Describe Azure Resource Manager and Azure ARM templates

6 minutes

Azure Resource Manager (ARM) is the deployment and management service for Azure. It provides a management layer that enables you to create, update, and delete resources in your Azure account. Anytime you do anything with your Azure resources, ARM is involved.

When a user sends a request from any of the Azure tools, APIs, or SDKs, ARM receives the request. ARM authenticates and authorizes the request. Then, ARM sends the request to the Azure service, which takes the requested action. You see consistent results and capabilities in all the different tools because all requests are handled through the same API.

Azure Resource Manager benefits

With Azure Resource Manager, you can:

- Manage your infrastructure through declarative templates rather than scripts. A Resource Manager template is a JSON file that defines what you want to deploy to Azure.
- Deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- Re-deploy your solution throughout the development life-cycle and have confidence your resources are deployed in a consistent state.
- Define the dependencies between resources, so they're deployed in the correct order.
- Apply access control to all services because RBAC is natively integrated into the management platform.
- Apply tags to resources to logically organize all the resources in your subscription.
- Clarify your organization's billing by viewing costs for a group of resources that share the same tag.

The following video provides an overview of how you can use different Azure tools with ARM to manage your environment:

ARM templates

Infrastructure as code is a concept where you manage your infrastructure as lines of code. Leveraging Azure Cloud Shell, Azure PowerShell, or the Azure CLI are some examples of using code to deploy cloud infrastructure. ARM templates are another example of infrastructure as code at work.

By using ARM templates, you can describe the resources you want to use in a declarative JSON format. With an ARM template, the deployment code is verified before any code is run. This ensures that the resources will be created and connected correctly. The template then orchestrates the creation of those resources in parallel. That is, if you need 50 instances of the same resource, all 50 instances are created at the same time.

Ultimately, the developer, DevOps professional, or IT professional needs only to define the desired state and configuration of each resource in the ARM template, and the template does the rest. Templates can even execute PowerShell and Bash scripts before or after the resource has been set up.

Benefits of using ARM templates

ARM templates provide many benefits when planning for deploying Azure resources. Some of those benefits include:

- **Declarative syntax**: ARM templates allow you to create and deploy an entire Azure infrastructure declaratively. Declarative syntax means you declare what you want to deploy but don’t need to write the actual programming commands and sequence to deploy the resources.
- **Repeatable results**: Repeatedly deploy your infrastructure throughout the development lifecycle and have confidence your resources are deployed in a consistent manner. You can use the same ARM template to deploy multiple dev/test environments, knowing that all the environments are the same.
- **Orchestration**: You don't have to worry about the complexities of ordering operations. Azure Resource Manager orchestrates the deployment of interdependent resources, so they're created in the correct order. When possible, Azure Resource Manager deploys resources in parallel, so your deployments finish faster than serial deployments. You deploy the template through one command, rather than through multiple imperative commands.
- **Modular files**: You can break your templates into smaller, reusable components and link them together at deployment time. You can also nest one template inside another template. For example, you could create a template for a VM stack, and then nest that template inside of templates that deploy entire environments, and that VM stack will consistently be deployed in each of the environment templates.

- **Extensibility**: With deployment scripts, you can add PowerShell or Bash scripts to your templates. The deployment scripts extend your ability to set up resources during deployment. A script can be included in the template or stored in an external source and referenced in the template. Deployment scripts give you the ability to complete your end-to-end environment setup in a single ARM template.

Next unit: Knowledge check

Continue >