasbergen

Azure Automation Script – Resource Group Export

- The value is part of the parameter name. Ex: if you have a web app named "007FFFWebAppDemo" it will generate a parameter called "sites_007FFFWebAppDemo_name";
- Not all resources can be scripted out in this manner yet. (You'll see a message at the top of the template pane when this occurs);
- Default values are overused;
- The admin password (when required) is not parameterized;
- There are a few inaccurate values that come out, which result in deployment failures when reusing the template as-is

ARM Templates flow to get kickstarted (what works for me...)

- Use Visual Studio 2017 as your « authoring » environment;
- Split out parameter file in a separate parameters.azuredeploy.json
- Fine-tune parameters according your needs
 - Matching naming conventions
 - Add metadata description for each parameter
 - Add parameters for anything that could vary between environments (dev, test, customer A, customer B,...)
 - NEVER store passwords as cleartext (rather securedstring with prompt or integrate with Azure Key Vault)
- Add variables where they make sense
- Fine-tune the Resources section
 - Use Resource Group inherited location
 - Use Tags (in relation to Azure Policies)

ARM Templates – GitHub Quickstart Templates

- <http://www.github.com/Azure/Quickstart-Templates>

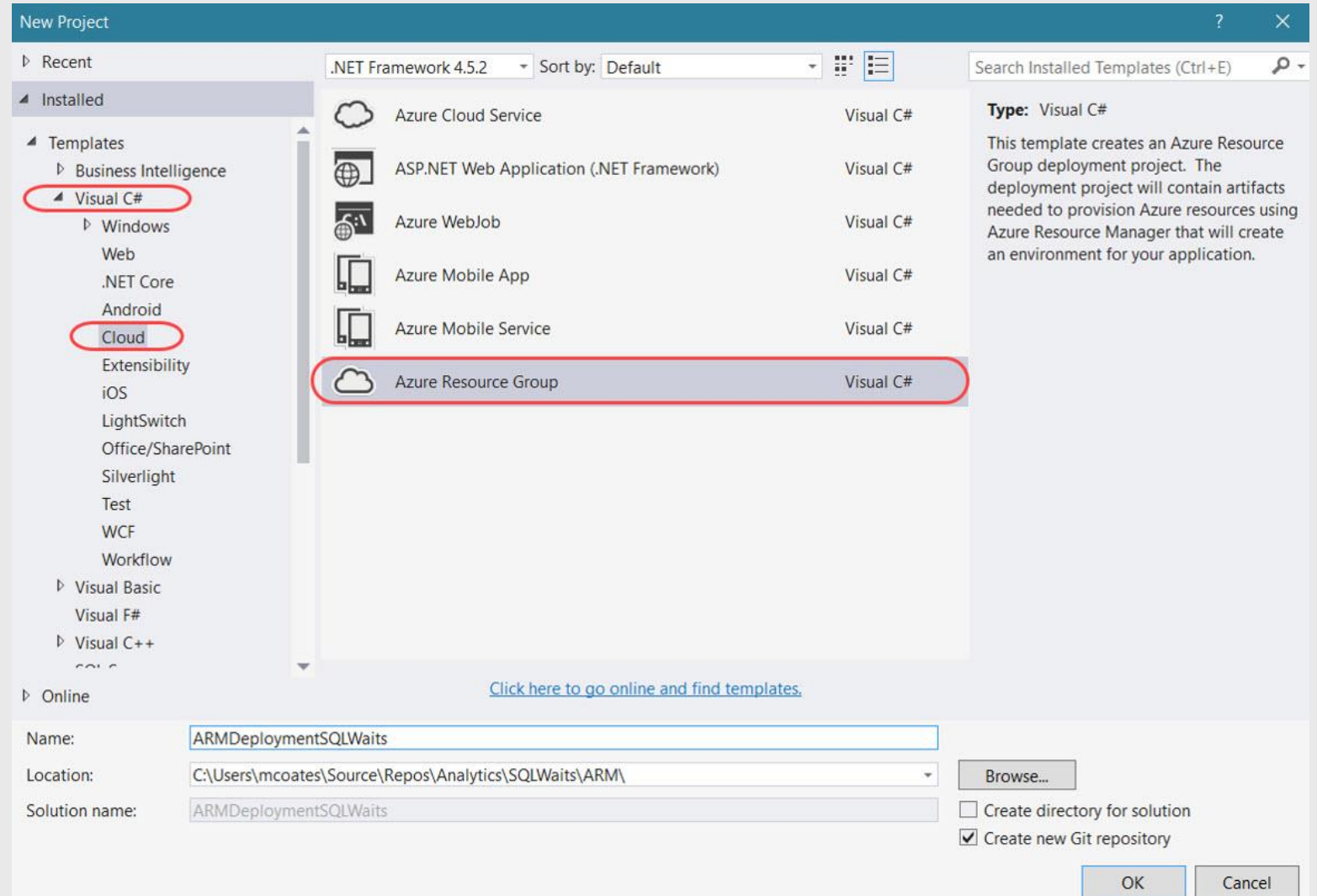
The screenshot shows the GitHub repository page for `Azure / azure-quickstart-templates`. The repository has 513 watchers, 3,835 stars, and 5,482 forks. It contains 425 issues, 34 pull requests, 0 projects, a Wiki, and Insights. The repository description is "Azure Quickstart Templates" with a link to <https://azure.microsoft.com/en-us/doc...>. The repository is categorized under `azure`, `templates`, and `arm`. It has 18,848 commits, 3 branches, 0 releases, 664 contributors, and is licensed under MIT. The current branch is `master`, and there is a button to create a new pull request. The repository includes a file browser showing the following files and their commit dates:

File Name	Commit Message	Commit Date
<code>.github</code>	Update stale.yml	May 22, 2018
<code>1-CONTRIBUTION-GUIDE</code>	text tweak	Jun 29, 2018
<code>100-blank-template</code>	added schema property to metadata.json	Jun 14, 2018
<code>100-marketplace-sample</code>	added schema property to metadata.json	Jun 14, 2018
<code>101-1vm-2nics-2subnets-1vnet</code>	added schema property to metadata.json	Jun 14, 2018
<code>101-Telegraf-InfluxDB-Grafana</code>	added schema property to metadata.json	Jun 14, 2018
<code>101-aci-dynamicsnav</code>	added schema property to metadata.json	Jun 14, 2018
<code>101-aci-linuxcontainer-public-ip</code>	added schema property to metadata.json	Jun 14, 2018

DEMO

GitHub Azure Quickstart templates

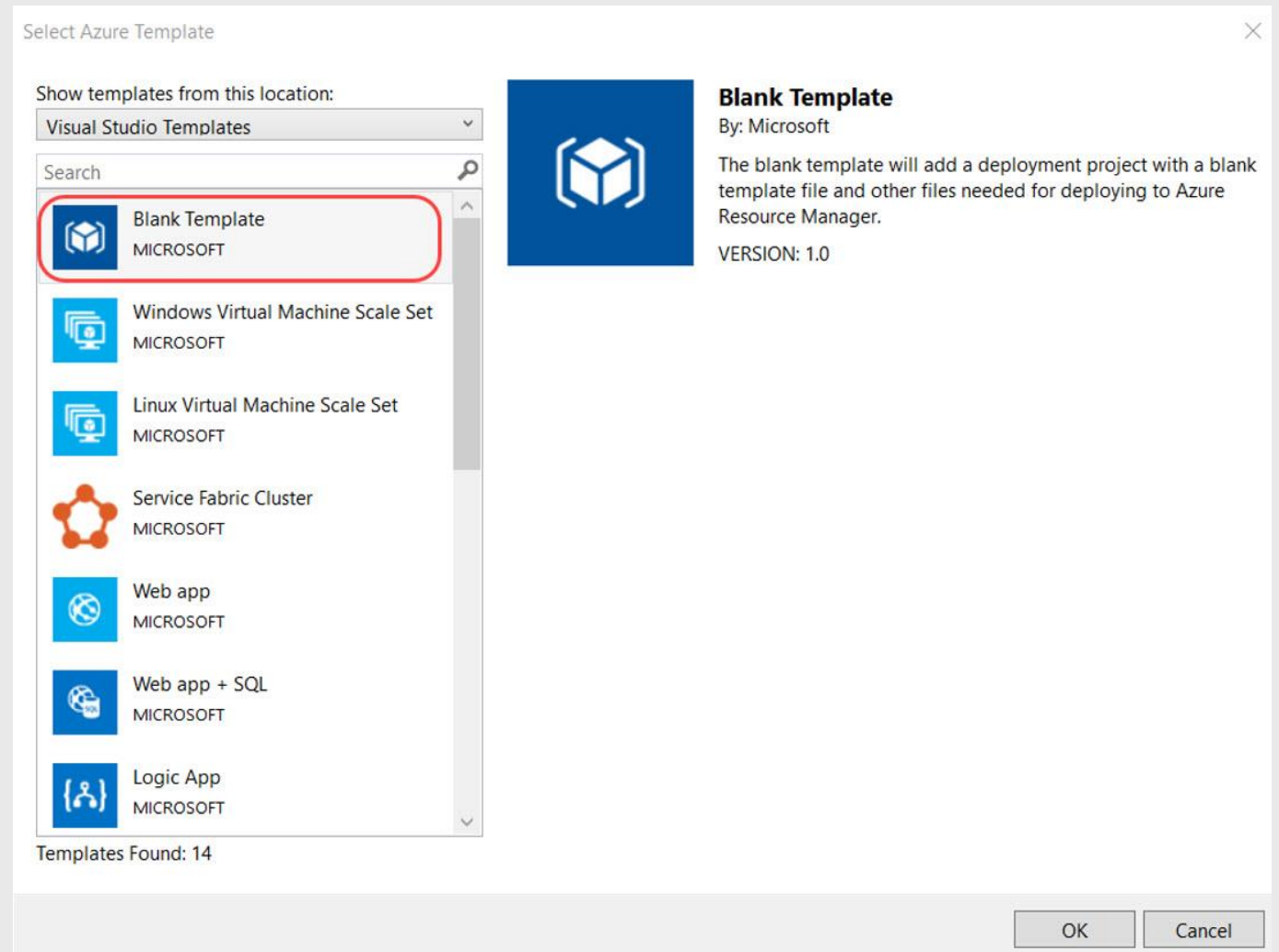
Create an ARM Project in VS2017



If you don't see this option, it means the Azure SDK has not been installed yet

Create an ARM Project in VS2017

**Start from a Blank Template or
use any of the GitHub
QuickStart Templates**

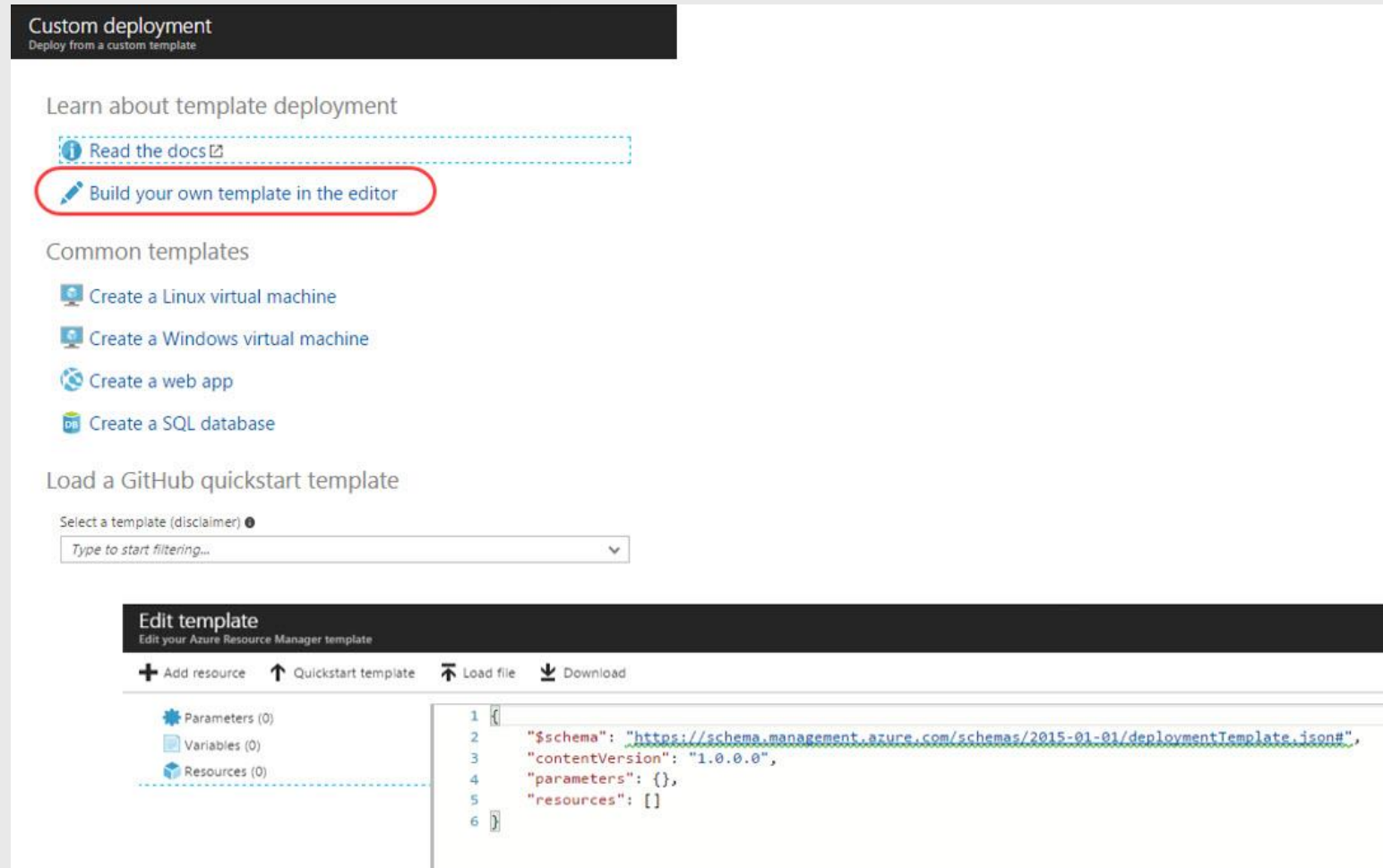


DEMO

ARM templates in Visual Studio

Running a template deployment from the Azure Portal

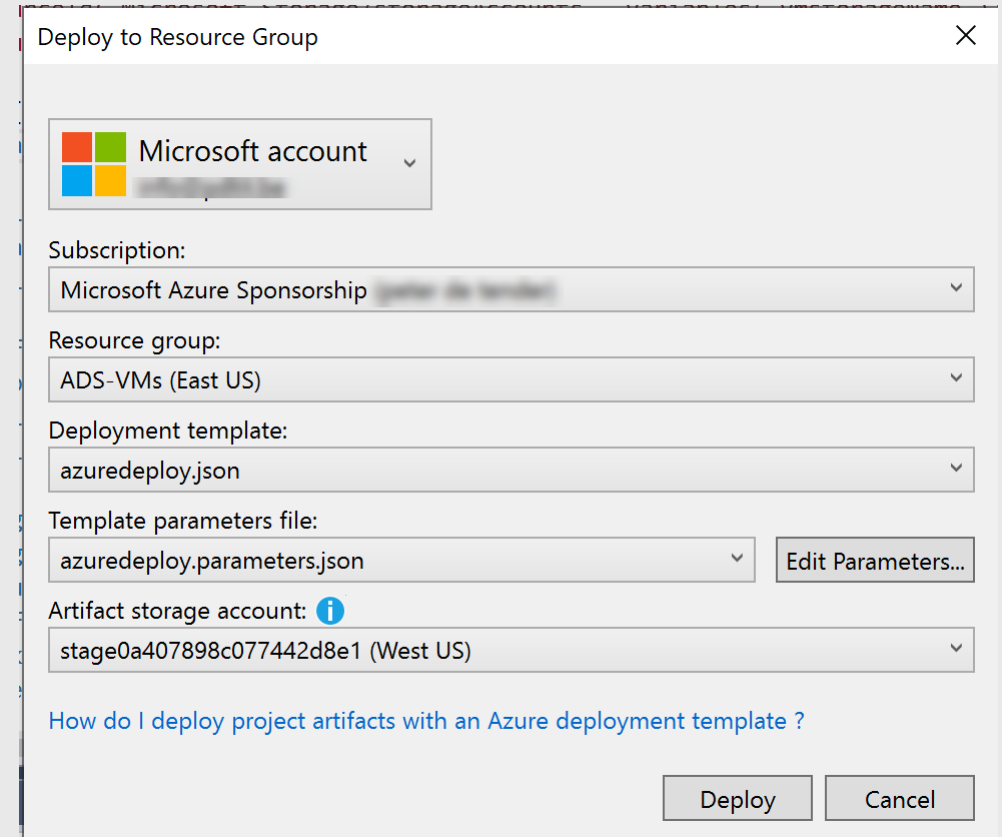
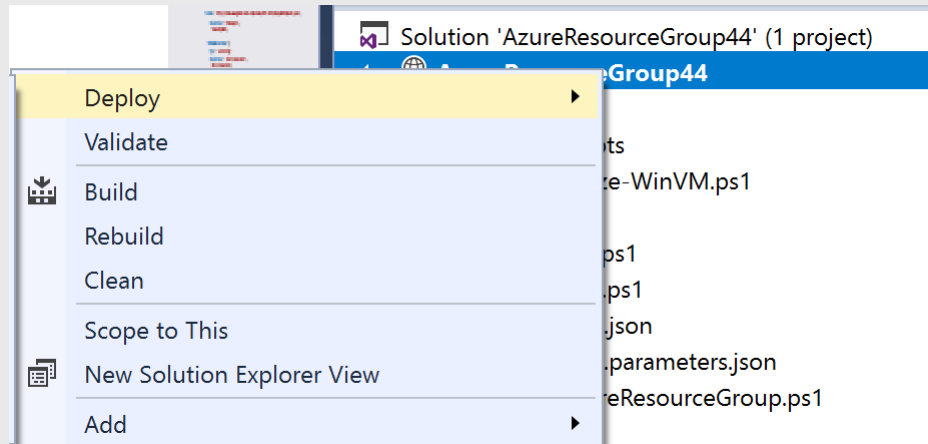
Add Resource / Template Deployment



The screenshot displays the 'Custom deployment' interface in the Azure Portal. At the top, a dark header reads 'Custom deployment' with the subtitle 'Deploy from a custom template'. Below this, a section titled 'Learn about template deployment' contains two links: 'Read the docs' (with an external link icon) and 'Build your own template in the editor' (with a pencil icon). The latter link is circled in red. Under the 'Common templates' section, there are four options: 'Create a Linux virtual machine', 'Create a Windows virtual machine', 'Create a web app', and 'Create a SQL database'. The 'Load a GitHub quickstart template' section features a dropdown menu labeled 'Select a template (disclaimer)' with the placeholder text 'Type to start filtering...'. The bottom half of the image shows the 'Edit template' editor, which has a dark header with the title 'Edit template' and subtitle 'Edit your Azure Resource Manager template'. It includes action buttons: '+ Add resource', '↑ Quickstart template', '↶ Load file', and '↴ Download'. On the left, a sidebar lists 'Parameters (0)', 'Variables (0)', and 'Resources (0)'. The main area on the right shows a JSON template structure with the following content:

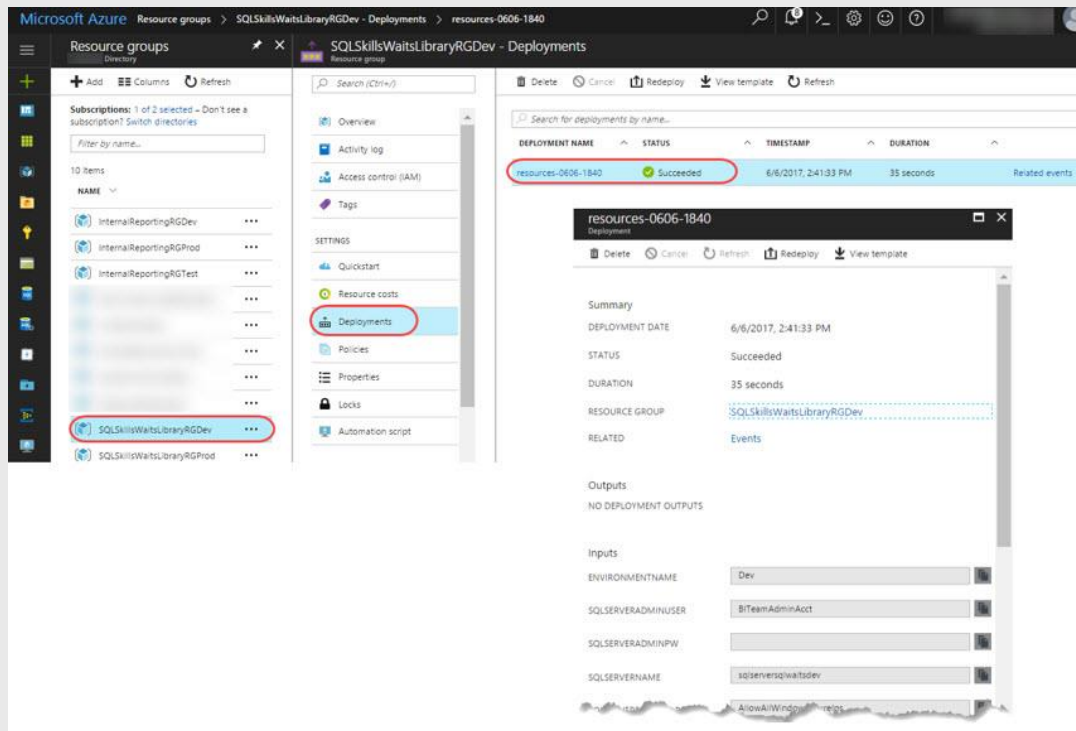
```
1 {  
2   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
3   "contentVersion": "1.0.0.0",  
4   "parameters": {},  
5   "resources": []  
6 }
```

Running a template deployment from VS2017

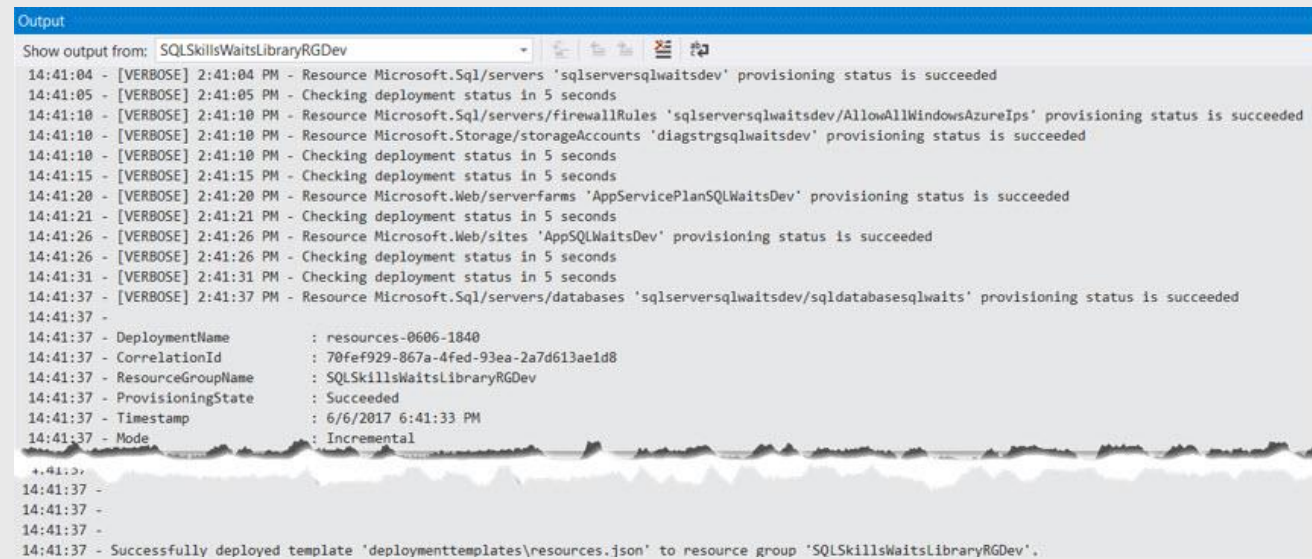


Monitoring ARM template deployment status

Azure Portal



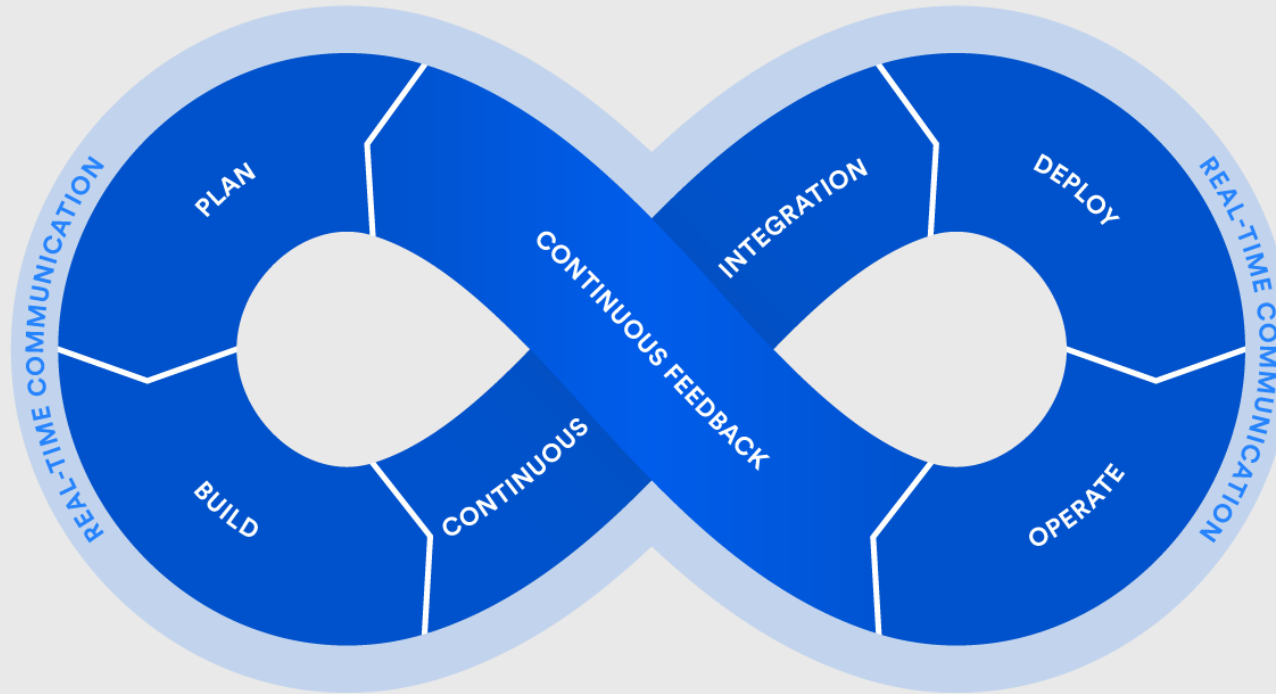
Visual Studio 2017 Output



DEMO

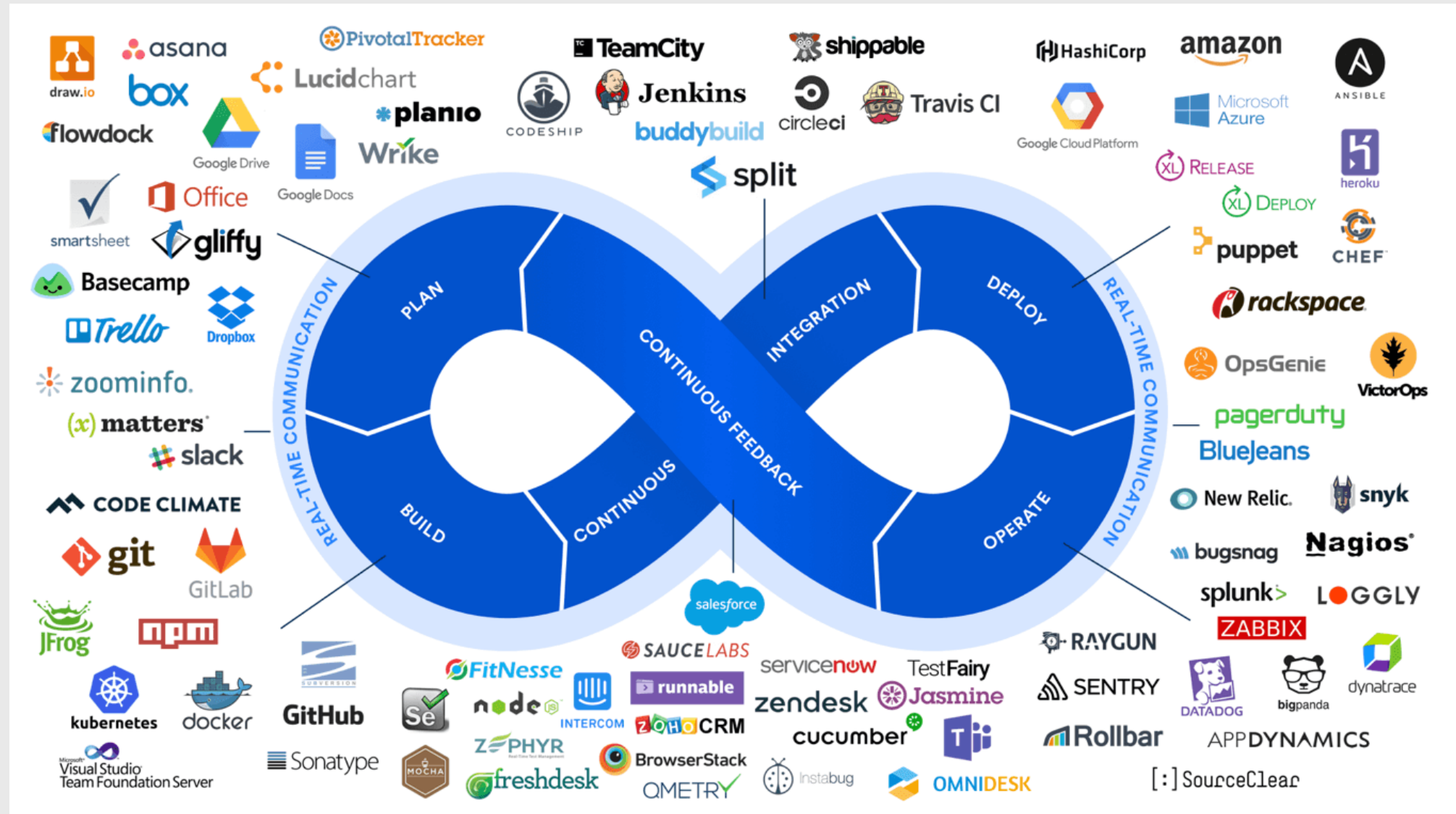
Monitoring / Troubleshooting ARM Template deployment

What is DevOps?



DevOps brings together **people, processes, and technology, automating software delivery to provide continuous value to your users.**

What is DevOps?



DevOps in an Azure World



Continuous integration (CI)

Take advantage of continuous integration to improve software development quality and speed. When you use Visual Studio Team Services or Jenkins to build apps in the cloud and deploy to Azure, each time you commit code, it's automatically built and tested—so bugs are detected faster.



Continuous Delivery (CD)

Ensure that code and infrastructure are always in a production-deployable state, with continuous delivery. By combining continuous integration and infrastructure as code (IaC), you'll achieve identical deployments and the confidence you need to manually deploy to production at any time.



Continuous Deployment (CI/CD)

With continuous deployment, you can automate the entire process from code commit to production if your CI/CD tests are successful. Using CI/CD practices, paired with monitoring tools, you'll be able to safely deliver features to your customers as soon as they're ready.

DevOps in an Azure World

Use your favorite DevOps toolchain – seamless integration

- Keep using the DevOps tools you know
- Get clear guidance and example architectures
- Deploy natively to Azure services
- Take advantage of the many tools available in the Azure Cloud Shell

Work with continuous integration and delivery tools

- Use [Visual Studio Team Services](#) or deploy directly to Azure infrastructure from your favorite continuous integration and continuous delivery tools, such as [Jenkins](#).



DevOps in an Azure World

Get the most from infrastructure automation and configuration management

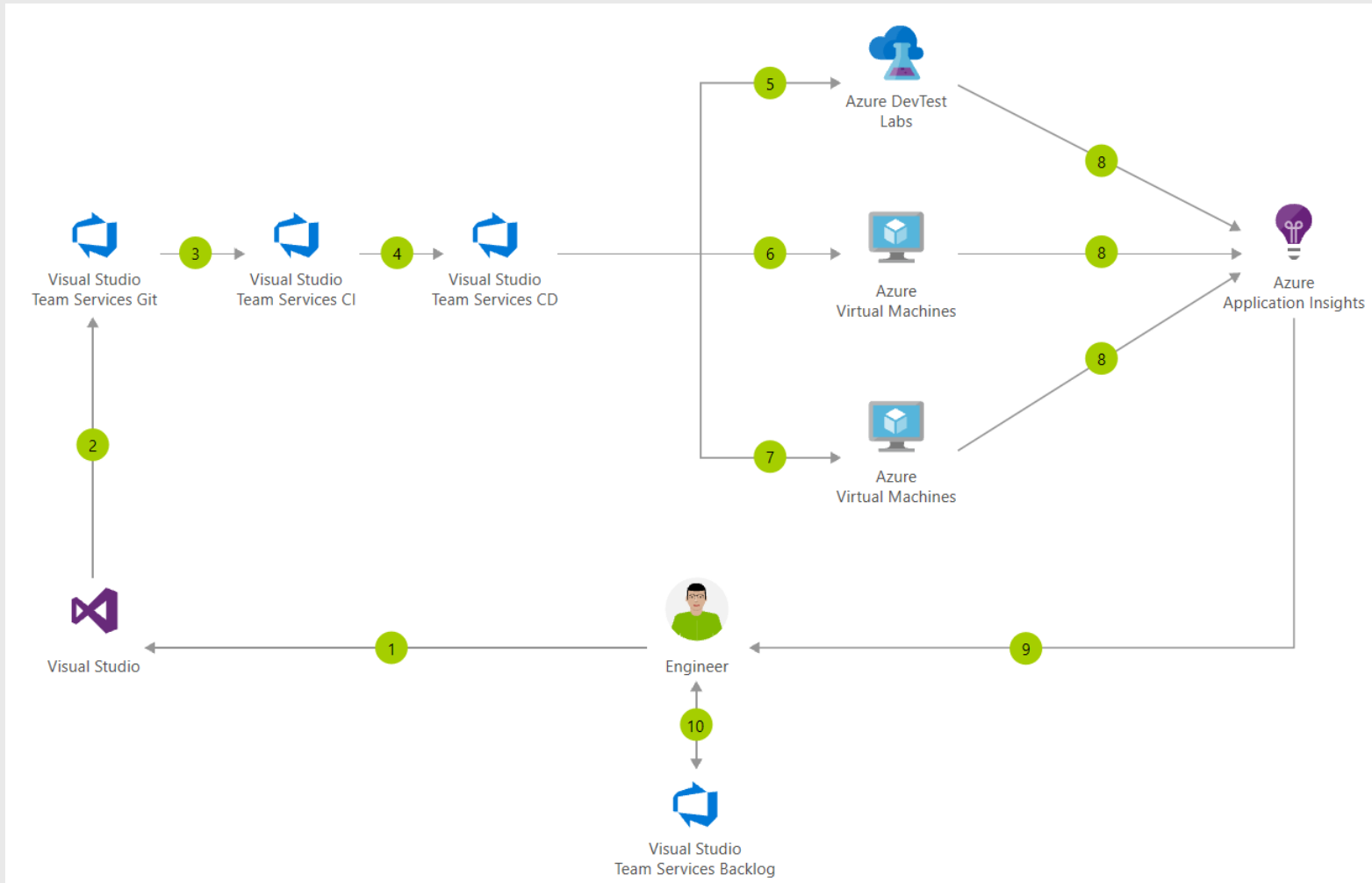
In addition to using [Azure Resource Manager](#) for infrastructure as code, you can provision and manage Azure infrastructure directly from your favorite third-party tools, such as [Ansible](#), Chef, Puppet, and [Terraform](#).

Gain clear guidance and example architectures

Azure services, third-party DevOps tools, and related products all work together to help meet the most common business needs and scenarios—including yours. Get started quickly with Azure [DevOps solutions](#) that give you access to architectures, tutorials, documentation, examples, templates, partners, and other resources.

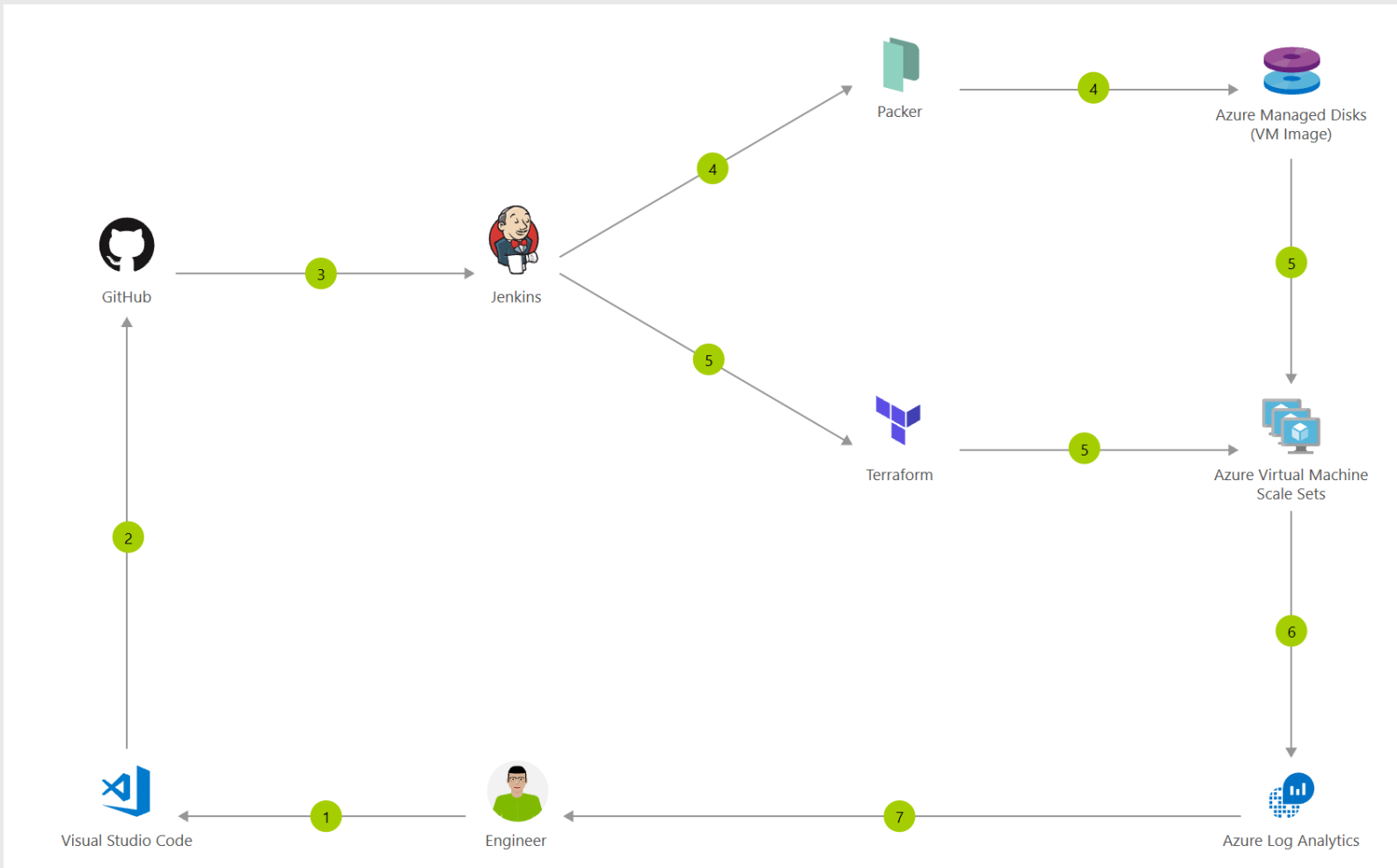


CI/CD for Azure VMs (VSTS example)



1. Change application source code
2. Commit Application Code and Azure Resource Manager (ARM) Template
3. Continuous integration triggers application build and unit tests
4. Continuous deployment trigger orchestrates deployment of application artifacts with environment specific parameters
5. Deployment to QA environment
6. Deployment to staging environment
7. Deployment to production environment
8. Application Insights collects and analyses health, performance and usage data
9. Review health, performance and usage information
10. Update backlog item

CI/CD for Azure VMs (Jenkins & TerraForm example)



1. Change application source code.
2. Commit code to GitHub.
3. Continuous Integration Trigger to Jenkins.
4. Jenkins triggers a Packer image build to create a VM and stores it as a VM image using Azure Managed Disks.
5. Jenkins triggers Terraform to provision a new Virtual Machine Scale Set using the Azure Managed Disks VM image.
6. Azure Log Analytics collects and analyzes logs.
7. Monitor application and make improvements.

<https://azure.microsoft.com/en-us/solutions/architecture/?solution=devops>

Questions Landing Spot

“...If you want good answers,
ask better questions...”

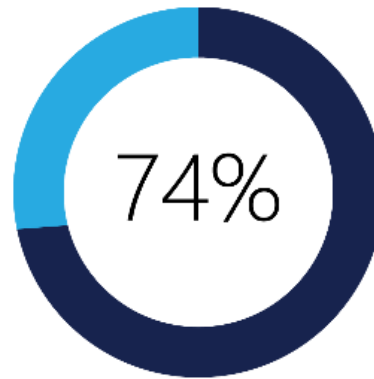
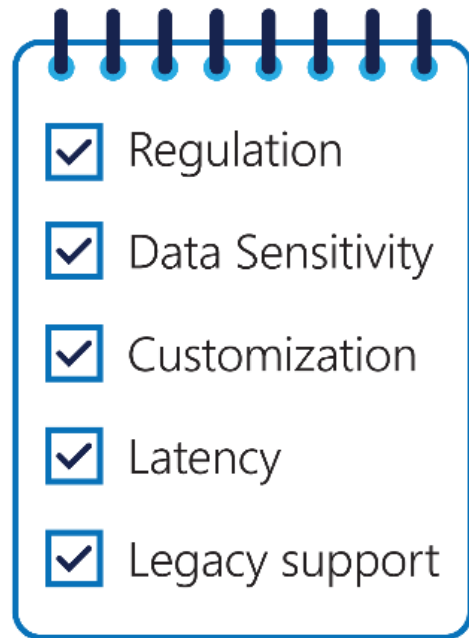
© Randy Glasbergen

**Why not just relying on Azure Resource Manager
Templates?**

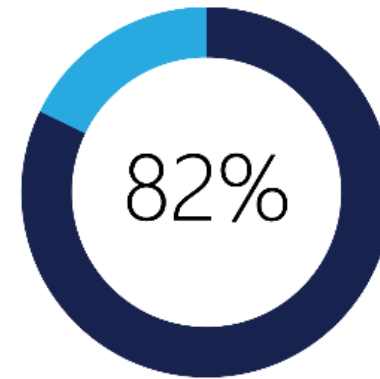
Hybrid Cloud, a reality today

Hybrid cloud, a reality today

Workload requirements



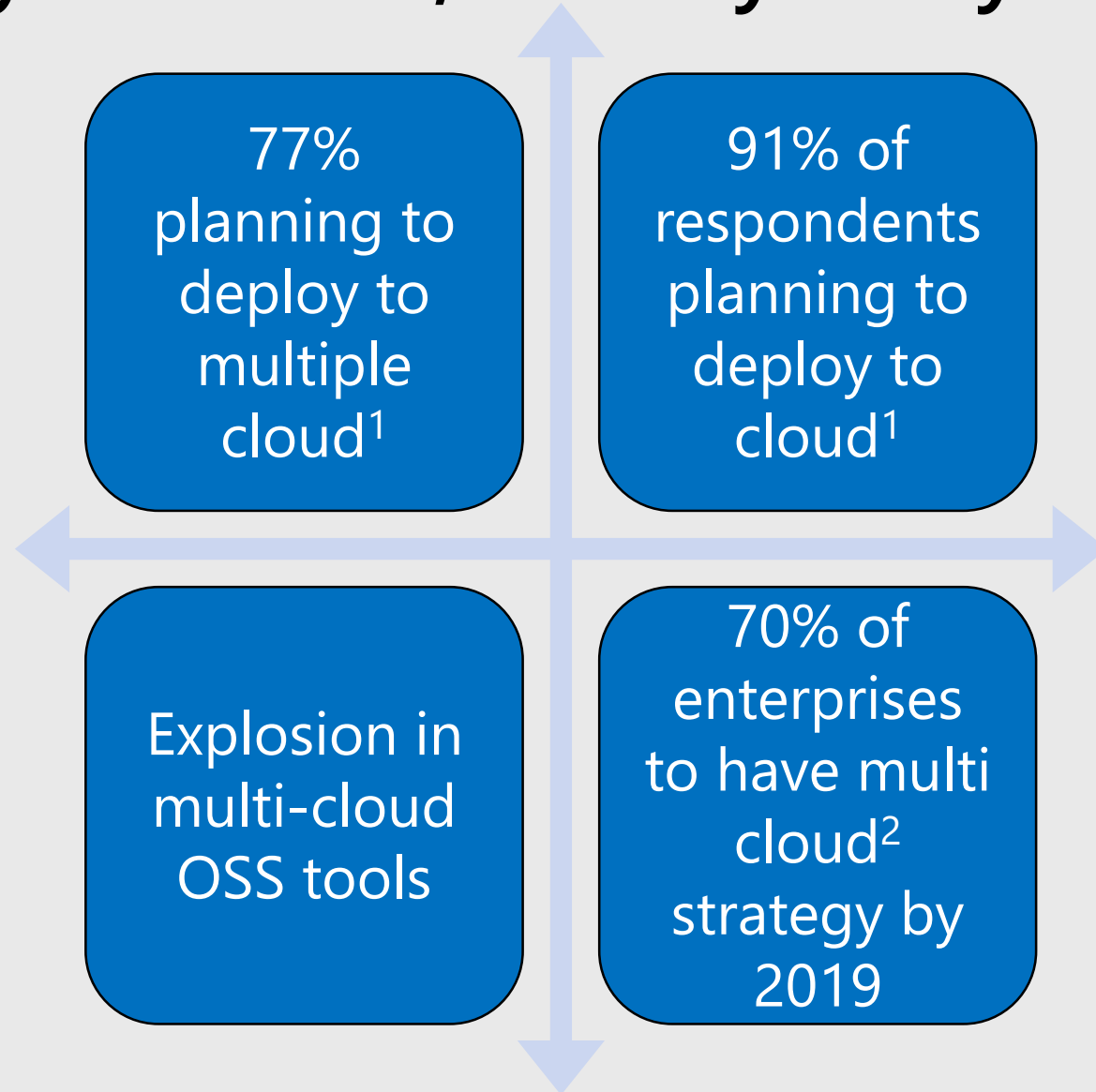
Enterprise believe a hybrid cloud will enable business growth¹



Enterprise have a hybrid cloud strategy, up from 74 percent a year ago²

Sources: 1. Avanade, Global Study: Hybrid Cloud-From Hype to Reality (Dec 2014); 2. IDC Cloud Prediction for 2015 (Dec 2014).

Hybrid Cloud, a reality today



1. [Dimensional Research study](#)
2. [Gartner Study of Future of Datacenter in Cloud Era](#)

Why Terraform

- Terraform is a product to **provision infrastructure and application resources** across private **cloud**, public cloud, and external services using a common workflow
- **Multi cloud**
- Easy to describe **json like** format call HCF
- Supports for both **on-prem and clouds**



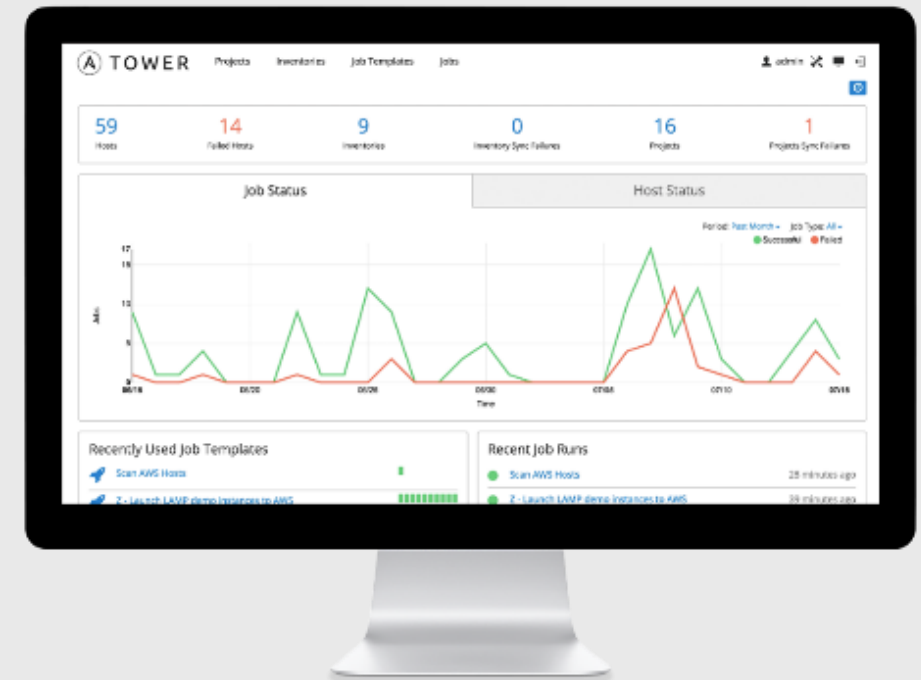
Provision Any Infrastructure For Any Application

Azure via Ansible



ANSIBLE

- **Modular**
 - Many built-in modules, or you can write your own
- **Agent-less**
 - Your Ansible controller will connect to hosts to run the tasks
- **SSH-based**
 - Connect to your hosts with SSH
 - Keys (recommended), passwords, or Kerberos



Azure via Ansible...Why?

- Use your **favorite** tooling
- You shouldn't have to **worry** about the "nooks and crannies" of Azure
- **Immutable**
- CI/CD integrated
- One Ring to Rule Them All (=both Deployment and Configuration Mgmt)

Ansible Azure Module support

Availability sets

DNS

Function App

Load balancer

Managed disk

Network

PublicIP

Security Group

Storage

Virtual Machines

Virtual Machine Scale Sets

VNET

...

Lab 1

Deploying your lab Virtual Machines environment

Lab 1 – Quick Instructions

1. Download the “Lab 1” Guide from GitHub (PDF)
2. Clone The LabFiles GitHub Repo
3. Task 1: Deploy a ‘JumpVM’, if you don’t have a VM with Visual Studio preinstalled
4. Task 2: Open de Visual Studio Project, become familiar with the ARM templates and deploy the infrastructure to your Azure subscription
5. When having questions: msdevseriesupport@007FFFlearning.com

Section Take-Aways

1. ARM templates are Azure's answer to IAC
2. ARM templates can do a lot more than just « deploy » Azure Resources
3. Besides ARM templates, there a lot of Open Source tools available allowing for IAC

Questions?

Peter De Tender

@pdtit

@007FFFlearning

Next Module...

SQL Database Migration to SQL PaaS



Peter De Tender

@pdtit

@007FFFlearning