**ARM Templates**

**IaC Runbook**

Last Modified Date: September 8, 2022

Author(s): Arnav Sharma

Version   
State: Final

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Changes** |
|  | August 2022 | Arnav S | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contents**

[1. Introduction 5](#_Toc113531663)

[1.1 Purpose 5](#_Toc113531664)

[1.2 Audience 5](#_Toc113531665)

[1.3 Document Syntax 5](#_Toc113531666)

[2. Infrastructure as a code (IaC) 6](#_Toc113531667)

[2.1 Introduction of IaC 6](#_Toc113531669)

[2.2 ARM Templates Overview 6](#_Toc113531670)

[2.3 Purpose of ARM templates 6](#_Toc113531671)

[2.4 Reference Architecture 7](#_Toc113531672)

[3 ARM Templates User Guide 8](#_Toc113531673)

[3.1 How do ARM templates work? 8](#_Toc113531674)

[3.2 Benefits of ARM Templates 8](#_Toc113531675)

[3.3 Tools needed for authoring ARM Templates 9](#_Toc113531676)

[3.4 Creating an ARM template 9](#_Toc113531677)

[3.4.1 ARM Template Syntax 9](#_Toc113531678)

[3.4.2 Generating Templates from Azure Portal 10](#_Toc113531679)

[3.4.3 Using Quick Start Templates 12](#_Toc113531680)

[3.5 Deploying an ARM template 12](#_Toc113531681)

[4 Azure DevOps for ARM Templates 13](#_Toc113531682)

[4.1 Creating a repository 13](#_Toc113531683)

[4.2 Service Connection 15](#_Toc113531684)

[5 YAML Pipeline in Azure DevOps 17](#_Toc113531685)

[5.1 What is a YML pipeline? 17](#_Toc113531686)

[5.2 Creating a YML pipeline in Azure DevOps 17](#_Toc113531687)

[5.3 Managing YML pipelines in Azure DevOps 17](#_Toc113531688)

[5.4 Best practices for using YML pipelines in Azure DevOps 18](#_Toc113531689)

[6 Deploy ARM Templates using Azure DevOps. 19](#_Toc113531690)

[6.1 Using the repo. 19](#_Toc113531691)

[6.2 Resource Pipelines 20](#_Toc113531692)

[6.3 Master Pipelines 20](#_Toc113531693)

[7 Reusing the Pipelines and Templates. 21](#_Toc113531694)

[7.1 Getting help: 21](#_Toc113531695)

[7.2 YAML-Templates 22](#_Toc113531696)

[7.3 To deploy specific resources: 24](#_Toc113531697)

# **Introduction**

## **Purpose**

The purpose of this document is to define the objective of the Infrastructure as a Code (IaC) ARM Templates assets and how to use them. It will provide the reader with the understanding of what these assets are envisioned for and wasn’t, providing insights of how to apply and accelerate their customer’s engagement.

## **Audience**

* Pre-Sales/Solution Architects
* DevOps Engineers
* Infrastructure Engineers

## **Document Syntax**

|  |  |
| --- | --- |
| ! | Important Note |

|  |  |
| --- | --- |
| ! | Critical Note |

|  |  |
| --- | --- |
| \* | Design Decision |

|  |  |
| --- | --- |
| ? | Business Decision Required |

# **Infrastructure as a code (IaC)**



## **Introduction of IaC**

Technological advances have led to a new way of thinking about how IT infrastructures are built and maintained. Instead of manually configuring each system, infrastructure can now be treated as code. This "infrastructure as code" approach has numerous benefits, including increased speed, improved consistency, and reduced errors. Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied. IaC is a key DevOps practice and is used in conjunction with continuous delivery.

Infrastructure as code defines the environment in a versioned
file

## **ARM Templates Overview**

An ARM template is a JSON file that defines the infrastructure and configuration for your Azure solution. The template includes a parameters file that specifies values for the variables in the template.

You can use an ARM template to deploy and manage Azure resources in a consistent, automated way. For example, you can use an ARM template to deploy multiple virtual machines (VMs) or web apps in parallel. You can also use an ARM template to provision resources for development or testing environments.

ARM templates are reusable so you can share them with other team members. You can also publish ARM templates in the Azure Marketplace so other subscribers can deploy them in their own Azure subscriptions.

## **Purpose of ARM templates**

ARM templates are used to provision Azure resources in a declarative way. This means that you define the desired state of your resources in the template, and then Azure will make sure that the actual state of your resources matches what is defined in the template.

You can use ARM templates to deploy an entire solution, or just deploy individual resources. For example, you could use a single template to deploy an entire application consisting of multiple web apps, databases, and storage accounts. Or, you could use separate templates for each resource type.

## **Reference Architecture**

ARM templates for the resources visible in the below reference Architecture have been created and the templates have been modularized for flexibility and re-useability.

Graphical user interface, diagram, application

Description automatically generated

# **ARM Templates User Guide**

## **How do ARM templates work?**

An ARM template is a JSON file that defines one or more resources to deploy to a resource group. The template can also specify parameters and variables to input when deploying the resource group.

When you deploy a resource group using an ARM template, Azure Resource Manager processes the template and creates the resources in the deployment order specified in the template. You can also use linked templates to reference other templates for modular deployments.

## **Benefits of ARM Templates**

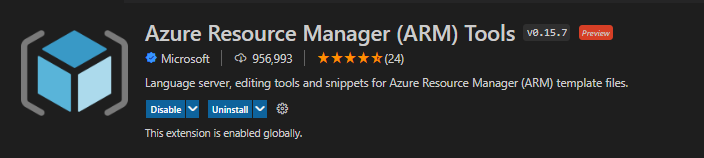
ARM templates are flexible and easy to use - making them ideal for DevOps processes.

With ARM templates, you can:

* Automate deployments: Deploy your applications and infrastructure with confidence knowing that what is deployed can be easily reproduced and updated.
* Control costs Reduce time spent on repetitive tasks and lower costs by using a standard, repeatable deployment process.
* Reduce the risk of unexpected resource consumption when you have visibility into how resources are being consumed and can plan for future resource usage.
* Deliver application changes quickly and reliably deploy updates to your applications with the click of a button.
* Scale application workloads Easily scale resources up or down as demand fluctuates, preventing resource contention, bottlenecks and disruptions in service.
* Reduce risk and increase efficiency Reduce the risk and impact of unexpected changes to your application environment, by maintaining a stable, consistent environment.

## **Tools needed for authoring ARM Templates**

Templates are JavaScript Object Notation (JSON) files. To create templates, you need a good JSON editor, the recommend one Visual Studio Code with the Azure Resource Manager Tools extension.



## **Creating an ARM template**

Creating an ARM template is a simple process. You start by creating a JSON file that holds the required parameters for the resources you want to create. You then save this file with a '.json' extension and upload it to Azure.

You can author your own templates, or use pre-built templates available from the Azure Marketplace or other locations. Once you have a template, you can deploy it using the Azure Portal, PowerShell, the Azure CLI, or REST API. ARM templates offer several benefits over other deployment methods. They allow you to reuse your templates for multiple deployments, and they provide a flexible way to set up resources in Azure.

When you create an ARM template, you specify the resources that make up your solution and their dependencies on one another. The resources can be deployed in parallel or serially. You also specify parameters that allow you to customize deployments for different environments (such as development, test, and production).

### **ARM Template Syntax**

The JSON file has these elements:

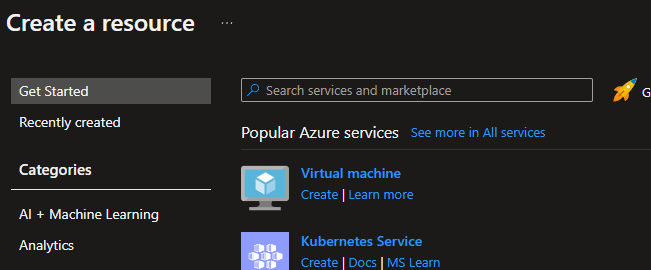
$schema: Specifies the location of the JSON schema file. The schema file describes the properties that are available within a template. The schema, for example, defines resources as one of the valid properties for a template. Don't worry that the date for the schema is 2019-04-01. This schema version is up to date and includes all of the latest features. The schema date hasn't been changed because there have been no breaking changes since its introduction.

contentVersion: Specifies the 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 you deploy resources using the template, you can use this value to make sure you're using the right template.

resources: Contains the resources you want to deploy or update.

### **Generating Templates from Azure Portal**

1. In Azure, click on Create a resource and type the resource you need to create the ARM template for.



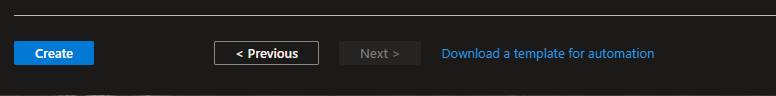
Here I will be deploying a storage account.

1. Input the required values (these can be changed later on)
2. Once you’rthe e in review page, ensure there are no errors:

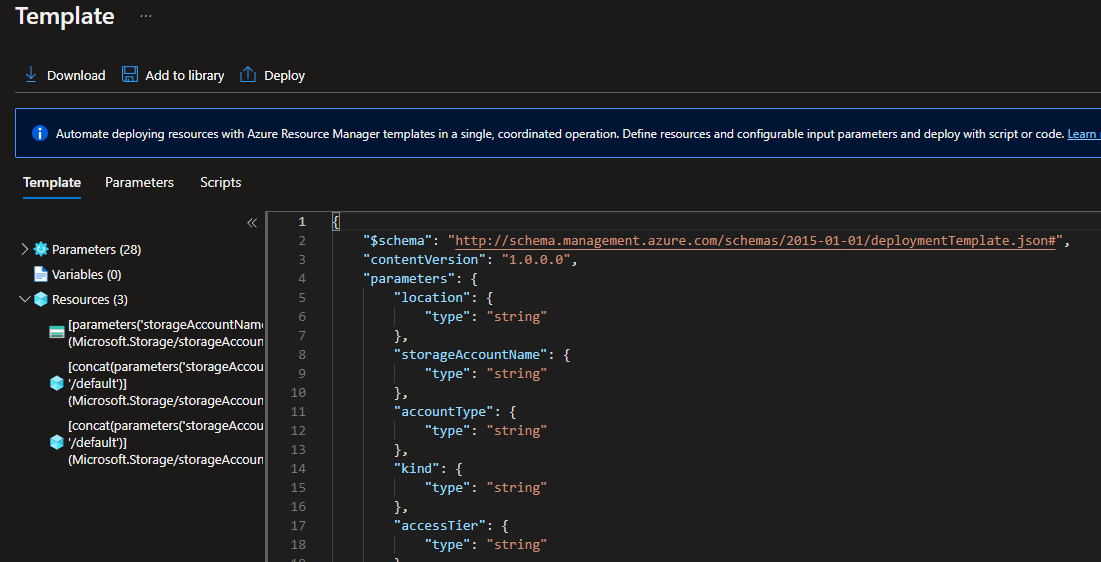
Text

Description automatically generated

1. At the botton of the page, click on Download a Template Option



1. This will open up a new page and will have option to save/download the template.



1. This can be used as a reference and a starting point for the templates.

### **Using Quick Start Templates**

Rather than manually building an entire ARM template, the easiest way is to get the templates from Quick Start repo. The Azure QuickStart Templates repo contains a large collection of templates that deploy common scenarios.

Link : <https://github.com/Azure/azure-quickstart-templates>

## **Deploying an ARM template**

To deploy an ARM template, you can use the Azure portal or the Azure CLI. When creating an ARM template, we recommend that you use a text editor (like VS Code) with JSON formatting to ensure that your JSON file is valid and easy to read.

You can deploy an ARM template in two ways:

- Use the Azure Portal

- Use the Azure CLI

When you deploy an ARM template, you specify the location of your template file and parameters that are used to customize your deployment. The following example shows how to deploy a basic VM using an ARM template:

az group deployment create --resource-group myResourceGroup --template-uri https://mystorageaccount.blob.core.windows.net/mycontainer/mytemplate.

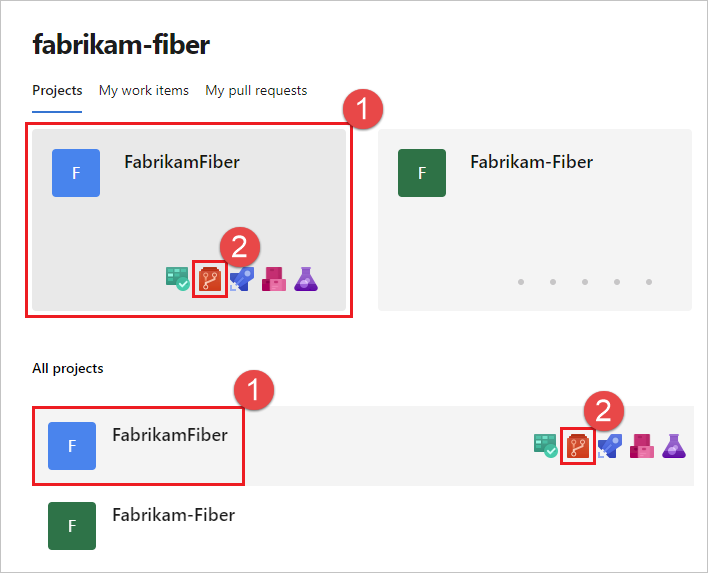
# **Azure DevOps for ARM Templates**

ARM Templates can be hosted and run using various DevOps tools. For the purpose of this document, we will use Azure DevOps.

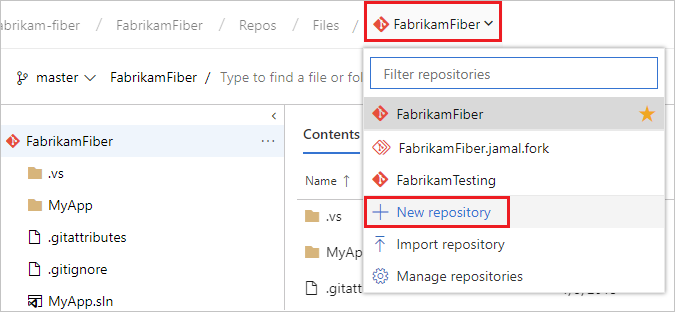
## **Creating a repository**

Repository provides version control for your artifacts, while providing capability for multiple users to collaborate in an independent and accountable manner.

1. Navigate to the **Repos** page in your project and hovering your mouse over the name of your project, then selecting the **Repos** icon. You can select it from the **All** projects list, or from the **Recent** projects list if you have accessed it recently.



1. From the repo drop-down, select **New repository**.



1. In the **Create a new repository** dialog, verify that Git is the repo type and enter a name for your new repo. You can also choose to add a README and create a .gitignore for the type of code you plan to manage in the repo. A README contains information about the code in your repo, and a .gitignore file tells Git which types of files to ignore, such as temporary build files from your development environment.

Graphical user interface, text, application, email

Description automatically generated

1. When you're happy with the repo name and choices, select **Create**.

A new empty Git repo is now created in your project.

You can now create and work with your ARM Templates configurations in that repo.

|  |  |
| --- | --- |
| ! | Important Note  *While Azure DevOps web interface supports creating and modification of text files like ARM Templates configurations, IDEs like Visual Studio Code is recommended.* |

## **Service Connection**

Your Azure DevOps require a credential to run that has permission to provision and modify Azure resources. In earlier steps, you have signed in using az cli using your own named account. For DevOps, a service account is required for this purpose and the credential must be safely stored in the DevOps project.

This is achieved via service connection in Azure DevOps.

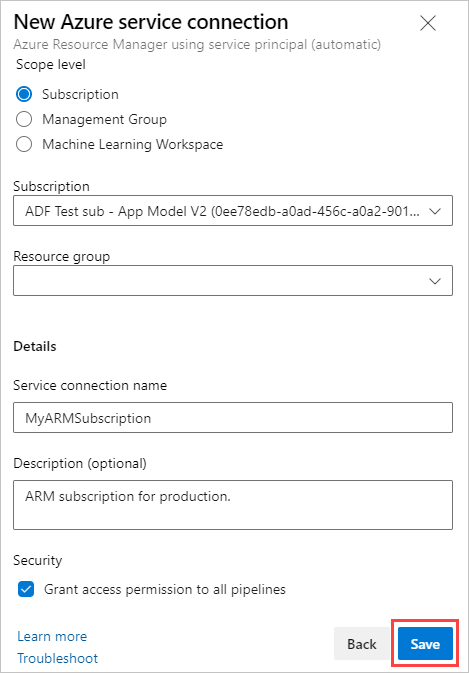
|  |  |
| --- | --- |
| ! | Important Note  *When using Service Principal (automatic) option described below to succeed, perform this action signed into the Azure DevOps with the named account that also has contributor access to the Azure subscription.* |

1. In Azure DevOps, select your project. Navigate to Project settings > Service connections
2. Select + New Service connection.

Graphical user interface, text, application, email

Description automatically generated

1. Select Azure Resource Manager
2. Select Service Principal (automatic)
3. Select Subscription as Scope level and wait for the Subscription name you have access to be retrieved
4. Select the desired subscription and populate the Service Connection Name



1. Click Save

# **YAML Pipeline in Azure DevOps**

## **5.1 What is a YML pipeline?**

In Azure DevOps, a YAML pipeline is a set of instructions that will be executed in an automated fashion when certain conditions are met. YAML stands for "YAML Ain't Markup Language", and is often used for configuration files. In a YAML pipeline, each step is defined in a YAML file, and these files are then stored in a central location (usually in source control). When someone makes a change to one of the files, the pipeline will automatically re-run the affected steps.

YAML pipelines have several advantages over traditional "build" pipelines. First, they are much easier to read and understand. Second, they are more flexible and can be easily modified without having to go through a lengthy approval process.

## **5.2 Creating a YML pipeline in Azure DevOps**

Pipelines in Azure DevOps are defined in YAML files, allowing for greater control and customization than the visual editor.

To create a new pipeline, select "New Pipeline" from the Pipelines page in Azure DevOps. This will open the "Create Pipeline" wizard. Choose "Starter pipeline" to have a basic YAML file generated automatically.

The generated file will be placed in the root of your repository and named azure-pipelines.yml. You can edit this file directly to add more stages, jobs, or steps to your pipeline.

To learn more about what you can do with YAML pipelines in Azure DevOps, check out the documentation: https://docs.microsoft.com/en-us/azure/devops/pipelines/yaml-schema?

## **5.3 Managing YML pipelines in Azure DevOps**

Azure DevOps makes it easy to manage YML pipelines. With a few clicks, you can create, edit, and delete YML pipelines. You can also view the history of your YML pipelines and see when they were last run.

To create a new YML pipeline, click the "New Pipeline" button in the Azure DevOps interface. This will open the "Create Pipeline" page. On this page, you will need to select a repository type (GitHub, Bitbucket, or Azure Repos), and then select a template (Basic or Starter). Once you have selected a template, click the "Create Pipeline" button.

To edit an existing YML pipeline, click the "Edit" button in the Azure DevOps interface. This will open the "Edit Pipeline" page.

## **5.4 Best practices for using YML pipelines in Azure DevOps**

There are a few best practices that you should follow when using YAML pipelines in Azure DevOps:

1. Use comments liberally to document your pipeline.azure-pipelines.yml file. This will help other developers understand what your pipeline is doing and why.

2. Check your pipeline into source control so that it can be easily reviewed and updated by other members of your team.

3. Be thoughtful about which tasks should be run in parallel and which tasks should be run sequentially.

# **Deploy ARM Templates using Azure DevOps.**

Azure DevOps is a great tool for automating the process of deploying ARM templates. In this article, we will walk through the steps necessary to deploy an ARM template using Azure DevOps.

First, you will need to create an Azure DevOps account and install the Azure Resource Manager extension. (if not done already)

Next, you will need to create a new Azure Pipelines build definition. In the build definition, you will need to add a task to deploy the ARM template.

Finally, you will need to queue a new build and wait for it to complete. Once the build completes successfully, you will have successfully deployed your ARM template using Azure DevOps!

## **6.1 Using the repo.**

All the pipelines to deploy resources are located here:

Text

Description automatically generated

## Resource Pipelines

Resource specific pipelines are available under specific resource

Text

Description automatically generated

## 6.3 Master Pipelines

The master pipelines has multiple resources and the resources can be added/removed as per requirement.

Text

Description automatically generated

# **Reusing the Pipelines and Templates.**

## Getting help

The read me files are under each resource – to check the dependencies, required parameters and optional ones.

A screenshot of a computer

Description automatically generated with medium confidence

## YAML-Templates

This folder contains all resource specific folders along with pipelines and then a master pipeline as shown:

Text

Description automatically generated

Text

Description automatically generated

The master pipeline contains all resources – for reference.

## To deploy specific resources:

1. Refer the read me file to know the dependencies:

Graphical user interface, application

Description automatically generated

1. Check the yaml pipeline for the specific resource inputs:

Graphical user interface, application

Description automatically generated

1. Add required resources to one pipeline and refer them to resource specific ones as shown:

Text

Description automatically generated

1. Deploy after updating the agent pool, parameters and variables / variable groups, along with service principle for connection:

Text

Description automatically generated