Using ARM Templates

John Savill
PRINCIPAL TECHNICAL ARCHITECT
@NTFAQGuy savilltech.com

Module Overview

- Anatomy of an ARM Template
- Sources of ARM Templates
- Template Deployment
- Azure Portal Template Repository
- Template Best Practices
- Handling Secrets

Anatomy of an ARM Template

An ARM Template uses JSON to document the desired end state

There are a number of elements to a template

The key elements you will most often work with are:

1. Parameters
2. Variables
3. Resources

The template should not have instance specific values hard coded. The goal is to be able to deploy a template unchanged between environments

Demo

Walking through a basic template

Where to get templates

The portal shows the JSON for resources

Deploy a custom deployment in the portal allows a template to be built from resource types

Templates can be created from scratch using Visual Studio but not pleasant

GitHub has a large selection of templates that make a great starting point

VS Code has an ARM Tools extension that provides Azure template language support

Deploying an ARM Template

Typically all resources in a template are deployed to a resource group

It is possible for a template to deploy across resource groups and even subscriptions however this is not common

Templates can be deployed via the portal, PowerShell, CLI and the REST API

Also via blueprints, build pipelines and more!

Demo

Deploying some templates

Types of Deployment

Deployments can run in one of two modes:

Incremental

Resources in the template are deployed. If the resource exists and matches the settings in the template it is unchanged. If settings are different it is updated if possible. Resources that exist but aren't defined in the template are ignored.

Complete

Same as incremental except resources that exist in the target resource group that are not defined in the template are removed.

Linked and Nested Templates

Simple

For simple deployments a single template is the easiest to understand

Complex

For more complex deployments having multiple, modular templates is more practical and encourages reuse

Main Template

A main template enables management of parameters and overall structure that calls other templates

Linked and Nested

Linked uses a separate template file while nested embeds the template in the main template

A Quick Word on Blueprints

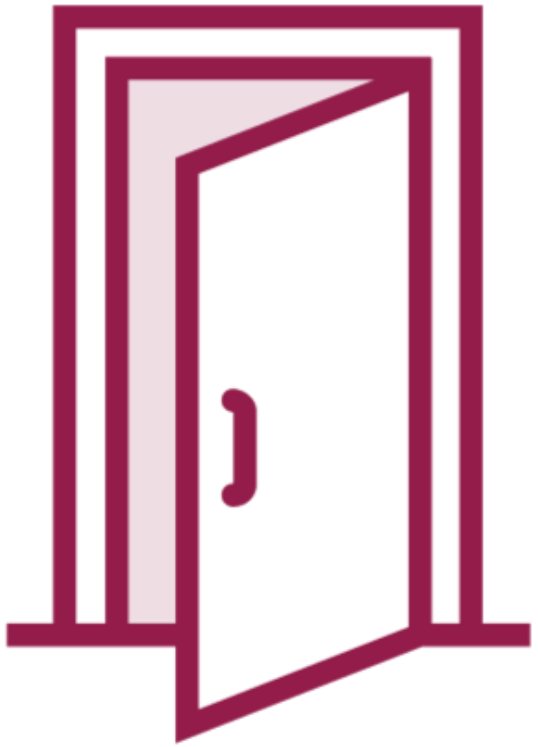
Blueprints bring together :

Templates + RBAC + Policy

Blueprints can be applied to “stamp” an environment with a complete configuration

If you have a complicated deployment instead of creating a large template another option may be to utilize a blueprint that can comprise of multiple, smaller, simpler templates

Templates in the Azure Portal



- Templates can be stored in the portal then easily deployed
- Library accessible when viewing templates in the portal
- Stored templates in the portal cannot access Key Vault for secrets
- Templates can be shared
- Been in preview for a number of years and lacks features of true repository

Keeping Secrets

NEVER put secrets in code

Use Azure Key Vault

Ensure access policy allows access for ARM template deployment

Access policy applies to types for entire vault, not per-secret

Reference the secret in template parameter file

Best Practices

There are limits for a template

Use parameters only as required

Use variables if the value is needed more than once or needs to be constructed

Use “comments” for resources

Don’t use DependsOn unless required as will reduce parallelism

Try and create a library of template components

Summary

- Anatomy of an ARM Template
- Sources of ARM Templates
- Template Deployment
- Azure Portal Template Repository
- Template Best Practices
- Handling Secrets

Next Up: Implementing Source Control for Templates