Introduction

Completed 100 XP

- 3 minutes

Scenario

Your company needs to ensure virtual machine deployments are consistent across the organization.

You use Azure Resource Manager templates to deploy resources including virtual machines.

Skills measured

Deploying resources using Azure Resource Manager templates is part of Exam AZ-104: Microsoft Azure Administrator.

Deploy and manage Azure compute resources (20–25%)

Automate deployment of virtual machines (VMs) by using Azure Resource Manager templates

- Modify an Azure Resource Manager template.
- Deploy from a template.
- Save a deployment as an Azure Resource Manager template.

Learning objectives

In this module, you'll learn how to:

- List the advantages of Azure templates.
- Identify the Azure template schema components.
- Specify Azure template parameters.
- Locate and use Azure Quickstart Templates.

Prerequisites

None

Next unit: Review Azure Resource Manager template advantages

Review Azure Resource Manager template advantages

Completed 100 XP

- 3 minutes

An **Azure Resource Manager template** precisely defines all the Resource Manager resources in a deployment. You can deploy a Resource Manager template into a resource group as a single operation.

Using Resource Manager templates will make your deployments faster and more repeatable. For example, you no longer have to create a VM in the portal, wait for it to finish, and then create the next VM. Resource Manager template takes care of the entire deployment for you.

Template benefits

- **Templates improve consistency**. Resource Manager templates provide a common language for you and others to describe your deployments. Regardless of the tool or SDK that you use to deploy the template, the structure, format, and expressions inside the template remain the same.
- **Templates help express complex deployments**. Templates enable you to deploy multiple resources in the correct order. For example, you wouldn't want to deploy a virtual machine prior to creating an operating system (OS) disk or network interface. Resource Manager maps out each resource and its dependent resources, and creates dependent resources first. Dependency mapping helps ensure that the deployment is carried out in the correct order.
- **Templates reduce manual, error-prone tasks**. Manually creating and connecting resources can be time consuming, and it's easy to make

mistakes. Resource Manager ensures that the deployment happens the same way every time.

- **Templates are code**. Templates express your requirements through code. Think of a template as a type of Infrastructure as Code that can be shared, tested, and versioned similar to any other piece of software. Also, because templates are code, you can create a "paper trail" that you can follow. The template code documents the deployment. Most users maintain their templates under some kind of revision control, such as GIT. When you change the template, its revision history also documents how the template (and your deployment) has evolved over time.
- **Templates promote reuse**. Your template can contain parameters that are filled in when the template runs. A parameter can define a username or password, a domain name, and so on. Template parameters enable you to create multiple versions of your infrastructure, such as staging and production, while still using the exact same template.
- **Templates are linkable**. You can link Resource Manager templates together to make the templates themselves modular. You can write small templates that each define a piece of a solution, and then combine them to create a complete system.
- **Templates simplify orchestration**. You only need to deploy the template to deploy all of your resources. Normally this would take multiple operations.

Next unit: Explore the Azure Resource Manager template schema

Explore the Azure Resource Manager template schema

- 3 minutes

Azure Resource Manager templates are written in JSON, which allows you to express data stored as an object (such as a virtual machine) in text. A JSON document is essentially a collection of key-value pairs. Each key is a string, whose value can be:

- A string
- A number
- A Boolean expression

- A list of values
- An object (which is a collection of other key-value pairs)

A Resource Manager template can contain sections that are expressed using JSON notation, but aren't related to the JSON language itself:

JSONCopy

```
{
    "$schema": "http://schema.management.azure.com/schemas/2019-04-
01/deploymentTemplate.json#",
    "contentVersion": "",
    "parameters": {},
    "variables": {},
    "functions": [],
    "resources": [],
    "outputs": {}
}
```

Element name	Required	Description
$schema	Yes	Location of the JSON schema file that describes the version of the template language. Use the URL shown in the preceding example.
contentVersion	Yes	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. This value can be used to make sure that the right template is being used.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group.
outputs	No	Values that are returned after deployment.

Next unit: Explore the Azure Resource Manager template parameters

Explore the Azure Resource Manager template parameters

Completed100 XP

- 3 minutes

In the parameters section of the template, you specify which values you can input when deploying the resources. The available properties for a parameter are:

JSONCopy

```
"parameters": {
    "<parameter-name>" : {
        "type" : "<type-of-parameter-value>",
        "defaultValue": "<default-value-of-parameter>",
        "allowedValues": [ "<array-of-allowed-values>" ],
        "minValue": <minimum-value-for-int>,
        "maxValue": <maximum-value-for-int>,
        "minLength": <minimum-length-for-string-or-array>,
        "maxLength": <maximum-length-for-string-or-array-parameters>,
        "metadata": {
        "description": "<description-of-the parameter>"
        }
    }
}
```

Here's an example that illustrates two parameters: one for a virtual machine's username, and one for its password:

JSONCopy

```
"parameters": {
  "adminUsername": {
    "type": "string",
    "metadata": {
      "description": "Username for the Virtual Machine."
    }
  },
  "adminPassword": {
    "type": "securestring",
    "metadata": {
      "description": "Password for the Virtual Machine."
    }
  }
}
```

Note

You're limited to 256 parameters in a template. You can reduce the number of parameters by using objects that contain multiple properties.

Next unit: Consider Bicep templates

Consider Bicep templates

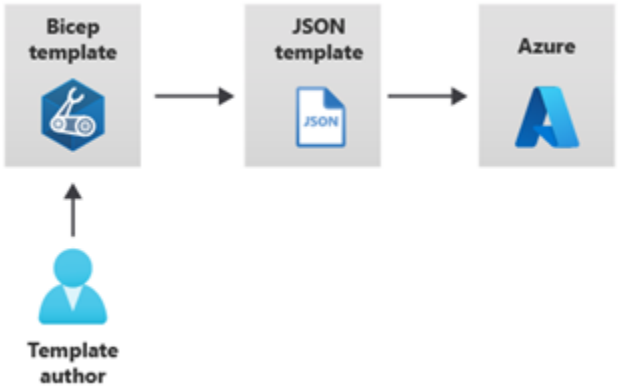
- 3 minutes

Azure Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. It provides concise syntax, reliable type safety, and support for code reuse.

You can use Bicep instead of JSON to develop your Azure Resource Manager templates (ARM templates). The JSON syntax to create an ARM template can be verbose and require complicated expressions. Bicep syntax reduces that complexity and improves the development experience. Bicep is a transparent abstraction over ARM template JSON and doesn't lose any of the JSON template capabilities.

How does Bicep work?

When you deploy a resource or series of resources to Azure, you submit the Bicep template to Resource Manager, which still requires JSON templates. The tooling that's built into Bicep converts your Bicep template into a JSON template. This process is known as transpilation. Transpilation is the process of converting source code written in one language into another language.



Bicep provides many improvements over JSON for template authoring, including:

- **Simpler syntax**: Bicep provides a simpler syntax for writing templates. You can reference parameters and variables directly, without using complicated functions. String interpolation is used in place of concatenation to combine values for names and other items. You can reference the properties of a resource directly by using its symbolic name instead of complex reference statements. These syntax improvements help both with authoring and reading Bicep templates.

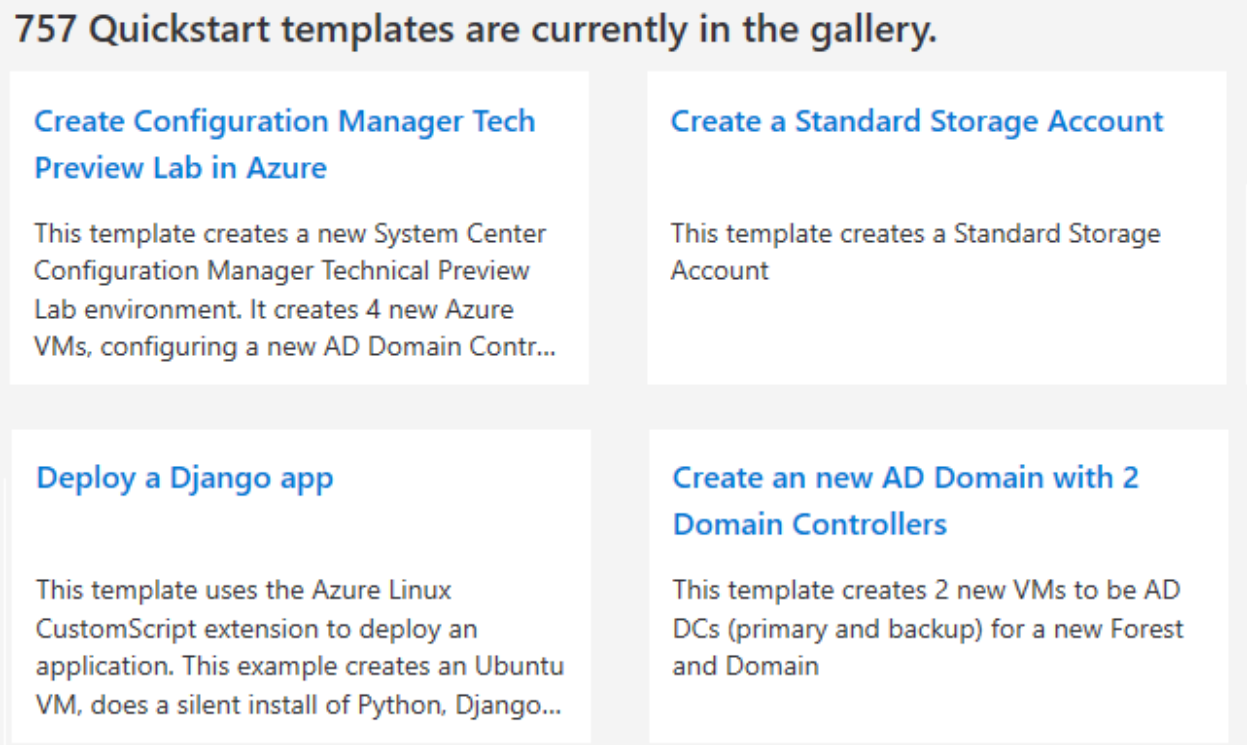
- **Modules**: You can break down complex template deployments into smaller module files and reference them in a main template. These modules provide easier management and greater reusability.
- **Automatic dependency management**: In most situations, Bicep automatically detects dependencies between your resources. This process removes some of the work involved in template authoring.
- **Type validation and IntelliSense**: The Bicep extension for Visual Studio Code features rich validation and IntelliSense for all Azure resource type API definitions. This feature helps provide an easier authoring experience.

Next unit: Review QuickStart templates

Review QuickStart templates

- 3 minutes

Azure Quickstart Templates are Azure Resource Manager templates provided by the Azure community.



Some templates provide everything you need to deploy your solution, while others might serve as a starting point for your template. Either way, you can study these templates to learn how to best author and structure your own templates.

- The README.md file provides an overview of what the template does.
- The azuredeploy.json file defines the resources that will be deployed.
- The azuredeploy.parameters.json file provides the values the template needs.

Note

Take a few minutes to browse the available templates. Anything of interest?

Next unit: Interactive lab simulation - templates

Interactive lab simulation - templates

Completed 100 XP

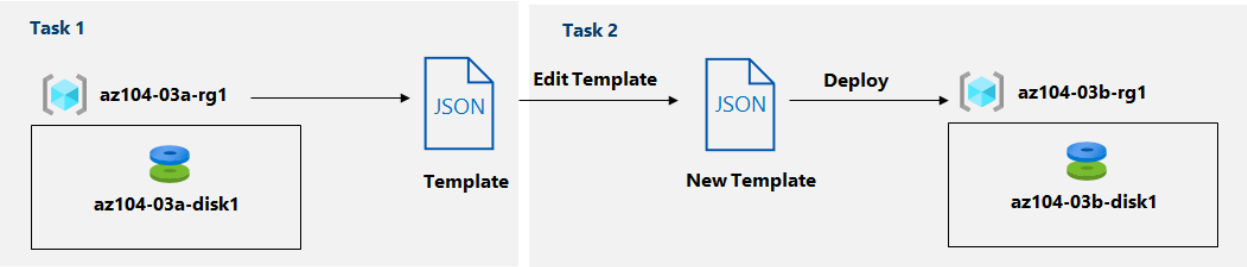
- 17 minutes

Scenario

Tailwind Traders are migrating their storage needs to Azure. You've successfully deployed a managed disk in a resource group. You've decided to create an Azure Resource Manager template to simplify the other disk deployments.

Architecture diagram

Your first disk deployment in the resource group az104-**03a**-rg1 is complete. You plan to customize the template and use it to deploy another disk in resource group az104-**03b**-rg1.



Tasks

Task 1: Review an ARM template for deployment of an Azure managed disk.

Task 2: Create an Azure managed disk by using an ARM template.

Task 3: Review the ARM template-based deployment of the managed disk.

Note

Click on the thumbnail image to start the lab simulation. When you're done, be sure to return to this page so you can continue learning.

Next unit: Knowledge check

Summary and resources

Completed 100 XP

- 3 minutes

To implement infrastructure as code for your Azure solutions, use Azure Resource Manager templates. The template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project. The template uses declarative syntax, which lets you state what you intend to deploy without having to write the sequence of programming commands to create it. In the template, you specify the resources to deploy and the properties for those resources.

You should now be able to:

- List the advantages of Azure templates.
- Identify the Azure template schema components.
- Specify Azure template parameters.
- Locate and use Azure Quickstart Templates.

Learn more

You can learn more by reviewing the following. A *sandbox* indicates a hands-on exercise.

- Azure Resource Manager template documentation
- Azure Quickstart Templates
- Deploy Azure infrastructure by using JSON Azure Resource Manager templates (Sandbox)
- Create Azure resources using Azure Resource Manager templates
- Build your first Bicep template (Sandbox)

Module complete: