

Infrastructure as code with Azure DevOps

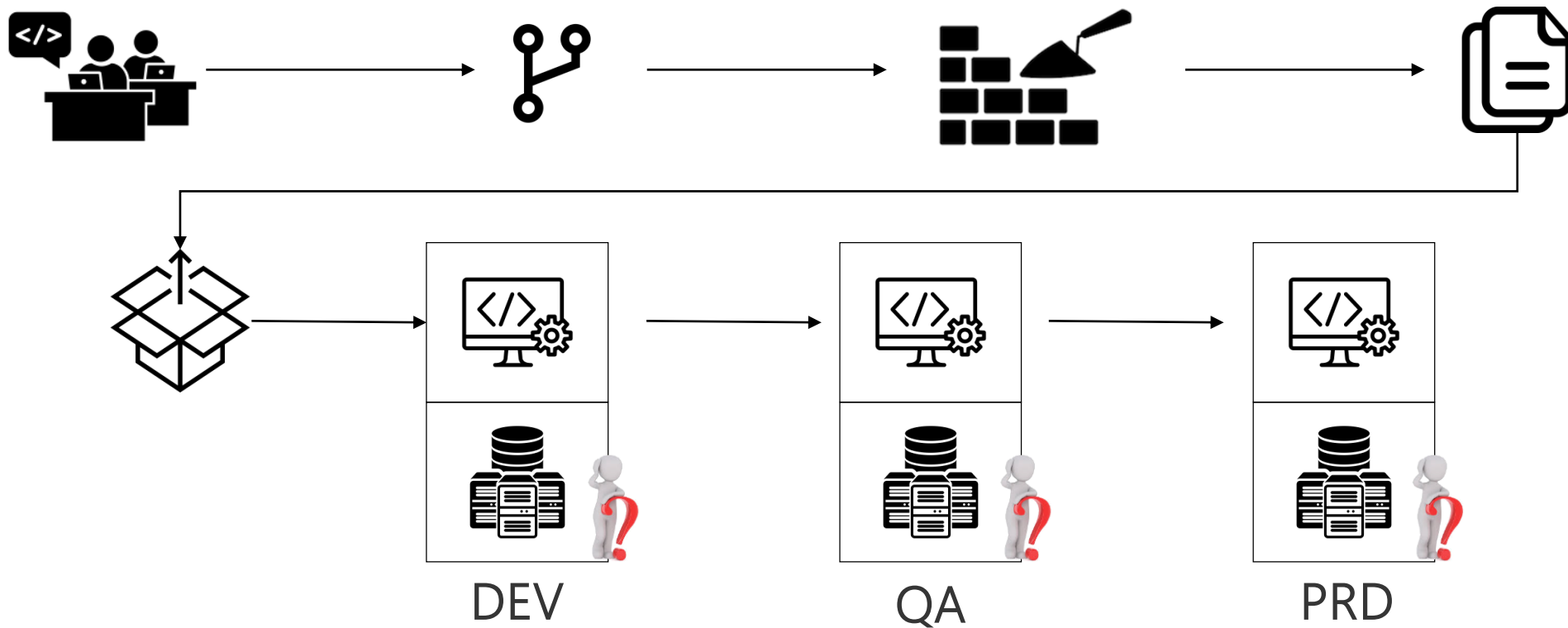
Martin Schwartzman

martin.schvartzman@microsoft.com

@martin77s



Application Development Approach



Infrastructure as Code (IaC)

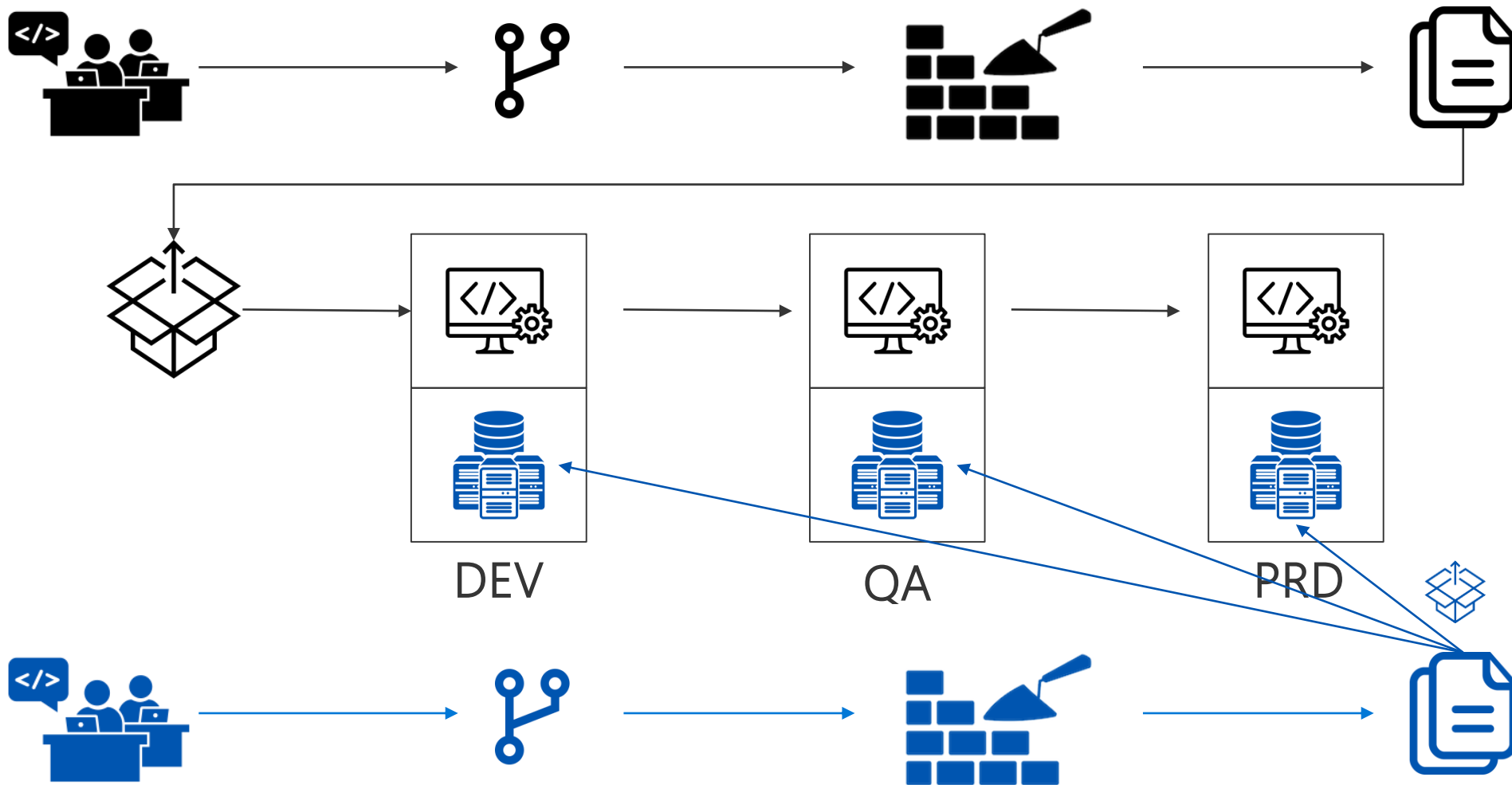
Infrastructure as Code is the **process** of managing and provisioning computing **infrastructure and its configuration** through machine-processable **definition files**.

It treats the infrastructure as a software system, applying **software engineering practices** to manage changes to the system in a **repeatable**, structured and safe way

- Declarative model (**not imperative**)
- Used in combination with continuous delivery
- Creates the same environment every time it is applied
(**idempotent**)

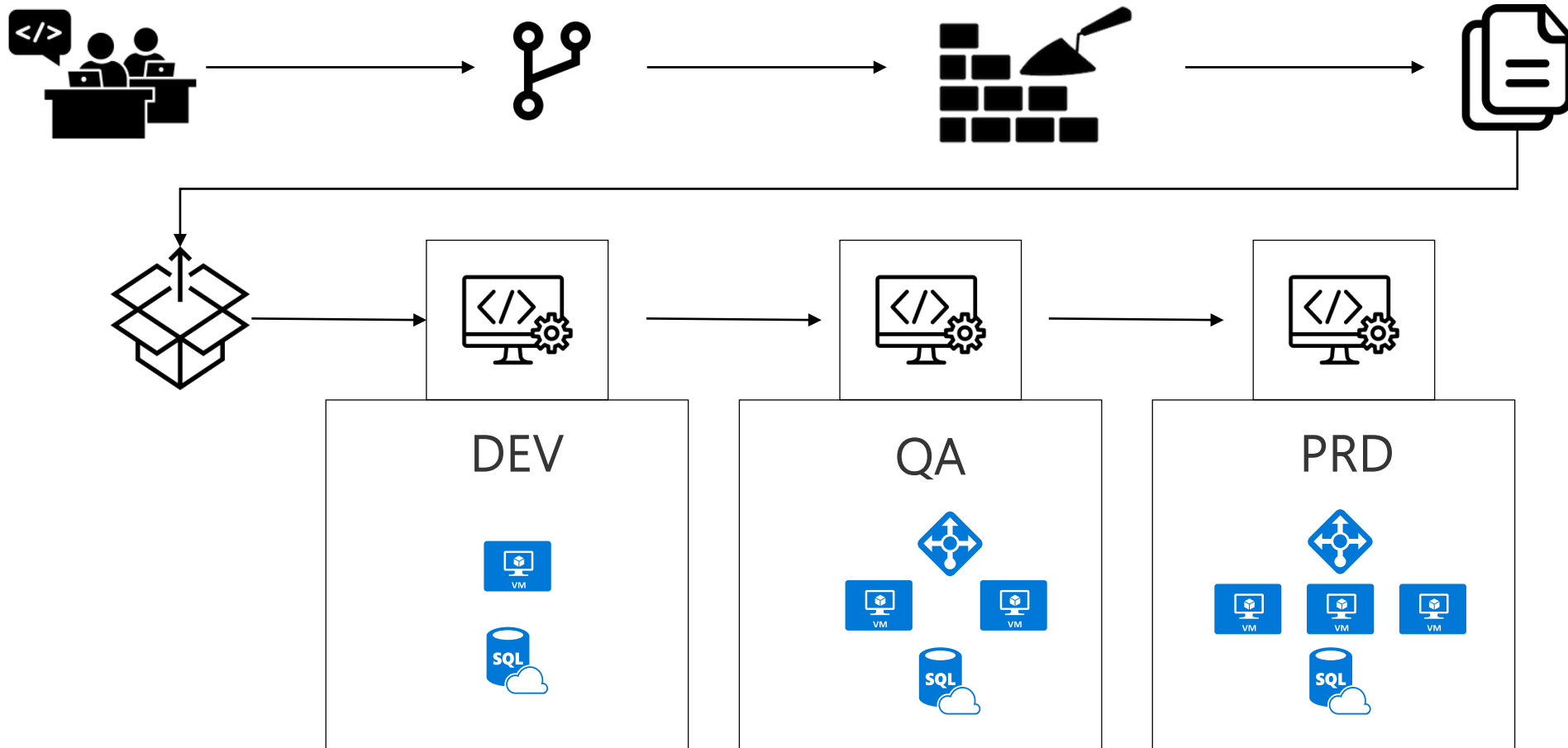


Application Development Approach + IaC



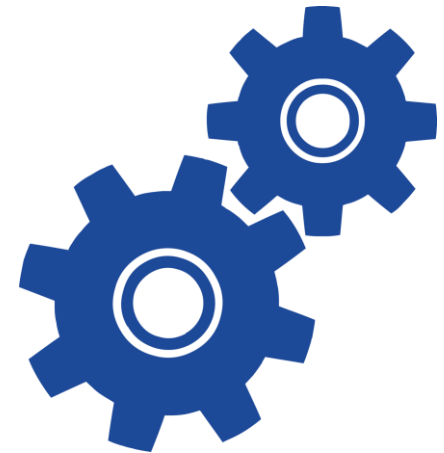
Application Development Approach + IaC

Environments can be different in terms of size and elements



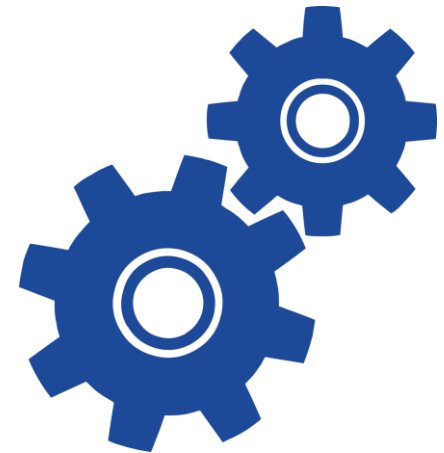
Benefits

- Simplify deployment and management
- Deliver stable environments quickly and at scale
- Decrease provisioning time
- Avoid configuration drift



Best Practices

- Maintain version control
- Test infrastructure
- Code infrastructure specifications in “configuration files”
- Keep secrets away (use KeyVaults)
- Use as little documentation as possible
 - Your code is a representation of your documentation



What are ARM Templates?

- Declarative files for creating Azure resources in a reliable, repeatable and auditable way.



Infrastructure as Code

Define Azure resources using text files.



Declarative Syntax

Declare how the resources should be and Azure Resource Manager “makes it so”.



JSON

ARM Templates are JSON format text files. Edit them in Visual Studio Code (or other text editors). Version control them.



Meta-Language

Contains some programming language constructs such as functions and loops.



<https://docs.microsoft.com/en-us/azure/templates/>

```
You, 10 months ago | 1 author (You)
You, 10 months ago • Initial Version
1  {
2    "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3    "contentVersion": "1.0.0.0",
4    "parameters": {
5      "aksResourceId": {
6        "type": "string",
7        "metadata": {
8          "description": "AKS Cluster resource Id"
9        }
10     },
11     "aksResourceLocation": {
12       "type": "string",
13       "metadata": {
14         "description": "Location of the AKS resource e.g. \"East US\""
15       }
16     },
17     "workspaceId": {
18       "type": "string",
19       "metadata": {
20         "description": "Azure Monitor Log Analytics resource id"
21       }
22     },
23     "workspaceRegion": {
24       "type": "string",
25       "metadata": {
26         "description": "Azure Monitor Log Analytics workspace region"
27       }
28     }
29   },
30   "resources": [{
31     "name": "[concat('Resource type', parameters('aksResourceId'), '/')[8]]",
32     "type": "Microsoft.ContainerService/managedClusters",
33     "location": "[parameters('aksResourceLocation')]"
```


Features of ARM Templates

Parameterized

Parameters can be used to configure resource attributes at deployment time. Allows generalization and re-use of ARM Templates.

Testable

Templates can be validated prior to deployment.

Modular

Templates can be broken into smaller, re-usable components and **linked** together at deployment time by using **Deployment** resources.

Templates can also be **nested** inside other templates.

Version Control

Using ARM Templates with version control allows your infrastructure to be reviewable, traceable and auditable.

Idempotent

Resources will only be changed or created if they have drifted out of state or need to be updated or created.



Demo

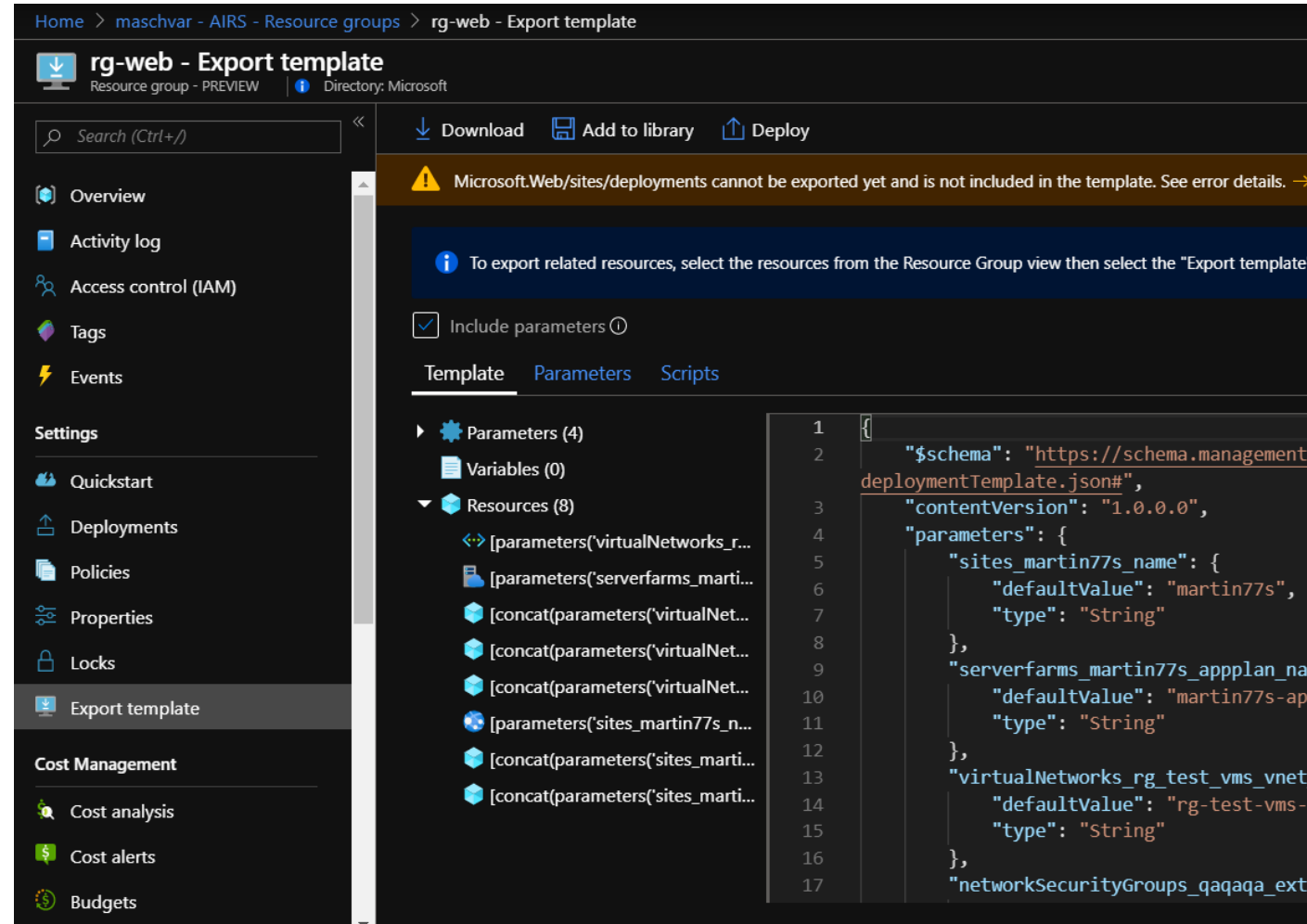
ARM Templates 101

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

Element name	Required	Description
\$schema	Yes	<p>Location of the JSON schema file that describes the version of the template language.</p> <p>For resource group deployments, use: https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#</p> <p>For subscription deployments, use: https://schema.management.azure.com/schemas/2018-05-01/subscriptionDeploymentTemplate.json#</p>
contentVersion	Yes	<p>Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.</p>
apiProfile	No	<p>An API version that serves as a collection of API versions for resource types. Use this value to avoid having to specify API versions for each resource in the template. When you specify an API profile version and don't specify an API version for the resource type, Resource Manager uses the API version for that resource type that is defined in the profile.</p>

Exporting ARM Templates

- Export from Resource Group
 - Export Specific Resources
 - Export from Deployments
-
- Generates a new ARM template
 - Simple, extendable patterns for
 - Azure CLI w/ Bash, PowerShell, .NET, Ruby
 - Not all resource types supported
 - Good starting point for automation exploration
 - Does not include changes post-deployment



Demo

ARM Templates 202

The screenshot displays the Visual Studio Code interface with an ARM template file open. The left pane shows the JSON code for the template, and the right pane shows the ARM Viewer visualization of the resources and their dependencies.

ARM Template Code (azuredeploy.json):

```
140     "name": "[variables('deployment-vnetName')]",
141     "type": "Microsoft.Network/virtualNetworks",
142     "location": "[resourceGroup().location]",
143     "apiVersion": "2016-03-30",
144     "dependsOn": [],
145     "properties": {
146       "addressSpace": {
147         "addressPrefixes": [
148           "[variables('deployment-vnetPrefix')]"
149         ]
150       },
151       "subnets": [
152         {
153           "name": "[variables('deployment-vnetSubnetName')]",
154           "properties": {
155             "addressPrefix": "[variables('deployment-vnetSubnetPrefix')]"
156           }
157         }
158       ],
159       "type": "Microsoft.Network/networkSecurityGroups",
160       "name": "[variables('deployment-nsgName')]",
161       "apiVersion": "2018-02-01",
162       "location": "[resourceGroup().location]",
163       "dependsOn": [
164         "[resourceId('Microsoft.Network/publicIPAddresses', variables('deployment-vnetName'))]"
165       ],
166       "properties": {
167         "securityRules": [
168           {
169             "name": "RDP",
```

ARM Viewer Visualization:

The ARM Viewer on the right shows a hierarchical diagram of the resources defined in the template. The resources are organized into groups and connected by dependency arrows. The resources include:

- serverfarms
- config
- sourcecontrols
- networkinterfaces
- virtualmachines
- extensions
- virtualnetworks
- networksecuritygroups

The status bar at the bottom indicates: Objects: 55 | Snap to grid: Off | Parameters: none | Filters: none. The bottom right corner shows the file path: Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Azure Resou.

Demo

laC w/ ADO

The screenshot displays the Azure DevOps web interface for configuring a pipeline named 'laC-Cl'. The left sidebar shows the navigation menu with 'Pipelines' selected. The main area shows the pipeline configuration with tabs for 'Tasks', 'Variables', 'Triggers', 'Options', 'Retention', and 'History'. The 'Tasks' tab is active, showing a list of tasks: 'Get sources' (laC, master) and 'Agent job 1' (Run on agent). Below 'Agent job 1', there is a task 'Create the WebApp' (Azure resource group deployment). The right sidebar shows the 'Pipeline' settings, including 'Name' (laC-Cl), 'Agent pool' (Azure Pipelines), 'Agent Specification' (vs2017-win2016), and 'Parameters'.

Azure DevOps

maschvar / laC / Pipelines

laC

Overview

Boards

Repos

Pipelines

Pipelines

Environments

Releases

Library

Task groups

Deployment groups

Test Plans

Artifacts

Compliance

Project settings

laC-Cl

Tasks Variables Triggers Options Retention History

Save & queue Discard Summary Queue

Pipeline Build pipeline

Get sources laC master

Agent job 1 Run on agent

Create the WebApp Azure resource group deployment

Name *

laC-Cl

Agent pool * ⓘ | Pool information

Azure Pipelines

Agent Specification *

vs2017-win2016

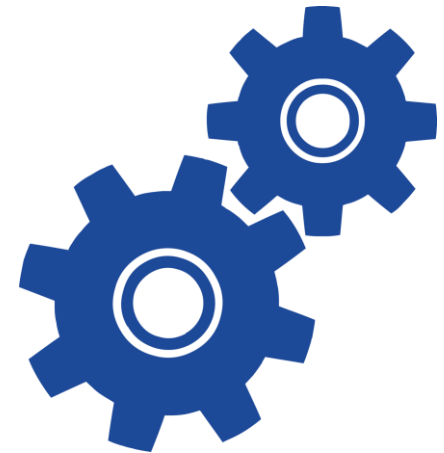
Parameters ⓘ

This pipeline doesn't have any pipeline settings between tasks and change the

[Learn more](#)

IaC everything?

- It depends. Sometimes the answer is no
- Some Azure components are deployed only one-time
- Coding infrastructure require time
 - Especially at the beginning, code only the components that can be reused
- Azure shared components can be prepared manually
 - ExpressRoute configuration, Hub configuration



Resources

Azure Resource Manager templates

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/template-deployment-overview>

Understand the structure and syntax of Azure Resource Manager templates

<https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-authoring-templates>

Azure Quickstart Templates

<https://azure.microsoft.com/en-us/resources/templates/>

CD of an Azure virtual machine using a Resource Manager template

<https://docs.microsoft.com/en-us/azure/devops/pipelines/apps/cd/azure/deploy-provision-azure-vm>

The Azure Superpowers workshop

<https://github.com/microsoft/AzureSuperpowers>

